Provably-Good Approximation Algorithms for Optimal Kinodynamic Robot Motion Plans

Patrick Gordon Xavier
Ph.D Thesis

92-1279
April 1992

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

PROVABLY-GOOD APPROXIMATION ALGORITHMS FOR OPTIMAL KINODYNAMIC ROBOT MOTION PLANS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Patrick Gordon Xavier

May 1992

PROVABLY-GOOD APPROXIMATION ALGORITHMS FOR OPTIMAL KINODYNAMIC ROBOT MOTION PLANS

Patrick Gordon Xavier, Ph.D.
Cornell University 1992

The kinodynamic planning problem is to synthesize a robot motion obeying simultaneous kinematic and dynamics constraints. To maximize robot performance we can consider *optimal* kinodynamic planning: for a given robot system, find a minimal-time trajectory that goes from a start state to a goal state, avoids obstacles by a speed-dependent safety margin, and respects the dynamics laws and dynamics bounds governing the system. In general, previous work on algorithmic motion planning does not address dynamics; furthermore, even in simple cases, finding exact globally-optimal solutions is $\mathcal{NP}$-hard. In response, we obtain provably-good, polynomial-time approximation algorithms that synthesize optimal kinodynamic trajectories. These algorithms forge new mathematical links between control theory and complexity theory, and our analysis investigates how discrete-control trajectories can approximate optimal solutions.

We cast optimal kinodynamic planning into the form of an ϵ-approximation problem, in which $\epsilon > 0$ characterizes closeness to optimality in terms of trajectory time, observance of the safety margin, and closeness to the start and goal states.

If T_{opt} is the time of an optimal trajectory, then an ϵ-optimal trajectory takes at most $(1 + \epsilon)T_{opt}$ time. We present (pseudo)-polynomial-time ϵ-approximation algorithms for a family of robot classes, including fully-controllable open kinematic chains. These algorithms run in time polynomial in $\frac{1}{\epsilon}$ and the geometric complexity of the constraints.

The basic idea behind the algorithms is to reduce the trajectory planning problem to a shortest-path problem on a polynomial-sized reachability graph embedded in the robot state space. These graphs are generated by control primitives and a timestep that the algorithm chooses to ensure ϵ-optimality. To obtain our complexity and approximation results, we introduce both continuous and combinatorial tools to analyze the robot's dynamical system. These include *scaling-tracking proof methods* that capture the key insight necessary for provably-good results, *tracking lemmas* on how closely we can approximate an optimal or time-rescaled optimal trajectory, *constructive trajectory proofs*, *adversary game proofs*, and *Time-Safety planning trade-offs*. We also describe an implementation and experiments in a restricted domain.

Biographical Sketch

Patrick Gordon Xavier was born on November 17, 1962, in Euclid, Ohio. It is only natural that his thesis research has a geometrical flavor.

On the brink of adulthood, Patrick learned the art of cooking omelettes and completed high school in 1981 at Choate Rosemary Hall in Wallingford, Connecticut.

Patrick ventured into Physics, Chinese cooking, and Computer Science at Harvard University, where he was awarded the degree of Bachelor of Arts, *magna cum laude*, in Applied Mathematics in June 1985. Later that year he bought a heavy steel wok and moved to Ithaca, New York, to pursue graduate studies at Cornell University. Faced with an electric range in 1986, he became adept at preparing many foods on the barbeque grill by the time Cornell awarded him a Master of Science degree in Computer Science in August 1988. In the meantime, he discovered a love for bicycling.

Over the next few years, Patrick experimentally determined that Pop-Tarts and Fig Newtons are good foods to eat on a century ride, and he also figured out how to use his wok on an electric range. He eventually wrote this dissertation, for which Cornell University is awarding him a Ph.D. in Computer Science in May 1992.

To my parents, Cecilia Marie Xavier and Vincent Louis Xavier, Jr.

Acknowledgements

Many thanks to my patient committee members, Bruce Donald, Allen Hatcher, Prakash Panangaden, and Alberto Segre and to proxy member Dan Huttenlocher. Bruce has been a fine advisor and friend, and I'm proud to have been his first Ph.D. student. Prakash deserves special gratitude for his friendship and valuable counsel, especially in my early years at Cornell. Alex Nicolau and Jon Solworth also have my gratitude for their guidance during my pre-robotics days. Although Dexter Kozen was never on my committee, he was supportive of my progress as a graduate student beyond his duties as GFR.

Outside of the Cornell Computer Science faculty, Jeff Koechling and Dinesh Pai deserve thanks for much information and advice about robotics and research. When I had a computational geometry question I could always turn to Robert Freimer and Christine Piatko. Bill Aitken and I had many valuable discussions on research and its presentation, as well as many enjoyable conversations on various and sundry topics.

Laura Sabel and Mike Hopcroft were ideal officemates during the last crucial months. They remained congenial, if not downright friendly, even when my thesis-related sprawl threatened to cover every inch of spare deskspace. Also, Laura's tape and scissors have been invaluable in the preparation of certain pages of this

document as well as several papers.

While I was in Ithaca, I was fortunate to have many friends who helped make my stay enjoyable and without whom keeping life in perspective would have been a trying task. Most notable in deserving my thanks are former housemates Bob Mihailovich and Harry Barshatzky, and Lynne Hsu, Bill Aitken, Mike DiMilia, Deb Hild, Steve Rapkin, and Kim Williams. I'm also grateful to the Ithaca Hash House Harriers for 26 fun-filled hash runs and associated activities, and to the Robotlab Lunch Society for many entertaining meals.

Finally, thanks to Mom and Dad for sending me down the right track, from teaching me the alphabet (which I still use today) to paying for college.

Table of Contents

List of Figures

Chapter 1

Introduction

To be useful, a robot must be capable of moving without colliding with objects in its environment or damaging itself. If, furthermore, a robot is to be productive in its function, its motions should take as little time as possible. Thus, in addition to competence ("Can your robot do X?"), we must be concerned with performance ("How long does it take for your robot to do X?").

A basic problem is to plan motions for a robot in a known environment. Minimum-time obstacle-avoiding motions are determined by the dynamical and control characteristics of the robot as well as the shapes and locations of the obstacles. Ideally, we would like to guarantee a robot's performance relative to its intrinsic physical, control, and sensory capabilities. Since the computation time required by a motion planner itself will limit how the robot can interact with the world, a planner that quickly synthesizes minimum-time motions would be desirable.

Unfortunately, even finding a shortest path can be computationally intractable in the $\mathcal{NP}$-hard sense [CR87]. Since it can be shown that finding a minimum-time motion is at least as hard,[1] *approximation* algorithms are a reasonable approach to planning time-optimal motions.

Natural questions arise about any approximation algorithm for finding time-optimal motions:

- How close to optimal is the solution?
- What is the running time (computational complexity) of the algorithm?

[1]See appendix A. Intuitively, we might expect the minimal-time problem to be harder, since it takes into account dynamics, and the shortest-path problem does not.

- Can the running time of the algorithm be related to the guaranteed solution accuracy?

Approximation and heuristic techniques have been proposed. However, prior to the results in this thesis, there existed no bounds on the goodness of the resulting solutions, or on the time-complexity of the algorithms. Note that by "time-optimal", we will always mean *globally* time-optimal, that is, time-optimal among all possible trajectories.[2]

1.1 An Approach

1.1.1 Observations

There is a wide spectrum of situations in which it is important for robot motions to be time-efficient. How long it takes to complete given tasks in a factory largely determines how productive and profitable that factory will be. While minimizing the time of obstacle-avoiding robot motions can speed up an assembly process only to a point, in certain settings — say, warehouses, shipping rooms, and docks — these motions would be a major bottleneck. In other situations the benefits of a robot being able to minimize its motion time would not just be monetary: how quickly toxic, radioactive, explosive, or otherwise hazardous materials can be contained or removed at the site of an accident or leak, and even how quickly the site can be surveyed, is important in limiting the damage to life, property, and the environment. In general, the less time it takes for robots to perform tasks, the less often people will need to subject themselves to dangerous environments. For example, in outer space, robots that are both competent *and fast* would enable missions to be accomplished within time constraints with a minimum of human extra-vehicular activity.

Dynamics laws, applied controls, and constraints determine the motion of a robot.[3] If a motion is to be reliably executable by a robot, that motion must obey constraints on dynamical and control quantities, such as acceleration, velocity, and applied and internal torques and forces. While these constraints may directly

[2] We strongly distinguish *time-optimal* from *locally*-time-optimal.

[3] While the effects of dynamics on a planned robot motion appear whenever inertia is high compared to available torques and forces, they are most dramatically illustrated when one considers using the Remote Manipulator System on the NASA Space Shuttle to move an object matching its payload rating — nearly half the mass of the Shuttle. [LLZ90].

reflect physical and control limitations, we might also impose "tighter" constraints — to reduce wear or power requirements or to use a simpler physical model, for example. Bounds on the velocities and forces that a payload or end-effector can tolerate would also result in additional constraints.

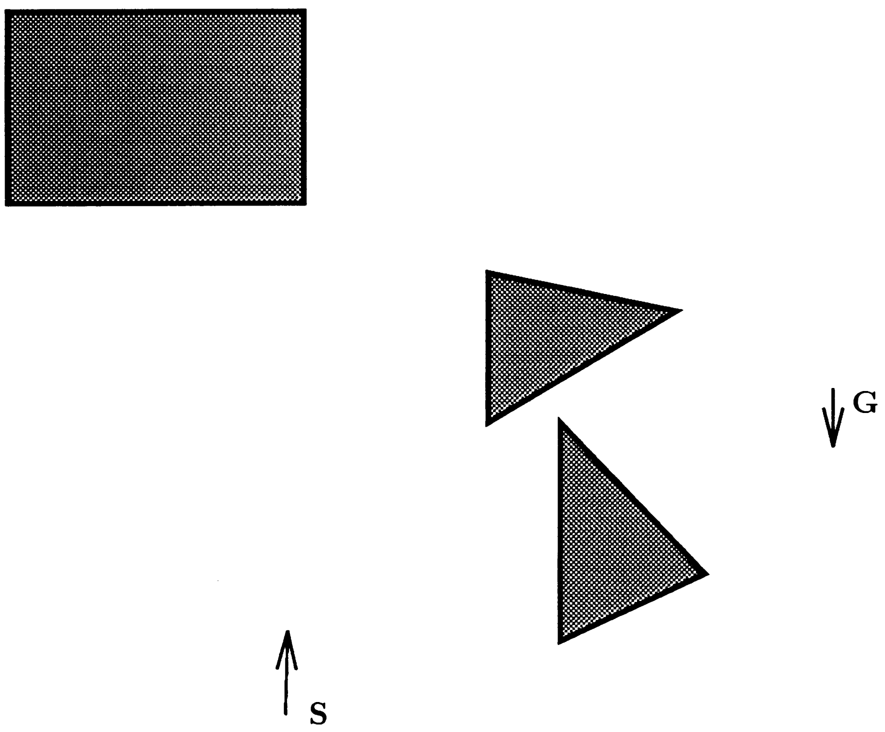


Figure 1.1: A kinodynamic planning problem for a point mass robot, showing the obstacles, the start $\mathbf{S} = (\mathbf{s}, \dot{\mathbf{s}})$, the goal $\mathbf{G} = (\mathbf{g}, \dot{\mathbf{g}})$. The point is subject to a maximum speed $\overline{v}$ and a maximum acceleration $\overline{a}$. Solutions and a speed-dependent safety margin are illustrated in figure 1.2.

Execution time, obstacle-avoidance, and dynamics are not the only aspects we need to consider about robot motions. We must also take into account how much time is spent planning them. If a motion task is to be repeated often enough, then it might be profitable spend a long time to plan nearly time-optimal movements. If a task will be executed just once, but as a part of a planned mission whose total time we wish to minimize, we might allocate planning time to a motion in proportion to how much its execution time will affect the total mission time. When the the usefulness of a motion will be short-lived or when the robot environment can change quickly, the motion planning problem is inherently "on-line", and it is the time required to plan *plus* the time required execute the motion that should be minimized. Thus, we would like to plan time-efficient robot motions as best we can *in an amount of time reasonable for the problem.* We therefore would hope for an analytical relationship between the running time bound of an approximation algorithm and the guaranteed quality, compared to the theoretical optimal, of a motion it plans.

1.1.2 Approximation Algorithms

Using techniques from several fields — including mechanical engineering, control theory, and computer science — researchers have introduced algorithms to generate robot motions that, to various degrees, are approximately time-optimal. Combinatorial optimization provides a general framework for analyzing the relationship between the running time of an *approximation algorithm* for an optimization problem and the quality of the solutions it is guaranteed to find. In particular, for optimization problems over discrete (*i.e.,* integer) variables, there is the notion of *ϵ-approximation algorithms* [PS82], which we will borrow and extend.

Consider an algorithm for some optimization problem in which a cost L is to be minimized. If for each problem instance $\mathcal{K}_i$ and approximation parameter $\epsilon > 0$ the algorithm is guaranteed to find a solution with cost no more than $(1+\epsilon)L_i$, where L_i is the cost of the optimum solution, then it is an *ϵ-approximation algorithm.* The *(time-)complexity* of the algorithm is the worst-case bound on its running time over all problem instances in terms of ϵ, the length of the encoding[4] of the problem instance, and the largest number in the encoding. Thus, the complexity of an ϵ-approximation algorithm describes how quickly its running time can grow as we demand solutions guaranteed to be closer and closer to optimal in cost.

If the complexity of the algorithm is polynomial in $\frac{1}{\epsilon}$, the length of the encoding, and the largest number in the encoding, then the algorithm is called a *(pseudo-)polynomial-time approximation scheme.* The algorithm is a *fully polynomial-time approximation scheme* if its complexity is polynomial in terms of both $\frac{1}{\epsilon}$ and the length of the encoding.[5]

[4]By an *encoding* we simply mean a representation using a fixed, finite alphabet.

[5]*I.e.,* $O((|\mathcal{K}_i|^{O(1)}(\frac{1}{\epsilon})^{-O(1)})$ [PS82].

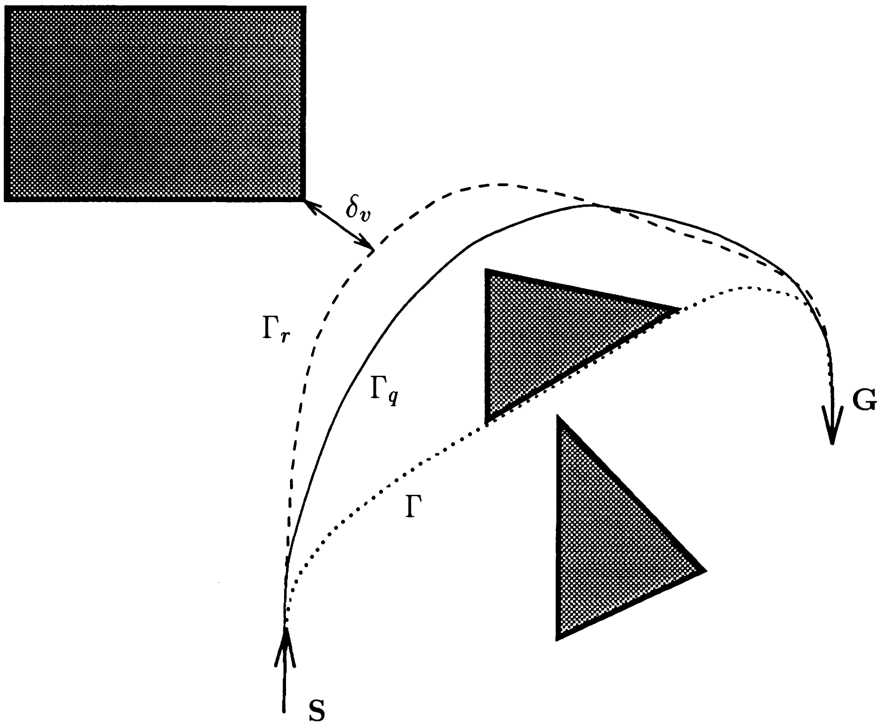


Figure 1.2: A kinodynamic planning problem for a point robot, which obeys particle dynamics ($\mathbf{f} = m\mathbf{a}$). This figure shows the obstacles, the start **S**, the goal **G**, and three illustrated trajectories: time-optimal Γ, optimal (safe) kinodynamic Γ_r, and approximately optimal kinodynamic Γ_q, which happens to be exact at the start and goal. All three obey the velocity and acceleration bounds $\overline{v}$ and $\overline{a}$. However, the time-optimal *motion* Γ leaves no margin for obstacle avoidance, whereas Γ_r avoids obstacles by a margin δ_v, and Γ_q avoids them by a smaller distance $(1 - \epsilon)\delta_v$.

1.1.3 Provably Good Approximation Algorithms for Kinodynamic Planning

Because of control error, sensing uncertainty, and errors in the geometric models of robots and their environments, we cannot expect the motion *executed* by a robot to be exactly the same as the *planned* motion. Thus, the optimality of planned motions should be defined *relative* to these factors. In addition to including robot dynamics and obstacles, our *kinodynamic planning* formulation[6], proposed with Canny, Donald, and Reif in [CDRX88], accounts for control, modeling, and sensing error by demanding that a planned motion avoid all obstacles by a speed-dependent "safety margin". From the kinodynamic planning standpoint an *optimal kinodynamic trajectory* is therefore a time-optimal motion plan. (See figures 1.1 and 1.2.)

This thesis presents provably good approximation algorithms for a set of optimal kinodynamic planning problems within an ϵ-approximation framework. Essentially, we will say that a planned motion is ϵ-optimal if it (a) takes at most an ϵ-factor more time than an optimal motion and (b) avoids obstacles by a safety margin that is at most an ϵ-factor smaller than the desired margin. (See figure 1.2.) For each class of robots considered, the results depend on four basic assumptions:

1. a bounded environment,
2. velocity bounds,
3. force or acceleration bounds,
4. a positive, possibly speed-dependent, safety-margin, and
5. polyhedral obstacles (which may overlap arbitrarily).

The algorithms described here run in time polynomial in both $\frac{1}{\epsilon}$ and the geometric complexity of the problems. The most general class of robots we consider is that of fully-controllable (actuated in every degree of freedom and able to overcome gravity statically) d-link open kinematic chains.[7] (See figure 1.3.) We believe that the techniques developed to prove the correctness and complexity of our *provably-good polynomial-time approximation algorithms* for kinodynamic planning can be

[6] For a formal definition, see Section 2.1.1. This formulation is extended in Section 4.1.

[7] Thus, their dynamical behavior obeys a Lagrangian dynamics equation for (an open-kinematic chain of) rigid bodies with gravity. See equation (4.1).

further generalized, if not applied directly, to other motion planning algorithms and other classes of robots.

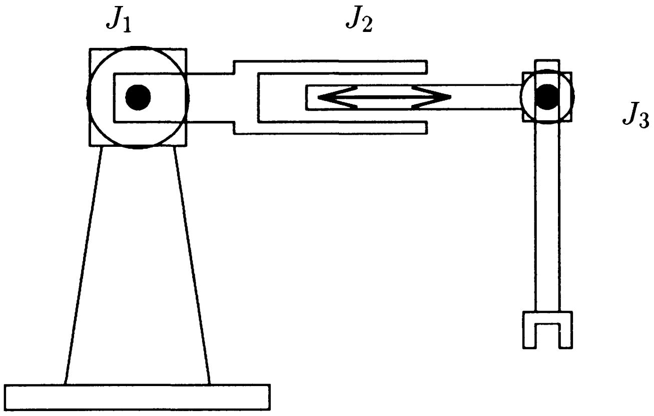


Figure 1.3: A planar, three-degree of freedom open-kinematic-chain ("open-chain") manipulator with two revolute joints (J_1, J_3) and one translational joint (J_2). An open kinematic chain is a serial linkage of rigid bodies, *i.e.*, it contains no loops. The most general class of robots we consider have are three-dimensional (*i.e.*, full-dimensional).

1.2 Kinodynamic Planning

1.2.1 Overview

A *kinodynamic trajectory* is a motion that obeys simultaneous *kinodynamic constraints*.[8] These constraints include *kinematic constraints*, which limit the configurations[9] a robot may assume, and *dynamics constraints*, which include bounds on velocity, acceleration, and other quantities that determine a robot's motion, as well as the dynamics laws that apply. Dynamics constraints are independent of

[8]The term "kinodynamic" describes the constraints relevant to time-optimal motion plans. Thus, optimal kinodynamic planning, as we describe later, is a *specific* constrained optimal control problem, as distinguished from general constrained optimization or problems obtained by "factoring" the time-optimal trajectory problem described in Section 1.2.3.

[9]A configuration is the generalized position of a robot, which specifies its location and pose.

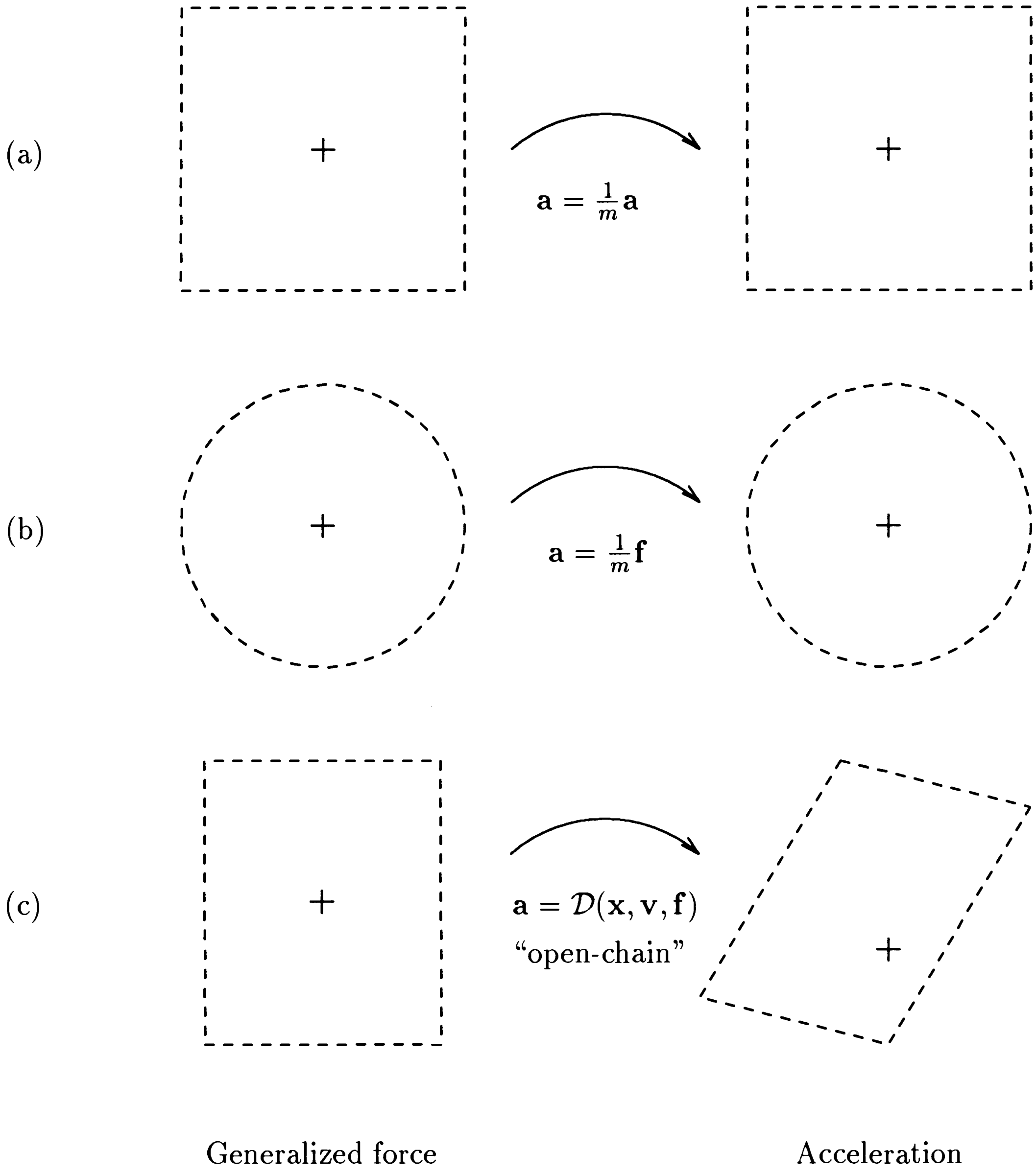


Figure 1.4: Maps between force bounds and acceleration bounds. Results cover robots with state-independent dynamics under (a) L_∞ and (b) L_2 dynamics bounds and (c) robots obeying open-kinematic-chain dynamics. See Section 4.1 and equation (4.7) for "open-chain" dynamics.

a robot's environment. Finally, *strictly kinodynamic constraints* are environment-dependent constraints that apply to the robot's state (configuration and velocity) and its time-derivatives but do not fall into either of the previous categories.

The kinodynamic planning framework encompasses several classes of motion planning problems. *Path planning problems* require one to find motions that satisfy kinematic constraints. In *trajectory planning* one attempts to find motions that satisfy kinematic constraints and dynamics constraints. This informal introduction to kinodynamic planning will review these problem classes first (in Sections 1.2.2 and 1.2.3), to provide a background for the kinodynamic planning problem that corresponds to the problem of finding a time-optimal motion plan.

Although simple examples in two dimensions will be used to help motivate and illustrate the kinodynamic planning framework, it generalizes to motion planning in arbitrary degrees of freedom. A typical example is seen in figure 1.1, where a point mass in $\Re^2$ must move from a start state (configuration and velocity) to a goal state while avoiding contact with a set of polyhedral obstacles.

1.2.2 Path Planning

Path planning considers only the kinematic aspects of motion planning. Thus, the path planning problem is to find an obstacle-avoiding path between two robot configurations, or, for a point mass robot in the plane, between two positions. Mathematically, a path is a continuous map

$$\mathbf{p} : [0, T_f] \to C,$$

where C is the space of robot configurations and $T_f \geq 0$. The kinematic constraints of this problem are simply the obstacles, which the path[10] must not intersect.

1.2.2.1 Shortest Paths

An "efficient motion" in the context of path planning is a *shortest path*, *i.e.*, a minimal arc-length motion. In two dimensions, a shortest path between two points amongst polygonal obstacles with geometric complexity (number of vertices) N can be found in time $O(N^2)$ by constructing a "visibility graph" and searching for a shortest path in the graph [GM87]. (See figure 1.5.) Although [Can88a] shows that there is an algorithm for finding *some* path in time polynomial in the geometric

[10]Strictly speaking in mathematical terms, this is actually its image.

complexity and the algebraic complexity of the obstacles, and singly-exponential in the degrees of freedom, the *shortest*-path problem for a point among polyhedra in three dimensions is $\mathcal{NP}$-hard [CR87].

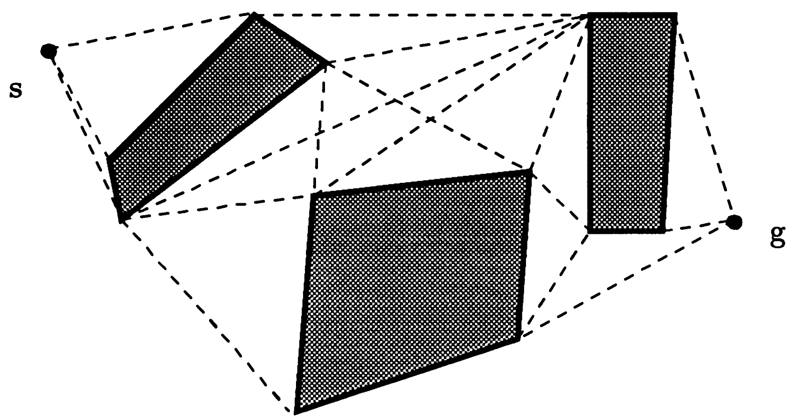


Figure 1.5: The *visibility graph* for a shortest-path problem in two dimensions. Obstacle edges are also graph edges, and all other graph edges are shown as dashed lines.

1.2.2.2 ϵ-Approximately Shortest Paths

Since the shortest-path problem for a point robot amongst polyhedra in $\Re^3$ is $\mathcal{NP}$-hard, one might consider a corresponding approximation problem. For a given problem instance, a shortest path between the start and goal positions is an optimal solution. If a path between the start and goal has arc length no more than an ϵ factor greater than the shortest, then it would be natural to consider it "within ϵ" of being optimal. Thus, the ϵ-approximation problem for shortest-path planning is, for given start and goal positions and a given set of obstacles, to find an obstacle-avoiding path between the start and goal whose arc-length is at most $(1+\epsilon)L_{opt}$, where L_{opt} is the length of a minimal arc-length solution. A fully-polynomial-time ϵ-approximation algorithm for the point robot problem in $\Re^3$ is given by [Pap85].

1.2.3 Trajectory Planning

A motion planning problem with dynamics constraints is a trajectory planning problem; that is, to find a motion that is a function of time. Because the term "time-optimal trajectory" has come to mean several things in the computer science, robotics, and control communities, it is necessary to describe what is meant by that term in this thesis.

1.2.3.1 Basic Trajectory Planning

Let us return to the path planning problem in $\Re^2$ and "add dynamics". First, we change the endpoint conditions of the motion to include velocity, so the motion must now go from a start state and a goal state.[11] Then, we introduce a bound on the speed of the motion and a bound on its acceleration. Since a point mass obeys the elementary dynamics law $\mathbf{f} = m\mathbf{a}$, an acceleration bound is equivalent to a bound on applied force.

For a point in the $\Re^2$, a *trajectory* taking time T_f is given by a map

$$\Gamma : [0, T_f] \to \Re^2 \times \Re^2,$$

where $\Gamma(t) = (\mathbf{p}(t), \dot{\mathbf{p}}(t))$ for some path $\mathbf{p} : [0, T_f] \to \Re^2$. The dynamics bounds require these maps to obey

$$\begin{aligned} \|\dot{\mathbf{p}}\|_\infty &\leq \overline{v}, \quad \text{and} \\ \|\ddot{\mathbf{p}}\|_\infty &\leq \overline{a}, \end{aligned}$$

where the bounds $\overline{v}$ and $\overline{a}$ are given in the problem instance. The resulting problem is a *trajectory planning* problem, and with appropriate modifications, such as the addition of control forces and more complicated dynamics laws, it can be generalized to more realistic robot models.

In this thesis, a basic *trajectory planning* problem instance will have four parts:

1. a start and goal state for the robot,
2. obstacles that the robot must avoid,
3. a dynamics law relating the robot's state, acceleration, and applied (control) generalized forces
4. dynamics bounds, such as bounds on the robot's velocity, acceleration, and the applied forces.

[11] Recall that a state is a *(configuration, velocity)* pair.

A solution to a problem instance is a trajectory from the start state to the goal state that avoids the obstacles such that under the dynamics law the velocity, acceleration, and force bounds are not violated. The dynamics law and the dynamics bounds are the *dynamics constraints.*

1.2.3.2 Planning Time-Optimal Trajectories

The term "time-optimal trajectory" (with respect to given problem constraints) has several meanings in the robotics and control literature.

1. It can mean a trajectory that is time-optimal among the trajectories that trace out a specified path.

2. It can mean a trajectory that is time-optimal in a class of trajectories that correspond to some parameterized set of functions.

3. It can mean a trajectory that is time-optimal in the set of all trajectories (that obey the constraints).

It does not help matters that the word "optimal" is often used to mean "locally optimal" by the robotics and control communities and "globally optimal" by the computer science and operations research communities, even though each community has a reasonable justification for its choice of meaning.

In this thesis, a time-optimal trajectory will always mean a trajectory that is *globally time-optimal in the set of all trajectories* that obey a given set of constraints and endpoint conditions. We call the problem of finding time-optimal trajectory with respect to obstacles and dynamics constraints the *time-optimal trajectory planning problem*, and it is the natural analogue of the shortest-path planning problem. It should be noted that finding shortest paths does not generally help one to find time-optimal trajectories. (See figures 1.6 and 1.7.) By choosing a maximum acceleration sufficiently large compared to the maximum speed, we can use the result and proof of [CR87] to show that this problem is $\mathcal{NP}$-hard for a point robot amongst polyhedral obstacles in three dimensions.[12]

It should be made clear that in this thesis does not consider the problem of synthesizing an optimal control law, another important research area. However, the kinodynamic planning problem formulation allows for control error in following a planned trajectory.

[12]See Appendix A.

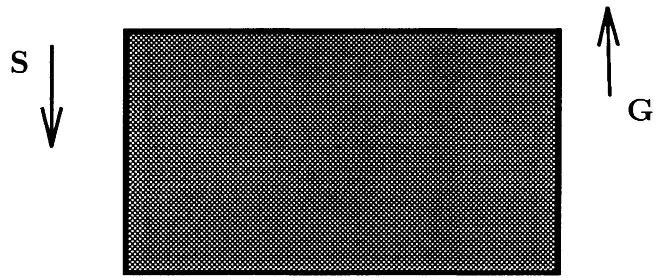


Figure 1.6: A simple trajectory planning problem. Which way around the rectangle is minimal-time will depend on the acceleration and velocity bounds.

1.2.3.3 Approximately Time-Optimal Trajectories

Since the time-optimal trajectory problem is generally intractable, we have pursued approximation algorithms. As described in Section 1.4, certain approximation algorithms for finding time-optimal robot trajectories in the presence of obstacles existed previous to the results presented here. While some of these earlier methods are useful for synthesizing robot trajectories, there are currently no proofs of how close to optimal the solutions they find will be. In fact, it is doubtful that an appropriate approximation problem had been formulated prior to [CDRX88].

One obvious way to cast the time-optimal trajectory planning problem into an optimization problem is to use the time of a trajectory as its cost, while demanding that solutions avoid the obstacles and obey the endpoint conditions and dynamics constraints. Thus, if a time-optimal trajectory takes time T_{opt}, an ϵ-time-optimal trajectory would take at most time $(1+\epsilon)T_{opt}$.

While this is the direct analogue of the ϵ-optimal shortest-path problem, it differs in a fundamental way. Suppose we have a shortest path $\mathbf{p}_{opt}$, pick any point $\mathbf{x}$ on it, and then choose another point $\mathbf{x}'$ near $\mathbf{x}$ such that the line segment between them intersects no obstacles. Then the shortest path between the start and goal that passes through $\mathbf{x}'$ will have a length bounded by $2\|\mathbf{x}-\mathbf{x}'\|$ plus the length of the original path $\mathbf{p}_{opt}$. In a region of obstacle-free space, a shortest-path solution $\mathbf{p}_{opt}$ can always be subjected to "perturbations" to obtain solutions that are infinitesimally longer than the shortest path. Thus, the ϵ-optimal shortest-path problem is intuitively "reasonable" in the sense that whenever a solution exists, ϵ-optimal solutions that are *not* exactly optimal will also exist.

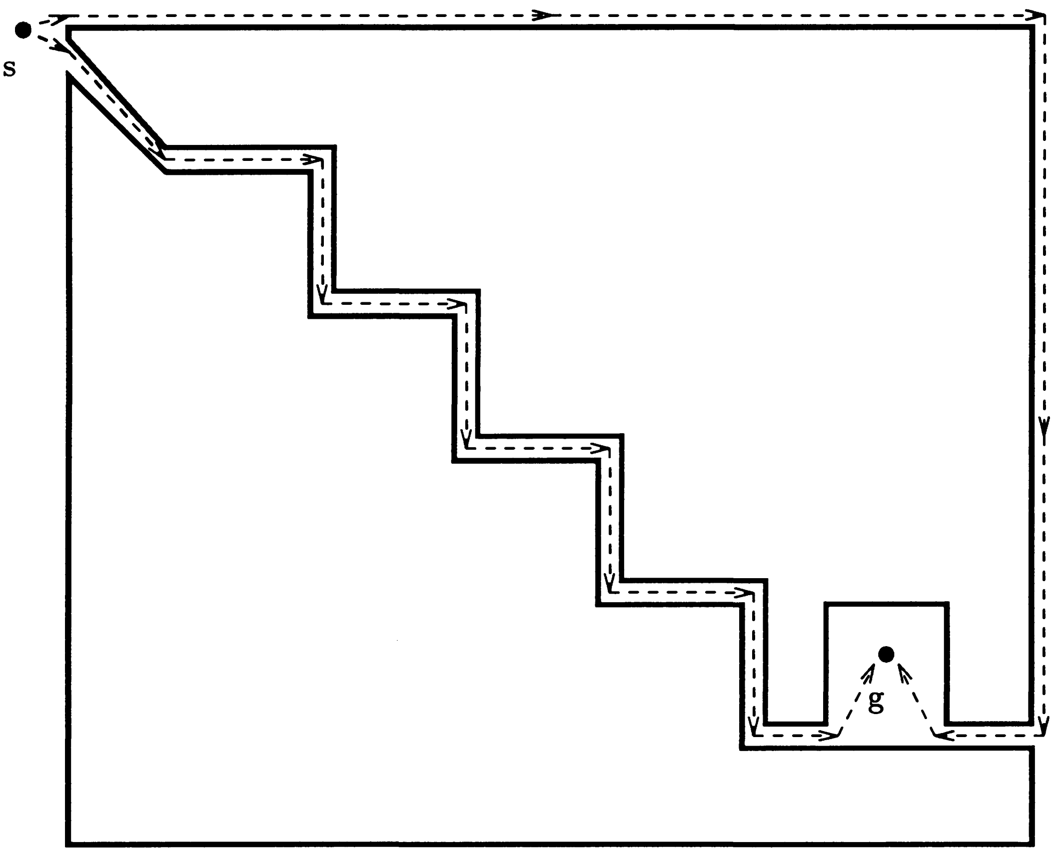


Figure 1.7: Another path planning problem. The shortest path through the "zig-zag" is the shortest path from **s** to **g** under both the L_2-norm and the L_∞-norm. Now, consider the related trajectory-planning problem with zero start and goal velocity, and with acceleration and velocity bounds $\overline{a}$ and $\overline{v}$. If $\frac{\overline{v}^2}{\overline{a}}$ is large enough compared to the width of the zig-zag, then the minimal-time trajectory through the zig-zag will take longer than the minimal-time trajectory that goes from **s** to **g** by taking the outside route. Thus, finding a shortest path might not even help to find the homotopy class of the minimum-time motion.

The same would not be true for the obvious ϵ-time-optimal trajectory problem. Consider the example in figure 1.8. The trajectory shown is the optimal trajectory. It follows full negative x-acceleration for its duration, and its y-acceleration is full negative, followed by full positive. A viable trajectory from **S** that deviates from this trajectory anywhere, even infinitesimally, must take substantially longer to reach the goal state; otherwise it either will fail to achieve the goal state exactly or will crash into the obstacle.

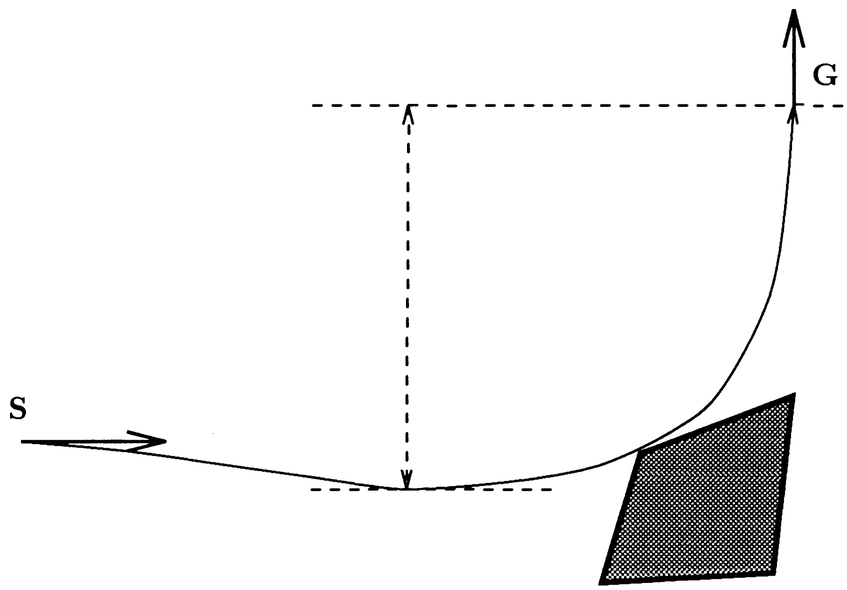


Figure 1.8: Difficulties with a naive approximation problem formulation. This (illustrated) optimal particle-dynamics trajectory follows full negative x-acceleration for its duration. The y-acceleration is full negative, followed by full positive. A viable trajectory from **S** that deviates a small distance from this either will fail to achieve the goal state exactly, will crash into the obstacle, or will have to go around it. In fact, all trajectories that enter a region of state space nearby a segment of this trajectory are doomed to crash into the obstacle. A state where an arbitrarily small deviation forces qualitative changes on trajectories otherwise beginning at it is a *critical point*.

1.2.4 Kinodynamic Planning

This thesis focuses on one class of problems within the kinodynamic framework. The problem formulation accounts for errors in control, sensing, and models of robot and obstacle geometry by including a *speed-dependent safety margin* as a problem parameter in addition to the kinematic and dynamics constraints allowed in the problem classes discussed above. Thus, for a given robot, a motion that is time-optimal with respect to these constraints will be time-optimal in the set of trajectories that the robot could attempt to follow while being guaranteed not to collide with obstacles in the environment.

1.2.4.1 Safety Functions and Kinodynamic Planning

Following a theoretically time-optimal trajectory closely enough to avoid obstacles may require unrealizable precision in control or sensing. The exact time-optimal trajectory may thus be unexecutable by a physical robot. If the trajectory is planned using models of the robot and the environment that contain some error, then the likelihood of the executed motion avoiding collisions is even less.

For these reasons, an optimal *planned* trajectory should avoid obstacles by a safety margin δ_v. In the kinodynamic planning formulation here, this margin is allowed to be a positive affine function of speed, with function coefficients chosen *a priori*. Specifically,

$$\delta_v(c_0, c_t)(v(t)) = c_0 + c_1|v(t)|,$$

where the coefficients $c_0 > 0$ and $c_1 \geq 0$ are contained in the algorithm input. Thus, a trajectory that is "$\delta_v(c_0, c_1)$-safe", or "δ_v-safe" for short, will avoid all obstacles by a constant margin plus a margin linear in speed. Since whether a robot state is safe depends on the robot velocity as well as position and the presence of obstacles, this characterization of safety is a strictly kinodynamic constraint, as opposed to a pure kinematic or pure dynamics constraint.

This first-order choice roughly corresponds to how accurately and quickly a robot senses its position and velocity, combined with how much time and space it requires to "survive" velocity errors and correct for them. Figure 1.9 illustrates some motivation this. For additional motivation, one might consider a one-dimensional system tracking a planned trajectory. Recall the relation between energy and momentum: $\frac{\partial E}{\partial v} = mv$, and suppose that the control system allows a maximum velocity error of Δv, and F_{res} force is available for correcting velocity errors. Then, the tracking error in position can grow by $\frac{mv\Delta v}{F_{res}}$ before the velocity error can be corrected. Concisely stated, c_1 characterizes how accurately the robot can control its energy consumption.

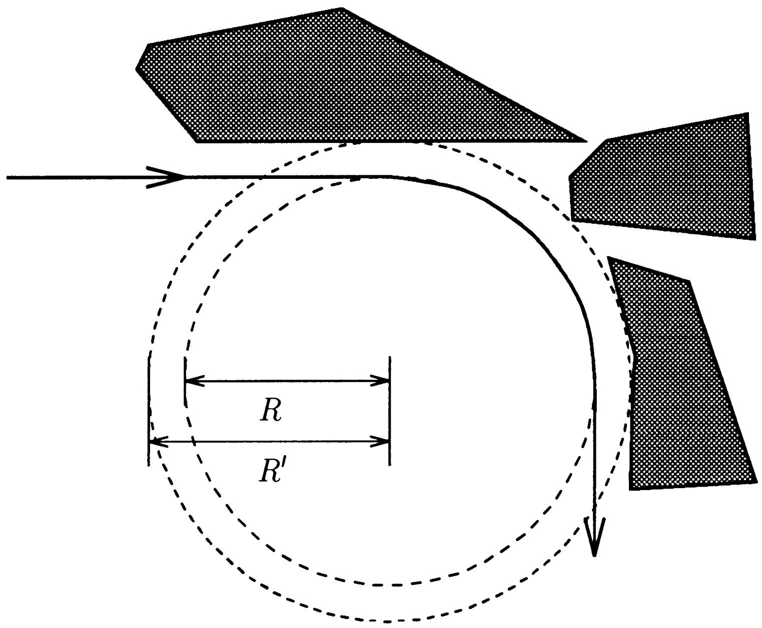


Figure 1.9: Some intuition behind a speed-dependent safety margin $c_0 + c_1(\|\mathbf{v}\|)$. Suppose following the path at planned speed v demands maximum acceleration $\bar{a}$ radially inward. Taking the derivative of $R = \frac{v^2}{\bar{a}}$, we see that velocity error of magnitude Δv can increase the curve's radius by $\frac{2(\Delta v)}{\bar{a}}v$. A change in "stopping distance" due to velocity error would also be linear in speed. Errors in static position sensing and geometric models are accounted for by c_0.

The safety margin ensures the existence of a "tube" or family of trajectories that are "nearby" a time-optimal δ_v-safe trajectory and approximate it. The paths traced out by such motions would lie inside a tube of diameter δ_v, or a *δ_v-safety tube* centered about path traced out by an optimal planned motion. (See figures 1.10 and 1.11.)

For a given robot and environment, a *δ_v-safe kinodynamic trajectory* avoids obstacles by a δ_v-safety margin and obeys the robots dynamics constraints. The *δ_v-safe kinodynamic planning problem* is to find a δ_v-safe kinodynamic trajectory

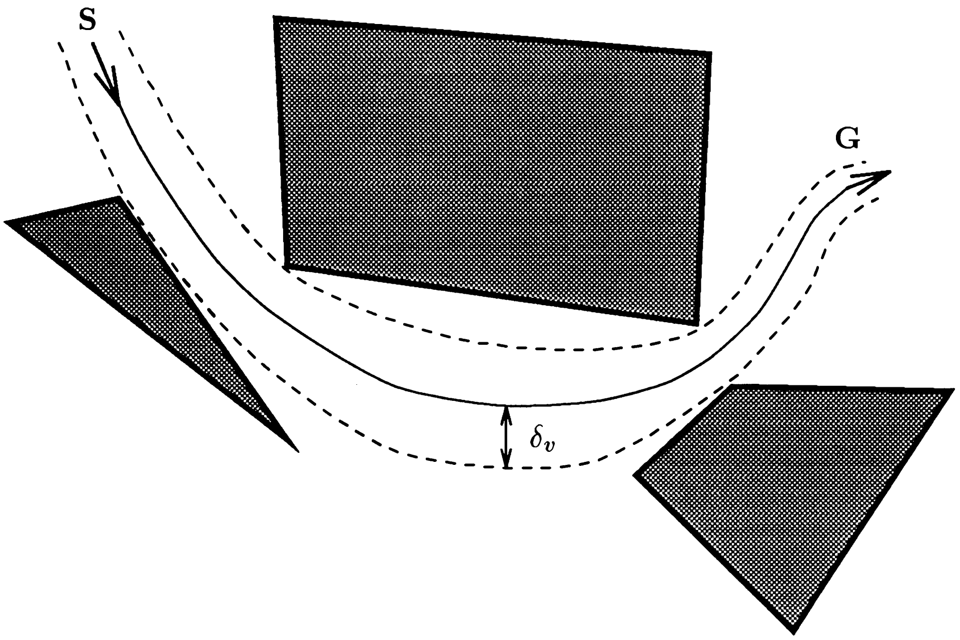


Figure 1.10: A δ_v-safe trajectory avoids contact with obstacles by a speed-dependent margin δ_v.

(a)

(b)

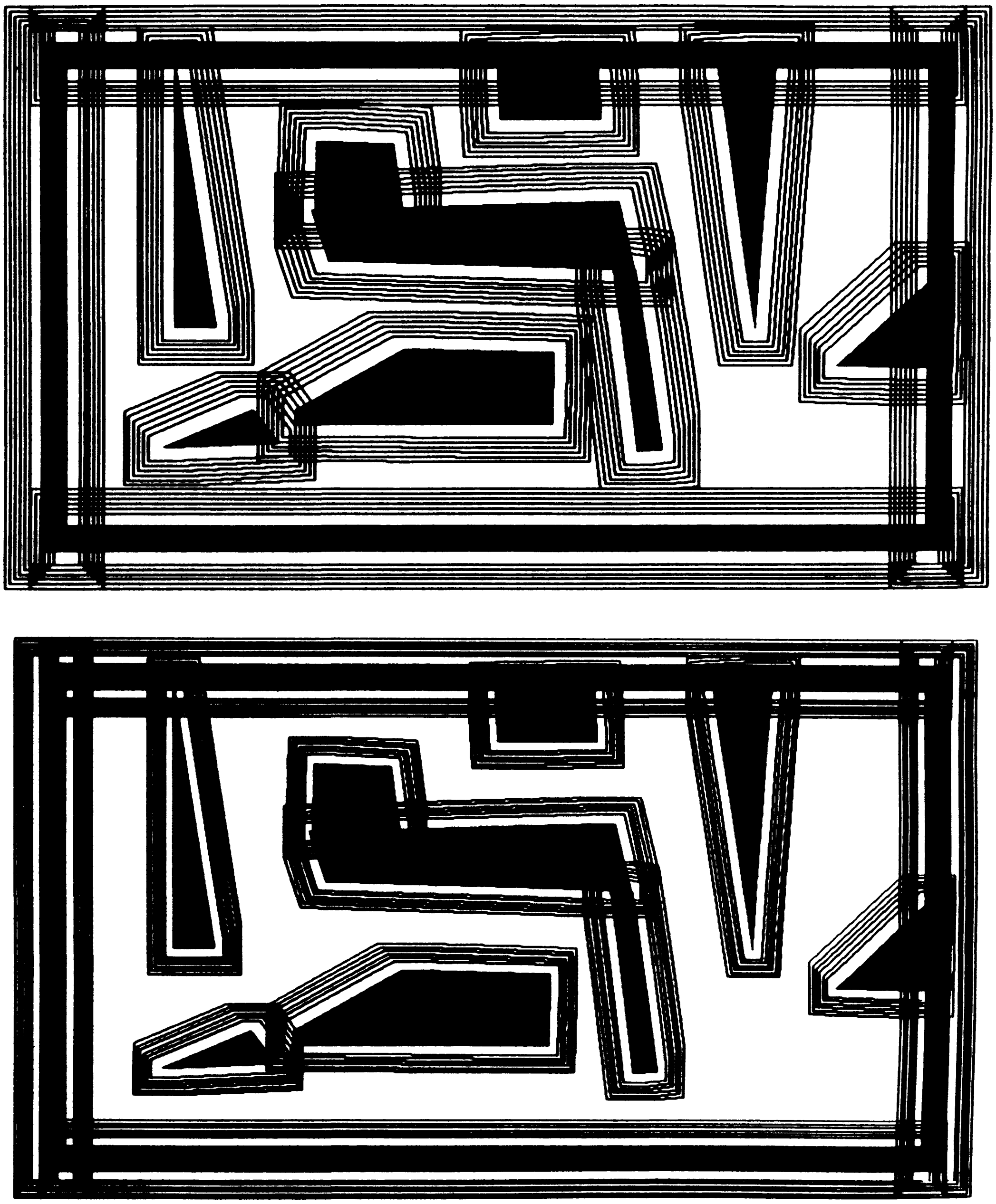


Figure 1.11: Obstacles and safety margins for a planar point robot problem. The concentric boundaries correspond to different robot speeds. (a) shows the original safety margins. (b) shows a safety margin a factor of 0.3 smaller than in the first.

from a start state to a goal state. Unless noted, the term "kinodynamic planning problem" will refer to this problem.

An *optimal kinodynamic trajectory* is a time-optimal δ_v-safe kinodynamic trajectory, and an optimal solution to a given kinodynamic planning problem is a time-optimal δ_v-safe kinodynamic trajectory from the start state to the goal state. The *optimal kinodynamic planning problem* is to find an optimal solution to a kinodynamic planning problem.

1.2.4.2 Approximately Optimal Kinodynamic Plans

We will choose our definition of an ϵ-optimal kinodynamic trajectory to meet two criteria. First, it should be intuitive and have the flavor of a conventional ϵ-approximation formulation. An ϵ-optimal trajectory must be feasible according to the kinematic and dynamics constraints, and we must be able to specify a trajectory "arbitrarily close to optimal" by choosing an adequately small ϵ. Second, it should be feasible to find trajectories that are provably ϵ-optimal solutions to a kinodynamic planning problem whenever an optimal solution exists. In particular, it should be possible to give a procedure that will provably find ϵ-optimal solutions to kinodynamic planning problems having optimal solutions. We must therefore guarantee that some finitely-parameterized trajectory will be ϵ-optimal, even for problems where no closed-form optimal solution is known to exist.

Toward the first criterion, if an optimal solution to an instance of the δ_v-kinodynamic planning problem takes time T_{opt}, then we allow an *ϵ-optimal solution* can take up to $(1+\epsilon)T_{opt}$ time. Towards the second, we will allow ϵ-optimal trajectories to avoid obstacles by a margin given by $(1-\epsilon)\delta_v$; this is sometimes called "δ'_v-safety", *i.e.*,

$$\delta'_v = (1-\epsilon)\delta_v.$$

This ϵ-factor of slack in the safety margin ensures that ϵ-optimal trajectories will not be forced to follow the same path as an optimal solution. Also for the purpose of meeting the second criterion, we will allow ϵ-optimal trajectories found by an algorithm to be *ϵ-close* to the specified start state and goal state,[13] by which we mean that its initial and final states approximate these states to within $O(\epsilon)$.

An algorithm guaranteed to find ϵ-optimal solutions to instances of an optimal kinodynamic planning problem is called a *provably good (ϵ-)approximation algorithm* for that problem. We must note that a trajectory planned by such an

[13]This was not discussed in [CDRX88], but included in [DX89b,DX90a].

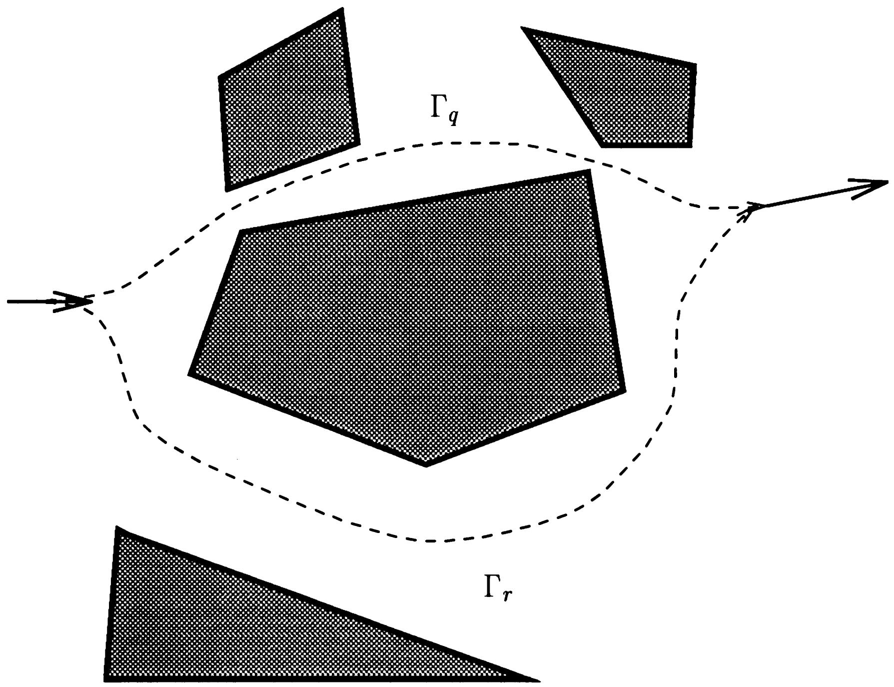


Figure 1.12: An ϵ-optimal trajectory can be topologically different from the truly optimal kinodynamic trajectory. In this illustration, Γ_r obeys the original safety-margin, and Γ_q obeys a margin a factor of ϵ smaller (with a fairly large $\epsilon < 1$.

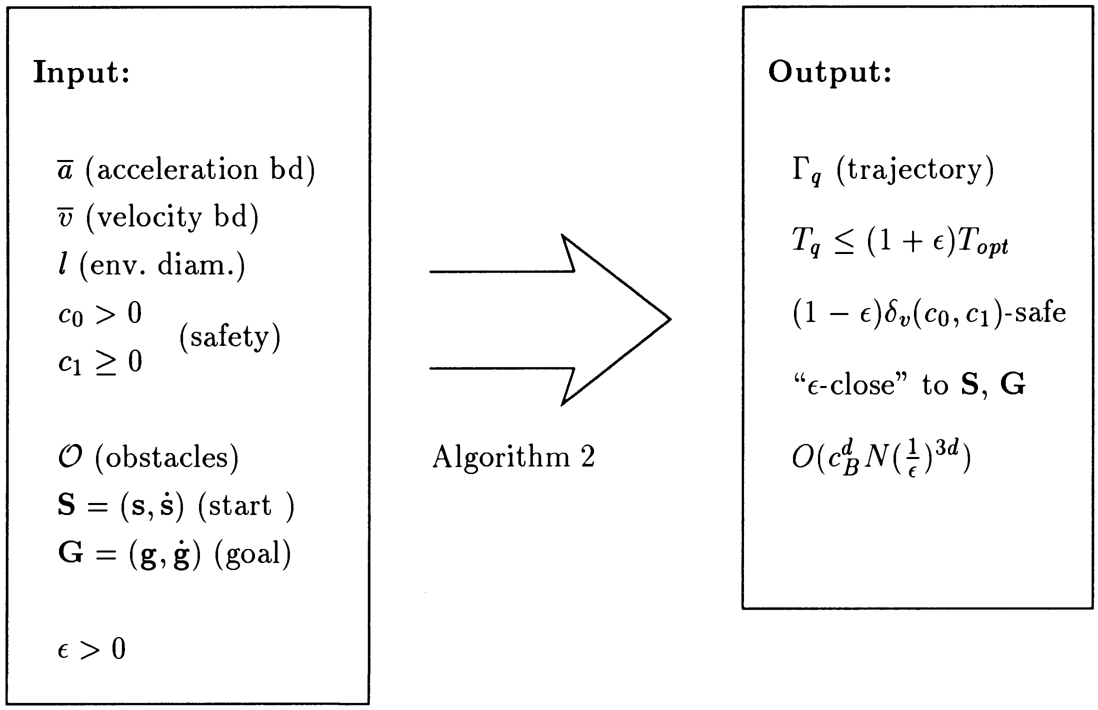


Figure 1.13: Algorithm 2 (from Chapter 3) is a *provably good polynomial-time ϵ-approximation algorithm* for optimal kinodynamic planning. As input, the algorithm is given an instance of the Optimal Basic Kinodynamic Planning Problem (see Chapter 2) in d degrees of freedom and an approximation parameter $\epsilon > 0$. If a solution exists, the algorithm is guaranteed to find an ϵ-optimal solution Γ_q in time polynomial in $\frac{1}{\epsilon}$ and N, the geometric complexity of the obstacles in the problem instance. c_B is a constant dependent on the robot's dynamics bounds (maximum speed $\overline{v}$ and maximum acceleration $\overline{a}$), the workspace diameter l, and the δ_v-safety parameters, c_0 and c_1.

algorithm is permitted to be quite different from an optimal solution, except at the start and goal. The ϵ-optimal kinodynamic planning problem permits this in two ways. First, because an ϵ-optimal trajectory can take more time than an exactly optimal trajectory, it can may trace out a longer path. Second, because ϵ-optimal trajectories obey slightly smaller safety margins, they can "squeeze through" some gaps between obstacles that optimal trajectories cannot, and they can squeeze through others at higher speeds. This illustrated in figure 1.12.

1.3 Techniques

1.3.1 Path Planning Observations

Even for path planning,[14] an apparent barrier to finding an ϵ-optimal motion is that the number of non-looping path-homotopy classes can be *at least* exponential in the number of the obstacles. Results in shortest-path planning originally motivated the study of kinodynamic planning in ϵ-approximation framework and our use of a state-space graph.

1.3.1.1 2D Path Planning

Once again, consider the problem of a planning shortest path in the plane for a point robot amongst polygonal obstacles with a N vertices. Any shortest path for such a robot is decomposable into line segments between obstacle vertices and the start and goal positions. For this reason, we can reduce two-dimensional path planning to finding the shortest path on a a visibility graph (recall figure 1.5), which will have at most $N+2$ vertices and can be constructed in time $O(N^2)$, as shown by Ghosh and Mount [GM87]. While the number of total paths through L vertices in the graph can be exponential in L, a breadth-first exploration of the visibility graph finds a shortest-path solution in $O(N^2)$ time.

1.3.1.2 3D Path Planning

Now, consider shortest-path planning in three dimensions. Here, we can still decompose any shortest path into straight line segments. However, a corollary of the $\mathcal{NP}$-hardness result in [CR87] is that finding these line segments is $\mathcal{NP}$-hard. On

[14] Recall that there are no dynamics in path planning.

the other hand, each segment endpoint either lies on an obstacle edge or is the start or goal.

Papadimtriou's fully-polynomial approximation algorithm [Pap85] is conceptually simple though involved in detail. It first divides up the obstacle edges into a number of segments, polynomial in $\frac{1}{\epsilon}$, having non-uniform lengths. Edges on the segment-segment visibility graph[15] correspond to equivalence classes of lines between points on obstacle edges, such that the lengths of any two lines in a class differ by no more than an ϵ-factor. Graph edge weights are segment-segment distances, and this graph is easily augmented to include the start and goal. Finding the shortest path between the start and goal vertices in this graph yields a path between the start and goal positions that is at most an ϵ factor longer than optimal. Since the number of vertices in the graph is polynomial in $\frac{1}{\epsilon}$, so is the cost of finding a shortest path between any two vertices in it.

1.3.2 Graphs Embedded In State Space

The basic idea behind the algorithms in this thesis is to reduce finding an ϵ-optimal solution to a kinodynamic planning problem to finding the shortest path in a directed *reachability graph* dependent on the problem instance. The vertices of the graph "discretize" the robot statespace,[16] and the edges of the graph correspond to trajectory segments that each take the same amount of time. The graph will have the provable property that if an optimal solution exists, then the shortest path in the reachability graph will directly correspond to an ϵ-optimal solution.

1.3.2.1 A Reachability Graph

To generate the reachability graph, we choose a state "ϵ-close" to the desired start state. This will be the *root vertex* of the graph. An ϵ-optimal trajectory will correspond to a the shortest path (in the graph) from the root vertex to a vertex or edge sufficiently close to the desired goal state. The degree of the graph and the number of vertices will be bounded by a polynomial in $\frac{1}{\epsilon}$. Thus, *instead of having to search a set of trajectories possibly exponential (or worse) in the number of obstacles, our algorithm searches a graph with size polynomial in* $\frac{1}{\epsilon}$ *and proportional to the "volume" of the statespace.* Except for dynamics laws, kinodynamic constraints are forbidden regions of this discretized statespace.

[15] Vertices in this graph correspond to obstacle-edge segments.

[16] The space of all robot states, *i.e.*, *configurations* $\times$ *velocities*

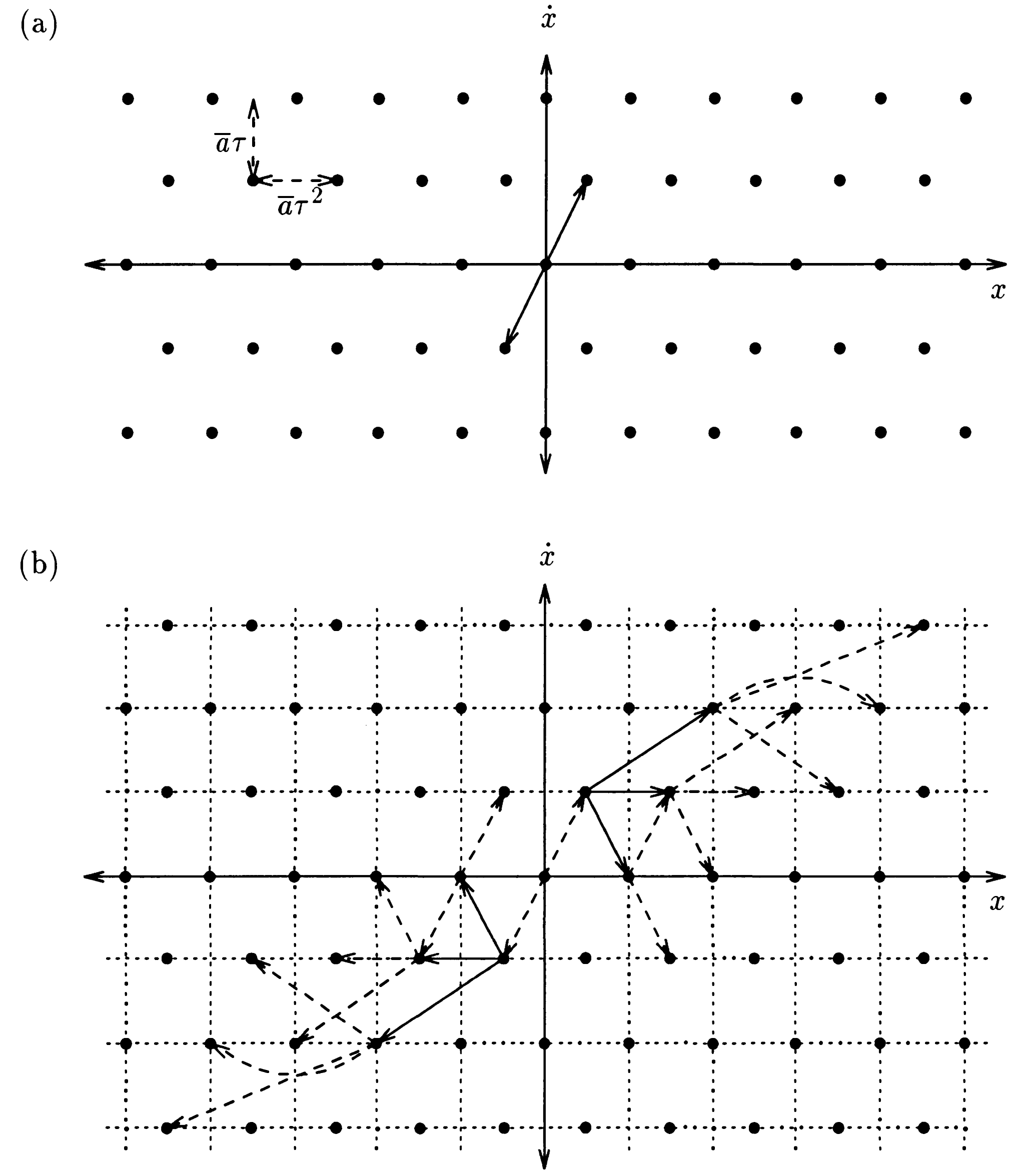


Figure 1.14: Growing a reachability graph in one-dimension, starting at the state $(0,0)$. Constant accelerations from the set $\{-\overline{a}, 0, \overline{a}\}$ applied for duration τ generate the graph. (a) shows the first generation. (b) shows the first three. States found will lie on an regular grid and "fill" much of the statespace region that obeys the kinodynamic constraints.

For any finite set of control-primitives[17] applied at the root vertex state for a timestep τ, we obtain a corresponding set of trajectory segments by integrating forward in time. If we prune this set of trajectory segments to eliminate those that violate the kinodynamic constraints, we obtain a set of trajectory segments that obey these constraints and that last for duration τ. These are the edges from the root vertex; they determine its neighbors in the graph. By applying the the control primitives on the newly-found graph vertices and taking the transitive closure, we generate a reachability graph. Thus, instead of being determined by spatial proximity, graph edges are determined by trajectory segments generated by control primitives, and each edge corresponds to τ elapsed time. Note that since the reachability graph is used to find ϵ-optimal trajectories, we only require the graph edges (trajectory segments) to obey the desired safety margin weakened by a factor of ϵ, *i.e.*, a $(1-\epsilon)\delta_v$ safety margin.

If the root vertex and control primitives are chosen carefully, then we might expect that the reachability graph would "fill" the region of statespace legal under the kinodynamic constraints. The intuition is that at each state corresponding to a vertex, the control primitives "sample" the vector fields permitted by the dynamics bounds. For example, let us consider an obstacle-free regions in one dimension. Suppose that $(0,0)$ is the root vertex, and suppose that we chose constant accelerations from the set $\{-\overline{a}, 0, \overline{a}\}$ applied for duration τ. This generates a reachability graph as shown in figure 1.14.

1.3.2.2 Graph Trajectories

We will call a trajectory that corresponds to a path in the reachability graph a *graph trajectory.* The smaller the timestep τ is, the better some graph trajectory will approximate an arbitrary trajectory in this simple state space space. In figure 1.15 we show the vertices from the first three generations of the reachability graph for a two-dimensional problem.

Limiting the set of control primitives to $\{-\overline{a}, \overline{a}\}$ yields a similar graph. For higher dimensions or more complicated dynamics bounds, the analogous set of control primitives consists of discretized control extremals or "near-extremals".

[17] A *control primitive* is function from time to acceleration (or to applied forces).

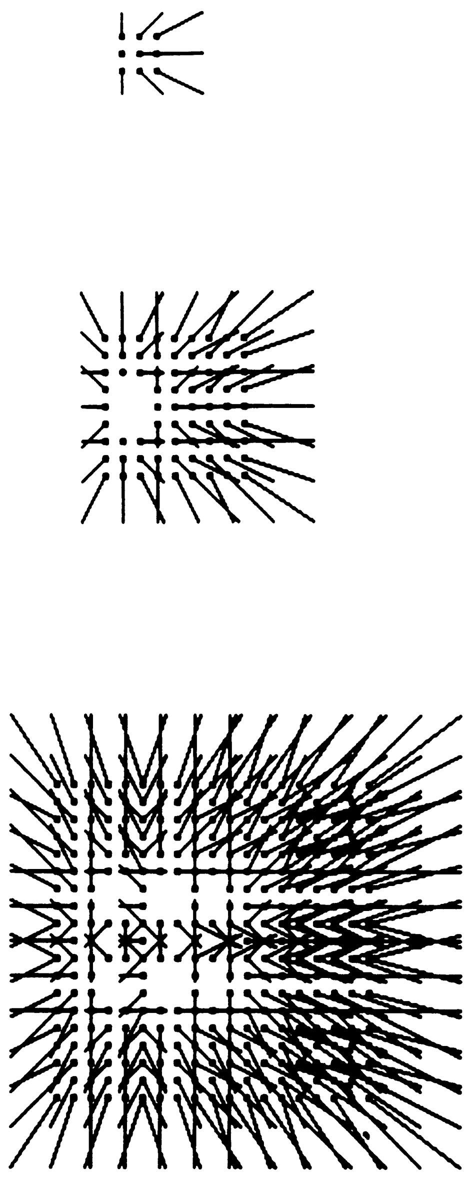


Figure 1.15: Vertices, or states, from the first three generations of a reachability graph (shown in succession). Dots show position, line-segments show velocity. Graph-generating parameters are chosen to guarantee that the graph will (discretely) fill all relevant regions of state space.

1.3.3 Graphs With Guarantees

While reachability graphs have intuitive appeal, we need to analytically relate the parameters that determine a reachability graph to the quality of the approximation obtained by searching it for a shortest path. In particular, how can we guarantee that some graph trajectory will be ϵ-optimal if a kinodynamic planning problem is solvable?

1.3.3.1 The Key Question: Tracking

A simple observation is that if a trajectory avoids obstacles by a δ_v margin, then there is an statespace region surrounding it in which all positions lie outside of all obstacles. A trajectory that stays in this region therefore avoids obstacles, as in figure 1.10. Furthermore, we can easily show that a trajectory that at all times *tracks*, or mimics, a δ_v-safe trajectory to within certain tolerances η_x in position and η_v in velocity will be $(1-\epsilon)\delta_v$-safe.[18]

If a trajectory tracks an optimal kinodynamic trajectory closely enough, to an *ϵ-safe tracking tolerance*, not only will it be $(1-\epsilon)\delta_v$-safe, but its time and endpoints will satisfy the conditions for ϵ-optimality. Ignoring reachability graph size for the moment, it would appear that the crucial question is how, given a tracking tolerance, to choose

1. a root vertex state,

2. an adequate set of control primitives, and

3. a timestep

that will generate a graph trajectory that tracks an optimal solution to this tolerance. It is not generally apparent whether a such a trajectory can be generated without violating the dynamics constraints. Specifically, we might worry that if the set of control primitives is finite, then any graph trajectory staying in the $(1-\epsilon)\delta_v$-safe region around the optimal trajectory might fall further and further behind it.

[18]This sort of tracking is depicted in figure 2.2.

1.3.3.2 Easier Tracking and Bounding the Graph Size

Suppose an optimal kinodynamic trajectory taking time T_{opt} is *uniformly* slowed down[19] by a factor of ϵ. Then the resulting "ϵ-slowed" trajectory will take time $(1+\epsilon)T_{opt}$, be within $O(\epsilon)$ of the start and goal,[20] and avoid obstacles by a $(1-\epsilon)\delta_v$ safety margin. Thus it will be ϵ-optimal. Furthermore, it will obey dynamics bounds that are tighter (*i.e.*smaller) than those obeyed by the optimal trajectory. This guarantees that there will be a closed region of states surrounding the image of the ϵ-slowed trajectory, such that under the original dynamics bounds, a trajectory beginning at any state in this region will be able to track the slowed trajectory to within a constant tolerance that depends only on the timestep and the dynamics bounds. In other words, under the original dynamics bounds, there will be a *dynamics advantage* that eliminates "critical points" from a closed region of states surrounding the ϵ-slowed trajectory image, relative to the ϵ-weakened safety margin. (Refer to figure 1.8.) Without knowledge of the actual optimal kinodynamic trajectory, we can exploit this advantage to choose the timestep, control primitives, and a root vertex that generate a trajectory that "tracks" the ϵ-slowed optimal trajectory closely enough to meet the ϵ-optimality criteria.

Finally, we must be able to bound the size of the graph search (for finding a shortest path in it). Two techniques prove useful. First, if we choose a finite set of constant accelerations as the control primitives, we can immediately compute the maximum size of the set of velocities and positions possible for any vertex state. Assuming a bounded world, this limits the size of the reachability graph itself. More generally, for appropriate primitive controls it is possible to use a density condition to prune the search to obtain polynomial complexity bounds, as we show in Chapter 4.

1.4 Previous and Related Work

Robotics is an inherently multidisciplinary area, and researchers from several different fields have been actively studying aspects of what we call kinodynamic planning for many years. While the path planning aspect of kinodynamic planning is computational-geometric by nature, finding an optimal kinodynamic trajectory

[19] That is, its velocity will be an ϵ-factor smaller. See equation (2.9) for mathematical expression.

[20] We require that $\epsilon < 1$, since we don't want to allow collisions.

obviously requires the solution of a constrained optimal control problem. We recognize the importance of the many previous results concerning the time-optimal trajectory problem that came from the field of optimal control, in addition to results from computer science. We have noted that in recent years other researchers have begun to explore the links between control theory, computational geometry, and computational optimization.

For a general review of issues in robotics and algorithmic motion planning, see [BHJ+82,Yap86,Bra89,BBD+90,Lat91].

1.4.1 Applied Control Theory

1.4.1.1 Optimal Control

There exists a large body of work on optimal control in the control theory and robotics literature.[21] Much of this work attempts an analytic characterization of time-optimal solutions. A major example is the well-known Pontryagin Maximum Principle [PBGM62], which gives an extremality condition necessary for optimal control and yields a corollary that can be used to solve a class of linear control problems. More recently Shin and McKay [SM85b] showed that the time of some geodesic trajectory will bound the minimum motion time and gave conditions under which a geodesic trajectory will be optimal. Sontag and Sussman [SS85,SS86] considered a class of dynamical systems and established sets of conditions under which (a) at least one control must be bang-bang, and (b) singular controls must be analytic. Schaettler [Sch87] considered a class of single-input linear control systems in $\Re^3$ and showed that every state has an open neighborhood in which a bang-bang trajectory with more than seven switchings cannot be optimal.

While these results are only a few among many in a rich and mathematically challenging field, the application of optimal control theory to motion planning has been difficult. Reasons for this include: state constraints (as opposed to control bounds), the generally non-linear dynamics of robot manipulators, and the absence of analytical solutions. Research in time-optimal trajectory planning for robotic manipulators dates back at least to the work of Kahn and Roth [Kah69,KR71], who linearized the equations of motion, otherwise approximated the dynamics, and assumed no kinematic constraints. Unfortunately, such simplifying assumptions concerning the dynamics are theoretically invalid [LWP80]. Even at present, there

[21]For a background on optimal control theory, terminology, and methods, see Leitman [Lei66], Athans [AF66], or Kirk [Kir70].

exist no exact algorithms for optimal kinodynamic planning with more than two degrees of freedom.

Studying restricted versions of the optimal trajectory problem has yielded numerical techniques that take into account the full dynamics of an open-chain manipulators. Luh, Walker, and Lin [LW77,LL81] minimize the motion time over paths consisting of straight line and circular arcs (in the space of Cartesian end-effector positions. Later, Bobrow, *et al* [Bob82,BDG85], and Shin and McKay [SM85a] provided numerical algorithms for finding the minimum-time trajectory *that traces a given input path with zero start and goal velocity.* Shiller and Dubowsky [SD85] extended the Bobrow result to include gripper and payload constraints. Hollerbach [Hol83] showed how to scale the velocity profile of a given trajectory to obtain a trajectory that obeys dynamics bounds. Slotine and Yang [SY89] later derived *characteristic switching points* and *limit curves* that allow faster computation of the minimum-time trajectory along a given path.

1.4.1.2 Near-Optimal Trajectories

Using some of these ideas, a number of authors have proposed (and often implemented) heuristic or approximate algorithms for what is hoped to be near time-optimal trajectory planning for robots with full dynamics and obstacle avoidance. Rajan [Raj85] optimized over two-dimensional joint-space paths parameterized by cubic splines and used the [Bob82,BDG85] algorithm to optimize the trajectory along each path. Dubowsky, Norris, and Shiller [DNS86] and Bobrow [Bob88] used parameterized cubic splines in the three-dimensional Cartesian space of the end-effector. Gilbert and Johnson [GJ85] also took a parameterization and optimization approach.

Sahar and Hollerbach [SH85] and Shiller and Dubowsky [SD88] both implemented algorithms which employ a fixed-resolution configuration-space (or state-space) grid on which path (or trajectory) knotpoints lie to compute near-minimal-time trajectories for robots with several degrees of freedom. Both groups used heuristic techniques to prune their searches, and they did not bound the goodness of their approximation, nor the running time of their algorithms. However, these grid methods take time which grows exponentially with the number of grid points or the resolution.

Another approach is to assume that some bang-bang control will be optimal[22]

[22]This is true in the linear case without state constraints, but not known to be true in general.

or nearly optimal. Niv and Auslander [NA84] and [SR84] follow this line of attack and use optimization techniques to search the space of actuator switching times (between bangs and coasts). Brown [Bro84] makes a similar assumption, but allows "coasts" (zero force) as well as bangs, chooses a time-interval (thus determining the set of possible switching times), rounds each trajectory segment endpoint to the nearest neighbor in a statespace grid, and searches the resulting network (graph). Of previous algorithms, Brown's are the most closely related to the family we present here. However, he makes no attempt to choose parameters that analytically guarantee approximation goodness or closeness to optimality.

While some of the optimization results mentioned above appear to have practical application, they all fail to provide the provable bounds we desire. Attempts to apply conventional optimization methods are prone to at least one of three shortcomings: 1) the guaranteed "optimality" of the solution is local to the parameterization, or even worse, local *within* the parameter space; 2) the cost-functional being minimized is not time but some functional that includes time; or 3) there is no analytical relationship between the size of the parameter-space over the which the minimization and how close to globally time-optimal the trajectory found could possibly be. For the specific instances of these "classical" optimization approaches, we never find an analytical relationship between the computational complexity and the accuracy of the solutions.

1.4.2 Algorithms With Provably Good Bounds

1.4.2.1 Path Planning

Canny and Reif [CR87] showed that shortest-path planning problem for a point amongst polyhedral obstacles in three dimensions is $\mathcal{NP}$-hard. The problem is equivalent to a basic optimal kinodynamic planning problem with the acceleration bound $\overline{a}$ set to infinity, and this observation may be used to extend [CR87] to show that in three dimensions, optimal kinodynamic planning is $\mathcal{NP}$-hard.[23] Papadimitiou [Pap85] gave an $O(N^4(B+\log\frac{N}{\epsilon})^2(\frac{1}{\epsilon})^2)$ algorithm for this shortest-path problem; here N is the number of features of the polyhedra and B is the bit-complexity of the largest integer used in describing them.[24] Clarkson [Cla87] described an alternative algorithm that is faster when $N\epsilon^3$ is sufficiently large and also presented an algorithm for the planar case that finds ϵ-shortest paths between

[23]See Appendix A.

[24]This is the algorithm we have previously described.

Table 1.1: Provably-good algorithms for obstacle-avoiding motion planning before the kinodynamic planning approach.

Problem		Provably-Good Algorithm	
Robot class	Minimize	Exact	ϵ-approximate
2DOF point	path length	yes	yes
3DOF point	path length	yes	yes
2DOF point	motion time	no	no
3DOF point	motion time	no	no
d-link open-chain	motion time	no	no

any two points in $O((\log N)/\epsilon)$ time with an $O(N^2 + N \log N/\epsilon)$-time preprocessing of the environment description. The fastest exact algorithm for planar shortest-path planning is $O(N^2)$, by Ghosh and Mount [GM87], as noted earlier. Sharir and Schorr [SS84] gave an exact $2^{2^{O(N)}}$ algorithm for the 3D shortest-path problem, and Reif and Storer [RS85] later lowered the bound with a $2^{N^{O(1)}}$ algorithm. This was subsequently bested by Canny, who found an $O(N^{O(N)})$ algorithm [Can88b].

The problem of finding *some* path between two positions in a polyhedral environment for d-link robots ("the Generalized Mover's Problem", or *GMP*) has been studied for a number of years; see Lozano-Pérez and Wesley [LPW79], Reif [Rei79], or Schwartz and Sharir [SS83,SS82]. Reif [Rei79] showed that *GMP* is $\mathcal{PSPACE}$-hard in the degrees of freedom. Lozano-Pérez [LP83] pioneered the use of configuration space, which has been subsequently used by many researchers. (See [Lat91] for a review.) Canny [Can88a] gave an algorithm that is singly exponential in the degrees of freedom and polynomial in the number of kinematic constraints; this is currently the lowest complexity algorithm for *GMP*.

1.4.2.2 Kinodynamic Planning

Ó'Dúnlaing [Ó87] provided an exact algorithm for one-dimensional pursuit game, a kind of kinodynamic planning. His methods may extend to the 2- and 3-dimensional cases as well. Kinodynamic planning in 2D is related to the problem of planning minimum-path motions with non-holonomic constraints, as studied by Fortune and Wilfong [FW88,Wil88], who give exact algorithms, and Jacobs and Canny [JC89] who describe an approximation algorithm. In the general problem, a robot with wheels and a bounded minimum turning radius must be moved. To make the analogy clear, in our case, the minimum turning radius is $\frac{\|\mathbf{v}\|^2}{\bar{a}}$.

Early results from this thesis come from research done in cooperation with Canny and Reif [CDRX88]. Specifically, we introduced the algorithmic approach to kinodynamic planning, by formalizing a simple problem and providing the first provably good approximation algorithm for 2- and 3-dimensional optimal kinodynamic planning.

Research in kinodynamic planning proceeded on the foundations laid out in [CDRX88]. Certain techniques presented here (and described in [DX90b,DX89a, DX90a,DX91]) to handle state-dependent dynamics are similar to those used by Jacobs, *et al* [JHCP89,HJCP90], who gave the first polynomial-time approximation algorithm for optimal kinodynamic planning for robots with open-chain dynamics. Their work introduces several techniques to the kinodynamic planning literature, including (a) discretizing acceleration-space according to the problem parameters and (b) reducing state-dependent dynamics to being locally constant. Our techniques result in lower complexity bounds for our algorithms. In concurrent independent work, Reif and Tate [RT90] used a parameter-dependent acceleration-space discretization implicitly to obtain a polynomial-time approximation algorithm for robots with decoupled dynamics, L_2 dynamics bounds, and polyhedral c-space obstacles.[25] Canny, Reif, and Rege [CRR90] use the first-order theory of real closed fields[26] to obtain an exact $\mathcal{PSPACE}$ algorithm for a planar optimal kinodynamic planning problem. At present, this is the only known algorithm for an exact optimal kinodynamic planning problem in more than one dimension.

1.4.3 Other Related Work

A relatively young research area examines problems from control theory using techniques from computational complexity theory; this has yielded results in both fields. For example, see the work of Papadimitriou and Tsitsiklis [PT86]. Recently, using the kinodynamic framework, Reif and Tate [RT91] defined a generalization of alternation (as in alternating Turing machines [CKS81]) to continuous domains. They consider a class of pursuit games in which a robot must not only obey kinodynamic constraints, but evade a moving adversary. For one such game in three dimensions, they show that the decision problem is $\mathcal{EXPTIME}$-hard, the first

[25] A configuration-space representation of obstacles.

[26] See, *e.g.*, [Yap86].

provable intractability result[27] for a robot motion planning problem with complete information. A notion of ϵ-approximation algorithms for these games is also defined[28], and approximation results are presented for robots that obey L_∞ and L_2 dynamics bounds.

Robot motion planning is often a more strongly "online" problem than an ϵ-approximation formulation effectively characterizes. In particular, the time spent on a planning problem can affect the usefulness of a solution or even the overall utility of the robot, especially when computation time is a shared resource. To address such issues, Dean and Boddy [DB88,BD89] have proposed an *expectation-driven refinement* framework for *time-dependent planning*, with an eye towards applying decision theory (*e.g.*, Horvitz, *et al.* [HBH88]) to robot planning. Their framework depends on the existence of *anytime algorithms* for individual problems. An anytime algorithm has two important characteristics:

1. it can return some solution whenever it is asked, and
2. the quality of its solution improves "nicely" with time.

This definition does not require that the solution quality converge to the optimal as the running time goes to infinity, so anytime algorithms generally cannot be used to find ϵ-approximations. At the same time, an ϵ-approximation algorithm will not necessarily be "convertible" to an anytime algorithm, except in the sense that the original algorithm could be run over and over with progressively smaller choices of ϵ. While this thesis does not consider anytime algorithms for kinodynamic planning, such algorithms (and an appropriate problem formulation) would be valuable. Finally, although ϵ-approximation algorithms and anytime algorithms are intended for different uses, algorithms that qualify as both would be an interesting research area.

1.5 Contributions and Thesis Outline

1.5.1 Algorithms

This thesis presents a series of provably-good polynomial time ϵ-approximation algorithms for optimal kinodynamic planning. The particular algorithms can be

[27]Here, we mean that the intractability is independent of whether $\mathcal{P} \neq \mathcal{NP}$.
[28]The measure of optimality concerns an evasion margin or a deadline time.

applied to robots characterized by three classes of dynamics laws and bounds: decoupled dynamics under L_∞-norm (*i.e.*, independent) bounds, decoupled dynamics under L_2-norm bounds, and fully-controllable open-chain manipulator dynamics.[29] Table 1.5.1 presents the asymptotic complexities of the algorithms in terms of the geometric complexity of the problem N, the number of degrees of freedom of the robot d, and the approximation parameter $\epsilon > 0$.

The algorithm from Chapter 2 has the distinction of being the first provably-good approximation algorithm for planning time-optimal robot trajectories. The two algorithms from Chapter 3, also for robots with decoupled dynamics and dynamics bounds, have the lowest known complexity for their classes. The (non-bang-bang) algorithms for robots with open-chain kinematics and dynamics, found in Chapter 4, also have the lowest known complexity for approximation algorithms in their class. The last algorithm listed (from Chapter 5) is the first provably-good approximation algorithm for planning time-optimal open-chain manipulator trajectories that generates bang-bang trajectories.[30]

1.5.2 Mathematical Contributions

The work in this thesis introduces several new concepts and techniques to the field of algorithmic motion planning. We briefly list highlights here. They are described in context in Section 1.5.3.

Problem Formulation. A constrained optimal trajectory planning problem is formulated to match the requirements of a time-optimal *planned trajectory*. In Appendix A it is shown that finding an exact optimal solution in 3D is $\mathcal{NP}$-hard. An ϵ-approximation version of this problem is formulated with the intention of allowing provably-good algorithms. A speed-dependent safety margin is included to account for errors in sensing, control, and modeling, and to ensure that the approximation problem is mathematically reasonable. See Chapters 2 and 4.

[29] *Dynamics laws* are *decoupled* if the map from forces to accelerations is constant and linear. (See figure 1.4.) Dynamic bounds are *independent* if there is a constant linear map to a coordinate system where the choice of acceleration and velocity along each single axis never affects the choices along the others. A *fully-controllable* manipulator is actuated at each joint and is able to overcome gravity at every possible pose.

[30] A trajectory and its controls are *bang-bang* if each control is fully-saturated (at a "corner" of the force bounds – see figure 1.4) at all but a finite number of points during the motion.

Table 1.2: Provably-good approximation algorithms for optimal kinodynamic planning in this thesis. Accuracy and asymptotic bounds shown. "Decpl" is an abbreviation for "decoupled". $c_A, ..., c_F$ are constants that depend on the robot and the algorithm, but that are independent of N and ϵ.

Ch	Robot Dynamics & Dynamics Bounds	Time	Safety	Algorithm Complexity
2	decpl, $d \leq 3$, L_∞-norm	$(1+\epsilon)T_{opt}$	$(1-\epsilon)\delta_v$	$O(c_A^d N(\frac{1}{\epsilon})^{6d})$
3	decpl, L_∞-norm	$(1+\epsilon)T_{opt}$	$(1-\epsilon)\delta_v$	$O(dc_B^d N(\frac{1}{\epsilon})^{3d})$
3	decpl, L_∞-norm	T_{opt}	$(1-\epsilon)\delta_v$	$O(dc_C^d N(\frac{1}{\epsilon})^{3d})$
4	decpl, L_2-norm	$(1+\epsilon)T_{opt}$	$(1-\epsilon)\delta_v$	$O(dc_D^d(\frac{1}{\epsilon})^{6d-1})$
4	open-chain	$(1+\epsilon)T_{opt}$	$(1-\epsilon)\delta_v$	$O(c_E^d N(d+\log N)(\frac{1}{\epsilon})^{6d-1})$
5	open-chain (bang-bang)	$(1+\epsilon)T_{opt}$	$(1-\epsilon)\delta_v$	$O(c_F^d N(d+\log N)(\frac{1}{\epsilon})^{7d})$

Scaling-Tracking Methods. All our proofs of algorithm accuracy and complexity follow a canonical structure: we show, if an optimal kinodynamic trajectory trajectory exists, how to choose parameters that ensure that some graph trajectory will track a "copy" of this trajectory that has been slowed down by a factor of ϵ closely enough to meet the conditions for ϵ-optimality. (For robots under decoupled laws and bounds, the optimal trajectory can be tracked.) Introduced in Chapter 2.

Tracking Lemmas. A series of tracking lemmas relate a tracking tolerance and dynamics constraints to control discretizations, root vertex choice, and time-step that guarantee an arbitrary trajectory will be tracked to the desired trajectory by a graph trajectory. The proof techniques for some of these lemmas are quite difficult. Introduced in Chapter 2 and found in all other technical chapters.

Constructive Trajectory Proofs. The proofs used to obtain the lowest known complexity bounds for kinodynamic planning for robots with decoupled dynamics and L_∞-dynamics bounds use inductive definition. Appears in proofs in Chapter 3 and following chapters.

Adversary Game Proofs. These are first used to prove tracking lemmas for robots with coupled dynamics bounds. In a basic version, rules in simultaneously-played games enforce the condition of extremal control. Next, a slightly harder game allows us to prune the search of an exponentially-sized reachability graph with a spatial hashing technique, while preserving the validity of the search. Appears first in Chapter 4.

Time-Safety-Planning Trade-offs. In Chapter 5 we re-formulate the approximation problem, splitting ϵ into (a) an ϵ_T that describes how close to time-optimal a kinodynamic trajectory is and (b) an ϵ_S that describes how well it observes the safety margin.

We note that proofs somewhat similar in flavor to the constructive trajectory proofs and adversary game proofs are found in recursion theory and game theory.

1.5.3 Thesis Outline

1.5.3.1 Basic Kinodynamic Planning

Chapter 2 presents the first probably-good approximation results for optimal kinodynamic planning in more than one dimension. The restricted situation of a point mass robot amongst polyhedral obstacles in two or three dimensions is considered, and L_∞-norm velocity and acceleration bounds are used. The optimal kinodynamic planning problem and the approximation problem are formalized for this class of robot and obstacles, and the approximation algorithm is described. The proof of the algorithm's correctness and of an $O(c^d N(\frac{1}{\epsilon})^{6d})$ time-complexity is given.

The general structure of this proof sets the pattern for most provably-good approximation algorithms for kinodynamic planning.

- First, a theorem shows how closely, for a given safety-margin δ_v and approximation parameter ϵ, a trajectory needs to "track" a δ_v-safe trajectory to guarantee that it will be $(1-\epsilon)$-safe itself.

- Second, it is verified that if a trajectory obeys the given dynamics bounds, then slowing it down uniformly by a factor of ϵ results in a trajectory that obeys dynamics bounds that are smaller in magnitude by a $\Omega(\epsilon)$ factor.

- Third, a *tracking lemma* shows how to choose the reachability graph parameters to so that in the absence of obstacles they will generate a trajectory that mimics this ϵ-slowed, or *ϵ-time rescaled trajectory*, to a desired tolerance. In particular, the size of the timestep shrinks polynomially with the closeness of the approximation.

For the simple problem discussed in Chapter 2, a sufficient set of primitive controls is the set of constant accelerations whose components are members of the set $\{-\overline{a}, 0, \overline{a}\}$. Using the steps together we show that a sufficiently small timestep is $\Omega(\epsilon^2)$. We also show that safety-checking can be done in time $O(N)$ for each graph edge.

1.5.3.2 Better Techniques For Better Bounds

In Chapter 3 we refine the both the complexity and accuracy results of the previous chapter. The results directly generalize to robots obeying decoupled dynamics and decoupled dynamics bounds.

First, we use constructive proof techniques to improve the earlier tracking lemma, showing that the size of the timestep will shrink linearly with how closely we require a graph trajectory to track an ϵ-time rescaled trajectory. This results in lowering the complexity bound of the algorithm to $O(c^d N(\frac{1}{\epsilon})^{3d})$.

Second, we show that for robots obeying L_∞-norm velocity and acceleration bounds the following is true: any δ_v-safe trajectory *obeying the original dynamics bounds* is tracked closely enough by some trajectory generated by the primitive controls above and an $\Omega(\epsilon)$ timestep, so that the tracking trajectory is $(1 - \epsilon)$-safe and meets the endpoint approximation conditions. Furthermore, the tracking trajectory will take the same amount of time as the original. This means that for robots obeying decoupled dynamics bounds, there is an approximation algorithm that generates kinodynamic trajectories that are not only ϵ-optimal, but that take the same amount of time as the optimal kinodynamic trajectories.

These algorithms have been implemented for point mass robots in two dimensions. We briefly describe an implementation and empirical results. Ours is the first implementation of a *provably-good* approximation algorithm for planning time-optimal motions in more than one dimension.

1.5.3.3 Kinodynamic Planning For Robots With Coupled Dynamics Bounds

Chapter 4 describes provably-good approximation algorithms for robots obeying coupled dynamics bounds, specifically for a class of robot systems that includes

(a) d-link open-chain manipulators with revolute and prismatic joints as well as (b) Cartesian robots obeying L_2-norm dynamics bounds. The algorithms find two types of trajectories: ones that obey piecewise-constant extremal controls, and ones that obey piecewise-constant near-extremal accelerations.

The keys to the correctness of these algorithms are two *Coupled Tracking Lemmas* (4.2.1 and 4.2.2). A tracking lemma for robots obeying L_∞ dynamics bounds obviously follows from a lemma for one dimension; however, for robots obeying coupled dynamics bounds the choice of one component of an acceleration (or velocity) vector can restrict the choices of the other components. Analyses of two games against an adversary are used to reduce the proofs to cases in one-dimension. Generating an extremal or near-extremal trajectory that tracks an ϵ-time-rescaled trajectory in d dimensions[31] directly corresponds to winning d simultaneous games against the adversary.

While it might appear that the state-dependence of the open-kinematic chain dynamics equation (4.1) would make proving the correctness of the algorithm difficult, the appropriate tracking lemmas limit the difficulty to detail. With the appropriate tracking lemmas, open-kinematic chain dynamics becomes conceptually simple because perturbations of size Δp in configuration and Δv in velocity perturb the boundary of the set of viable accelerations by a magnitude of $O(\Delta p + \Delta v)$.

1.5.3.4 Further Kinodynamic Planning

Chapter 5 contains two results in different directions of kinodynamic planning research.

By extending the techniques in Chapter 4, we show the existence of a provably good approximation algorithm for optimal kinodynamic planning that generates trajectories that obey *bang-bang* control. In proving the correctness and complexity of the algorithm, we show that if an optimal kinodyanmic trajectory exists, then there is an ϵ-optimal bang-bang trajectory with $o(\epsilon)$ time between switchings.

Finally, while we formulated the optimal kinodynamic planning problem to obtain an analytic trade-off relationship between the closeness to time-optimality of the motions found and the time used to plan them, we propose a practically-motivated extension of the formulation. We believe that algorithms that allow one to trade-off among time-optimality, safety, and running time would be desirable. Towards this goal, instead of using a single parameter ϵ to describe approxima-

[31] d is the dimension of the *configuration space*, *i.e.*, degrees of freedom of the robot.

tion goodness, we introduce two parameters ϵ_T and ϵ_S that (respectively) describe closeness to optimality in execution time and in observance of the safety margin. In addition to asymptotic results, we derive *equicomplexity curves* to show how our algorithms allow one to trade-off between time and safety.

1.5.4 Publication Note

Portions of the this material have been presented, or announced in abbreviated form, at conferences [CDRX88,DX89b,DX89a,DX90a,DX91]. Chapter 2 represents joint work with John Canny and John Reif.

Chapter 2

Cartesian Kinodynamic Planning

Given a robot system, we wish to find a minimal-time motion from a start state (position and velocity) to a goal state, while avoiding obstacles and respecting dynamic constraints on velocity and acceleration. Finding an exact solution for this problem is generally $\mathcal{NP}$-hard (in more than two dimensions), even for simple geometries and dynamics. Therefore, we turn to approximation algorithms to plan these motions. Ideally, we would like to find an algorithm for which one could guarantee the solution accuracy to some tolerance ϵ. While one might expect such an algorithm to run longer when a more accurate solution was required, a *probably-good polynomial-time approximation algorithm* would have a running-time bound polynomially dependent on $\frac{1}{\epsilon}$.

Because the problem is hard, we start with a simplified case. In this chapter, we consider the restricted situation of particle dynamics in two and three dimensions. This will also serve as a simple example of the kinodynamic planning framework. First, we define an appropriate optimal kinodynamic planning problem for two- and three-dimensional problems with polyhedral obstacles. This is the simplest formulation of the *Optimal Cartesian Kinodynamic Planning Problem.*

We describe a simple but provably-good polynomial-time ϵ-approximation algorithm for this problem. A proof of the complexity bounds and accuracy of the algorithm are presented. The high-level proof structure will be used and the problem formulation generalized in later chapters.

2.1 Kinodynamic Planning For A Simple Robot

2.1.1 Problem Formulation

We consider the following restricted problem. (See figure 1.2.) A point mass in $\Re^d$ ($d = 2, 3$) must be moved from a state $\mathbf{S} = (\mathbf{s}, \dot{\mathbf{s}})$ to a goal state $\mathbf{G} = (\mathbf{g}, \dot{\mathbf{g}})$. In the course of the motion, the point must avoid a set $\mathcal{O}$ of polyhedral obstacles, which, with world boundaries, constitute the *kinematic constraints of the problem.*

We will denote the configuration space $\Re^d$ by C, and its phase space by TC. Phase space TC is the robot state space and is isomorphic to $\Re^{2d}$. Thus, a point in TC is a *(position, velocity)* pair such as $\mathbf{S}$ or $\mathbf{G}$. By using a configuration space [LP83] approach, this is readily extended to cover a rigid non-rotating robot geometrically described by a union $\mathcal{R}$ of convex polyhedra.

Formally, a robot motion over a time interval $[0, T_f]$ can be specified by a twice-differentiable map $\mathbf{p} : [0, T_f] \to C$. This map is the *path* of the motion. In kinodynamic planning, the motion must obey dynamics and dynamics constraints, and it is convenient to specify $\dot{\mathbf{p}}$ explicitly. The *trajectory* of a robot motion taking time T_f is the map $\Gamma : [0, T_f] \to TC$ given by $\Gamma(t) = (\mathbf{p}(t), \dot{\mathbf{p}}(t))$. We denote the position and velocity components of a subscripted trajectory Γ_r by $\mathbf{p}_r$ and $\dot{\mathbf{p}}_r$, respectively, and its time by T_r. While a motion $\mathbf{p}$ can be given directly as a function of time, two equivalent specifications are useful: (a) an initial position $\mathbf{p}_0$ and a velocity function $\mathbf{v} = \dot{\mathbf{p}}$, and (b) an initial state $(\mathbf{p}_0, \mathbf{v}_0)$ and an acceleration function $\mathbf{a} = \ddot{\mathbf{p}}$.

Movement is controlled by applying forces or commanding accelerations, which are equivalent for a point mass. The motion must respect upper bounds on the magnitudes of the acceleration and velocity. At all times t the acceleration $\ddot{\mathbf{p}}(t)$ and the velocity $\dot{\mathbf{p}}(t)$ must obey

$$\|\dot{\mathbf{p}}(t)\|_\infty \leq \overline{v} \qquad \text{and} \tag{2.1}$$

$$\|\ddot{\mathbf{p}}(t)\|_\infty \leq \overline{a}. \tag{2.2}$$

Eqs. (2.1) and (2.2) are the *dynamics bounds.* The simple dynamics law $\mathbf{f} = m\mathbf{a}$ and the dynamics bounds comprise the *dynamics constraints* of the problem. We call a robot obeying these simple constraints *a Cartesian robot with L_∞ dynamics bounds*, or a *a Cartesian robot* for short.

We assume that the obstacles $\mathcal{O}$ are represented by a set of convex, possibly overlapping polyhedra. Suppose these convex polyhedra have a total of n faces overall. We call n the *combinatorial complexity* of $\mathcal{O}$. Note that n is also the number of bounding halfplanes of the obstacles. *Free space* is the complement of these obstacles. We assume that the set of free configurations is bounded by a d-cube of side length l. A *general kinodynamic planning problem*, then, is a tuple $(\mathcal{O}, \mathbf{S}, \mathbf{G}, l, \overline{a}, \overline{v})$.

An *exact solution* to the kinodynamic planning problem is a trajectory Γ such that $\Gamma(0) = \mathbf{S}$, $\Gamma(T_f) = \mathbf{G}$, and Γ obeys the kinodynamic constraints. That is, the path $\mathbf{p}$ avoids all obstacles, the velocity $\dot{\mathbf{p}}$ respects (2.1), and $\ddot{\mathbf{p}}$ respects (2.2). The *time-optimal* kinodynamic planning problem is to find a kinodynamic solution that takes a globally minimal time, that is, the fastest trajectory in the set of all kinodynamic solutions. A solution is represented as a suitable encoding of the start state $\Gamma(0)$ and the acceleration function $\mathbf{a}$.

A theoretically time-optimal solution may require unrealizable precision in control or sensing and thus be unexecutable by a physical robot. For this reason, an optimal solution should observe a safety margin; the margin we define is speed-dependent. Furthermore, the safety margin ensures the existence of a family of solutions "nearby" in time and in phase-space that "approximate" the optimal safe solution. Safety margins and the existence of such a "tube" of approximating are not only practically motivated, but mathematically necessary for our approach.

A *δ_v-safe kinodynamic solution* avoids all obstacles by a safety margin δ_v. In this thesis, we define this safety margin to be an affine function of the trajectory speed. Under a simple model, this corresponds to how accurately and quickly a robot senses its position and velocity, combined with how quickly it can correct for velocity errors. (See section 1.2.4.1.) Two positive scalars c_0 and c_1 parameterize this safety margin, which one can view as an obstacle-free tube centered about the path, as in Figure 1.10.

Formally, a *δ_v-safe kinodynamic solution* has the property that for all times t in $[0, T_f]$, there exists a ball about $\mathbf{p}(t)$ in free space of radius

$$\delta_v(c_0, c_1)(\dot{\mathbf{p}}(t)) = c_0 + c_1 \|\dot{\mathbf{p}}(t)\|. \tag{2.3}$$

We will drop the parameters c_0 and c_1 in the discussion when confusion will not arise. Note that δ_v-safety is is an example of a kinodynamic constraint that is neither a pure kinematic constraint nor a pure dynamics constraint. A *δ_v-safe* *kinodynamic planning problem*, then, is a tuple $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$. We call $\overline{a}$, $\overline{v}$, l, c_0, and c_1 the *kinodynamic bounds.*

For fixed c_0 and c_1, consider the class of all δ_v-safe kinodynamic solutions. We define an *optimal δ_v-safe kinodynamic solution* to be a solution whose time is minimal in this class. We will henceforth employ the term *optimal safe kinodynamic solution* or *optimal kinodynamic solution* since δ_v-safety is the only type of safety we consider here, and we will only consider planning trajectories that respect a safety margin. The problem of finding such a solution is called the *optimal kinodynamic planning problem.* When the robot is a Cartesian robot (with L_∞ dynamics bounds), we call the problem the *Optimal Cartesian Kinodynamic Planning Problem.* Note that the definition of *time-optimal* found earlier does not include a safety margin, while the definition of *optimal* here means *time-optimal with respect to a δ_v safety margin.*

We now specify what it means for a kinodynamic solution Γ_q to be *ϵ-optimal*, where a positive $\epsilon < 1$ parameterizes the closeness of the approximation. Recall our notational convention: trajectory $\Gamma_q = (\mathbf{p}_q, \dot{\mathbf{p}}_q)$ and takes time T_q.

First of all, Γ_q must obey the safety margin

$$\delta'_v(c_0, c_1)(\dot{\mathbf{p}}_q) = (1-\epsilon)\delta_v(c_0, c_1)(\dot{\mathbf{p}}_q). \tag{2.4}$$

Second, if an optimal safe trajectory takes time T_{opt}, then we require that

$$T_q \leq (1+\epsilon)T_{opt}.$$

Now, let us say that an approximating state $(\mathbf{x}', \dot{\mathbf{x}}')$ is "ϵ-close" to a reference state $(\mathbf{x}, \dot{\mathbf{x}})$ if

$$\begin{aligned} \|\mathbf{x} - \mathbf{x}'\| &= O(\epsilon), \qquad \text{and} && (2.5) \\ \left\|\dot{\mathbf{x}} - \dot{\mathbf{x}}'\right\| &= O(\epsilon). && (2.6) \end{aligned}$$

Our use of "ϵ"-closeness will always occur in a context where the origin of the constant omitted in the $O()$ notation is obvious or is shown in the discussion, such as in the analysis of an algorithm.

For the final criterion, we require that $\Gamma_q(0)$ and $\Gamma_q(T_q)$ be ϵ-close to the desired start and goal states $\mathbf{S}$ and $\mathbf{G}$, respectively.

In order to obtain our result, we must assume four things: a velocity bound, a workspace diameter bound, L_∞-norm acceleration, and safety. Each of these assumptions can be motivated in physical terms. For example, robots exist in the physical world, and of course any actual robot will have bounded maximum velocity and a bounded workspace. However, the proofs in this thesis do not go through if any of these assumptions is dropped. In later chapters, we relax the L_∞-norm. Safety, as we shall see, proves to be a crucial assumption.

2.1.2 Statement of Results

In section 2.2.2 we describe a provably good approximation algorithm for the optimal safe kinodynamic planning problem. Concisely stated, we show:

Theorem 2.1.1 *Let $\mathcal{K} = (\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ be an instance of the Optimal Cartesian Kinodynamic Planning Problem in two or three dimensions. Let $0 < \epsilon < 1$. Let n be the combinatorial complexity of the obstacles $\mathcal{O}$.*

Suppose there is a $\delta_v(c_0, c_1)$-safe trajectory that obeys the dynamics bounds $\overline{a}$ and $\overline{v}$ and goes from $\mathbf{S}$ to $\mathbf{G}$ in time T_{opt}. Then the algorithm finds a $\delta'_v(c_0, c_1)$-safe trajectory that obeys the dynamics bounds, takes at most time $(1+\epsilon)T_{opt}$, and goes from some $\mathbf{S}^*$ to some $\mathbf{G}^*$ such that $\mathbf{S}^*$ is ϵ-close to $\mathbf{S}$ and $\mathbf{G}^*$ is ϵ-close to $\mathbf{G}$.

The running time of the algorithm is

$$O\left(n\left[\frac{l\overline{v}\gamma^3}{\epsilon^6}\right]^d\right),$$

$d = 2, 3$, *where*

$$\gamma = \max\left(\frac{\overline{a}(c_1+1)}{c_0}, \sqrt{\frac{\overline{a}(c_1+1)}{2c_0}}, \frac{\overline{a}}{\overline{v}}\right).$$

Thus, $c_A = \frac{l\overline{v}\gamma^3}{\epsilon^6}$ in Table 1.5.1.

An Optimal Cartesian Kinodynamic Planning Problem has three complexity components. The combinatorial complexity of $\mathcal{K}$ is the number n of bounding halfplanes of the obstacles $\mathcal{O}$. The *algebraic complexity of the geometry* is the number of bits necessary to encode the coordinates of the vertices of $\mathcal{O}$ and the start and goal states. The *complexity of the kinodynamic bounds* is the number of bits necessary to encode the kinodynamic bounds $(\overline{a}, \overline{v}, c_1, c_0)$. In the language of combinatorial optimization [PS], we show that our algorithm is an ϵ-approximation scheme that is *fully polynomial* in the combinatorial and algebraic complexity of the geometry, and *pseudo-polynomial* in the kinodynamic bounds.

Note that we cannot claim that the approximately optimal safe solution is necessarily near to a truly optimal safe solution in position. In this respect it is useful to compare our result to Papadimitriou's fully polynomial approximation scheme for 3D Euclidean shortest path [Pap85]. Specifically, neither method necessarily finds a solution that is spatially close to the optimal path, but merely one that has a length (time) that is not too much longer than the optimal length (time). In fact, this must be so, because of the shortest path $\mathcal{NP}$-hardness result of [CR87].

These above results can be extended to a rigid, non-rotating robot whose geometry is given by a union $\mathcal{R}$ of convex polyhedra. This configuration space transformation has been discussed extensively in the literature (see, *e.g.*, [LP83,Don87]). The algorithm of [LP83] could be used as a preprocess to reduce the planning problem for $\mathcal{R}$ amidst $\mathcal{O}$ to the point navigation problem we discuss. Since the dynamics equations for such a robot are identical to those of a point robot, we only need to map the problem to this configuration space and apply the algorithm. This simple problem is identical to that of a two- or three-degree-of-freedom (3DOF) robot with constant, decoupled dynamics equations and decoupled dynamics bounds and polyhedral *configuration-space obstacles* with complexity N.

2.2 Algorithm and Analysis

2.2.1 The General Idea

Our approach transforms the problem of finding an approximately minimal-time trajectory to finding the shortest path in a directed graph. The vertices of the graph "discretize" the statespace TC, and the edges of the graph correspond to trajectory segments that each take time τ, a parameter computed by the algorithm.

Given the acceleration bounds $\overline{a}$, let $\mathcal{A}$ be the set of constant accelerations whose (vector) components are members of the set $\{-a, 0, a\}$. Let us choose a timestep τ such that velocity bound $\overline{v}$ is a multiple of $\overline{a}\tau$. Applying a member of $\mathcal{A}$ for duration τ is called an $(\overline{a},\tau)$-*bang*. (See figure 2.2.1.) We also use this term to refer to the resulting trajectory segment: we say that there is an $(\overline{a},\tau)$-*bang* from state $\mathbf{X}$ to state $\mathbf{Y}$ if following an $(\overline{a},\tau)$-bang control at state $\mathbf{X}$ results in state $\mathbf{Y}$.

Suppose $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*) \in TC$ such that $\dot{\mathbf{s}}^*$ is a vector of multiples of $\overline{a}\tau$. Suppose that $(\mathbf{x}, \dot{\mathbf{x}})$ is a state reachable from $\mathbf{S}^*$ by some sequence of $(\overline{a},\tau)$-bangs. Then for each coordinate i,

$$\begin{aligned} p_i &= s_i^* + \tfrac{m_i}{2}\overline{a}\tau^2 \quad \text{and} \\ \dot{p}_i &= \dot{s}_i^* + n_i\overline{a}\tau \end{aligned} \tag{2.7}$$

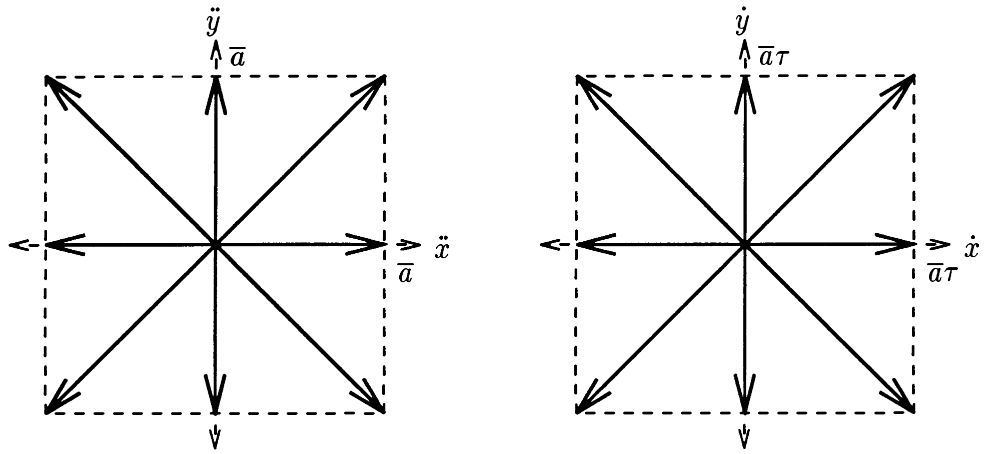


Figure 2.1: $(\overline{a}, \tau)$-bangs for a planar robot. At left, accelerations whose components are members of $\{-a, 0, a\}$ generate $(\overline{a}, \tau)$-bangs. At right: velocity changes corresponding to $(\overline{a}, \tau)$-bangs.

for some integers m_i and n_i. Thus, all states reachable from $\mathbf{S}^*$ under a sequence of $(\overline{a}, \tau)$-bangs belong to a set of states that lie at the interstices of an underlying, regular grid embedded in TC. This grid has spacings of $\frac{\overline{a}\tau^2}{2}$ in position and $\frac{\overline{a}\tau}{2}$ in velocity, as shown in figure 1.14. We call this set of intersticial states the *TC-grid*, and each of these states a *TC-gridpoint.* We call a trajectory that results from a sequence of $(\overline{a}, \tau)$-bangs between TC-gridpoints an *$(\overline{a}, \tau)$-grid-bang trajectory.* Its velocity function is called a *grid-bang velocity function.*

We say that state $(\mathbf{x}, \dot{\mathbf{x}})$ obeys δ'_v-safety if the ball of radius $\delta'_v(\|\dot{\mathbf{x}}\|)$ about $\mathbf{x}$ lies in free space. (Recall equation (2.4).) $\mathbf{S}^*$, $\overline{v}$, δ'_v, $\mathcal{O}$, and $(\overline{a}, \tau)$-bangs determine a *reachability graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$ embedded in TC. The vertices $v_i \in \mathcal{V}$ are the TC-gridpoints. The edges $e_j \in \mathcal{E}$ are[1] δ'_v-safe $(\overline{a}, \tau)$-bangs between pairs of these vertices. We say that τ, $\mathbf{S}^*$, $\overline{a}$, $\overline{v}$, c_0, c_1, $\mathcal{O}$, and ϵ *induce the reachability graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$. We call $\mathbf{S}^*$ the *root vertex state*, or *root vertex* of the reachability graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

The smaller τ is, the finer the underlying TC-grid, and the better some $(\overline{a}, \tau)$-grid-bang trajectory will approximate an arbitrary trajectory that starts at $\mathbf{S}^*$ and obeys the kinodynamic constraints. If τ is small enough and Γ_{opt} is an optimal

[1] Strictly speaking, they are identified with the trajectory segments, but we use the correspondence strongly and so often that not to say "are" would be cumbersome.

kinodynamic trajectory from a state $\mathbf{S}$, adequately approximated by $\mathbf{S}^*$, to a state $\mathbf{G}$, then it is intuitively plausible that there will be an $(\overline{a},\tau)$-grid-bang trajectory Γ_q going from $\mathbf{S}^*$ to a vertex state $\mathbf{G}^*$ approximating $\mathbf{G}$ in time

$$T_q \leq (1+\epsilon)T_{opt}.$$

It would follow that this would reduce the problem of finding an ϵ-optimal trajectory to the shortest path in $\mathcal{G}(\mathcal{V},\mathcal{E})$ from $\mathbf{S}^*$ to a vertex approximating $\mathbf{G}$, since each $(\overline{a},\tau)$-bang takes time τ. This is the basic idea behind the algorithm presented in this chapter.

2.2.2 The Algorithm

To explain the algorithm, and later, to prove its correctness and complexity, we need two more definitions. These definitions will be used throughout this thesis.

First, we introduce terminology to describe how closely one state approximates another.

Definition 2.2.1 *Let* $\mathbf{X} = (\mathbf{x},\dot{\mathbf{x}})$ *and* $\mathbf{Y} = (\mathbf{y},\dot{\mathbf{y}})$ *be two states, and let* η_x *and* η_v *be non-negative scalars. Suppose that*

$$\begin{aligned} \|\mathbf{x}-\mathbf{y}\|_\infty &\leq \eta_x \quad \text{and} \\ \|\dot{\mathbf{x}}-\dot{\mathbf{y}}\|_\infty &\leq \eta_v. \end{aligned}$$

Then we say that $\mathbf{X}$ is within (η_x,η_v) of $\mathbf{Y}$, *or that* $\mathbf{X}$ approximates $\mathbf{Y}$ to tolerance (η_x,η_v).

Second, we generalize this definition of approximation to trajectories, formalizing the notion of how closely one trajectory might "track" another.

Definition 2.2.2 *Consider two trajectories* Γ_a, $\Gamma_b : [0,T] \to TC$. *Given two non-negative scalars* η_x *and* η_v, *we say that* Γ_a approximately tracks Γ_b to tolerance (η_x,η_v) in the L_p-norm *if for all times* t,

$$\begin{aligned} \|\mathbf{p}_a(t)-\mathbf{p}_b(t)\|_p &\leq \eta_x, \quad \text{and} \\ \|\dot{\mathbf{p}}_a(t)-\dot{\mathbf{p}}_b(t)\|_p &\leq \eta_v. \end{aligned} \tag{2.8}$$

We will use the L_∞ norm except where otherwise noted. Figure 2.2 illustrates what this might look like in the phase-space.

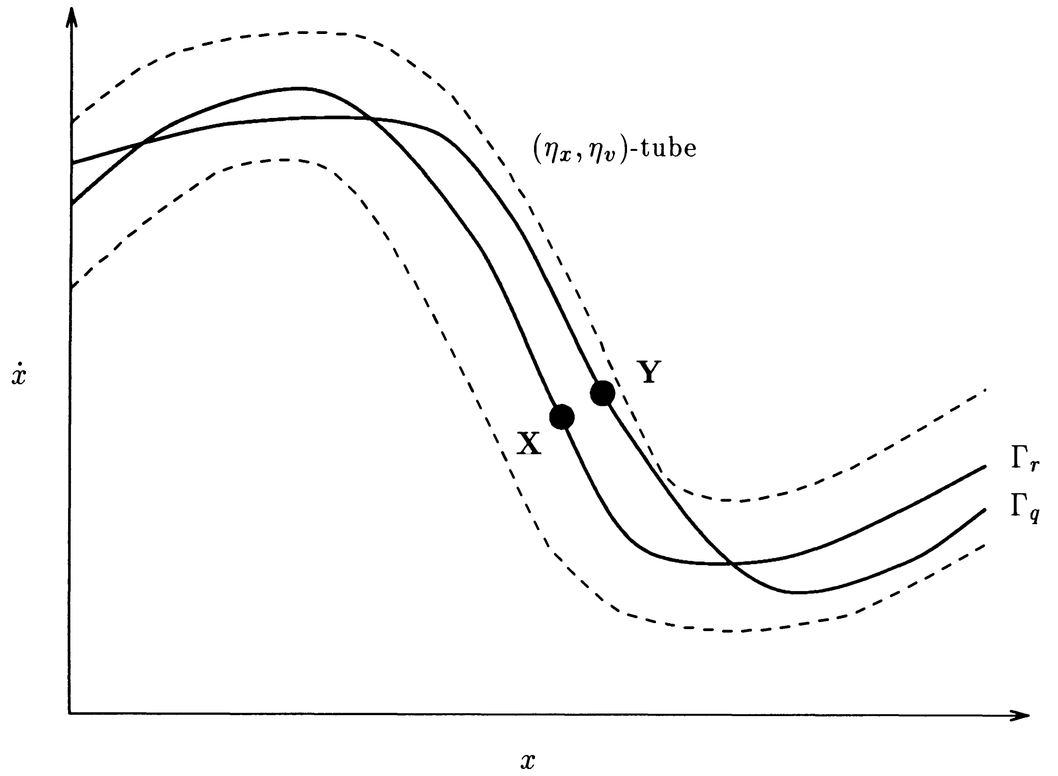


Figure 2.2: An (η_x, η_v)-tube projected into phase-space for one dimension. Trajectory Γ_q approximately tracks Γ'_r to tolerance (η_x, η_v).

We now give a high-level description of a provably-good polynomial approximation algorithm for optimal Basic Kinodynamic Planning. A method for checking the safety of $(\overline{a}, \tau)$-bangs is given in section 2.2.5. Given a problem instance $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ and an approximation parameter ϵ, the algorithm does the following:

Algorithm 1

1. Chooses a timestep τ as a function of $\overline{a}$, $\overline{v}$, ϵ, c_0, and c_1. Specifically, the algorithm chooses the largest τ such that $\tau \leq \frac{\epsilon \overline{v}}{\overline{a}}$, $\overline{a}\tau$ divides $\overline{v}$, and
$$\tau \leq \frac{\epsilon}{13} \min\left(\sqrt{\frac{2c_0\epsilon}{\overline{a}(c_1+1)}}, \frac{c_0\epsilon}{\overline{a}(c_1+1)}\right).$$
2. Next, the algorithm chooses the starting TC-gridpoint $\mathbf{S}^*$ according to the following: $\mathbf{s}^* = \mathbf{s}$, and $\dot{\mathbf{s}}^*$ is the multiple of $\overline{a}\tau$ closest to $\frac{\dot{\mathbf{s}}}{1+\epsilon}$.
3. It then searches for the shortest path in the induced embedded graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, described above, from $\mathbf{S}^*$ to any state (vertex) that is within $(\frac{\overline{a}\tau^2}{2}, \frac{\overline{a}\tau}{2})$ of $(\mathbf{g}, \frac{\dot{\mathbf{g}}}{1+\epsilon})$. The algorithm explores the graph using breadth-first search, checking the δ'_v-safety of each $(\overline{a}, \tau)$-grid-bang it considers.

To show the correctness and complexity of the algorithm we show how to choose τ so that the following holds: if there exists a δ_v-safe trajectory from $\mathbf{S}$ to $\mathbf{G}$ taking time T, then there also exists a δ'_v-safe $(\overline{a}, \tau)$-grid-bang trajectory between states ϵ-close to $\mathbf{S}$ and $\mathbf{G}$ that takes time $(1+\epsilon)T$.

We first observe that if trajectory Γ_{opt} obeys dynamics bounds $\overline{a}$ and $\overline{v}$, then there is a time-rescaled [Hol83] trajectory Γ'_{opt} that takes time $(1+\epsilon)T_{opt}$ and that obeys dynamics bounds $\frac{\overline{a}}{(1+\epsilon)^2}$ and $\frac{\overline{v}}{(1+\epsilon)}$. We then choose η_x and η_v that guarantee that if Γ_q tracks Γ'_{opt} to tolerance (η_x, η_v), then it will be δ'_v-safe. We then show there is a τ proportional to ϵ^2 such that there exists an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q that (a) approximately tracks Γ'_{opt} to this tolerance, and (b) is within $(\frac{\overline{a}\tau^2}{2}, \frac{\overline{a}\tau}{2})$ of Γ'_{opt} when $t = 0$ and when $t = (1+\epsilon)T_{opt}$. The latter implies ϵ-closeness.

Finally, we show that δ_v-safety-checking is $O(n)$ per $(\overline{a}, \tau)$-bang. Recalling that TC-gridpoints have the form (2.7), we find that the number of vertices in the induced statespace graph is $O\left(\frac{l\overline{v}}{\overline{a}^2\tau^3}\right)^d$. The definition of $(\overline{a}, \tau)$-bang implies that the maximal out-degree in $\mathcal{G}$ is 3^d, but since $d \leq 3$, we get the complexity bound in Theorem 2.1.1 .

2.2.3 Time-Rescaling and Safe Tracking

We say that a path $\mathbf{p}$ is *traversed* by a trajectory Γ if the image of the position component of Γ is equal to the image of $\mathbf{p}$.

Lemma 2.2.1 *If path $\mathbf{p}$ is traversed in time T_r by a trajectory Γ_r under acceleration bound $\overline{a}$ and velocity bound $\overline{v}$, then there exists some Γ_r' that traverses $\mathbf{p}$ in time $T_r(1+\epsilon)$ while obeying acceleration bound $\frac{\overline{a}}{(1+\epsilon)^2}$ and velocity bound $\frac{\overline{v}}{1+\epsilon}$. In particular, this is true of $\Gamma_r' = (\mathbf{p}_r', \dot{\mathbf{p}}_r')$, where*

$$\begin{array}{rcl} \mathbf{p}_r'(t) & = & \mathbf{p}_r\left(\frac{t}{1+\epsilon}\right); \\ \dot{\mathbf{p}}_r'(t) & = & \frac{1}{1+\epsilon}\dot{\mathbf{p}}_r\left(\frac{t}{1+\epsilon}\right). \end{array} \tag{2.9}$$

Proof: Follows from the results of [Hol83] or from direct computation using (2.9). □

To prove the main theorem we need to note that (2.9) preserves δ_v-safety:

Observation 2.2.2 *Suppose Γ_r is a $\delta_v(c_0, c_1)$-safe trajectory from $\mathbf{S}$ to $\mathbf{G}$ that takes time T_r and obeys bounds $\overline{v}$ and $\overline{a}$. Then Γ_r' as defined by (2.9) is $\delta_v(c_0, c_1)$-safe, obeys bounds $\frac{\overline{v}}{1+\epsilon}$ and $\frac{\overline{a}}{(1+\epsilon)^2}$, and goes from $\mathbf{S}' = (\mathbf{s}, \frac{\dot{\mathbf{s}}}{1+\epsilon})$ to $\mathbf{G}' = (\mathbf{g}, \frac{\dot{\mathbf{g}}}{1+\epsilon})$.*

Note that if it is known that ϵ is less than some constant C independent of problem instance, then $\mathbf{S}'$ and $\mathbf{G}'$ are ϵ-close to $\mathbf{S}$ and $\mathbf{G}$.

Intuitively, we expect that if a trajectory Γ_q tracks Γ_r' as defined in (2.9) closely enough, then Γ_q will be $(1-\epsilon)\delta_v$-safe. We have the following lemma, which is independent of norm:

Lemma 2.2.3 (The Safe Tracking Lemma) *Suppose that δ_v is specified by c_0 and c_1 and that Γ_r is a δ_v-safe trajectory. Let $0 < \epsilon < 1$, and let ${\delta_v}' = (1-\epsilon)\delta_v$. Then there exists a tolerance (η_x, η_v) such that for any trajectory Γ_q, the following hold:*

1. *If Γ_q tracks Γ_r to tolerance (η_x, η_v), then Γ_q is δ_v'-safe.*
2. *Furthermore, for any positive β, the following choices suffice:*

$$\begin{array}{rcl} \eta_v & \leq & \frac{c_0\epsilon}{c_1(1-\epsilon)+\beta} \\ \eta_x & \leq & \beta\eta_v. \end{array} \tag{2.10}$$

Proof: We find positive real numbers η_x and η_v such that if Γ_q tracks Γ_r to tolerance (η_x, η_v), then the δ'_v-tube induced by Γ_q lies entirely inside the δ_v-tube induced by Γ_r. If Γ_r is δ_v-safe, this assures that Γ_q is $\delta_v{}'$-safe. Henceforth, let $c'_0 = (1-\epsilon)c_0$ and $c'_1 = (1-\epsilon)c_1$.

Suppose that $\mathbf{x} \in C$ lies inside the δ'_v-tube induced by Γ_q. Then for some $t_x \in [0, T_r]$,

$$\|\mathbf{x} - \mathbf{p}_q(t_x)\| < c'_0 + c'_1 \|\dot{\mathbf{p}}_q(t_x)\|. \tag{2.11}$$

Let $B_\eta(\mathbf{x})$ denote the ball of radius η about $\mathbf{x}$. If Γ_q tracks Γ_r to tolerance (η_x, η_v), then $\mathbf{p}_q(t_x) \in B_{\eta_x}(\mathbf{p}_r(t_x))$ and $\dot{\mathbf{p}}_q(t_x) \in B_{\eta_v}(\dot{\mathbf{p}}_r(t_x))$, Therefore

$$\begin{aligned} \|\mathbf{x} - \mathbf{p}_r(t_x)\| &\leq \|\mathbf{x} - \mathbf{p}_q(t_x)\| + \eta_x, \quad \text{and} \\ \|\dot{\mathbf{p}}_q(t_x)\| &\leq \|\dot{\mathbf{p}}_r(t_x)\| + \eta_v. \end{aligned} \tag{2.12}$$

By substituting into (2.11) and adding η_x to the right-hand side,

$$\|\mathbf{x} - \mathbf{p}_r(t_x)\| < c'_0 + \eta_x + c'_1 \left(\|\dot{\mathbf{p}}_r(t_x)\| + \eta_v\right). \tag{2.13}$$

Now, suppose that $\beta > 0$ and that η_x and η_v satisfy (2.10). It follows that

$$\eta_x + (1-\epsilon)c_1\eta_v \leq \epsilon c_0,$$

and thus

$$c'_0 + \eta_x + c'_1\eta_v \leq c_0. \tag{2.14}$$

But then,

$$c'_0 + \eta_x + c'_1 \left(\|\dot{\mathbf{p}}_r(t_x)\| + \eta_v\right) \leq c_0 + c_1 \|\dot{\mathbf{p}}_r(t_x)\|. \tag{2.15}$$

This implies that $\|\mathbf{x} - \mathbf{p}_r(t_x)\| < c_0 + c_1\|\dot{\mathbf{p}}_r(t_x)\|$ via (2.13). Therefore, $\mathbf{x}$ lies inside the δ_v-tube induced by Γ_r.

Since $\mathbf{x}$ is an arbitrary point inside the δ'_v-tube induced by Γ_q, it follows that the δ'_v-tube induced by Γ_q lies entirely inside the δ_v-tube induced by Γ_r. □

For given δ_v-safety parameters c_0 and c_1 and an approximation parameter ϵ, we call the tolerances (η_x, η_v) that obey (2.10) *ϵ-safe-tracking tolerances.*

Because this lemma holds for any L_p-norm, our results are easily extendible to safety margins given in any such norm. As a direct corollary, we have

Corollary 2.2.4 *Let δ_v be specified by c_0 and c_1, and let $0 < \epsilon < 1$. Let the tracking tolerance (η_x, η_v) satisfy the condition*

$$\begin{aligned} \eta_v &\leq \tfrac{c_0\epsilon}{c_1+1}, \quad \text{and} \\ \eta_x &\leq \tfrac{c_0\epsilon}{c_1+1}. \end{aligned} \tag{2.16}$$

Suppose that Γ_r is a δ_v safe trajectory, and Γ'_r is obtained by ϵ-time-rescaling Γ_r (as in (2.9)). Suppose that Γ_q tracks Γ'_r to tolerance (η_x, η_v). Then the δ'_v-tube induced by Γ_q lies within the δ_v-tube induced by Γ_r.

2.2.4 The Simple Tracking Lemma

The Simple Tracking Lemma relates a timestep size τ to a tracking tolerance. In particular, it tells how to choose τ to assure that in the absence of obstacles, for every Γ_u that obeys dynamics bounds $\frac{\overline{a}}{(1+\epsilon)^2}$ and $\frac{\overline{v}}{1+\epsilon}$ there will exist an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q that tracks Γ_u to tolerance (η_x, η_v).

2.2.4.1 Knotpoints

We first show that if Γ_u obeys dynamics bounds $\frac{\overline{a}}{(1+\epsilon)^2}$ and $\frac{\overline{v}}{1+\epsilon}$, then at "knotpoints" separated in time by an adequately large multiple of τ, some $(\overline{a}, \tau)$-grid-bang trajectory Γ_q will be within $(\frac{\overline{a}\tau^2}{2}, \frac{\overline{a}\tau}{2})$ of Γ_u. That is, for a large enough integer N, $\Gamma_q(kN\tau)$ will approximate $\Gamma_u(kN\tau)$ to tolerance $(\frac{\overline{a}\tau^2}{2}, \frac{\overline{a}\tau}{2})$ for all relevant k.

Lemma 2.2.5 *Let $\epsilon > 0$, and let Γ_u be a trajectory respecting dynamics bounds $\frac{\overline{a}}{(1+\epsilon)^2}$ and $\frac{\overline{v}}{1+\epsilon}$. Let $\tau \leq \frac{\epsilon\overline{v}}{\overline{a}}$, and suppose that $\overline{a}\tau$ divides $\overline{v}$.*

Suppose that

$$N \geq \frac{6}{\epsilon}. \tag{2.17}$$

Then if $(\mathbf{p}_{q0}, \dot{\mathbf{p}}_{q0})$ is a TC-gridpoint such that $\|\mathbf{p}_{q0} - \mathbf{p}_u(0)\| \leq \frac{\overline{a}\tau^2}{2}$ and $\|\dot{\mathbf{p}}_{q0} - \dot{\mathbf{p}}_u(0)\| \leq \frac{\overline{a}\tau}{2}$, there exists an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q such that $\Gamma_q(0) = (\mathbf{p}_{q0}, \dot{\mathbf{p}}_{q0})$ and that $\Gamma_q(N\tau)$ is within $(\frac{\overline{a}\tau^2}{2}, \frac{\overline{a}\tau}{2})$ of $\Gamma_u(N\tau)$.

Proof:

Since we are using the L_∞-norm, it is sufficient to consider the case of a one-dimensional configuration space C. We show that the lower bound on N given by (2.17) is sufficiently large to guarantee that if Γ_u meets the hypotheses of the lemma then some $(\overline{a}, \tau)$-grid-bang trajectory can meet the endpoint conditions.

Let $\epsilon > 0$, $\tau \leq \frac{\epsilon\overline{v}}{\overline{a}}$, and $\overline{a}\tau | v$. Let $p_u(0)$, $\dot{p_u}(0)$, and $\dot{p_u}(N\tau)$ be fixed, and consider some Γ_u that satisfies these endpoint conditions and the hypotheses of the lemma. Let $(p_{q0}, \dot{p}_{q0})$ be a TC-gridpoint within $(\frac{\overline{a}\tau^2}{2}, \frac{\overline{a}\tau}{2})$ of $\Gamma_u(0)$. To find a sufficiently large N, we introduce variables b and $\mathcal{Q}$. These variables depend on Γ_u. Let b be an integer such that such that

$$|\dot{p}_{q0} + b\overline{a}\tau - \dot{p}_u(N\tau)| \leq \frac{\overline{a}\tau}{2}.$$

Now, define $\mathcal{Q}$ to be the collection of all $(\overline{a}, \tau)$-grid-bang trajectories of time length $N\tau$ starting at $(p_{q0}, \dot{p}_{q0})$ with net velocity change $b\overline{a}\tau$. The positions reached by the different trajectories in $\mathcal{Q}$ at time $N\tau$ form a set of discrete points spaced $\overline{a}\tau^2$ apart. Call these positions the *$\mathcal{Q}$-positions.* If the range of $\mathcal{Q}$-positions spans the range of possible $\Gamma_u(N\tau)$, then for some trajectory $\Gamma_q \in \mathcal{Q}$, $|p_q(N\tau) - p_u(N\tau)| \leq \frac{\overline{a}\tau^2}{2}$.

We show how to choose N so that the maximum $\mathcal{Q}$-position exceeds the maximum possible $p_u(N\tau)$. A similar argument shows that the same N guarantees that the minimum $\mathcal{Q}$-position is less than the minimum possible $p_u(N\tau)$. For the remainder of this proof, we will refer to the two trajectories that achieve the maximum $\mathcal{Q}$-position and the maximum possible $p_u(N\tau)$ as Γ_q and Γ_u, respectively.

The Γ_q that attains the maximum $\mathcal{Q}$-position obeys either (a) full positive acceleration, possibly followed by zero acceleration for one timestep, followed by full negative acceleration[2], or (b) full positive acceleration until its velocity is $\overline{v}$, followed by zero acceleration, followed by full negative acceleration. Similarly, a Γ_u that maximizes $p_u(N\tau)$ obeys either (c) full positive acceleration followed by full negative acceleration, or (d) full positive acceleration until its velocity is $\frac{\overline{v}}{(1+\epsilon)}$, followed by zero acceleration, followed by full negative acceleration.

Consider $\dot{p}_q$ and $\dot{p}_u$ and their role in determining $p_q(N\tau)$ and $p_u(N\tau)$. In the worst case, $\dot{p}_{q0} = \dot{p}_u(0) - \frac{\overline{a}\tau}{2}$ and $\dot{p}_q(N\tau) = \dot{p}_u(N\tau) - \frac{\overline{a}\tau}{2}$. If $p_q(N\tau)$ is to be greater than $p_u(N\tau)$, then we can divide the interval $[0, N\tau]$ into three intervals during which Γ_q "loses ground to" Γ_u, "gains on" Γ_u, and "loses ground to" Γ_u. (See figure 2.3.) In other words, there are times t_c and t_l, $0 < t_c < t_l < N\tau$, such that

$$\begin{array}{lcll} \dot{p}_q(t) & < & \dot{p}_u(t) & \text{if } 0 < t < t_c; \\ \dot{p}_q(t) & > & \dot{p}_u(t) & \text{if } t_c < t < t_l; \quad \text{and} \\ \dot{p}_q(t) & < & \dot{p}_u(t) & \text{if } t_l < t < N\tau. \end{array}$$

Now, when Γ_q is accelerating full-positive,

$$\ddot{p}_q(t) - \ddot{p}_u(t) \geq a\left(1 - \frac{1}{(1+\epsilon)^2}\right) > \frac{\epsilon a}{2}.$$

Similarly, when Γ_q is accelerating full-negative,

$$\ddot{p}_q(t) - \ddot{p}_u(t) < \frac{\epsilon a}{2}.$$

[2]The zero acceleration timestep in the first case occurs if $N - \frac{\dot{p}_q(N\tau) - \dot{p}_{q0}}{\overline{a}\tau}$ is odd.

Hence, when $0 \le t < t_c$, $\ddot{p}_q(t) - \ddot{p}_u(t) > \frac{\epsilon a}{2}$, and when $t_l < t \le N\tau$, $\ddot{p}_q(t) - \ddot{p}_u(t) < -\frac{\epsilon a}{2}$. Then, because $\dot{p}_{q0} \ge \dot{p}_u(0) - \frac{\overline{a}\tau}{2}$ and $\dot{p}_q(N\tau) \ge \dot{p}_u(N\tau) - \frac{\overline{a}\tau}{2}$,

$$\begin{aligned} \textstyle\int_0^{t_c}(\dot{p}_u(t) - \dot{p}_q(t))dt &< \tfrac{\overline{a}\tau^2}{4\epsilon}, \quad \text{and} \\ \textstyle\int_{t_l}^{N\tau}(\dot{p}_u(t) - \dot{p}_q(t))dt &< \tfrac{\overline{a}\tau^2}{4\epsilon}. \end{aligned}$$

Therefore, it is sufficient to choose an N that guarantees

$$\int_{t_c}^{t_l}(\dot{p}_q(t) - \dot{p}_u(t))dt \ge \frac{\overline{a}\tau^2}{2\epsilon}. \tag{2.18}$$

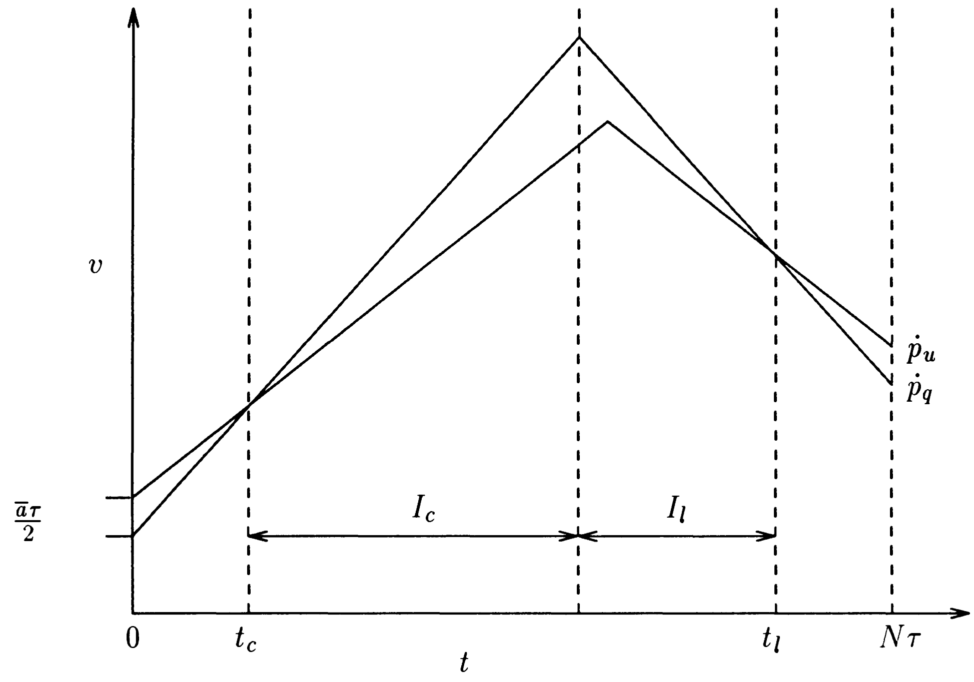


Figure 2.3: Bounding position changes subject to velocity knotpoints. An example of $\dot{p}_u$ and $\dot{p}_q$ that achieve the maximum position subject to conditions at times 0 and $N\tau$. In this case, $\dot{p}_q$ never reaches the maximum allowed velocity $\overline{v}$. N must be large enough so that the distance Γ_q gains over Γ_u during I_c and I_l makes up for the distance Γ_q loses to Γ_u during $[0, t_c]$ and $[t_l, N\tau]$.

Consider the behavior of $\dot{p}_q(t) - \dot{p}_u(t)$ between times t_c and t_l. For now, suppose that $\dot{p}_q(t) < v$ during this time. Then, for an interval of time I_c beginning with t_c, both $\dot{p}_q(t)$ and $\dot{p}_q(t) - \dot{p}_u(t)$ increase; for an interval I_l beginning at most one timestep after I_c and ending with t_l, both $\dot{p}_q(t)$ and $\dot{p}_q(t) - \dot{p}_u(t)$ decrease. Furthermore,

$$\begin{array}{rcll} \dot{p}_q(t) - \dot{p}_u(t) & > & \frac{\overline{a}\epsilon}{2}(t - t_c) & \text{during } I_c, \text{ and} \\ \dot{p}_q(t) - \dot{p}_u(t) & > & \frac{\overline{a}\epsilon}{2}(t_l - t) & \text{during } I_l. \end{array}$$

Using some manipulation, we then see that if $\dot{p}_q(t) < v$ for all $t \in [t_c, t_l]$, the condition

$$t_l - t_c \geq \frac{2\tau}{\epsilon} + 1 \tag{2.19}$$

guarantees that (2.18) is true.

Now, suppose $\dot{p}_q(t) = v$ for some interval $I_m \subset [t_c, t_l]$. (See figure 2.4.) Then, for an interval of time I_c immediately preceding I_m and beginning with t_c, both $\dot{p}_q(t)$ and $\dot{p}_q(t) - \dot{p}_u(t)$ increase; for an interval I_l immediately after I_m and ending with t_l, both $\dot{p}_q(t)$ and $\dot{p}_q(t) - \dot{p}_u(t)$ decrease. However, during I_m, $\dot{p}_q(t) - \dot{p}_u(t) \geq \overline{a}\tau$, since $\tau \leq \frac{\epsilon\overline{v}}{\overline{a}}$. It follows that (2.19) again guarantees that (2.18) is true.

We observe that

$$\begin{array}{rcl} t_c & \leq & \frac{\tau}{\epsilon}, \text{ and} \\ N\tau - t_l & \leq & \frac{\tau}{\epsilon}. \end{array}$$

Recalling (2.19), we see that the following choice of N guarantees that the range of $\mathcal{Q}$-positions will be adequate:

$$N = 4\left\lceil \frac{1}{\epsilon} \right\rceil + 1.$$

Using the fact that $0 < \epsilon < 1$, we obtain the sufficient condition (2.17). □

2.2.4.2 Proving The Simple Tracking Lemma

By bounding the maximum tracking error between knotpoints, we can establish the following.

Lemma 2.2.6 (The Simple Tracking Lemma) *Let $\epsilon > 0$. Let Γ_u be a trajectory that respects dynamics bounds $\frac{\overline{a}}{(1+\epsilon)^2}$ and $\frac{\overline{v}}{1+\epsilon}$ and takes time T_u. Let η_x and η_v be positive. Let $\tau \leq \frac{\epsilon\overline{v}}{\overline{a}}$.*

Suppose that

$$\tau \leq \frac{\epsilon}{13}\min\left(\sqrt{\frac{2\eta_x}{\overline{a}}}, \frac{\eta_v}{\overline{a}}\right), \tag{2.20}$$

and $\overline{a}\tau | v$. Furthermore, let N be given by (2.17), and suppose that $T_u \geq N\tau$. Then in the absence of obstacles there exists an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q respecting *dynamics bounds* $\overline{v}$ *and* $\overline{a}$ *that approximately tracks* Γ_u *to tolerance* (η_x, η_v) *during* $[0, T_u]$ *and obeys the following endpoint conditions:*

$$\begin{aligned} \mathbf{p}_q(0) &= \mathbf{p}_u(0), \\ \|\dot{\mathbf{p}}_q(0) - \dot{\mathbf{p}}_u(0)\|_\infty &\leq \tfrac{\overline{a}\tau}{2}, \\ \|\mathbf{p}_q(T_u) - \mathbf{p}_u(T_u)\|_\infty &\leq \overline{a}\tau^2, \quad \text{and} \\ \|\dot{\mathbf{p}}_q(T_u) - \dot{\mathbf{p}}_u(T_u)\|_\infty &\leq \overline{a}\tau. \end{aligned} \tag{2.21}$$

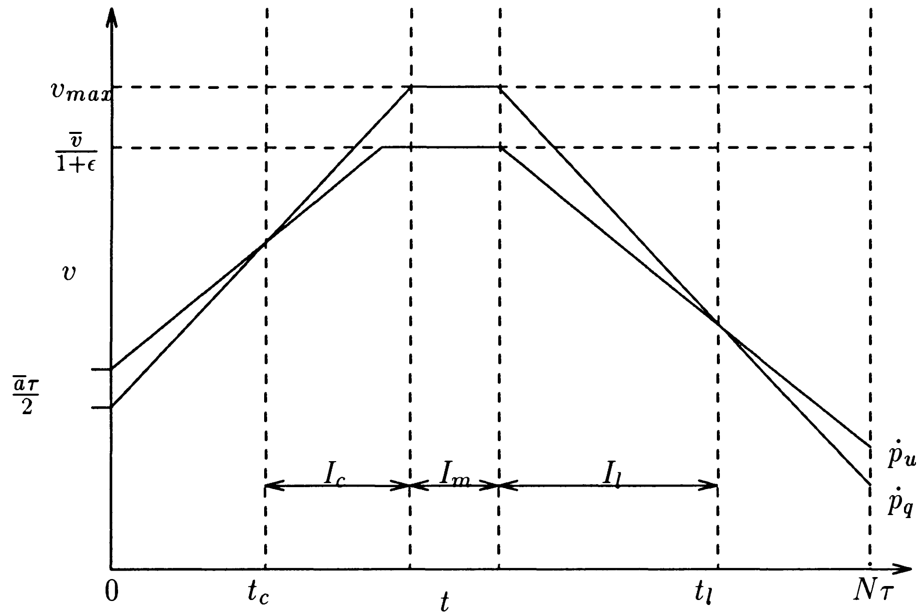


Figure 2.4: Bounding position changes subject to velocity knotpoints, another case. Further towards proving Lemma 2.2.5. An example of $\dot{p}_u$ and $\dot{p}_q$ that achieve the maximum position subject to conditions at times 0 and $N\tau$. In this case, $\dot{p}_q$ sustains the maximum allowed velocity $\overline{v}$ for the interval I_m. N must be large enough so that the distance Γ_q gains over Γ_u during I_c, I_m, and I_l makes up for the distance Γ_q loses to Γ_u during $[0, t_c]$ and $[t_l, N\tau]$. Note that in this figure, the condition $\tau \leq \frac{\epsilon\overline{v}}{\overline{a}}$ is not met.

Proof:

We show that the the upper bound (2.20) is correct. Let the hypotheses of the lemma be satisfied. Let N be given by (2.17). Then it follows from Lemma 2.2.5 that there is an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q such that for any positive integer k

satisfying $kN\tau \leq T_u$,

$$\begin{aligned} \|\mathbf{p}_q(kN\tau) - \mathbf{p}_u(kN\tau)\|_\infty &\leq \tfrac{\overline{a}\tau^2}{2}, \text{ and} \\ \|\dot{\mathbf{p}}_q(kN\tau) - \dot{\mathbf{p}}_u(kN\tau)\|_\infty &\leq \tfrac{\overline{a}\tau}{2}. \end{aligned} \tag{2.22}$$

This can be shown by induction on k.

Now, for all t, $\|\ddot{\mathbf{p}}_u(t) - \ddot{\mathbf{p}}_q(t)\|_\infty \leq 2a$. By considering the relative velocity in the worst case, where along some axis

$$\begin{aligned} |p_q(kN\tau) - p_u(kN\tau)| &= \tfrac{\overline{a}\tau^2}{2}, \\ |\dot{p_q}(kN\tau) - \dot{p_u}(kN\tau)| &= \tfrac{\overline{a}\tau}{2}, \\ |p_q((k+1)N\tau) - p_u((k+1)N\tau)| &= -\tfrac{\overline{a}\tau^2}{2}, \text{ and} \\ |\dot{p_q}((k+1)N\tau) - \dot{p_u}((k+1)N\tau)| &= -\tfrac{\overline{a}\tau}{2}, \end{aligned}$$

we conclude that for all $t \in [0, T_u]$,

$$\|\mathbf{p}_q(t) - \mathbf{p}_u(t)\|_\infty < \frac{\overline{a}(N+1)^2\tau^2}{2} \tag{2.23}$$

To guarantee that the right-hand side of (2.23) is less than η_x, it is sufficient that

$$\tau < \sqrt{\frac{2\eta_x}{\overline{a}}}\left(\frac{1}{N+1}\right). \tag{2.24}$$

Since $\|\dot{\mathbf{p}}_q(t) - \dot{\mathbf{p}}_u(t)\|_\infty < a(N+1)\tau$ when $0 \leq t \leq T_u$, for the velocity case we require that

$$\tau \leq \frac{\eta_v}{\overline{a}}\left(\frac{1}{N+1}\right). \tag{2.25}$$

If $T_u > N\tau$ is not a multiple of $N\tau$, then for some natural numbers $n < N$ and k, $(kN+n)\tau$ is within $\frac{\tau}{2}$ of T_u. Substituting twice the value for N from Lemma 2.2.5 and rounding to simplify yields the condition (2.20). □

2.2.5 Basic Safety Checking

We describe an $O(n)$ time to check whether an $(\overline{a}, \tau)$-bang violates the speed-dependent safety-margin $\delta_v(c_0, c_1)$. We review some basic computational geometry, describe the special case when $c_1 = 0$, and then extend the method to the general case. (See [LP83] for some background on the basic computational-geometric techniques we employ.)

As noted above, we assume that obstacles are the union of (possibly-overlapping) convex polyhedra. For now, let the safety margin be a constant $c_0 > 0$ (*i.e.*,

$c_1 = 0$), and define the B_{c_0} to be the L_∞ ball with radius c_0. Staying c_0-safe relative to a convex polyhedron A is then equivalent to avoiding $\overline{A} = A \oplus B_{c_0}$, where "$\oplus$" denotes the Minkowski sum. Since B_{c_0} is a d-cube, $\overline{A}$ is also a convex polyhedron and has $O(|faces(A)|)$ faces. By taking the Minkowski sum of each of the obstacles with B_{c_0} we obtain the *expanded obstacles.*

Suppose $\overline{A}$ has faces $\{ F_0, \ldots, F_m \}$ lying on the boundary planes of the closed half-spaces $\{ H_0, \ldots, H_m \}$. The boundary plane of each H_i is the kernel of an affine function f_i. If $\mathbf{n}_i$ is a unit vector in the outward normal direction from the boundary plane of H_i, and $\mathbf{y}_i$ is any point on this boundary then

$$f_i(\mathbf{x}) = \langle \mathbf{n}_i, \mathbf{x} \rangle - \langle \mathbf{n}_i, \mathbf{y}_i \rangle. \tag{2.26}$$

The polyhedron $\overline{A}$ is thus described by a set of functions $\mathcal{F} = \{ f_0, \ldots, f_m \}$.

A point $\mathbf{x}$ is on the boundary of $\overline{A}$ if and only if it lies on some *closed* face F_k of $\overline{A}$. Equivalently, $f_k(\mathbf{x}) = 0$, and for all f_j that determine an edge of F_k, $f_j(\mathbf{x}) \leq 0$. Since for a convex polyhedron the numbers of edges and faces are linearly related and an edge is common to two faces, determining whether $\mathbf{x}$ lies on the boundary of any of the expanded of obstacles takes total time $O(n)$.

Without loss of generality, suppose that $(\overline{a}, \tau)$-bang $\mathbf{p}$ begins at $t = 0$ and that $\mathbf{p}(0)$ is c_0-safe. We then can check the c_0-safety of $\mathbf{p}(t)$ by determining whether $\mathbf{p}(t)$ violates the boundary of an expanded obstacle. For a face F_k of $\overline{A}$, we only need to solve $f_k(\mathbf{p}(t)) = 0$, and for each solution t_s check whether $f_j(\mathbf{p}(t_s)) > 0$ for some f_j that determines an edge of F_k with f_k.

Now, consider speed-dependent safety function δ_v. $\mathbf{p}(t)$ is δ_v-safe relative to a convex polyhedron A if and only if it avoids the expanded obstacle $\tilde{A}(\dot{\mathbf{p}}(t)) = A \oplus B_{\delta_v}(\dot{\mathbf{p}}(t))$, where $B_{\delta_v}(\dot{\mathbf{p}}(t))$ is the L_∞ ball with radius $\delta_v(\dot{\mathbf{p}}(t))$. $\tilde{A}(\dot{\mathbf{p}}(t))$ is described similarly to $\overline{A}$, by a set of functions $\tilde{\mathcal{F}} = \{ \tilde{f}_0, \ldots, \tilde{f}_m \}$. For each $f_i \in \mathcal{F}$,

$$\tilde{f}_i(\mathbf{p}(t), \dot{\mathbf{p}}(t)) = \langle \mathbf{n}_i, \mathbf{p}(t) \rangle - \langle \mathbf{n}_i, \mathbf{y}_i + \mathbf{q}_i \|\dot{\mathbf{p}}(t)\|_\infty \rangle, \tag{2.27}$$

where $\mathbf{q}_i$ is a constant vector that depends only on $\mathbf{n}_i$. To check whether $\mathbf{p}(t)$ violates the faces of $\tilde{A}$, we use the $\tilde{f}_i \in \tilde{\mathcal{F}}$ the same way we use the $f_i \in \mathcal{F}$ above.

By the same argument as in checking whether a point is in on an expanded obstacle boundary, we need to solve $O(n)$ equations and check $O(n)$ inequalities, overall. It follows that the cost of safety-checking is $O(n)$ per $(\overline{a}, \tau)$-bang. Since $\mathbf{p}(t)$ is quadratic, $\tilde{f}_k(\mathbf{p}(t), \dot{\mathbf{p}}(t)) = 0$ has solutions of the form $t = a + \sqrt{b}$. When computing the inequalities we can square twice to eliminate the radical, and thus it is adequate to compute square roots symbolically. This implies that safety checking never requires numbers longer (in the number of bits) than a constant-multiple of the length of the longest number in the input. Therefore, the complexity of safety-checking is $O(n)$ under both the Real-RAM and bit-complexity models.

2.2.6 Proving the Main Theorem

We can now prove Theorem 2.1.1.

Proof of Theorem 2.1.1:

Let $\mathcal{K} = (\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ be an instance of the optimal safe kinodynamic planning problem. Let $0 < \epsilon < 1$.

Suppose Γ_{opt} is a $\delta_v(c_0, c_1)$-safe trajectory that obeys the dynamics bounds $\overline{a}$ and $\overline{v}$ and goes from $\mathbf{S}$ to $\mathbf{G}$ in time T_{opt}. By Lemma 2.2.1 and Observation 2.2.2 that follows it, there is a trajectory Γ'_{opt} that is $\delta_v(c_0, c_1)$-safe, obeys bounds $\frac{\overline{v}}{1+\epsilon}$ and $\frac{a}{(1+\epsilon)^2}$, takes time $(1+\epsilon)T_{opt}$, and goes from $\mathbf{S}' = (\mathbf{s}, \frac{\dot{\mathbf{s}}}{1+\epsilon})$ to $\mathbf{G}' = (\mathbf{g}, \frac{\dot{\mathbf{g}}}{1+\epsilon})$.

Now suppose we run the algorithm described in Section 2.2.2. The choice of τ in the algorithm matches the conditions in Lemma 2.2.6 when the values of η_x and η_v from equation (2.16) in Corollary 2.2.4 are substituted into equation (2.20). Furthermore, the algorithm's choice of root vertex state $\mathbf{S}^*$ obeys the condition on $(\mathbf{p}_{q0}, \dot{\mathbf{p}}_{q0})$ in Lemma 2.2.5. Therefore, some $(\overline{a}, \tau)$-grid-bang trajectory beginning at $\mathbf{S}^*$ tracks Γ'_{opt} closely enough to be $\delta'_v(c_0, c_1)$-safe, to obey the dynamics bounds $\overline{a}$ and $\overline{v}$, and to take time $(1+\epsilon)$ to reach a state $\mathbf{G}^*$ within tolerance $(\frac{\overline{a}\tau^2}{2}, \frac{\overline{a}\tau}{2})$ of $\mathbf{G}'$.

Breadth-first search guarantees such a trajectory will be found if there is no $\delta'_v(c_0, c_1)$-safe $(\overline{a}, \tau)$-grid-bang trajectory beginning at $\mathbf{S}^*$ that obeys the dynamics bounds and comes adequately close to $\mathbf{G}'$ in less time. Thus, the algorithm will find a trajectory meeting the conditions of the theorem.

To establish the time complexity of the algorithm, we now bound size of the reachability graph $\mathcal{G}$. The number of vertices is bounded by the number $G_\infty(\overline{a}, \tau, \overline{v}, l, d)$ of TC-gridpoints for a point robot with maximum (L_∞) speed $\overline{v}$ and acceleration $\overline{a}$ in a d-dimensional free-space of diameter l. Without loss off generality, let us choose $\mathbf{s}^*$ to be the "zero" position. Recalling the canonical form of a TC-gridpoint from (2.7), we conclude

$$G_\infty(\overline{a}, \tau, \overline{v}, l, d) = O\left(\left(\frac{vl}{\overline{a}^2\tau^3}\right)^d\right). \tag{2.28}$$

The out-degree of the graph is the number of $(\overline{a}, \tau)$-bangs from a state, 3^d. The cost of checking the safety of a bang is $O(n)$. Therefore, the total complexity of the algorithm can be obtained by substituting τ from (2.20) into (2.28). Since $\epsilon < 1$, we can use

$$\tau \leq \frac{\epsilon^2}{13} \min\left(\sqrt{\frac{2c_0}{\overline{a}(c_1+1)}}, \frac{c_0}{\overline{a}(c_1+1)}\right)$$

instead of (2.20) to get the bound in the theorem. □

2.3 Summary

This chapter formally introduced the kinodynamic planning formulation of optimal trajectory planning. The specific problem, Optimal Cartesian Kinodynamic Planning, considers a point mass robot obeying L_∞ dynamics bounds in an environment of polyhedral obstacles. Although this problem appears simple, finding an exactly optimal solution is $\mathcal{NP}$-hard in three dimensions. Furthermore, the algorithmic result can be generalized to robots obeying decoupled dynamics and dynamics bounds.

The simple algorithm presented was the first *provably-good polynomial-time approximation algorithm* for planning time-optimal robot motions in more than one dimension. It searches for a shortest path in a reachability graph generated by using elementary control primitives, $(\overline{a}, \tau)$-bangs. The trajectory segments generated by these primitives sufficiently "discretize" the vector fields possible under the kinodynamic constraints, to guarantee that a trajectory corresponding to a path in the reachability graph will be ϵ-optimal if the problem has a solution.

The "Scaling-Tracking" method used in proof of the algorithm's correctness and complexity bounds is very important to later chapters of this thesis. This proof structure is characterized by three main features:

1. A lemma relates the safety margin and ϵ to $\Omega(\epsilon)$ ϵ-safe-tracking parameters η_x and η_v ensuring that any trajectory tracking a δ_v-safe trajectory to tolerance (η_x, η_v) will be $(1-\epsilon)\delta_v$-safe, or δ_v'-safe.

2. It is observed that if an optimal kinodynamic trajectory is uniformly slowed down by a factor of ϵ, or ϵ-time-rescaled, then the resulting trajectory will be ϵ-optimal, observe δ_v-safety, and obey tighter dynamics bounds.

3. Given a tracking tolerance (η_x, η_v), where $\eta_x, \eta_v > 0$, a tracking lemma provides criteria for choosing reachability graph parameters guaranteeing that in the absence of obstacles, any trajectory obeying these tighter bounds and starting appropriately close to the root vertex will be tracked to tolerance (η_x, η_v) by some graph trajectory. The graph parameters will be polynomial in η_x and η_v, and the graph size will be polynomial in $\frac{1}{\eta_x}$ and $\frac{1}{\eta_v}$.

Chapter 3

Better Bounds for Cartesian Kinodynamic Planning

3.1 Improved Techniques

3.1.1 Introduction

In the previous chapter, we formalized the optimal and ϵ-optimal Basic Kinodynamic Planning Problems. The provably-good polynomial time approximation algorithm described is simple, and the overall structure of the proof we give is intuitive. Almost all complicated details are contained in the proof the Simple Tracking Lemma (2.2.6), which includes the supporting Lemma 2.2.5.

Although the proof of the lemma may seem tortuous, the strategy behind it is rather naive: it simply focuses on tracking trajectory position. The velocity tracking falls out because the "knotpoints" are too close together for too much velocity error to accumulate, provided that the knotpoint velocity condition (2.22) and the acceleration bound are obeyed. As one might expect, improved complexity and accuracy bounds can be obtained through an analysis that carefully considers tracking position and velocity together. While our proof of the Simple Tracking Lemma combines knotpoints and structural induction to prove the existence of the necessary graph trajectories, we can prove stronger tracking lemmas by giving inductive definitions that simultaneously enforce both position and velocity conditions. These lemmas enable us to improve the ϵ-approximation results for Cartesian robots obeying L_∞ dynamics bounds in both accuracy and complexity.

We modify Algorithm 1 and apply new analytical techniques to lower the complexity of finding ϵ-optimal kinodynamic trajectories. We halve the exponent in the polynomial bounds from $6d$ to $3d$, showing that this second algorithm, Algorithm 2, has complexity $O\left(dc_B^d N \left(\frac{1}{\epsilon}\right)^{3d}\right)$, where N is the geometric complexity of the environment and c_B is a robot-dependent constant. Further modifications yield an algorithm that finds solutions more accurately approximating the start and goal. While the previous algorithms are guaranteed to find solutions that take at most $(1+\epsilon)T_{opt}$ when an optimal solution taking time T_{opt} exists, Algorithm 3 is guaranteed to find a solution that will take *no more than* T_{opt} *time.* The complexity of this algorithm is the same as the second with respect to the approximation parameter ϵ and the geometric complexity N, but it is more sensitive to the dynamics bounds.

The complexity bounds indicate that implementations of these theoretical algorithms could be feasible. We briefly report on a *Common LISP* implementation of the improved algorithms. This implementation is used for the simple experiments in Chapter 5.

3.1.2 Improved Results

We describe two provably good approximation algorithms for the optimal Basic Kinodynamic Planning Problem $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ in d dimensions.

The first algorithm improves on the complexity bounds of Algorithm 1 from Chapter 2, lowering the exponent of $\frac{1}{\epsilon}$ from $6d$ to $3d$.

Theorem 3.1.1 *Let* $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ *be an optimal kinodynamic planning problem for a point mass robot in d dimensions obeying* L_∞ *dynamics bounds. Let* $0 < \epsilon < 1$.

Suppose there is a $\delta_v(c_0, c_1)$*-safe trajectory from* $\mathbf{S}$ *to* $\mathbf{G}$ *taking time* T_{opt}. *Then* Algorithm 2 *finds a* $(1-\epsilon)\delta_v(c_0, c_1)$*-safe trajectory going from some* $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*)$ *to some* $\mathbf{G}^* = (\mathbf{g}^*, \dot{\mathbf{g}}^*)$ *within* $O(\epsilon)$ *of* $\mathbf{S}$ *and* $\mathbf{G}$*, respectively, and taking time at most* $(1+\epsilon)T_{opt}$. *The approximation error at* $\mathbf{S}$ *and* $\mathbf{G}$ *is independent of* $\overline{v}$ *and* l.

The running time of the algorithm is

$$O\left(\tilde{c}_B^d N \left[\frac{\overline{v}l(c_1\overline{a} + \sqrt{c_0\overline{a}})^3}{\overline{a}^2 c_0^3 \epsilon^3}\right]^d\right),$$

where N *is the geometric complexity of the problem and* $\tilde{c}_B$ *is a constant of the algorithm.*

Thus, with obvious substitutions, the algorithm has an asymptotic time bound of $O(d\tilde{c}_B^d N(\frac{1}{\epsilon})^{3d})$, as in Table 1.5.1. Note that the the constant $\tilde{c}_B$ is used to simplify the expression that results from substituting in the timestep τ from equation (3.10) into the bound (3.16) on the number of reachability graph vertices and then factoring in the branching factor of graph.

The second algorithm improves on both the accuracy and the complexity.

Theorem 3.1.2 *Let* $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ *be an optimal kinodynamic planning problem for a point mass robot in d dimensions obeying* L_∞ *dynamics bounds. Let* $0 < \epsilon < 1$.

Suppose there is a $\delta_v(c_0, c_1)$*-safe trajectory from* $\mathbf{S}$ *to* $\mathbf{G}$ *taking time* T_{opt}. *Then* Algorithm 3 *finds a* $(1-\epsilon)\delta_v(c_0, c_1)$*-safe trajectory going from some* $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*)$ *to some* $\mathbf{G}^* = (\mathbf{g}^*, \dot{\mathbf{g}}^*)$ *within* $O(\epsilon)$ *of* $\mathbf{S}$ *and* $\mathbf{G}$, *respectively, and taking time at most* T_{opt}. *The approximation error at* $\mathbf{S}$ *and* $\mathbf{G}$ *is independent of* $\overline{v}$ *and* l.

The running time of the algorithm is

$$O\left(d\tilde{c}_C^d N \left[\frac{\overline{v}(\overline{a}c_1 + \overline{v})^3 l}{\overline{a}^2 c_0^3 \epsilon^3}\right]^d\right),$$

where N *is the geometric complexity of the problem and* $\tilde{c}_C$ *is a constant of the algorithm.*

Thus, with obvious substitutions, the algorithm has an asymptotic time bound of $O(d\tilde{c}_C^d N(\frac{1}{\epsilon})^{3d})$, as in Table 1.5.1. Note that Algorithm 3 will find a trajectory that takes at most as long as the optimal trajectory but that is not guaranteed to be as safe. $\tilde{c}_C^d$ is obtained similarly to $\tilde{c}_B^d$, but using equation (3.11) instead of (3.10).

Theorems 3.1.1 and 3.1.2 are similar in flavor to Theorem 2.1.1. They analytically relate the running times of the algorithms to guaranteed error bounds, and they state that the algorithms run in time polynomial in the geometric complexity N and in the resolution $(\frac{1}{\epsilon})$. The complexity bounds in both theorems are a significant improvement over the $O(N(\frac{1}{\epsilon})^{6d})$ result in Chapter 2. The $d\tilde{c}_B^d$ (or $d\tilde{c}_C^d$) factor in the complexity comes from removing the restriction that $d = 2$ or 3. Because the results identically extend to robots with decoupled dynamics and decoupled dynamics bounds when the c-space obstacles are unions of simple convex polytopes, we use N instead of n to denote geometric complexity, *i.e.*, N denotes the number of c-space obstacle surfaces.

Recall from Section 2.1.2 that an optimal kinodynamic planning problem $\mathcal{K}$ has three components: the combinatorial complexity of the geometry, the algebraic complexity of the geometry, and the algebraic complexity of the kinodynamic bounds. The two algorithms we present in this chapter are ϵ-approximation schemes fully polynomial in the combinatorial and algebraic complexity of the geometry, and pseudo-polynomial in the complexity of the kinodynamic bounds. Again, the approximately optimal solutions are not necessarily spatially near to the (true) optimal safe solution, except at the endpoints.

3.2 Kinodynamic Planning with L_∞ Dynamics Bounds

3.2.1 An Algorithm for Basic Kinodynamic Planning, Revisited

The basic idea behind our approach is again to reduce the problem of finding an approximately minimal-time trajectory to finding the shortest path in a directed graph. The vertices of the graph "discretize" the statespace TC, and the edges of the graph correspond to trajectory segments that each take time τ, which is a parameter calculated by the algorithm. As in the previous chapter, the algorithm will choose a root vertex state $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*)$ and use δ'_v-safe $(\overline{a}, \tau)$-bangs to induce a reachability graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Again, all the components of $\dot{\mathbf{s}}^*$ will be divisible by $\overline{a}\tau$, so that the vertices $v_i \in \mathcal{V}$ will lie at the interstices of an underlying grid in TC. (See section 2.2.1.)

The smaller τ is, the finer the underlying TC-grid, and the better some $(\overline{a}, \tau)$-grid-bang trajectory will approximate any arbitrary trajectory that starts at $\mathbf{S}^*$ and obeys the kinodynamic constraints. Thus, it is intuitively plausible that if τ is small enough and Γ_{opt} is an optimal trajectory from a state $\mathbf{S}$ sufficiently near $\mathbf{S}^*$ to a state $\mathbf{G}$, then there will be an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q going from $\mathbf{S}^*$ to a state $\mathbf{G}^*$ near $\mathbf{G}'$ in time

$$T_q \leq (1 + \epsilon) T_{opt}.$$

A naive algorithm might therefore do the following. First it would choose a time-step τ as a function of $\overline{a}$, $\overline{v}$, ϵ, c_0, and c_1. Then, it would choose a start state $\mathbf{S}^*$ that approximates $\mathbf{S}$, with the restriction that $\overline{a}\tau$ divides $\dot{s}^*$. Finally, it would search for the shortest path in the induced reachability graph to any of a set of vertices that approximate **G**, and return the trajectory corresponding to this path. The closeness of the approximations to **S** and **G** would improve as τ decreases.

This intuition is correct, and it matches Algorithm 1 from Section 2.2.2. Again, the main burden in proving correctness is to show how to choose an adequately small τ that induces a reachability graph only polynomially large in $\frac{1}{\epsilon}$. The disadvantage of the naive algorithm is that the proofs that lower its asymptotic complexity to $O(n(\frac{1}{\epsilon})^{3d})$ would only guarantee it to approximate the goal to $O(\overline{v}\tau)$ in position.

3.2.2 Improving Algorithm 1

Recall that edges in the reachability graph correspond to $(\overline{a}, \tau)$-bangs between vertices. A state in the image of one of these trajectory segments is called an *edge-state* if it is not a vertex. Since an $(\overline{a}, \tau)$-bang trajectory might come closest to the goal state at a time that is not a multiple of τ, we might expect that an algorithm that checks the closeness of edge-states to the goal will find a better approximation than the naive algorithm would find. (See figure 3.1.) This intuition is correct, and by considering edge-states as well as vertices, this simply modified algorithm can find a trajectory that approximates the ϵ-time-rescaled goal state $\mathbf{G}' = (\mathbf{g}, \frac{1}{1+\epsilon}\dot{\mathbf{g}})$ to within $O(\overline{a}\tau^2)$ in position and $O(\overline{a}\tau)$ in velocity.

We define a *graph trajectory* to be a trajectory that begins at the root $\mathbf{S}^*$ of the reachability graph $\mathcal{G}$ and is a sub-trajectory of some trajectory corresponding to a path in the graph. Thus, the algorithm looks for a minimal-time graph trajectory to an edge-state or vertex within the appropriate closeness of $\mathbf{G}'$.

If for some coordinate i, $\dot{s}_i$ is not a multiple of $\overline{a}\tau$, then it is hard to choose the root vertex $\mathbf{S}^*$ that approximates **S** and that is the best for finding an approximately optimal graph trajectory, without a cumbersome case analysis. We attempt to simplify the description by using the following trick: whenever some $\dot{s}_i$ is not a multiple of $\overline{a}\tau$, the algorithm chooses a root vertex $\mathbf{S}^*$ so that **S** will be approximated by $\Gamma_q(\tau)$, *for any graph trajectory* Γ_q *beginning at* $\mathbf{S}^*$. If Γ_q is the minimal-time graph trajectory to a state adequately close to $\mathbf{G}'$, the algorithm will then return the sub-trajectory of Γ_q that begins at time τ.

3.2.2.1 An Improved Algorithm for Basic Kinodynamic Planning

Recall that $\mathbf{S}' = (\mathbf{s}, \frac{1}{1+\epsilon}\dot{\mathbf{s}})$ and $\mathbf{G}' = (\mathbf{g}, \frac{1}{1+\epsilon}\dot{\mathbf{g}})$, where $\mathbf{S} = (\mathbf{s}, \dot{\mathbf{s}})$ and $\mathbf{G} = (\mathbf{g}, \dot{\mathbf{g}})$ are the start and goal states, respectively. **Algorithm 2** does the following:

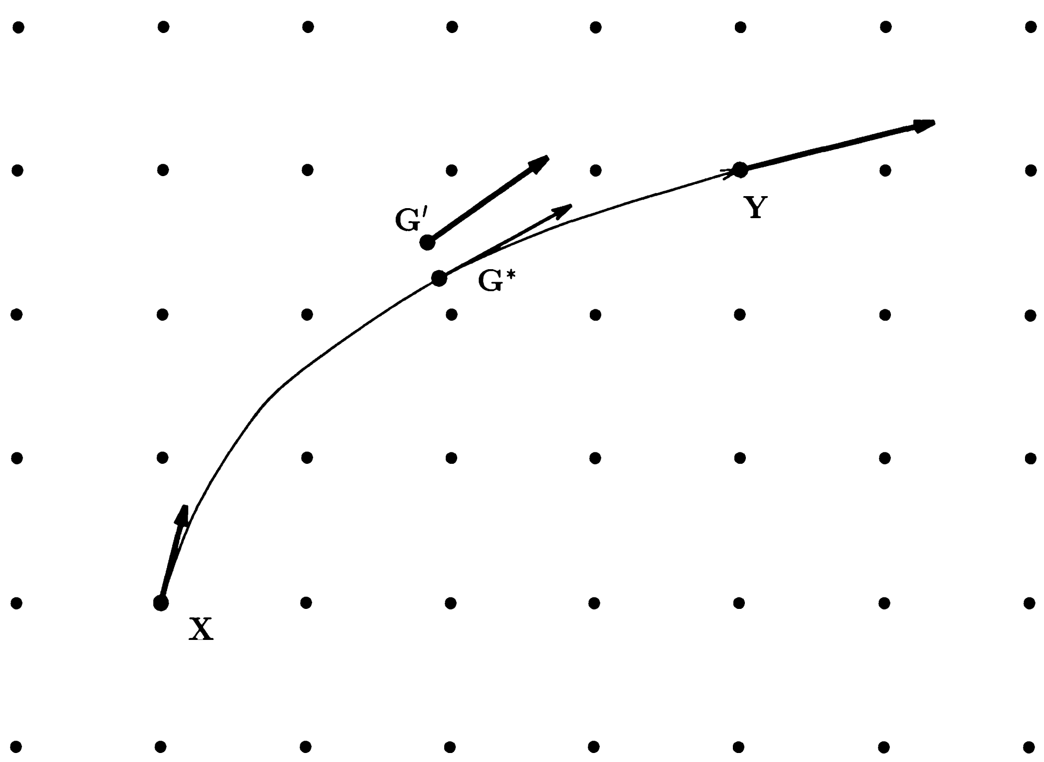


Figure 3.1: Edge-states can allow better approximation. State $\mathbf{G}^*$ on the trajectory of the $(\overline{a}, \tau)$-bang from $\mathbf{X}$ to $\mathbf{Y}$ lies much closer to $\mathbf{G}'$ than either $\mathbf{X}$ or $\mathbf{Y}$ do.

Algorithm 2

1. Chooses a timestep τ as a function of $\overline{a}$, $\overline{v}$, ϵ, c_0,and c_1. Specifically, the algorithm chooses the largest τ such that $\overline{a}\tau|\overline{v}$ and

$$\tau \leq \frac{8c_0\epsilon}{\overline{a}\left(7c_1(1-\epsilon)+\sqrt{49{c_1}^2(1-\epsilon)^2+\frac{48c_0(3+10\epsilon+5\epsilon^2)}{(2+\epsilon)\overline{a}}}\right)}.$$

2. Next, it chooses the starting TC-gridpoint $\mathbf{S}^*$ that serves as the root vertex of the reachability graph. For each i:

$$\begin{aligned} \dot{s}_i^*(0) &= \text{the multiple of } \overline{a}\tau \text{ closest to } \tfrac{1}{1+\epsilon}\dot{s}_i \\ s_i^*(0) &= s_i(0) - \tfrac{\tau}{2}(\dot{s}_i(0) + \dot{s}_i^*(0)). \end{aligned}$$

3. It then searches for a minimal-time graph trajectory from $\mathbf{S}^*$ to any state that is within $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$ of $\mathbf{G}'$. This search is basically a breadth-first search

of the induced reachability graph $\mathcal{G}$. In the n^{th} generation of the search the algorithm finds vertices n edges away from $\mathbf{S}^*$. However, as the algorithm explores an edge out of a vertex, it checks whether the corresponding $(\overline{a}, \tau)$-bang comes adequately close to $\mathbf{G}'$. If during a generation n more than one such $(\overline{a}, \tau)$-bang is are found, the algorithm chooses one that comes adequately close to $\mathbf{G}'$ at the earliest time. The algorithm uses backpointers to construct a minimal-time graph trajectory when its search succeeds.

4. Let Γ_q be the minimal-time graph trajectory found above. If $\mathbf{S}^* = \mathbf{S}'$, then the algorithm returns Γ_q; otherwise, the algorithm returns Γ_q^τ, defined by:

$$\Gamma_q^\tau(t) = \Gamma_q(t + \tau). \tag{3.1}$$

3.2.2.2 Analyzing the Algorithm

We claim that if an optimal δ_v-safe trajectory Γ_{opt} from $\mathbf{S}$ to $\mathbf{G}$ takes time T_{opt}, then Algorithm 2 will find a δ_v'-safe trajectory taking at most time $(1+\epsilon)T_{opt}$ and approximating $\mathbf{S}'$ and $\mathbf{G}'$ to within tolerances that are $O(\epsilon)$ and that do not grow with l or $\overline{v}$. This accuracy bound is similar to that in the previous chapter.

Applied in conjunction with Lemma 2.2.3 (the Safe Tracking Lemma), a key lemma (the Efficient Tracking Lemma, 3.3.1) shows how to choose a timestep τ that is $\Omega(\epsilon)$ such that the choice of $\mathbf{S}^*$ above assures that there will be a δ_v'-safe graph trajectory Γ_q taking time

$$T_q \leq (1+\epsilon)T_{opt}$$

and going from $\mathbf{S}^*$ to a state $\mathbf{G}^*$ approximating $\mathbf{G}'$ to the appropriate tolerance given in step 3 above. This shows that searching for the shortest graph trajectory will be adequate. The lemma also bounds $\|\Gamma_q(\tau) - \mathbf{S}'\|$ in the case when $\mathbf{S}^* \neq \mathbf{S}'$. Note that the corresponding lemma (the Simple Tracking Lemma, 2.2.6) in the previous chapter requires a timestep τ that is $O(\epsilon^2)$. Algorithm 2's choice of an $\Omega(\epsilon)$ timestep is responsible for the improvement in complexity.

As before, the overall complexity of the algorithm is the product of the size of the reachability graph $\mathcal{G}$ and the cost of checking whether an $(\overline{a}, \tau)$-bang (candidate graph edge) is δ_v'-safe. The maximal out-degree in $\mathcal{G}$ is 3^d, and the number of vertices is bounded by the number of TC-gridpoints, which is proportional to $\left(\frac{l\overline{v}}{\overline{a}^2\tau^3}\right)^d$.

Safety-checking using simple computational-geometric techniques described in Section 2.2.5 costs $O(N)$ per $(\overline{a}, \tau)$-bang for polyhedral obstacles. In d dimensions, for obstacles that are the unions of simple convex polytopes, safety-checking costs $O(dN)$ per $(\overline{a}, \tau)$-bang, with the d factor coming from the cost of computing dot-products. With minor modifications, the technique for checking safety with respect to a single obstacle is also used to check closeness to the goal. Absorbing constant factors from the timestep choice into a robot dependent constant c_B, we get the complexity bound $O(dc_B^d N(\frac{1}{\epsilon})^{3d})$.

3.2.2.3 Improving Accuracy Bounds

The accuracy bounds of the algorithm can be improved while maintaining the $(1-\epsilon)\delta_v$ safety bounds. With easy changes to the choice of root vertex and the search-termination condition, a modified algorithm can find a trajectory that approximates **S** and **G** to $(O(\overline{a}\tau^2), O(\overline{a}\tau))$ tolerances and that takes no more time than the optimal, exact solution.

The Strong Tracking Lemma (3.3.9) is crucial to this improvement in algorithm accuracy. Specifically, whereas Tracking Lemmas 2.2.6 and 3.3.1 show how to track an ϵ-time-rescaled trajectory, the Strong Tracking Lemma provides parameters that guarantee some graph trajectory can track an arbitrary trajectory to a desired tolerance, provided that the latter trajectory obeys the dynamics bounds obeys a simple start condition.

Very similar to Algorithm 2, **Algorithm 3** does the following

Algorithm 3

1. Chooses a timestep τ as a function of $\overline{a}$, $\overline{v}$, ϵ, c_0, and c_1. Specifically, the algorithm chooses the largest τ such that $\overline{a}\tau | \overline{v}$ and $\tau \leq \frac{c_0\epsilon}{2\overline{a}c_1(1-\epsilon)+5\overline{v}}$.

2. Next, it chooses the root vertex state $\mathbf{S}^*$ of the reachability graph. For each i:
$$\begin{aligned} \dot{s}_i^*(0) &= \text{the multiple of } \overline{a}\tau \text{ closest to } \dot{s}_i \\ s_i^*(0) &= s_i(0) - \frac{\tau^2}{2}(\dot{s}_i(0) + \dot{s}_i^*(0)). \end{aligned}$$

3. It then searches for a minimal-time graph trajectory from $\mathbf{S}^*$ to any state that is within $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$ of **G**. The search is a modified breadth-first search, just like step 3 in Algorithm 2. The algorithm uses backpointers to construct a minimal-time graph trajectory when its search succeeds.

4. Let Γ_q be the minimal-time graph trajectory found above. If $\mathbf{S}^* = \mathbf{S}$, then the algorithm returns Γ_q; otherwise, the algorithm returns Γ_q^τ, defined by:

$$\Gamma_q^\tau(t) = \Gamma_q(t+\tau). \tag{3.1}$$

3.3 Better Bounds for a Cartesian Robot with L_∞ Dynamics Bounds

3.3.1 Overview

We now outline the arguments proving Theorem 3.1.1 and Theorem 3.1.2. We will show that the parameters chosen by Algorithms 2 and 3 that guarantee the approximation closeness and the complexity bounds in the theorems.

The general idea of the proof of Theorem 3.1.1 similar to the proof of Theorem 2.1.1. The choice of timestep τ and root vertex $\mathbf{S}^*$ will guarantee that for every trajectory obtained by ϵ-time-rescaling a δ_v-safe trajectory beginning at start state $\mathbf{S}$, there will be a graph trajectory that tracks the rescaled trajectory closely enough to be $(1-\epsilon)\delta_v$-safe. In the proof of Theorem 3.1.2 there is no ϵ-time-rescaling: the choice of timestep and root vertex guarantees that for every δ_v-safe trajectory beginning at start state $\mathbf{S}$, there will be a graph trajectory that tracks it closely enough.

3.3.1.1 A Proof for Improved Complexity

To show Theorem 3.1.1, we will use the structure of the proof of Theorem 2.1.1 but substitute in a stronger tracking lemma. Suppose that Γ_r is a trajectory from $\mathbf{S}$ to $\mathbf{G}$ taking time T_r, and suppose furthermore that Γ_r obeys dynamics bounds $\overline{a}$ and $\overline{v}$ and is $\delta_v(c_0, c_1)$-safe. Let $0 < \epsilon < 1$.

We first recall that the Safe Tracking Lemma (2.2.3) yields a one-parameter family of *ϵ-safe-tracking tolerances* (η_x, η_v) such that if trajectory Γ_a tracks a $\delta_v(c_0, c_1)$-safe trajectory Γ_b to a tolerance in this family, then Γ_a will be $(1-\epsilon)\delta_v(c_0, c_1)$-safe. Again, letting Γ_r' denote Γ_r ϵ-time-rescaled, as in equation (2.9), if a trajectory Γ_q tracks Γ_r' to any ϵ-safe-tracking tolerance, then Γ_q will be $(1-\epsilon)\delta_v(c_0, c_1)$-safe.

We then show how to choose a timestep τ and a root vertex state $\mathbf{S}^*$ so that in the absence of obstacles, (a) for any graph trajectory Γ beginning at $\mathbf{S}^*$ and all $t \in [0, \tau]$, $\Gamma(t)$ is within (η_x, η_v) of $\mathbf{S}$, and (b) there will be some graph-trajectory Γ_q

such that Γ_q^τ tracks[1] Γ_r' to tolerance (η_x, η_v). (Lemma 3.3.1, the Efficient Tracking Lemma, from Section 3.3.2.) Γ_q and Γ_q^τ will therefore be $(1-\epsilon)\delta_v(c_0, c_1)$-safe even in the presence of the obstacles.

Finally, we show how to choose a safe tracking tolerance (η_x, η_v) so that there will be a timestep τ that both is $\Omega(\epsilon)$ and that satisfies the Efficient Tracking Lemma. It follows that the number of TC-gridpoints will be $O\left(c^d \left(\frac{1}{\epsilon}\right)^{3d}\right)$, for some robot-dependent constant c. Since each TC-graph vertex has a maximum out-degree of 3^d, and checking the $(1-\epsilon)\delta_v(c_0, c_1)$-safety of an $(\overline{a}, \tau)$-bang is $O(Nd)$, it follows that searching the TC-graph for a shortest path from $\mathbf{S}^*$ to $\mathbf{G}^*$ in the TC-graph takes time $O\left(dc^d 3^d N \left(\frac{1}{\epsilon}\right)^{3d}\right)$. Letting $c_B = 3c$, we get the complexity $O\left(dc_B^d \left(\frac{1}{\epsilon}\right)^{3d}\right)$. The lemmas and theorem (3.3.16) that follow, together with the terms in the algorithm, give us Theorem 3.1.1.

3.3.1.2 A Proof for Improved Accuracy

Although Theorem 3.1.2 states that if an exact optimal solution takes time T_{opt} then Algorithm 3 will find an ϵ-optimal solution that take at most time T_{opt} — as opposed to at most $(1+\epsilon)T_{opt}$ — our proof of the theorem is structurally simpler than our proof of Theorem 3.1.1. In particular, we show how to choose a timestep τ and a root vertex state $\mathbf{S}^*$ that guarantee some graph trajectory will track *an optimal trajectory, not merely an optimal trajectory that has been ϵ-time-rescaled,* closely enough to be $(1-\epsilon)\delta_v(c_0, c_1)$-safe and ϵ-close at the start and goal.

We again use the family of ϵ-safe-tracking tolerances (η_x, η_v) given by the Safe Tracking Lemma. However, Lemma 3.3.9 (the Strong Tracking Lemma) provides conditions for a timestep and a root vertex that guarantee (a) for any graph trajectory Γ beginning at $\mathbf{S}^*$ and all $t \in [0, \tau]$, $\Gamma(t)$ is within (η_x, η_v) of $\mathbf{S}$, and (b) there will be some graph-trajectory Γ_q such that Γ_q^τ tracks Γ_r to tolerance (η_x, η_v). To obtain the final bounds, we show how to select (η_x, η_v) that result in a timestep that is $\Omega(\epsilon)$. Except for details, the rest of the proof for Theorem 3.1.2 is the same as for Theorem 3.1.1.

[1]Recall that $\Gamma_q^\tau(t+\tau) = \Gamma_q(t)$ from equation (3.1).

3.3.2 The Efficient Tracking Lemma

3.3.2.1 Preliminary Discussion of the Lemma

Our tighter approximation and time-complexity bounds in Theorem 3.1.1 rely on following.

Lemma 3.3.1 (The Efficient Tracking Lemma) *Suppose a trajectory Γ_r respects acceleration bound $\frac{\overline{a}}{(1+\epsilon)^2}$ and velocity bound $\overline{v}$ and takes time T_r. Furthermore, suppose that $\overline{a}\tau$ divides $\overline{v}$. Then, in the absence of obstacles, the following hold.*

1. *For any positive η_x, η_v, there exists a timestep size τ and a choice of root vertex $\mathbf{S}^*$ such that*
   - *(a) for any graph trajectory Γ and all $t \in [0, \tau]$, $\Gamma(t)$ is within (η_x, η_v) of $\Gamma_r(0)$, and $\Gamma(\tau)$ is within $(\overline{a}\tau^2, \frac{3\overline{a}\tau}{2})$ of $\Gamma_r(0)$;*
   - *(b) there is a graph-trajectory Γ_q such that $\Gamma_q(t+\tau)$ is within tolerance (η_x, η_v) of $\Gamma_r(t)$ for all $t \in [0, T_r]$, and $\Gamma_q(T_r+\tau)$ is within $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$ of $\Gamma_r(T_r)$*
2. *Moreover, τ is only polynomial small in η_x, η_v, and $\frac{1}{\overline{a}}$. Specifically, τ can be chosen as the largest τ such that $\overline{a}\tau$ divides $\overline{v}$ and*

$$\tau \leq \min\left(\sqrt{\frac{4\epsilon(2+\epsilon)\eta_x}{3\overline{a}(3+10\epsilon+5\epsilon^2)}}, \frac{4\eta_v}{7\overline{a}}\right) \tag{3.2}$$

As in the proof of Lemma 2.2.6, it will be sufficient to consider the one-dimensional case to prove this lemma, since we are using the L_∞-norm for dynamics bounds. As in the previous chapter, let us call a function that describes the velocity of some grid-bang trajectory a *grid-bang velocity function.*

We consider an arbitrary trajectory Γ_r in the absence of obstacles that obeys the velocity bound $\overline{v}$ and the acceleration bound $\frac{\overline{a}}{(1+\epsilon)^2}$. We initially assume that $T_r = N\tau$ for some integer N, and that $\overline{v}$, $v_r(0)$ and $v_r(T_r)$ are multiples of $\overline{a}\tau$, where $v_r = \dot{p}_r$.

The proof has four stages. First, we show that there exists a pair of grid-bang velocity functions bounding v_r from above and below while staying within a constant of v_r. (See figure 3.2.) Second, we use these bounding trajectories to show there exists a grid-bang trajectory $\tilde{\Gamma}_q$ that tracks Γ_r to a tolerance that is a function of the dynamics bounds and τ. (See figure 3.4.) Third, we show that there exists a grid-bang trajectory Γ_q that tracks Γ_r only slightly less closely than $\widetilde{\Gamma}_q$ but that approximates Γ_r better at T_r. (See figure 3.5.) Finally, we relax our assumptions about Γ_r, except for the condition that $\overline{a}\tau$ divide $\overline{v}$. To relax the assumptions, we show how to prepend and append trajectory segments of length τ or less to Γ_r to reduce the proof to the restricted case. (See Figures 3.6 and 3.7.)

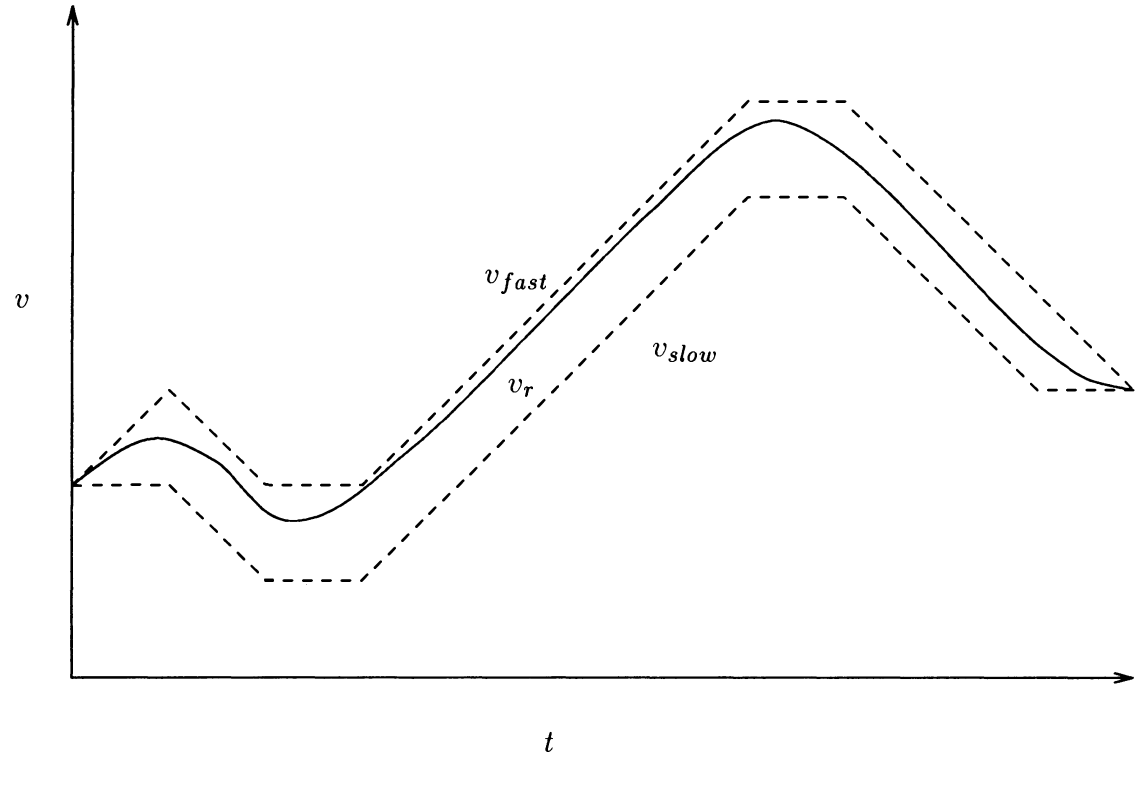


Figure 3.2: First step in proving the Efficient Tracking and Strong Tracking Lemmas (3.3.1 and 3.3.9). Assume that trajectory Γ_r obeys the velocity and acceleration bounds. In addition, for now assume that $T_r = N\tau$ for some integer N, and that $\overline{v}$, $v_r(0)$ and $v_r(T_r)$ are multiples of $\overline{a}\tau$. Then v_r is bounded above and below by $(\overline{a}, \tau)$-bang velocity functions v_{fast} and v_{slow}, which stay within $\frac{7\overline{a}\tau}{4}$ of v_r.

3.3.2.2 Bounding Velocity Functions

We now introduce a notation to describe grid-bang trajectories. Observe that a grid-bang trajectory Γ lasting for time $N\tau$ is uniquely described by the initial state $(\mathbf{p}(0), \dot{\mathbf{p}}(0))$ and a sequence of N vectors $\{\mathbf{c}^0, \ldots, \mathbf{c}^{N-1}\}$ over $\{-1, 0, 1\}^d$. Then we have for for $t \in [n\tau, (n+1)\tau)$:

$$\begin{aligned} \mathbf{v}(t) &= \mathbf{v}(n\tau) + (t - n\tau)\overline{a}\mathbf{I}\mathbf{c}^n \\ \mathbf{p}(t) &= \mathbf{p}(n\tau) + \textstyle\int_{n\tau}^{t} \mathbf{v}(s)ds, \end{aligned} \tag{3.3}$$

where $\mathbf{I}$ is the identity matrix[2].

The following lemma shows the existence of two "bounding" grid-bang velocity functions for every one-dimensional velocity function v_r that obeys bounds $\overline{v}$ and $\overline{a}$. (See figure 3.2.) For a velocity function $\mathbf{v}_r$ in d dimensions there will exist d such pairs of functions. In the construction of an $(\overline{a}, \tau)$-bang trajectory that approximates Γ_r, we will use these bounding functions to guarantee velocity-tracking.

Lemma 3.3.2 *Let τ be fixed such that $\overline{a}\tau | \overline{v}$. Let v_r be a velocity function obeying dynamics bounds $\overline{v}$ and $\overline{a}$, and let $\tau | T_r$. Suppose that $\overline{a}\tau$ divides $v_r(0)$ and $v_r(T_r)$.*

Then there exist two grid-bang velocity functions $v_{slow}, v_{fast} : [0, N\tau] \to [-\overline{v}, \overline{v}]$ that satisfy the following five conditions:

1. $v_{slow}(0) = v_r(0) = v_{fast}(0)$,
2. $v_{slow}(T_r) = v_r(T_r) = v_{fast}(T_r)$
3. $v_{slow}(t) \le v_r(t) \le v_{fast}(t)$ *for all* $t \in [0, T_r]$, *and*
4. $v_{fast}(t) - v_r(t) \le \frac{7}{4}\overline{a}\tau$ *and* $v_r(t) - v_{slow}(t) \le \frac{7}{4}\overline{a}\tau$ *for all* $t \in [0, T_r]$.
5. *for all* n, $v_{fast}(n\tau) - v_r(n\tau) \le \frac{3}{2}\overline{a}\tau$ *and* $v_r(n\tau) - v_{slow}(n\tau) \le \frac{3}{2}\overline{a}\tau$

Proof: We first define a grid-bang velocity function $\tilde{v}_r$ that approximates v_r. (See figure 3.3.) Let us define $\tilde{v}_r(n\tau)$ to be the integral multiple of $\overline{a}\tau$ that most closely approximates[3] $v_r(n\tau)$. We then inductively construct $\tilde{v}_r$:

- Initialization: $\tilde{v}_r(0) = v_r(0)$.

---

[2]If instead of L_∞ bounds $\overline{v}$ and $\overline{a}$, we have vectors of decoupled dynamics bounds, we replace $\overline{a}\mathbf{I}$ with the diagonal matrix of acceleration bounds $\mathbf{A}$, *i.e.*, $\mathbf{A}_{ii}$ is the acceleration bound in the i^{th} coordinate direction.

[3]In cases when $v_r(\tau) - \left\lfloor \frac{v_r(n\tau)}{\overline{a}\tau} \right\rfloor \overline{a}\tau = \frac{\overline{a}\tau}{2}$ the floor or ceiling must be chosen consistently.

- Stage n: for $t \in [n\tau, (n+1)\tau)]$,

$$\tilde{v}_r(t) = \tilde{v}_r(n\tau) + \frac{(\tilde{v}_r((n+1)\tau) - \tilde{v}_r(n\tau))(t - n\tau)}{\tau}. \tag{3.4}$$

Thus, for all n, $|v_r(n\tau) - \tilde{v}_r(n\tau)| \leq \frac{\overline{a}\tau}{2}$. It follows that $\tilde{v}_r$ also obeys acceleration bound $\overline{a}$, or else v_r would have to violate $\overline{a}$. Furthermore, because of the closeness at multiples of τ, $|v_r(t) - \tilde{v}_r(t)| \leq \frac{3\overline{a}\tau}{4}$ for all $t \in [0, T_r]$. It is obvious that $\tilde{v}_r$ obeys velocity bound $\overline{v}$.

We now define:

1. $\tilde{v}_{fast}(t) = \min(\overline{v}, \tilde{v}_r(t) + \overline{a}\tau)$.

2. $\tilde{v}_{slow}(t) = \max(-\overline{v}, \tilde{v}_r(t) - \overline{a}\tau)$.

It is straightforward to verify that these grid-bang velocity functions satisfy the last three claims of the lemma.

v_{fast} and v_{slow} are similar to $\tilde{v}_{fast}$ and $\tilde{v}_{slow}$. First, we set

$$v_{fast}(0) = v_{slow}(0) = \tilde{v}_r(0), \text{and}$$

$$v_{fast}(T_r) = v_{slow}(T_r) = \tilde{v}_r(T_r).$$

We then construct the functions forwards in time from $t = 0$ and backwards from $t = T_r$ until they match $\tilde{v}_{fast}(t)$. (See figure 3.3.) For the v_{fast} case at the $t = 0$ end, we set

$$\frac{d}{dt}(v_{fast}(t)) = \min(1, \frac{d}{dt}\tilde{v}_{fast}(t) + 1)$$

until the minimal time when $v_{fast}(t) = \tilde{v}_{fast}(t)$. If there is no such time, then $v_{fast} = \tilde{v}_r$. The other cases are similar. □

3.3.2.3 Tracking a Restricted Trajectory

Given a trajectory Γ_r that begins and ends with velocities that are multiples of $\overline{a}\tau$ and that takes a multiple of τ time, we show that there exists an $(\overline{a}, \tau)$-bang trajectory that matches it at the start, approximates it at the end, and tracks it to a tolerance that is independent of trajectory time. Lemma 3.3.3 establishes that the start state can be matched and that the tracking tolerance can be obeyed for an arbitrarily long trajectory. The trajectories v_{slow} and v_{fast} are used in the construction as "markers" to bound velocity tracking error and enforce exact matches at $t = 0$ and $t = T_r$. Lemma 3.3.4 states that with an increase in tracking error, an $(\overline{a}, \tau)$-bang trajectory can also approximate $\Gamma_r(T_r)$ to within $\overline{a}\tau^2$ in position.

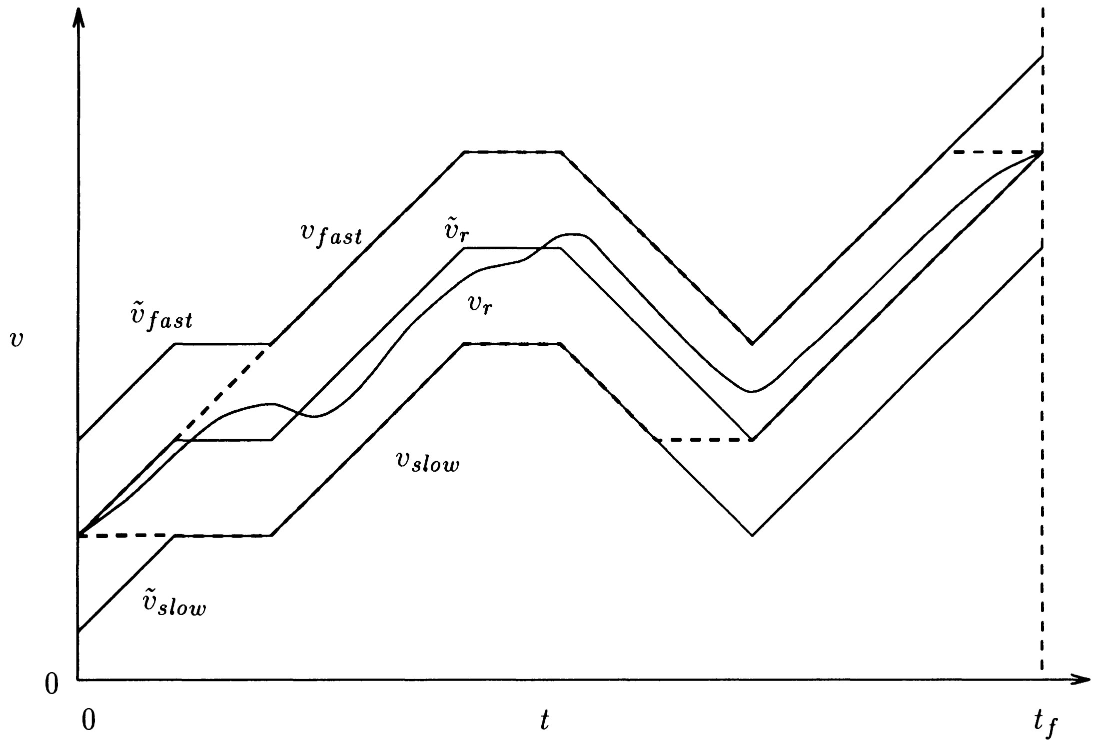


Figure 3.3: Constructing bounding velocity functions for Lemma 3.3.2. $\tilde{v}_r$ matches v_r at $t = 0, T_f$. v_{fast} and v_{slow} are visible as dashed lines where they differ from $\tilde{v}_{fast}$ and $\tilde{v}_{slow}$, respectively.

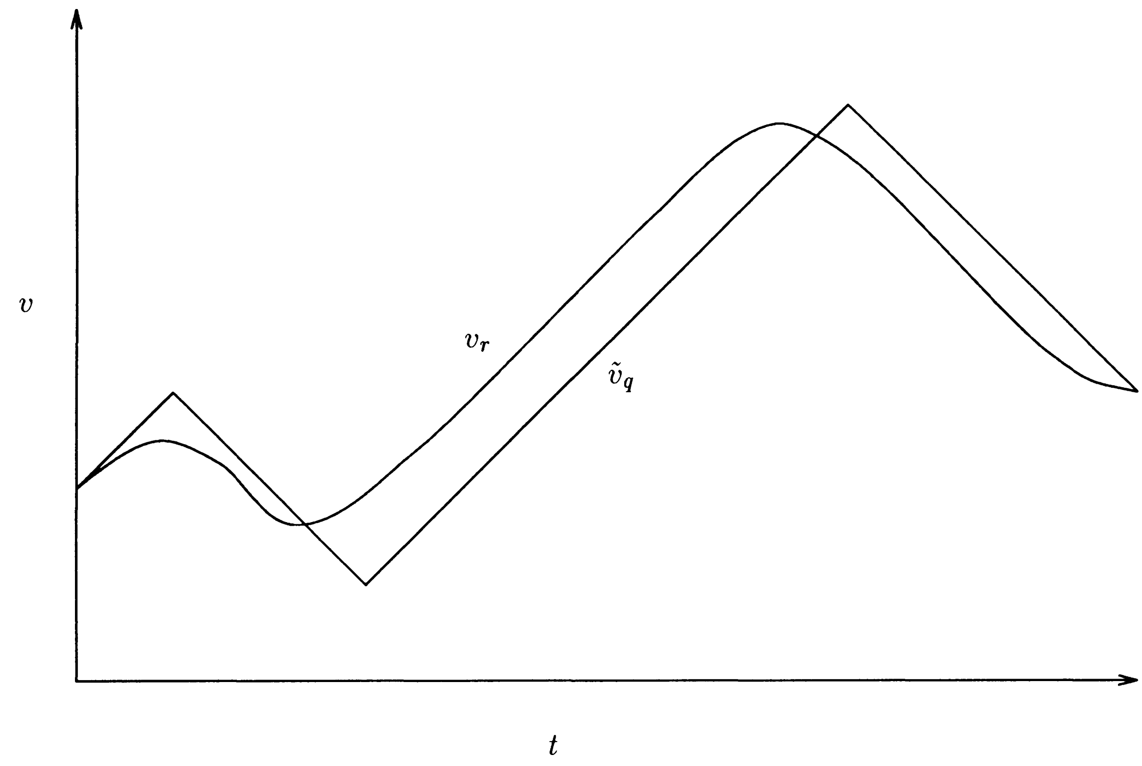


Figure 3.4: Second step in proving the Efficient Tracking and Strong Tracking Lemmas (3.3.1 and 3.3.9). We constructively show the existence of an $(\overline{a}, \tau)$-bang trajectory that tracks Γ_r. $\tilde{p}_q(0) = p_r(0)$, and $|p_r(t) - \tilde{p}_q(t)| < \frac{\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{8\epsilon(2+\epsilon)}$ in the proof of the Lemma 3.3.1. $\tilde{v}_q$ is bounded above and below by $(\overline{a}, \tau)$-bang velocity functions v_{fast} and v_{slow}, which stay within $\frac{7\overline{a}\tau}{4}$ of v_r.

Lemma 3.3.3 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$, and let $\epsilon > 0$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\frac{\overline{a}}{(1+\epsilon)^2}$, and let $\tau|T_r$. Suppose $\overline{a}\tau$ divides $v_r(0)$ and $v_r(T_r)$.*

Then there is an $(\overline{a},\tau)$-bang trajectory $\widetilde{\Gamma}_q$ such that for all times $t \in [0, T_r]$,

$$|v_r(t) - \tilde{v}_q(t)| \;\leq\; \frac{7\overline{a}\tau}{4}, \quad \textit{and} \tag{3.5}$$

$$|p_r(t) - \tilde{p}_q(t)| \;<\; \frac{\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{8\epsilon(2+\epsilon)} \tag{3.6}$$

Proof:

We inductively define a $(\overline{a},\tau)$-bang trajectory $\widetilde{\Gamma}_q$ that tracks Γ_r to the tolerance in the lemma. Our choice of $\widetilde{\Gamma}_q(0)$ serves as the root vertex of the reachability graph and determines the position-space alignment of the TC-grid coordinates. Let $\tilde{v}_r$, v_{fast} and v_{slow} be defined as in the previous lemma.

- Initially, set $\tilde{p}_q(0) = p_r(0)$ and $\tilde{v}_q(0) = \tilde{v}_r(0)$. The origin of the grid is implicitly chosen so that $(\tilde{p}_q(0), \tilde{v}_q(0))$ is a gridpoint.

- Stage n (≥ 0): We choose $c^n \in \{-1, 0, 1\}$, and set $\tilde{v}_q(t) = \tilde{v}_q(n\tau) + c^n\overline{a}(t - n\tau)$ for $t \in [n\tau, (n+1)\tau)$. c_n is simply chosen to minimize

  $$|p_r((n+1)\tau) - \tilde{p}_q((n+1)\tau)|$$

  subject to the condition that

  $$v_{slow}(t) \leq \tilde{v}_q(t) \leq v_{fast}(t) \quad \text{for all } t \in [n\tau, (n+1)\tau].$$

We claim that the trajectory $\widetilde{\Gamma}_q$ thus defined will meet the conditions of the lemma. $\tilde{v}_q$ obeys (3.6) by inspection. To see that $\tilde{p}_q$ obeys (3.6), we consider how large $|p_r(t) - \tilde{p}_q(t)|$ can grow.

First, we observe by inspection that if for integer k

$$\begin{aligned} |p_r(k\tau) - \tilde{p}_q(k\tau)| \;&\leq\; \tfrac{\overline{a}\tau^2}{2} \quad \text{and} \\ |p_r((k+1)\tau) - \tilde{p}_q((k+1)\tau)| \;&\leq\; \tfrac{\overline{a}\tau^2}{2}, \end{aligned}$$

then for all $t \in [k\tau, (k+1)\tau]$, $|p_r(t) - \tilde{p}_q(t)| \leq \overline{a}\tau^2$.

Now, suppose that $|p_r(k\tau) - \tilde{p}_q(k\tau)| \leq \frac{\overline{a}\tau^2}{2}$ but $|p_r((k+1)\tau) - \tilde{p}_q((k+1)\tau)| > \frac{\overline{a}\tau^2}{2}$. Because of the second rule, we conclude that

$$\begin{aligned} \operatorname{sgn}(p_r(k\tau) - \tilde{p}_q(k\tau)) \;&=\; \operatorname{sgn}(v_r(k\tau) - \tilde{v}_q(k\tau)) \\ &=\; \operatorname{sgn}(p_r((k+1)\tau) - \tilde{p}_q((k+1)\tau))\ . \\ &\neq\; 0 \end{aligned}$$

Furthermore, since Γ_r obeys the acceleration bound $\frac{\overline{a}}{(1+\epsilon)^2}$,

$$[\mathrm{sgn}(p_r((k+1)\tau) - \tilde{p}_q((k+1)\tau))](\tilde{a}_q(t) - a_r(t)) \geq \frac{\overline{a}\epsilon(2+\epsilon)}{(1+\epsilon)^2}$$

from time $k\tau$ until the least later T_w such that $\tilde{v}_q(T_w) = v_r(T_w)$, *i.e.*, the time when $|p_r(t) - \tilde{p}_q(t)|$ stops growing.

Since $|v_r(k\tau) - \tilde{v}_q(k\tau)| \leq \frac{3\overline{a}\tau}{2}$ and Γ_r obeys the acceleration bound $\frac{\overline{a}}{(1+\epsilon)^2}$, it follows that

$$T_w \leq \frac{3(1+\epsilon)^2\tau}{2\epsilon(2+\epsilon)},$$

and

$$|p_r(T_w) - \tilde{p}_q(T_w)| \leq \frac{\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{8\epsilon(2+\epsilon)}. \quad \square$$

Using the trajectory $\tilde{\Gamma}_q$ defined above, we show the existence an $(\overline{a},\tau)$-bang trajectory Γ_q that track the respective Γ_r a little less closely in position but that is within $(\overline{a}\tau^2, \overline{a}\tau)$ of Γ_r at time T_r. Specifically, we modify the velocity function $\tilde{v}_q$ to improve the approximation while staying between v_{fast} and v_{slow}. (See figure 3.5.)

Lemma 3.3.4 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\frac{\overline{a}}{(1+\epsilon)^2}$, and let $\tau|T_r$. Suppose $\overline{a}\tau|v_r(0), v_r(T_r)$.*

Then, there exists a grid-bang Γ_q that tracks Γ_r to tolerance

$$\left(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4}\right)$$

such that $\Gamma_q(0) = \Gamma_r(0)$, $v_q(T_r) = v_r(T_r)$, and $|p_q(T_r) - p_r(T_r)| < \overline{a}\tau^2$.

Proof: Given Γ_r obeying the hypothesis of the lemma, let us define v_{fast}, v_{slow}, and $\tilde{\Gamma}_q$ as defined in the proof of Lemma 3.3.3. We show by construction how to incrementally modify $\tilde{\Gamma}_q$ into a Γ_q that satisfies the lemma. Let $y = p_r(T_r) - \tilde{p}_q(T_r)$.

- Initialization: Let $\Gamma_q^{(0)} = \tilde{\Gamma}_q$. Go to stage 0.
- Stage $n \geq 0$: If $|p_r(T_r) - p_q^{(n)}(T_r)| < \overline{a}\tau^2$, then $\Gamma_q = \Gamma_q^{(n)}$. Otherwise, we obtain $\Gamma_q^{(n+1)}$ by modifying $v_q^{(n-1)}$. Let m be the least integer such that

$$\begin{aligned} \mathrm{sgn}(y)\,(v_r(m\tau) - \tilde{v}_q(m\tau)) &> 0 \\ \mathrm{sgn}(y)c^m &\geq 0, \quad \text{and} \\ \mathrm{sgn}(y)c^{m-1} &\leq 0. \end{aligned}$$

  Set $c^{m-1} = c^{m-1} + \mathrm{sgn}(y)$ and $c^m = c^m - \mathrm{sgn}(y)$. Proceed to the next stage.

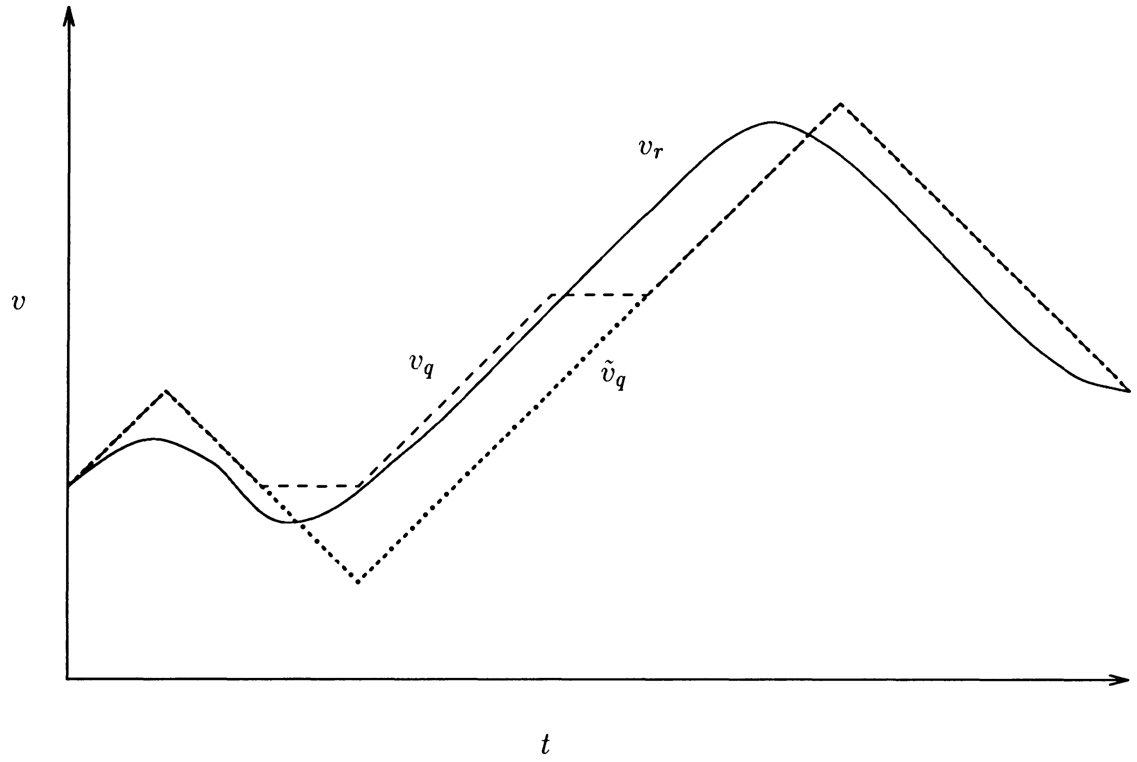


Figure 3.5: Third step in our proofs of Lemmas 3.3.1 and 3.3.9. $\tilde{v}_q$ (from Lemma 3.3.3 or 3.3.10) is modified to obtain v_q such that $\int_0^{T_r} v_q(t)dt$ is within $\overline{a}\tau^2$ of $\int_0^{T_r} v_r(t)dt$.

For $n > 0$ we verify that at stage n we can always find the necessary integer m. We describe the case where $y > 0$; the $y < 0$ case is similar. If $v_q^{(n)} = v_{fast}$ uniformly, then $p_q^{(n)}(T_r) \geq p_r(T_r)$. We show that as long as $v_q^{(n)}(t) < v_{fast}(t)$ at some time t, the stage can proceed. Suppose that we have followed the construction through $n-1$ stages, reach stage n, and find that $p_r(T_r) - p_q^{(n)}(T_r) \geq \overline{a}\tau^2$. Then, since $v_q^{(n)}(0) = v_{fast}(0)$, there is a least integer k such that $v_q^{(n)}(k\tau) < v_{fast}(k\tau)$. Either $c^{k-1} = 0$ or $c^{k-1} = -1$. Suppose that $c^{k-1} = -1$. Then there must be a least integer $m \geq k$ such that $c_m \geq 0$, since $v_q^{(n)}(k\tau) < v_{fast}(k\tau)$ but $v_q^{(n)}(T_r) = v_{fast}(T_r)$. Suppose that $c^{k-1} = 0$. If $c_k \geq 0$, then $m = k$. Otherwise, by the previous argument, there must be a least integer $m > k$ such that $c_m \geq 0$.

By inspection, we can see that for each n and all $t \in [0, T_r]$,

$$\begin{aligned} p_q^{(n)}(t) \leq p_q^{(n+1)}(t) &\leq p_q^{(n)}(t) + \overline{a}\tau^2, \quad \text{and} \\ p_q^{(n+1)}(T_r) &= p_q^{(n)}(T_r) + \overline{a}\tau^2. \end{aligned}$$

This assures that the construction reaches termination and the position-tracking bounds are achieved at T_r. The procedure guarantees that for $t \in [0, T_r]$,

$$|\tilde{p}_q(t) - p_q(t)| \leq \left\lceil \frac{9(1+\epsilon)^2}{8\epsilon(1+\epsilon)} \right\rceil \overline{a}\tau^2.$$

This and the properties of $\tilde{\Gamma}_q$ assure that Γ_q meets the position-tracking conditions of the lemma everywhere. The velocity-tracking bounds follow from the fact that $v_{slow}(t) \leq v_q(t) \leq v_{fast}(t)$ for all t. □

The following corollary is immediate.

Corollary 3.3.5 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\frac{\overline{a}}{(1+\epsilon)^2}$, and let $\tau|T_r$. Suppose that $\Gamma_r(T_r)$ is a TC-gridpoint relative to $\Gamma_r(0)$, τ, and the dynamics bounds.*

Then, there exists a grid-bang trajectory Γ_q that tracks Γ_r to tolerance

$$\left(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4}\right)$$

such that $\Gamma_q(0) = \Gamma_r(0)$ and $\Gamma_q(T_r) = \Gamma_r(T_r)$

Proof: Lemma 3.3.4 asserts all the claims except that $p_q(T_r) = p_r(T_r)$. The corollary follows from the property that $(\overline{a}, \tau)$-grid-bang trajectories with the same start states and final velocities will have final positions that differ by a multiple of $\overline{a}\tau^2$. □

3.3.2.4 Removing Restrictions

We will now remove certain restrictions on the Γ_r. (See figures 3.6 and 3.7.) First, we remove the restriction that T_r be a multiple of τ. The basic idea is that if we extend the trajectory and obtain a trajectory that takes $\left\lceil \frac{T_r}{\tau} \right\rceil \tau$ time, we can obtain a trajectory that satisfies the hypotheses of Lemma 3.3.11. The acceleration bound and the timestep then limit how much greater the tracking error is at time T_r than at $\left\lceil \frac{T_r}{\tau} \right\rceil \tau$.

We now present three lemmas relating to trajectories that obey dynamics bounds $\frac{\overline{a}}{(1+\epsilon)^2}$ and $\overline{v}$, followed by corresponding lemmas relative to trajectories obeying dynamics bounds $\overline{a}$ and $\overline{v}$.

Corollary 3.3.6 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\frac{\overline{a}}{(1+\epsilon)^2}$ such that $\overline{a}\tau|v_r(0)$ and $\overline{a}\tau|v_r(T_r)$.*

Then, there exists a grid-bang Γ_q that tracks Γ_r to tolerance

$$(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4})$$

such that $\Gamma_q(0) = \Gamma_r(0)$, $|v_q(T_r) - v_r(T_r)| < \overline{a}\tau$, and $|p_q(T_r) - p_r(T_r)| < \frac{3\overline{a}\tau^2}{2}$.

Proof: Suppose Γ_r is a trajectory obeying the hypotheses of the corollary. Define $T_r' = \left\lceil \frac{T_r}{\tau} \right\rceil \tau$. Then there is a trajectory Γ_r' such that:

$$\begin{aligned} \Gamma_r'(t) &= \Gamma_r(t) \quad \text{if } t \in [0, T_r], \\ v_r'(t) &= v_r(T_r) \quad \text{if } t \in [T_r, T_r']. \end{aligned}$$

By Lemma 3.3.4, there is an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q obeying $\overline{v}$ that tracks Γ_r' to tolerance $(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4})$ such that $\Gamma_q(0) = \Gamma_r'(0)$, $v_q(T_r') = v_r'(T_r')$, and $|p_q(T_r') - p_r'(T_r')| < \overline{a}\tau^2$. Since $\Gamma_r(T_r) = \Gamma_r'(T_r)$ and $v_r'(t) = v_r(T_r) = v_q(T_r')$ for all $t \in [T_r, T_r']$, and since v_q obeys acceleration bound $\overline{a}$, it follows that $|p_r(T_r) - p_q(T_r)| < \frac{3\overline{a}\tau^2}{2}$ and $|v_r(T_r) - v_q(T_r)| < \overline{a}\tau$. $\square$

Next, we remove the restriction that $\overline{a}\tau|v_r(T_r)$. Again, we show how to modify the "tail" of Γ_r to get a trajectory that satisfies the hypotheses of Lemma 3.3.4.

Corollary 3.3.7 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\frac{\overline{a}}{(1+\epsilon)^2}$ such that $\overline{a}\tau|v_r(0)$.*

Then, there exists a grid-bang Γ_q that tracks Γ_r to tolerance

$$(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4})$$

such that $\Gamma_q(0) = \Gamma_r(0)$, $|v_q(T_r) - v_r(T_r)| < \frac{7\overline{a}\tau}{4}$, *and* $|p_q(T_r) - p_r(T_r)| < \frac{5\overline{a}\tau^2}{2}$.

Proof: Suppose Γ_r is a trajectory obeying the hypotheses of the corollary. Define $T'_r = \left\lceil \frac{T_r}{\tau} \right\rceil \tau$, and $T''_r = T'_r - \tau$. Define v_0 to be the multiple of $\overline{a}\tau$ closest to $v_r(T_r)$ subject to the condition that $|v_r(T''_r) - v_0| \leq \overline{a}\tau$.

We then define a trajectory Γ'_r:

$$\begin{aligned} \Gamma'_r(t) &= \Gamma_r(t) && \text{if } \ t \in [0, T''_r], \ \text{and} , \\ v'_r(t) &= v'_r(T''_r) + \frac{(t-T''_r)}{\tau}(v_0 - v'_r(T''_r)) && \text{if } \ t \in [T''_r, T'_r]. \end{aligned}$$

Γ'_r obeys $\overline{a}$. It follows that $|p_r(T_r) - p'_r(T_r)| < \overline{a}\tau^2$, since $T''_r - T''_r = \tau$. By Lemma 3.3.4, there is an $(\overline{a}, \tau)$-grid-bang trajectory Γ_q obeying $\overline{v}$ that tracks Γ'_r to tolerance $(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4})$ such that $\Gamma_q(0) = \Gamma'_r(0)$, $v_q(T'_r) = v'_r(T'_r)$, and $|p_q(T'_r) - p'_r(T'_r)| < \overline{a}\tau^2$. It follows that $|p_q(t) - p'_r(t)| < \frac{5\overline{a}\tau^2}{2}$ for all $t \in [T''_r, T_r]$. Because $|v_q(T''_r) - v_r(T''_r)| \leq \frac{7\overline{a}\tau}{4}$ and $|v_q(T'_r) - v_r(T'_r)| \leq \overline{a}\tau$, it follows that $|v_q(t) - v_r(t) < \frac{7\overline{a}\tau}{4}$ for all $t \in (T''_r, T_r]$. □

Finally, we remove the restriction that $v_r(0)$ be a multiple of $\overline{a}\tau$. We modify Γ_r by prepending a trajectory segment that has an initial velocity that is a multiple of $\overline{a}\tau$, ends at $\Gamma_r(0)$, and takes time τ. (This construction "symmetrical" to the construction in the proof of Corollary 3.3.6. See Figure 3.7.) The modified trajectory satisfies the hypotheses of Corollary 3.3.7.

Corollary 3.3.8 *Let* τ *be fixed such that* $\overline{a}\tau | \overline{v}$. *Let* Γ_r *have velocity function* v_r *obeying dynamics bounds* $\overline{v}$ *and* $\frac{\overline{a}}{(1+\epsilon)^2}$.

Suppose that

$$\begin{aligned} \dot{s}^* &= \text{the multiple of } \overline{a}\tau \text{ closest to } \ v_r(0) \\ s^* &= p_r(0) - \tfrac{\tau}{2}(v_r(0) + \dot{s}^*). \end{aligned}$$

Then, there exists a grid-bang Γ_q *such that*

1. $\Gamma_q(0) = (s^*, \dot{s}^*)$;
2. $\Gamma_q(\tau)$ *approximates* $\Gamma_r(0)$ *to tolerance* $(\overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$;
3. *For all* $t \in [0, T_r]$, $\Gamma_q(t + \tau)$ *tracks* $\Gamma_r(t)$ *to tolerance* $(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4})$; *and*
4. $\Gamma_q(T_r + \tau)$ *approximates* $\Gamma_r(T_r)$ *to tolerance* $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$.

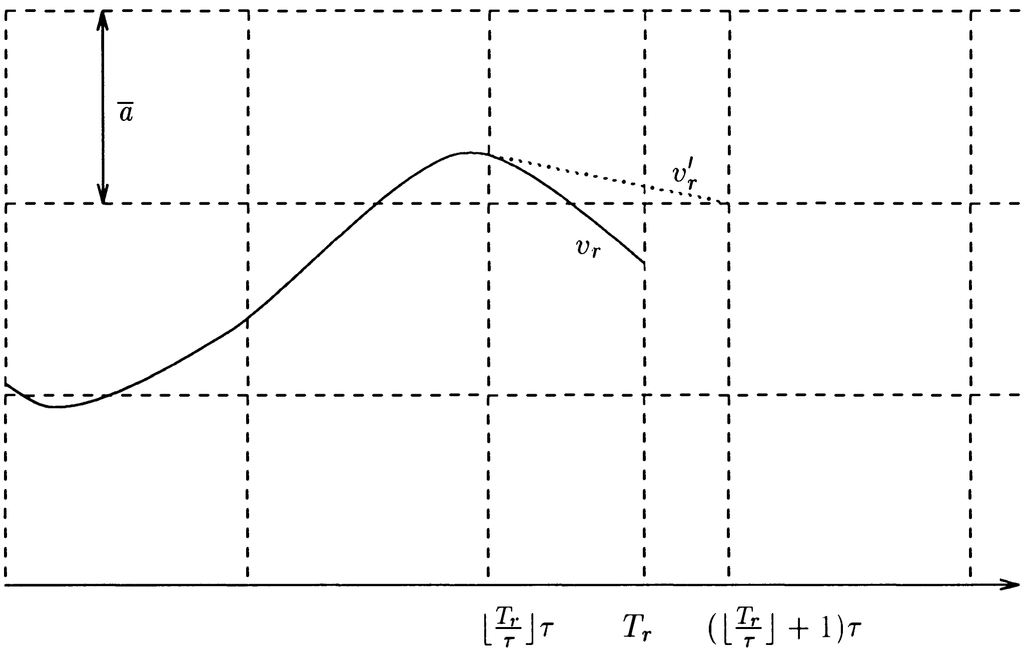


Figure 3.6: Proving Corollaries 3.3.5 through 3.3.14 and 3.3.5 through 3.3.14: if $\overline{a}\tau \not| v_r(T_r)$, then there is some $\overline{a}$-bounded v'_r such that $v'_r(t) = v_r(t)$ for $t \in [0, \left\lfloor \frac{T_r}{\tau} \right\rfloor]$ and $\overline{a}\tau$ divides $v'_r((\left\lfloor \frac{T_r}{\tau} \right\rfloor + 1)\tau)$. If $p_r(0) = p'_r(0)$, then a trajectory that tracks Γ'_r also tracks Γ_r almost as closely.

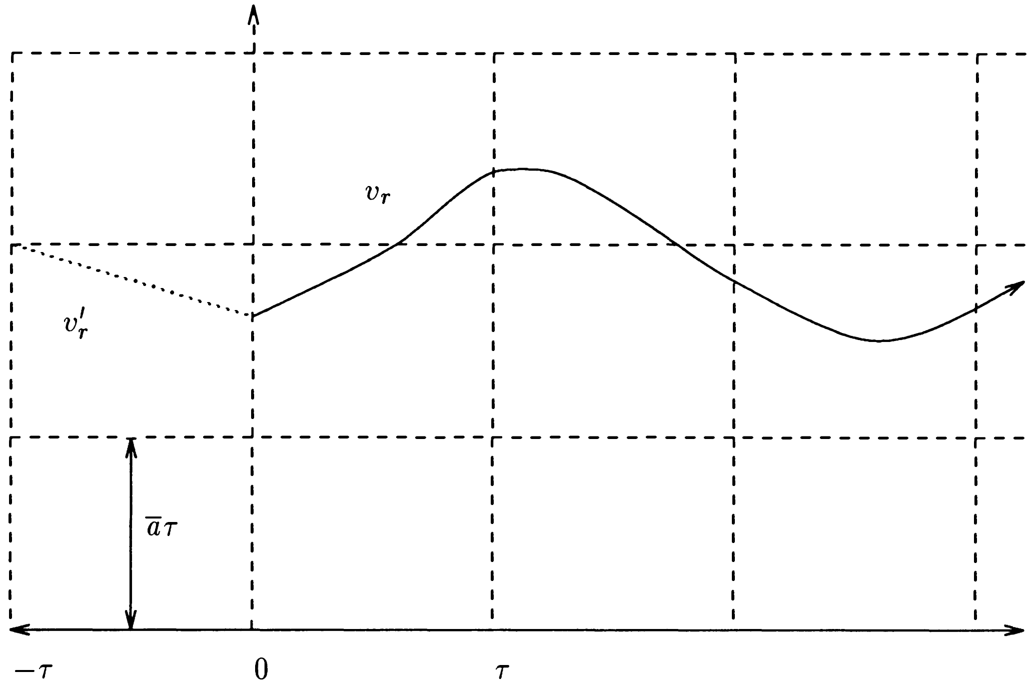


Figure 3.7: Prepending a trajectory segment in proving Lemma 3.3.8 or 3.3.15: if $\overline{a}\tau \nmid v_r(0)$, then there is some $\overline{a}$-bounded v'_r such that $v'_r(t+\tau) = v_r(0)$ for $t \in [0, T_r]$ and $\overline{a}\tau$ divides $v'_r(0)$. Here the time-scale for v'_r is offset by $-\tau$ to show this relationship.

Furthermore, $\Gamma_q(0)$ *can be calculated from* $\overline{a}$, $\overline{v}$, τ, *and* $\Gamma_r(0)$.

Proof: Suppose Γ_r is a trajectory obeying the hypotheses of the corollary, and suppose $v_r(0)$ is not a multiple of $\overline{a}\tau$. Let v_0 be the multiple of $\overline{a}\tau$ closest to $v_r(0)$. We can then define a trajectory Γ'_r such that $\Gamma'_r(\tau) = \Gamma_r(0)$:

$$\begin{array}{rcll} v_r(0) & = & v_0, & \\ v'_r(t) & = & v_0 + \frac{t}{\tau}(v_r(0) - v_0) & \text{if} \quad t \in [0, \tau], \\ p'_r(0) & = & p_r(0) - \int_0^\tau v'_r(t)dt, & \text{and,} \\ \Gamma'_r(t) & = & \Gamma_r(t - \tau) & \text{if} \quad t \in [\tau, T_r + \tau]. \end{array}$$

Clearly, $\Gamma'_r(0) = (s^*, \dot{s}^*)$ as defined in the lemma. By Corollary 3.3.7, there is an $(\overline{a}, \tau)$-bang trajectory Γ_q such that

1. Γ_q tracks Γ'_r to tolerance $(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4})$
2. $|p_q(T_r + \tau) - p'_r(T_r + \tau)| < \frac{5\overline{a}\tau^2}{2}$, and
3. $|v_q(T_r + \tau) - v'_r(T_r + \tau)| < \frac{7\overline{a}\tau}{4}$.

Since $\Gamma'_r(t) = \Gamma_r(t - \tau)$ for all $t \in [\tau, T_r + \tau]$, this establishes the last three conditions of the corollary. Since $\Gamma_q(0) = \Gamma'_r(0)$ and $\Gamma'_r(\tau) = \Gamma_r(0)$, the first two conditions follow from the $\overline{a}$ bound. □

3.3.2.5 Proving the Efficient Tracking Lemma

We now apply the lemmas and corollaries of the previous two sections to prove the Efficient Tracking Lemma (3.3.1).

Proof (of lemma 3.3.1): We apply lemma 3.3.4 and the corollaries that follow.

Specifically, suppose that trajectory Γ_r respects acceleration bound $\frac{\overline{a}}{(1+\epsilon)^2}$ and velocity bound $\overline{v}$ and takes time T_r, and that $\overline{a}\tau$ divides $\overline{v}$. Using Corollary 3.3.8, for any τ such that $\overline{a}\tau$ divides $\overline{v}$, we can compute the components $(s_i^*, \dot{s}_i^*)$ of a state $\mathbf{S}^*$ such that for some $(\overline{a}, \tau)$-bang trajectory Γ_q beginning at $\mathbf{S}^*$,

1. $\Gamma_q(\tau)$ approximates $\Gamma_p(0)$ to tolerance $(\overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$;
2. $\Gamma_q(t + \tau)$ tracks $\Gamma_r(t)$ to tolerance $(\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, \frac{7\overline{a}\tau}{4})$;
3. and $\Gamma_q(T_r + \tau)$ approximates $\Gamma_p(T_r)$ to tolerance $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$.

Therefore, if $\frac{3\overline{a}\tau^2}{2} + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)} \leq \eta_x$ and $2\overline{a}\tau \leq \eta_v$, then $\Gamma_q(t+\tau)$ tracks $\Gamma_r(t)$ to tolerance (η_x, η_v) over the interval $[0, T_r]$. Thus the choice of τ in equation (3.2) is sufficient. □

3.3.3 The Strong Tracking Lemma

The proof structure and specific techniques from Section 3.3.2 can be used to show the following lemma, which is essential to our proof of Theorem 3.1.2 using Algorithm 3.

Lemma 3.3.9 (The Strong Tracking Lemma) *Suppose trajectory Γ_r respects acceleration bound $\overline{a}$ and velocity bound $\overline{v}$ and takes time T_r. Furthermore, suppose that $\overline{a}\tau$ divides $\overline{v}$. Then in the absence of obstacles, the following hold.*

1. *For any positive η_x, η_v, there exists a timestep size τ and a choice of root vertex $\mathbf{S}^*$ such that*

   (a) *for any graph trajectory Γ and all $t \in [0, \tau]$, $\Gamma(t)$ is within (η_x, η_v) of $\Gamma_r(0)$, and $\Gamma(\tau)$ is within $(\overline{a}\tau^2, 2\overline{a}\tau)$ of $\Gamma_r(0)$;*

   (b) *there is a graph-trajectory Γ_q such that $\Gamma_q(t+\tau)$ is within tolerance (η_x, η_v) of $\Gamma_r(t)$ for all $t \in [0, T_r]$, and $\Gamma_q(T_r + \tau)$ is within $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$ of $\Gamma_r(T_r)$*

2. *Moreover, τ is only polynomially small in η_x, η_v, $\frac{1}{\overline{v}}$, and $\frac{1}{\overline{a}}$. Specifically, τ can be chosen as the largest τ such that $\overline{a}\tau$ divides $\overline{v}$ and*

$$\tau \leq \min\left(\frac{\eta_x}{5\overline{v}}, \frac{\eta_v}{2\overline{a}}\right) \tag{3.7}$$

As in the proof of the Efficient Tracking Lemma, it is sufficient examine the one-dimensional case. We consider an arbitrary trajectory Γ_r in the absence of obstacles, obeying dynamics bounds $\overline{v}$ and $\overline{a}$. We again initially assume that $T_r = N\tau$ for some integer N, and that $\overline{v}$, $v_r(0)$ and $v_r(T_r)$ are multiples of $\overline{a}\tau$.

Using the "bounding" velocity functions v_{fast} and v_{slow} from Section 3.3.2.2, we show there exists a grid-bang trajectory $\tilde{\Gamma}_q$ that tracks Γ_r to a tolerance dependent on $\overline{a}$, $\overline{v}$, and τ. We then show that there exists a grid-bang trajectory that is similar to Γ_q but approximates Γ_r better at T_r. Finally, we relax our assumptions about Γ_r, except for the condition that $\overline{a}\tau$ divide $\overline{v}$.

3.3.3.1 Tracking A Restricted, But Not ϵ-Time-Rescaled, Trajectory

The intuition behind the next lemma is similar that for Lemma 3.3.3: if at some time difference in position between the "reference" trajectory Γ_q and the "tracking" trajectory Γ_q is big enough and the difference in their velocities has the same sign, then we can bound the duration before the tracking trajectory catches up in velocity and thus begins to "catch up" in position. However, instead of using a difference in maximum accelerations to calculate this bound, we will use the amount of time a maximum acceleration can be maintained without violating the velocity bounds.

Lemma 3.3.10 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\overline{a}$, and let $\tau|T_r$. Suppose $\overline{a}\tau$ divides $v_r(0)$ and $v_r(T_r)$.*

Then there is an $(\overline{a},\tau)$-bang trajectory $\tilde{\Gamma}_q$ such that for all times $t \in [0, T_r]$,

$$\begin{aligned} |v_r(t) - \tilde{v}_q(t)| &\leq \frac{7\overline{a}\tau}{4}, \quad \textit{and} && (3.8) \\ |p_r(t) - \tilde{p}_q(t)| &< 2\overline{v}\tau + \overline{a}\tau^2. && (3.9) \end{aligned}$$

Proof:

We inductively define a $(\overline{a},\tau)$-bang trajectory $\tilde{\Gamma}_q$ that tracks Γ_r to the tolerance in the lemma. In stage n of the construction, we define $\tilde{v}_q(t)$ for all $t \in [n\tau, (n+1)\tau)$ and possibly alter a previously defined section of the function.[4] Our choice of $\tilde{\Gamma}_q(0)$ serves as the root vertex of the reachability graph and determines the position-space alignment of the TC-grid coordinates. Let $\tilde{v}_r$, v_{fast} and v_{slow} be defined as in Lemma 3.3.2.

- Initially, set $\tilde{p}_q(0) = p_r(0)$ and $\tilde{v}_q(0) = \tilde{v}_r(0)$. The origin of the grid is implicitly chosen so that $(\tilde{p}_q(0), \tilde{v}_q(0))$ is a gridpoint.

- Stage n (≥ 0): We choose $c^n \in \{-1, 0, 1\}$, and set $\tilde{v}_q(t) = \tilde{v}_q(n\tau) + c^n\overline{a}(t - n\tau)$ for $t \in [n\tau, (n+1)\tau)$. The following rules determine c^n.

  1. Initially, c^n is chosen to minimize $|p_r((n+1)\tau) - \tilde{p}_q((n+1)\tau)|$ subject to the condition $v_{slow}(t) \leq \tilde{v}_q(t) \leq v_{fast}(t)$, and subject to the previously chosen $c^0, \ldots, c^{n-1}$.

[4] The definition works for semi-infinite (*i.e.*,infinitely long) trajectories, and is in this way similar to simple finite injury constructions from recursion theory.

2. Let $y^{(n)} = p_r((n+1)\tau) - \tilde{p}_q((n+1)\tau)$. We call $y^{(n)}$ the *lag during stage* n. If $|y^{(n)}| > 2\overline{v}\tau$, then

   (a) for the greatest integer $m < n$ such that $c^m \neq \mathrm{sgn}(y^{(n)})$, set $c^m = c^m + \mathrm{sgn}(y^{(n)})$,

   (b) if there is a greatest integer $j, m < j \leq n$, such that setting $c^j = c^j + \mathrm{sgn}(y^{(n)})$ minimizes $|p_r((n+1)\tau) - \tilde{p}_q((n+1)\tau)|$, or if otherwise $\tilde{v}_q((n+1)\tau)$ would not be bounded by $v_{fast}((n+1)\tau)$ and $v_{slow}((n+1)\tau)$, then set $c^j = c^j - \mathrm{sgn}(y^{(n)})$.

By this definition, for all $t \in [0, T_r]$, $v_{slow}(t) \leq \tilde{v}_q(t) \leq v_{fast}(t)$, so $\tilde{v}_q(t)$ satisfies the first claim in the lemma.

Before we show $\tilde{v}_q$ satisfies the second claim, we observe that if we only use the first rule at each stage n, then

$$|p_r(t) - \tilde{p}_q(t)| < \frac{7\overline{v}\tau}{2} + \overline{a}\tau^2$$

for all $t \in [0, T_r]$. To see this, first note that a Γ_r can maintain maximum acceleration for at most time $\frac{2\overline{v}}{\overline{a}}$, since otherwise it would violate the velocity bounds. Therefore, the rule implies that if at time t_b

$$p_r(t_b) - \tilde{p}_q(t_b) \geq \overline{a}\tau^2,$$

then $\tilde{v}_q$ will equal or exceed v_r at some time no later than $t_b + \frac{2\overline{v}}{\overline{a}}$.

We now verify that the second rule for stage n of the construction is consistent. *Wlog*, we will assume that the rule is applied when the lag $y^{(n)}$ exceeds $2\overline{v}\tau$; the negative lag case is symmetrical.

Let

$$\widehat{\Gamma}_q^{(n)} = (\hat{p}_q^{(n)}, \hat{v}_q^{(n)})$$

be the trajectory just before the rule is applied. Since $y^{(n)} > 2\overline{v}\tau$, there is some minimal t_w such that

$$\hat{v}_q^{(n)}(t) \leq v_{fast}(t) \quad \text{for all } t \in [t_w, n\tau).$$

Otherwise, the first rule would have been violated at a previous stage. Furthermore, there is an integer m such that $m\tau \leq t_w < (n-1)\tau$, $c^m \leq 0$, and $c^{m+1}, ..., c^{n-1} = 1$. Thus, we can set $c^m = c^m + 1$.

Finally, we consider the effect of the rule at stage n of the construction, assuming $y^{(n)} > 2\overline{v}\tau$ and that $\tilde{p}_q^{(n)}(t)$ obeys (3.9) for all $t \in [0, n\tau]$. In particular, we must ensure that this does not cause the new $\tilde{p}_q^{(n)}(t)$ to become too different from $p_r(t)$ in the interval $[m\tau, (n+1)\tau]$. Let $\tilde{\Gamma}_q^{(n)}$ denote the trajectory at the end of stage n. First, for $t \in [n\tau, (n+1)\tau]$,

$$\frac{\overline{a}\tau^2}{2} \leq \tilde{p}_q^{(n)}(t) - \hat{p}_q^{(n)}(t) \leq 2\overline{v}\tau.$$

Now recall that for all integers k, $v_r(k\tau) - v_{slow}(k\tau) \leq \frac{3\overline{a}\tau}{2}$. Since $n\tau - m\tau \leq \frac{2\overline{v}}{\overline{a}}$ and $y^{(n)} > 2\overline{v}\tau$,

$$p_r(m\tau) - \tilde{p}_q^{(n)}(m\tau) \geq -\frac{\overline{v}\tau}{2}.$$

Since

$$\tilde{p}_q^{(n)}(t) - \hat{p}_q^{(n)}(t) \leq (t - m\tau)\overline{a}\tau$$

for $t \in [m\tau, (n+1)\tau]$, the rule therefore does not cause a violation of (3.9) for any $t \in [m\tau, n\tau]$ Thus $\tilde{p}_q^{(n+1)}(t)$ obeys (3.9) for all[5] $t \in [0, (n+1)\tau]$. □

3.3.3.2 Improving Approximation and Removing Restrictions

A lemma similar to Lemma 3.3.4 and a series of corollaries similar to Corollaries 3.3.5—3.3.5 can be shown with the most minor of changes to the corresponding proofs.

Lemma 3.3.11 *Let τ be fixed such that $\overline{a}\tau | \overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\overline{a}$, and let $\tau | T_r$. Suppose $\overline{a}\tau | v_r(0), v_r(T_r)$.*

Then, there exists a grid-bang Γ_q that tracks Γ_r to tolerance $(4\overline{v}\tau + \overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$ such that $\Gamma_q(0) = \Gamma_r(0)$, $v_q(T_r) = v_r(T_r)$, and $|p_q(T_r) - p_r(T_r)| < \overline{a}\tau^2$.

Proof: The construction is identical to that in the proof of Lemma 3.3.4, except that $|\tilde{p}_q(t) - p_q(t)| \leq 2\overline{v}\tau$. □

Corollary 3.3.12 *Let τ be fixed such that $\overline{a}\tau | \overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\overline{a}$, and let $\tau | T_r$. Suppose that $\Gamma_r(T_r)$ is a TC-gridpoint relative to $\Gamma_r(0)$, τ, and the dynamics bounds.*

Then, there exists a grid-bang trajectory Γ_q that tracks Γ_r to tolerance $(4\overline{v}\tau + \overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$ such that $\Gamma_q(0) = \Gamma_r(0)$ and $\Gamma_q(T_r) = \Gamma_r(T_r)$

[5]The $2\overline{v}\tau$ has been chosen conservatively to shorten an already long proof.

Proof: Same as for Corollary 3.3.5, except that Lemma 3.3.11 is applied instead of Lemma 3.3.4. $\square$

Corollary 3.3.13 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\overline{a}$ such that $\overline{a}\tau|v_r(0)$ and $\overline{a}\tau|v_r(T_r)$.*

Then, there exists a grid-bang Γ_q that tracks Γ_r to tolerance $(4\overline{v}\tau + \overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$ such that $\Gamma_q(0) = \Gamma_r(0)$, $|v_q(T_r) - v_r(T_r)| < \overline{a}\tau$, and $|p_q(T_r) - p_r(T_r)| < \frac{3\overline{a}\tau^2}{2}$.

Proof: Same as for Corollary 3.3.6, except that Lemma 3.3.11 is applied instead of Lemma 3.3.4. $\square$

Corollary 3.3.14 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$ and $\overline{a}$ such that $\overline{a}\tau|v_r(0)$.*

Then, there exists a grid-bang Γ_q that tracks Γ_r to tolerance $(4\overline{v}\tau + \overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$ such that $\Gamma_q(0) = \Gamma_r(0)$, $|v_q(T_r) - v_r(T_r)| < \frac{7\overline{a}\tau}{4}$, and $|p_q(T_r) - p_r(T_r)| < \frac{5\overline{a}\tau^2}{2}$.

Proof: Same as for Corollary 3.3.7, except we apply 3.3.11 instead of Lemma 3.3.4. $\square$

Corollary 3.3.15 *Let τ be fixed such that $\overline{a}\tau|\overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $\overline{v}$.*

Suppose that

$$\begin{aligned} \dot{s}^* &= \text{the multiple of } \overline{a}\tau \text{ closest to } \quad v_r(0) \\ s^* &= p_r(0) - \tfrac{\tau}{2}(v_r(0) + \dot{s}^*). \end{aligned}$$

Then, there exists a grid-bang Γ_q such that

1. *$\Gamma_q(0) = (s^*, \dot{s}^*)$;*
2. *$\Gamma_q(\tau)$ approximates $\Gamma_r(0)$ to tolerance $(\overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$;*
3. *For all $t \in [0, T_r]$, $\Gamma_q(t+\tau)$ tracks $\Gamma_r(t)$ to tolerance $(4\overline{v}\tau + \overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$; and*
4. *For all $t \in [0, T_r]$, $\Gamma_q(T_r+\tau)$ approximates $\Gamma_r(T_r)$ to tolerance $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$.*

Furthermore, $\Gamma_q(0)$ is computable from $\overline{a}$, $\overline{v}$, τ, and $\Gamma_r(0)$.

Proof: Same as for Corollary 3.3.8, but we use Corollary 3.3.14 instead of Corollary 3.3.7. $\square$

3.3.3.3 Proving the Strong Tracking Lemma

Proof (of lemma 3.3.9): We apply lemma 3.3.11 and the corollaries that follow.

Suppose that a trajectory Γ_r respects acceleration bound $\overline{a}$ and velocity bound $\overline{v}$ and takes time T_r, and that $\overline{a}\tau$ divides $\overline{v}$. By Corollary 3.3.15, for any τ such that $\overline{a}\tau$ divides $\overline{v}$, we can compute $\mathbf{S}^*$ such that for some $(\overline{a}, \tau)$-bang trajectory Γ_q beginning at $\mathbf{S}^*$

1. $\Gamma_q(\tau)$ approximates $\Gamma_p(0)$ to tolerance $(\overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$;
2. $\Gamma_q(t+\tau)$ tracks $\Gamma_r(t)$ to tolerance $(4\overline{v}\tau + \overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$;
3. and $\Gamma_q(T_r+\tau)$ approximates $\Gamma_p(T_r)$ to tolerance $(\frac{5\overline{a}\tau^2}{2}, \frac{7\overline{a}\tau}{4})$.

If $4\overline{v}\tau + \overline{a}\tau^2 \leq \eta_x$ and $2\overline{a}\tau \leq \eta_v$, then $\Gamma_q(t+\tau)$ tracks $\Gamma_r(t)$ to tolerance (η_x, η_v).
□

3.3.4 δ_v'-Safe ("Also Safe") Grid-Bang Trajectories and Deriving Final Bounds

3.3.4.1 Final Timestep Choices

Recall that the Efficient Tracking and Strong Tracking Lemmas (3.3.1 and 3.3.9) specify timesteps τ that are polynomially dependent on η_x and η_v. Applying these lemmas and Lemma 2.2.3 and choosing β to maximize the upper bound on τ yields the following theorems.

Theorem 3.3.16 *Given dynamics bounds $\overline{a}$ and $\overline{v}$, obstacles $\mathcal{O}$, and positive scalars $\epsilon \leq 1$, c_0, and c_1, let τ be chosen such that $\overline{a}\tau$ divides $\overline{v}$ and*

$$\tau \leq \frac{8c_0\epsilon}{\overline{a}\left(7c_1(1-\epsilon) + \sqrt{49{c_1}^2(1-\epsilon)^2 + \frac{48c_0(3+10\epsilon+5\epsilon^2)}{(2+\epsilon)\overline{a}}}\right)}. \tag{3.10}$$

Then for any start state $\mathbf{S}$ we can choose a root vertex state $\mathbf{S}^*$ such that if Γ_r begins at $\mathbf{S}$, is δ_v-safe, and takes time T_r, there exists a δ_v'-safe trajectory Γ_q with the following properties:

1. $\Gamma_q(0) = \mathbf{S}^*$.
2. *$\Gamma_q(\tau)$ approximates $\Gamma_r(0)$ to within $(\overline{a}\tau^2, 2\overline{a}\tau)$, and*

3. $\Gamma_q(T_r+\tau)$ *approximates* $\Gamma_r(T_r)$ *to within* $(\frac{5\overline{a}\tau^2}{2}, 2\overline{a}\tau)$.

Theorem 3.3.17 *Given dynamics bounds* $\overline{a}$ *and* $\overline{v}$, *obstacles* $\mathcal{O}$, *and positive scalars* $\epsilon \leq 1$, c_0, *and* c_1, *let* τ *be chosen such that* $\overline{a}\tau$ *divides* $\overline{v}$ *and*

$$\tau \leq \frac{c_0\epsilon}{2\overline{a}c_1(1-\epsilon)+5\overline{v}}. \tag{3.11}$$

Then for any start state $\mathbf{S}$ *we can choose a root vertex state* $\mathbf{S}^*$ *such that if* Γ_r *begins at* $\mathbf{S}$, *is* δ_v*-safe, and takes time* T_r, *there exists a* δ_v'*-safe trajectory* Γ_q *with the following properties:*

1. $\Gamma_q(0) = \mathbf{S}^*$.
2. $\Gamma_q(\tau)$ *approximates* $\Gamma_r(0)$ *to within* $(\overline{a}\tau^2, 2\overline{a}\tau)$, *and*
3. $\Gamma_q(T_r+\tau)$ *approximates* $\Gamma_r(T_r)$ *to within* $(\frac{5\overline{a}\tau^2}{2}, 2\overline{a}\tau)$.

Details of the proof of Theorem 3.3.17 are simpler than those in the proof of Theorem 3.3.16. Therefore, we will prove Theorem 3.3.17 first.

Proof (of Theorem 3.3.17): Suppose Γ_r is a $\delta_v(c_0, c_1)$-safe trajectory taking time T_r obeying acceleration bound $\overline{a}$. Then by lemma 2.2.3 the choice of a tolerance (η_x, η_v) given in (2.10) ensures that if a trajectory Γ_q approximately tracks Γ_r to tolerance (η_x, η_v), then the δ_v'-tube induced by Γ_q lies entirely inside the δ_v-tube induced by Γ_r. Since the δ_v-tube induced by Γ_r intersects no obstacles in $\mathcal{O}$, Γ_q is therefore δ_v'-safe. Given the tolerance (η_x, η_v), we use the Strong Tracking Lemma to choose the time-spacing τ and the root vertex $\mathbf{S}^*$ (via Corollary 3.3.15.)

To get the desired bounds, we must choose β so that using (3.11) yields a maximal τ as given by (3.7). Let us therefore define for $\beta \geq 0$

$$\begin{aligned} \tau_x(\beta) &= \frac{c_0\epsilon\beta}{5\overline{v}(c_1(1-\epsilon)+\beta)} \\ \tau_v(\beta) &= \frac{c_0\epsilon}{2\overline{a}(c_1(1-\epsilon)+\beta)} \\ \tau(\beta) &= \min(\tau_x(\beta), \tau_v(\beta)). \end{aligned} \tag{3.12}$$

By inspection, $\tau_x(0) < \tau_v(0)$, τ_x is monotonically increasing, and τ_v is monotonically decreasing. Thus, $\tau(\beta)$ is maximized when $\tau_x(\beta) = \tau_v(\beta)$. Requiring β to be positive and doing a little computation, we find that $\tau(\beta)$ is maximized when

$$\beta = \frac{2\overline{v}}{5\overline{a}}. \tag{3.13}$$

Applying either τ_x or τ_v to this β yields the desired τ in (3.11). Finally, we see that the choice of τ and the closeness of $\Gamma_q(\tau)$ to $\mathbf{S}$ guarantee that the trajectory segment from the root vertex to $\Gamma_q(\tau)$ will be δ'_v-safe. $\square$

To prove Theorem 3.3.16, we show how to choose a root vertex $\mathbf{S}^*$ and a timestep τ so that some graph trajectory will track *an ϵ-time-rescaled* $\delta_v(c_0, c_1)$-safe trajectory to an ϵ-safe-tracking tolerance.

Proof (of Theorem 3.3.16): Suppose Γ_r is a $\delta_v(c_0, c_1)$-safe trajectory taking time T_r obeying acceleration bound $\overline{a}$. Then the ϵ-time-rescaled trajectory Γ'_r given by

$$\begin{aligned} \mathbf{p}'_r(t) &= \mathbf{p}_r\left(\tfrac{t}{1+\epsilon}\right); \\ \dot{\mathbf{p}}'_r(t) &= \tfrac{1}{1+\epsilon}\dot{\mathbf{p}}_r\left(\tfrac{t}{1+\epsilon}\right). \end{aligned}$$

obeys velocity bound $\frac{\overline{v}}{1+\epsilon}$ and acceleration bound $\frac{\overline{a}}{(1+\epsilon)^2}$. Furthermore, Γ'_r is $\delta_v(c_0, c_1)$-safe and takes time $(1+\epsilon)T_r$.

By definition, if a trajectory Γ_q approximately tracks Γ'_r to an ϵ-safe-tracking tolerance (η_x, η_v), then Γ_q is δ'_v-safe. Given the a one-parameter family of ϵ-safe-tracking tolerances (η_x, η_v) from Lemma 2.2.3 (2.10), we use the Efficient Tracking Lemma (3.3.1) to choose the timestep τ and the root vertex $\mathbf{S}^*$.

To get the desired bounds, we must choose β to maximize τ as given by (3.2). Define for $\beta \geq 0$:

$$\begin{aligned} \tau_x(\beta) &= \sqrt{\tfrac{4\epsilon(2+\epsilon)(c_0\epsilon\beta)}{3\overline{a}(3+10\epsilon+5\epsilon^2)(c_1(1-\epsilon)+\beta)}} \\ \tau_v(\beta) &= \tfrac{4c_0\epsilon}{7\overline{a}(c_1(1-\epsilon)+\beta)} \\ \tau(\beta) &= \min(\tau_x(\beta), \tau_v(\beta)). \end{aligned} \tag{3.14}$$

By inspection, $\tau_x(0) < \tau_v(0)$, τ_x is monotonically increasing, and τ_v is monotonically decreasing. Thus, $\tau(\beta)$ is maximized when $\tau_x(\beta) = \tau_v(\beta)$. Consulting *Mathematica* for an analytical solution, we find the maximum value of $\tau(\beta)$ is given by (3.10).

Finally, we see that the choice of τ and the closeness of $\Gamma_q(\tau)$ to $\mathbf{S}'$ guarantee that the trajectory segment from the root vertex to $\Gamma_q(\tau)$ will be δ'_v-safe. $\square$

3.3.4.2 Approximation Goodness and Overall Complexity Bounds

As the choice of timestep and root vertex in Algorithm 2 agrees with Theorem 3.3.16 and Corollary 3.3.8, we have shown the approximation bounds in Theorem

3.1.1. The same is true for Algorithm 3, Theorem 3.3.17, Corollary 3.3.15, and Theorem 3.1.2.

To determine the total complexity of the algorithms we must bound the number of $(\overline{a}, \tau)$-gridpoints for a Cartesian robot with maximum (L_∞) speed $\overline{v}$ in a d-dimensional free-space of diameter l. As the results in this Chapter are used in the implementation discussed in Section 5.2, the discussion will be more detailed than in Chapter 2.

First we consider the $d = 1$ case. Recall from section 2.2.6 that $G_\infty(\overline{a}, \tau, \overline{v}, l, d)$ denotes the maximum number of TC-gridpoints for a point robot with maximum (L_∞) speed $\overline{v}$ and acceleration $\overline{a}$ in a d-dimensional free-space of diameter l. It is clear that $G_\infty(a, \tau, \overline{v}, l, 1)$ is equal to the maximum number of possible velocities at any given time $k\tau$ multiplied by the maximum number of possible positions at that time. Since at each timestep the change in velocity is $\overline{a}\tau$, $-\overline{a}\tau$, or 0, the number of possible velocities is at most $\frac{2\overline{v}}{\overline{a}\tau} + 1$. To see that the number of possible positions at a given velocity is at most $\frac{l}{\overline{a}\tau^2} + 1$, let $v^{(i)}$ denote the velocity and $x^{(i)}$ the position at timestep i for all i. Then $x^{(k+1)} = v^{(k)}\tau + \sigma(k)\frac{\overline{a}\tau^2}{2}$, where $\sigma(k) \in \{-1, 0, 1\}$. *Wlog*, let $v^{(0)} = 0$ and $x^{(0)} = 0$. Since $v^{(k)} = c^{(k)}\overline{a}\tau$ for some integer $c^{(k)}$, by using induction we can show that

$$\begin{aligned} x^{(k)} &= \frac{(2\Upsilon^{(k)}+1)\overline{a}\tau^2}{2} \quad \text{if } c^{(k)} \text{ odd} \\ x^{(k)} &= \frac{2\Upsilon^{(k)}\overline{a}\tau^2}{2} \quad \text{if } c^{(k)} \text{ even,} \end{aligned} \tag{3.15}$$

for some integer $\Upsilon^{(k)}$. It follows that

$$G_\infty(\overline{a}, \tau, \overline{v}, l, d) = \left(\left(\frac{2\overline{v}}{\overline{a}\tau} + 1\right)\left(\frac{l}{\overline{a}\tau^2} + 1\right)\right)^d \tag{3.16}$$

Hence, in a bounded workspace with velocity limits, we can use a polynomial-sized reachability graph to find an approximate optimal safe solution. Given a problem for which δ_v-safe solution exists, the algorithm finds a trajectory Γ_q that satisfies the time approximation $T_q \leq T$, in addition to respecting the kinodynamic constraints and being δ'_v-safe. Noting that the maximum out-degree of the reachability graph is 3^d and that checking the safety of trajectory segments with respect to c-space obstacles that are (the union of) simple convex polytopes requires time $O(Nd)$, we substitute the algorithm's choice of τ, (3.10) or (3.11), into (3.16) to obtain the complexity bounds in Theorem 3.1.1 and Theorem 3.1.2.

3.4 An Implementation for the L_∞ Cartesian Robot

3.4.1 Brief Description

To get some idea for how our algorithms for Cartesian robots obeying L_∞ dynamics bounds might behave in practice, we have completed a prototype *Common Lisp* implementation for a point robot in two dimensions. This planner, which we have run on a Sun SparcStation2, is the first implementation of any algorithm that generates provably good, provably near-optimal kinodynamic plans for problems in more than one dimension.[6]

Our planner is a simple implementation and uses no search-pruning heuristics. Depending on how several parameters are set, it executes a version of Algorithm 2 or Algorithm 3; it can also be used to execute a version of the L_∞ Cartesian Robot algorithm described in Chapter 5. The planner basically does a breadth-first graph search of the reachability graph, computing the neighbors of a state (vertex) only when it is on the search frontier. A bit array is used to record whether a state has been reached, and each vertex found keeps a pointer to its "parent". The planner enforces δ'_v-safety, so each edge is a δ'_v-safe $(\overline{a}, \tau)$-bang.

Although the planner can be easily modified to execute the theoretical algorithms exactly, the present version uses simplified methods to choose the root vertex and and to check closeness to the goal. In addition, an inexpensive "discretizing" preprocessing step[7] allows the implementation to reduce its use of the safety-checking procedure from Section 2.2.5 (implemented using floating-point operations, however) to the obstacles "relevant" to each trajectory segment.

The implementation retains provable ϵ-approximation properties that can be shown by extending techniques presented in this thesis. The principal difference is that for each example, if T_{opt} is the optimal kinodynamic trajectory time for the problem instance (assuming it is solvable) and $\tau = O(\epsilon)$ is the timestep chosen by the algorithm, the implementation will first find a trajectory taking time at most $(1+\epsilon)T_{opt} + \tau$ time (or $T_{opt} + \tau$ time for Algorithm 3), and then proceed to search for a trajectory that comes within a "grid-spacing" of the ϵ-time-rescaled goal state at a multiple of τ. For all the runs we tried, this took at most three

[6]Actually, it's the second incarnation of our first implementation.

[7]For all the runs done to this day, the preprocessing took at most a few seconds. Details are rudimentary in principle and beyond the scope of this thesis.

more time steps. Thus, the trajectories shown take time at most $(1+\epsilon)T_{opt}+4\tau$ (or $T_{opt}+4\tau$).

3.4.2 Simple Examples

An example of a solution found by the planner running Algorithm 2 is shown in Figure 3.8. The start state, at left, has an upward velocity of 0.1, and the goal state has a downward velocity of 0.1. The small dots indicate the position component of the planned trajectory at each time step. The "lines" or "spikes" attached to each dot indicate the velocity at that position. In this problem, the world is 3.5 by 1.9 units. For this run $\epsilon = 0.6$, $c_0 = 0.2$, $c_1 = 0.5$, $\overline{a} = 0.2$, and $\overline{v} = 0.3$. The planner run took 4 minutes and 32 seconds of run time (5 m. 31 sec. total elapsed time) on a 32Mbyte SparcStation2 to find the trajectory in the figure.[8]

Two examples of solutions found by our planner running Algorithm 3 are shown in fig. 3.9. Although the asymptotic complexity of this algorithm in the resolution $\frac{1}{\epsilon}$ and the geometry N the problem is the same as that of Algorithm 2, comparing (3.11) to (3.10) reveals that it is more sensitive to the kinodynamic bounds ($\overline{a}$, $\overline{v}$, c_0, and c_1). In this problem instance, the world is 3.5 by 1.9 units; $\overline{v} = 0.12$; $\overline{a} = 0.1$; $c_0 = 0.31$; $c_1 = 0.1$; and $\epsilon = 0.8$. Note that the larger c_0 is compared to the maximum velocity, the larger the ϵ-safe tracking tolerance, and hence the easier the problem. The start position is at lower left, and the start velocity is in 0.08 the positive y direction; the goal, at the upper right, has 0.08 velocity in the negative y direction. τ was chosen as 0.4, resulting in a TC-grid of approximately 800,000 states. The implementation searched over 740,000 states and required approximately 50 minutes of run time[9] on a 48Mbyte SparcStation2 to generate this solution.

3.4.3 Discussion

As we stated above, we implemented the algorithms to see how they might behave in practice. Because there were no prior provably good approximation algorithms for 2D optimal kinodynamic planning, *ad hoc* methods previously had to be used

[8] "Run time" is the time the process is actually active, *i.e.*, computer time. "Total elapsed time" is how much real time passes on a wall clock; it thus includes time the process is idle, *e.g.*, swapping.

[9] Total elapsed time averaged about three hours, but this varied greatly, since the machine was shared.

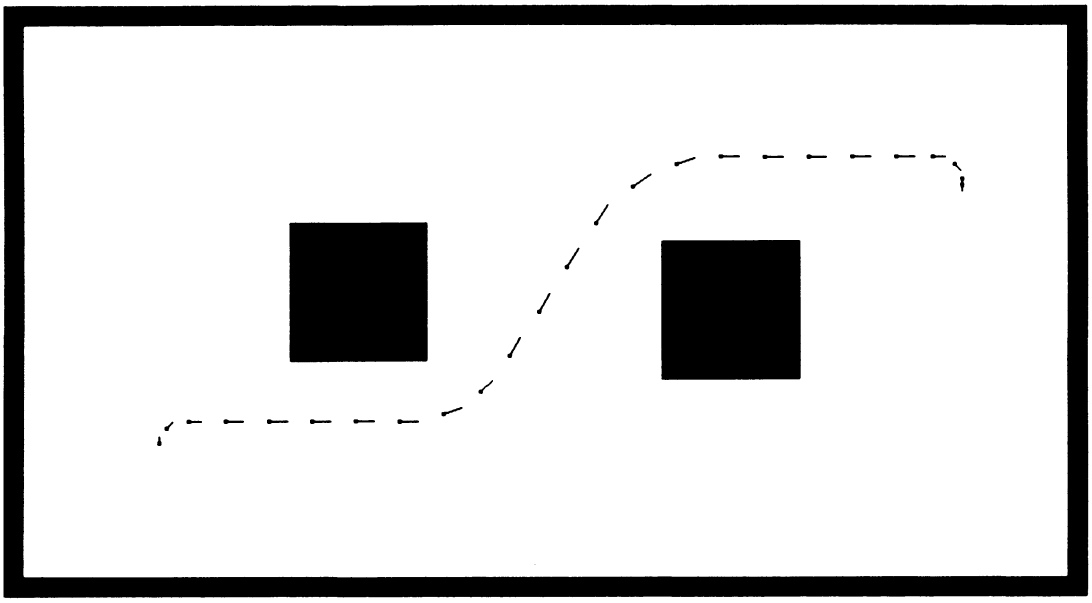
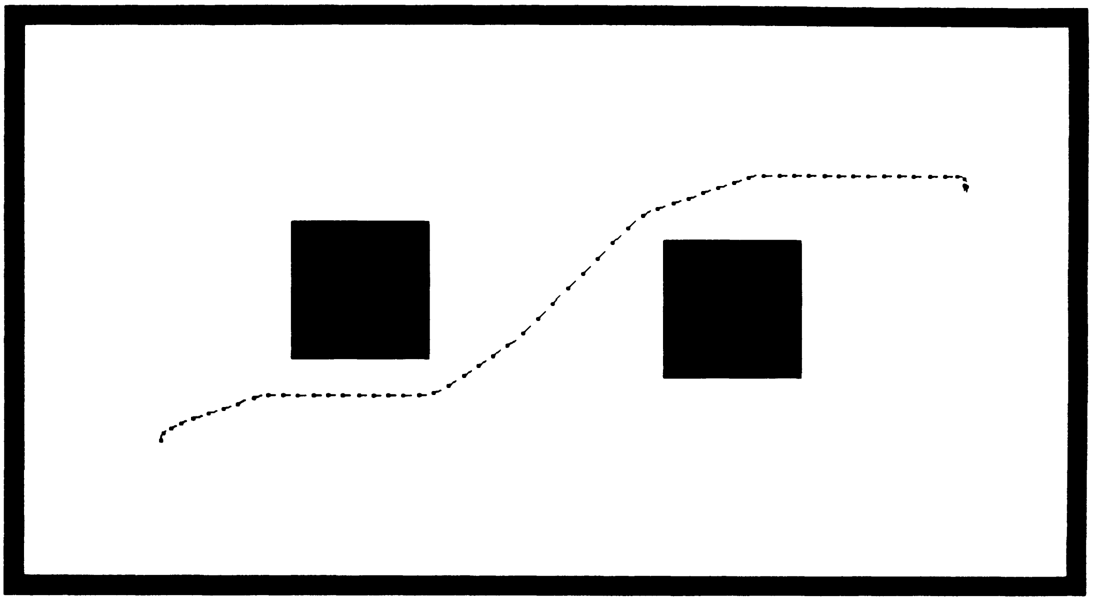


Figure 3.8: A trajectory found by our implementation running Algorithm 2. Here, the world size is 3.5 by 1.9 units. $\epsilon = 0.6$, $c_0 = 0.2$, $c_1 = 0.5$, $\overline{a} = 0.2$, and $\overline{v} = 0.3$. Start is at lower left, goal at upper right. (The "spikes" indicate the velocities). Our implementation took 4 minutes and 32 seconds of run time on a 32Mbyte SparcStation2 to plan the trajectory above.

(*a*)



(*b*)

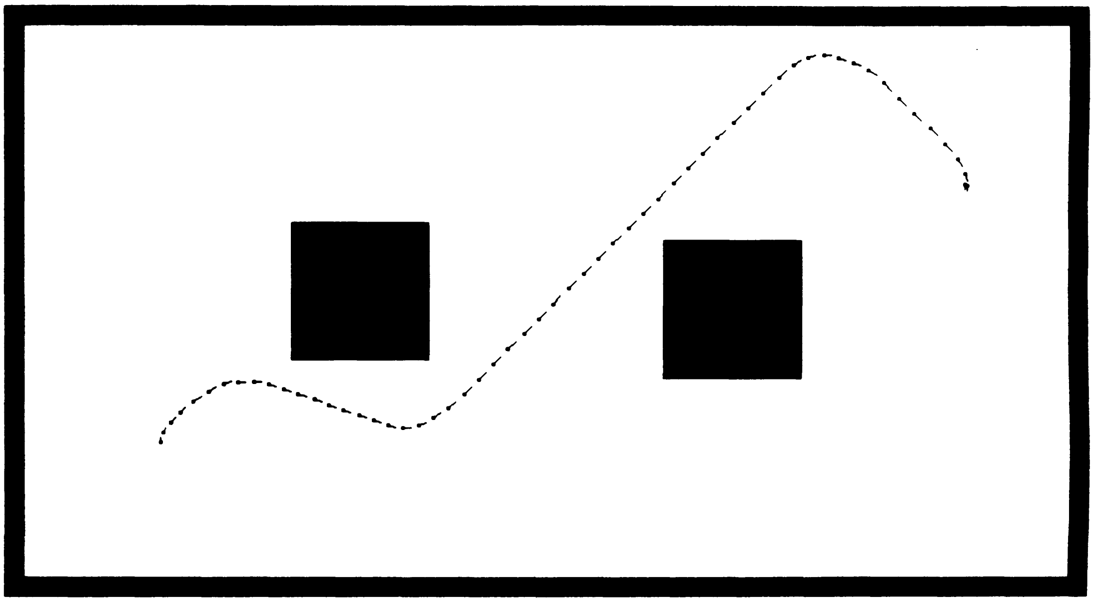
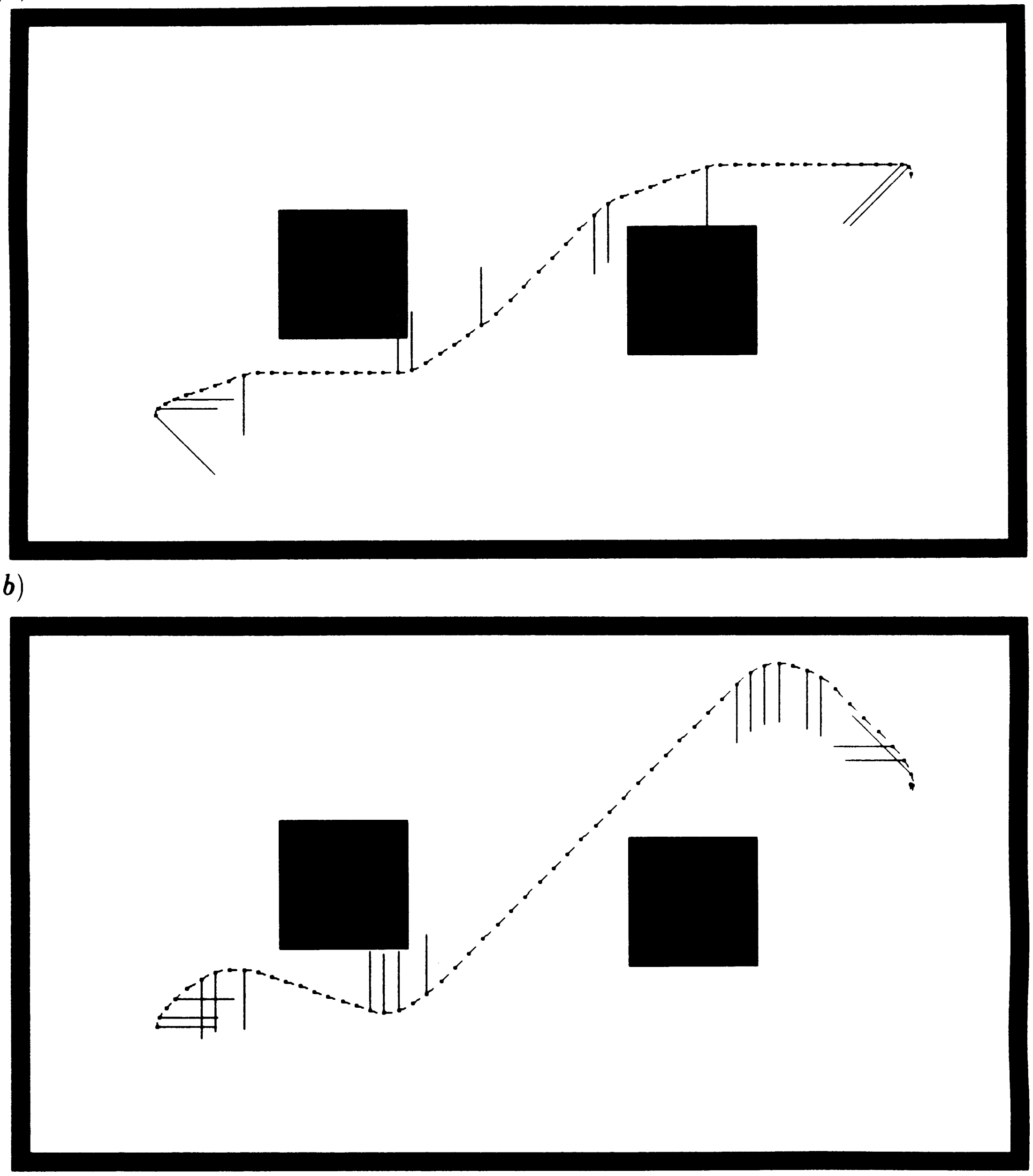


Figure 3.9: Near-optimal trajectories that were found by two versions of our implementation running Algorithm 3. If we choose to minimize (3.17) as a secondary performance measure, the implementation returns the solution in (a), which is less "curvy" than the solution in (b).

(a)



(b)

Figure 3.10: Trajectories from previous figure, with commanded accelerations. Thus, this figure represents the "plan" of commanded forces. The "diagonal" accelerations correspond to L_∞-norm $(\overline{a}, \tau)$-bangs that are fully-saturated in along both the x and y axes.

to argue the near-optimality of trajectories generated by other algorithms. Before our implementation, we had no way to study what provably good kinodynamic solutions look like without first having to solve problems by hand.

Despite the slowness of the our current implementation, we do not believe that the algorithms, Algorithm 2 in particular, are inherently impractical. Allowing solutions that take at most time $(1+\epsilon)T_{opt}$ instead of T_{opt}, where T_{opt} is the time for an optimal kinodynamic trajectory, results in a larger timestep size for a given set of kinodynamic bounds. This dramatically reduces the size of the reachability graph and the running time. Later, in Chapter 5, we use two approximation parameters ϵ_S and ϵ_T to describe the quality of approximation in terms of safety and time-optimality separately; this enables the user to trade off trajectory safety, trajectory time, and planning time to a certain extent.

In a practical implementation we could exploit parallelism at two levels of our algorithms. At a high level, since the TC-grid is a low-degree directed graph, it can be explored using parallel search techniques. Because of the spatial properties of the reachability graph and its underlying regular structure, we expect reasonable speedups on both SIMD and MIMD machines when the number of processors is appropriately smaller than the size of the TC-grid. At a lower level, we observe that checking the safety of $(\overline{a}, \tau)$-bangs can be done in parallel, since detecting a possible collision with one obstacle face is completely independent of checking for collisions with other faces.

One way in which we have experimented with the implementation is in the use of a secondary performance measure. In general, there may be many minimal-time $(\overline{a}, \tau)$-bang trajectories from a vertex $\mathbf{S}^*$ to a vertex $\mathbf{G}^*$, and the algorithm as described in previous sections might find any one of these. These trajectories can differ from each other in many properties, such as homotopy class, maximum curvature, and average speed. Since the algorithms each return only one trajectory, we would like to have a secondary optimization criterion that yields trajectories that are in some sense "good". One quantity of interest is

$$\int_0^T \|\mathbf{a}(t)\|_1 dt. \tag{3.17}$$

Minimizing this quantity as the secondary optimization criterion appears to produce trajectories with minimal chatter. (Chatter is a potential problem with "bang-bang" trajectories.) More importantly, it also seems to select trajectories that are "globally the least curvy" and that agree with our intuition of "good trajectories".

(Compare the trajectories in fig. 3.9.)

To minimize (3.17) we associate a unit "fuel cost" with axial bangs and a 2-unit cost with diagonal bangs. Let E_a denote the total fuel cost of a trajectory Γ_a. The total cost of a trajectory combines time with fuel consumption by lexicographic ordering on *time* $\times$ *fuel-cost*: $\mathcal{C}(\Gamma_a) < \mathcal{C}(\Gamma_b)$ if (a) $T_a < T_b$ or (b) $T_a = T_b$ and $E_a < E_b$. At the start of any timestep n, the implemented algorithm knows a min-fuel min-time $(\overline{a}, \tau)$-bang trajectory to each state it has found. The implementation finds a min-fuel min-time $(\overline{a}, \tau)$-bang trajectory to a state $\mathbf{Y}$ in the breadth-first search frontier for the stage by extending a previous min-fuel min-time trajectory with an $(\overline{a}, \tau)$-bang to reach $\mathbf{Y}$ with minimal total fuel cost. Thus, for a given problem, our implementation will return minimal-time $(\overline{a}, \tau)$-bang trajectory with the lowest fuel cost.

We can, in general, incorporate a secondary cost function L using lexicographic ordering when the following is true of L: if $\Gamma = \Gamma_1 * \Gamma_2$, then $L(\Gamma) = L(\Gamma_1) + L(\Gamma_2)$, where "$*$" denotes trajectory concatenation. This follows from the mapping of trajectory segments to graph edges. However, we will only be guaranteed to find the $(\overline{a}, \tau)$-*bang trajectory* Γ_q with minimal L-cost.

Let $\mathcal{T}$ denote the class of minimal-time $(\overline{a}, \tau)$-bang trajectories from $\mathbf{S}^*$ to $\mathbf{G}^*$, and let Let $\Gamma_{L_{opt}}$ be the time-optimal δ_v-safe trajectory with minimal L-cost. We would like to compare $L(\Gamma_q)$ to $L(\Gamma_{L_{opt}})$, but for an arbitrary cost functional L, the lexicographic ordering technique does not imply any particular relationship. To prove for some predicate $\mathcal{R}$ that $\mathcal{R}(\Gamma_q, \Gamma_{L_{opt}})$, we need to extend the tracking lemma to state that if $\Gamma_{L_{opt}}$ is an L-minimal time-minimal trajectory, then there is a trajectory Γ_q in $\mathcal{T}$ such that $\mathcal{R}(\Gamma_q, \Gamma_{L_{opt}})$. General techniques for proving such lemmas would be of considerable interest.

3.5 Summary

In this chapter we again considered optimal kinodynamic planning for Cartesian robots obeying L_∞ dynamics bounds. While we retained the problem formulation and made only minor modifications to the algorithm from the previous chapter, we proved stronger tracking lemmas to improve the complexity and accuracy bounds of the new algorithms significantly. Exploiting the decoupled dynamics and decoupled dynamics bounds of Cartesian robots, we are able to (a) tighten the complexity bound to $O\left(dc^d N \left(\frac{1}{\epsilon}\right)^{3d}\right)$, where ($c = c_B$ or c_C) is a constant polynomially dependent on the robot dynamics bound and the environment diameter, and (b) show that Algorithm 3 finds a trajectory taking at most time T_{opt} (the optimal time) instead of merely time $T_{opt}(1+\epsilon)$.

We prove the vital tracking lemmas by construction. Given a velocity function v_r obeying the relevant acceleration bounds, we first show the existence of velocity functions that are realizable by $(\overline{a},\tau)$-bangs, bound the given velocity function v_r above and below, and track it to within $\frac{7}{4}\overline{a}\tau$. Subsequent constructions prove the existence of a velocity functions whose integrals track the integral of the given function to the required tolerances while staying between the "bounding functions." Similar techniques will be used in proofs in later chapters.

We have described a preliminary implementation and reported the results of some simple experiments. This is the first implementation of polynomial-time, provably-good approximation algorithms for kinodynamic planning. While the current implementation runs slowly, eventually an improved implementation may be reasonable for practical off-line motion planning.

Chapter 4

Kinodynamic Planning With Coupled Dynamics Bounds

We have so far discussed optimal kinodynamic planning for a class of robots where the dynamics constraints allow us to reduce most of the necessary analysis to a simple one-dimensional system. Specifically, for Cartesian robots with L_∞ dynamics bounds, the acceleration and velocity bounds act independently in each coordinate direction. Furthermore, the set of instantaneous accelerations allowed by the dynamics constraints is constant.

We extend the results from previous chapters to robots that obey more general dynamics laws and dynamics bounds — in particular, d-link 3D robots with full rigid-body dynamics amidst obstacles. Specifically, we describe polynomial-time approximation algorithms for Cartesian robots obeying L_2 dynamics bounds and for open-kinematic-chain manipulators with revolute and prismatic joints. (See figures 1.3 and 1.4.) The correctness and complexity of these algorithms rely on new trajectory tracking lemmas for robots with coupled dynamics bounds. We obtain $O\left(c^d N(d + \log N)(\frac{1}{\epsilon})^{6d-1}\right)$ asymptotic bounds, where N measures the geometric complexity of the problem and c is a robot-dependent constant.

4.1 Kinodynamic Motion Planning for Robots With Coupled Dynamics Bounds

4.1.1 A More General Kinodynamic Planning Problem

We now reformulate the optimal and ϵ-optimal kinodynamic planning problems to accommodate a wider class of robots.

In a kinodynamic planning problem, a robot with d degrees of freedom must move from a start state $\mathbf{S} = (\mathbf{s}, \dot{\mathbf{s}})$ to a goal state $\mathbf{G} = (\mathbf{g}, \dot{\mathbf{g}})$ while avoiding a set of obstacles and configuration (*e.g.* joint) limits; these are the *kinematic constraints.*

We will denote robot configuration space by C, and its phase space, the robot state space, by TC. A robot motion taking time T_f can be specified by a twice-differentiable map $\mathbf{p} : [0, T_f] \rightarrow C$. This map is the *path* of the motion. The *trajectory* of a robot motion is the map $\Gamma : [0, T_f] \rightarrow TC$ given by $\Gamma(t) = (\mathbf{p}(t), \dot{\mathbf{p}}(t))$. Thus, a motion is determined by an initial state $(\mathbf{p}_0, \mathbf{v}_0)$ and an *acceleration function* $\mathbf{a} = \ddot{\mathbf{p}}$. We denote the position and velocity components of a subscripted trajectory Γ_r by $\mathbf{p}_r$ and $\dot{\mathbf{p}}_r$, respectively.

The robot motion is governed by a *dynamics law*, which relates applied generalized forces $\mathbf{f}$ to states, accelerations, and forces $\mathbf{G}(\mathbf{p})$ induced by gravity. For open kinematic chains [AS86,Hol83]:

$$\mathbf{f}(t) = \mathbf{M}(\mathbf{p}(t))\mathbf{a}(t) + [\dot{\mathbf{p}}^T(t)\mathbf{C}(\mathbf{p}(t))\dot{\mathbf{p}}(t)] + \mathbf{G}(\mathbf{p}(t)). \tag{4.1}$$

$\mathbf{M}(\mathbf{p}(t))$, the robot inertia tensor, is orthogonal, symmetric, and positive-definite. $\mathbf{C}(\mathbf{p}(t))$ is a tensor of rank three, and $[\dot{\mathbf{p}}^T(t)\mathbf{C}(\mathbf{p}(t))\dot{\mathbf{p}}(t)]$ denotes the column vector in which

$$\begin{aligned} [\dot{\mathbf{p}}^T(t)\mathbf{C}(\mathbf{p}(t))\dot{\mathbf{p}}(t)]_i &= \dot{\mathbf{p}}^T(t)\mathbf{C}^i(\mathbf{p}(t))\dot{\mathbf{p}}(t), \quad \text{where} \\ \mathbf{C}^i(\mathbf{p}(t))_{j,k} &= \textstyle\sum_{j,k} \left\{ \frac{\partial M_{jk}(\mathbf{p})}{\partial p_k} - \frac{1}{2}\frac{\partial M_{jk}(\mathbf{p})}{\partial p_i} \right\} \end{aligned} \tag{4.2}$$

(See [AS86] for a derivation.)

We call a robot whose inertia tensor is constant and whose dynamics law simplifies to

$$\mathbf{f}(t) = \mathbf{M}\mathbf{a}(t) \tag{4.3}$$

a *Cartesian robot.*

A robot motion $\mathbf{p}$ obeys dynamics bounds $(\overline{\mathbf{v}}, \overline{\mathbf{f}})$ if for all times t the joint velocities $\dot{\mathbf{p}}$ and the applied generalized forces $\mathbf{f}$ obey the following at each joint i:

$$\begin{aligned} |\dot{p}_i(t)| &\leq \overline{v}_i, \quad \text{and} \\ |f_i(t)| &\leq \overline{f}_i, \end{aligned} \tag{4.4}$$

These bounds imply global acceleration bounds $\overline{\mathbf{a}}$, and we define

$$A_{max} = \max_i \overline{a}_i.$$

For a Cartesian robot, acceleration can be substituted for force, and all these bounds are usually given in an L_p-norm. We consider the L_2-norm here, whereas in previous chapters we used the L_∞-norm. The dynamics laws and dynamics bounds that apply to a robot are its *dynamics constraints.* (See figure 1.4.) Note that when the norm makes a difference, we will specify it, except in the L_∞-case.

Generally, the problem parameters must include an encoding $\mathcal{M}$ of the robot's dynamics equation.[1] In addition, there must be an encoding $\mathcal{O}$ of the workspace obstacles. An instance of the *general kinodynamic planning problem*, then, is a tuple $\mathcal{K} = (\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{\mathbf{f}}, \overline{\mathbf{v}}, \mathcal{M})$. An *exact solution* is a a trajectory Γ such that $\Gamma(0) = \mathbf{S}$, $\Gamma(T_f) = \mathbf{G}$ at some time T_f, and Γ obeys the kinematic and dynamics constraints. Thus, the corresponding map $\mathbf{p}$ must avoid all obstacles and respect (4.1) and (4.4). The *time* for solution Γ is simply T_f. The *time-optimal kinodynamic planning problem* is to find a minimal-time solution, which is represented as a suitable encoding of the start state and the acceleration function $\mathbf{a}$.

We assume that the robot and obstacles are polyhedral, and that they have been mapped to configuration-space (c-space) obstacles with geometric complexity N in the configuration space, *i.e.*, the c-space constraints consist of N facets (c-space surfaces) as in [LP83,Don87,CD88]. *Free space* is the complement of the c-space obstacles in C. For a polyhedral robot of geometric complexity m and a set of polyhedral obstacles with geometrical complexity n, the number of configuration space constraints[2] $N = O(m(m+n))$ [LP83]. Finally, we assume that all linear degrees of freedom are above bounded by a length l.

[1] Since the general form of the equation is given by equation (4.1), this involves supplying an encoding of the matrices $\mathbf{M}(\mathbf{p})$ and $\mathbf{C}(\mathbf{p})$ and the vector $\mathbf{G}(\mathbf{p})$ in terms of configuration $\mathbf{p} \in C$. Thus, for a given link of a manipulator, $\mathcal{M}$ include constants such as the mass, center of mass, inertia, and the distance and relative orientation between joint axes. For an example of what must be encoded in $\mathcal{M}$, see section C.1.1 in Appendix C.

[2] An arm must avoid self collisions.

4.1.2 Safe and Near-Optimal Kinodynamic Plans

A theoretically time-optimal solution may require unrealizable precision in control or sensing to execute. Because we wish to concisely associate a deviation from a safe trajectory with a decrease in safety, we again require a safe solution to avoid obstacles by a margin that is a strictly-positive affine function of speed.[3] As before, we define a trajectory to be *δ_v-safe* if and only if for all times t in $[0, T_f]$, there exists a ball about $\mathbf{p}(t)$ in free space of radius

$$\delta_v(c_0, c_1)(\dot{\mathbf{p}}(t)) = c_0 + c_1 \|\dot{\mathbf{p}}(t)\|, \tag{2.3}$$

where $c_0 > 0$ and $c_1 \geq 0$ are problem parameters. For any scalars $c_0 > 0$ and $c_1 \geq 0$, we define an *optimal (δ_v-safe) kinodynamic solution* to be a δ_v-safe solution whose time is minimal. An instance of the optimal (safe) kinodynamic planning problem is a tuple $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{\mathbf{f}}, \overline{\mathbf{v}}, l, \mathcal{M}, c_0, c_1)$. $\overline{\mathbf{f}}$, $\overline{\mathbf{v}}$, c_0, and c_1 the *kinodynamic bounds.* Together, the kinodynamic bounds and $\mathcal{M}$ are the *kinodynamic specifications* of the robot.

As before, the quality of a solution is measured in terms of an approximation parameter ϵ. A solution $\Gamma_q : [0, T_q] \to TC$ is ϵ-optimal if (a) $\Gamma_q(0)$ and $\Gamma_q(T_q)$ are within $O(\epsilon)$ tolerances of $\mathbf{S}$ and $\mathbf{G}$, respectively; (b) $T_q \leq (1+\epsilon)T_{opt}$, where T_{opt} is the time of an optimal solution; and (c) Γ_q observes $\delta'_v(c_0, c_1)$-safety, as in equation (2.4). (See section 2.1.1.)

4.1.3 Statement of Results

4.1.3.1 Robots With Coupled Dynamics Bounds

A robot system obeys *coupled dynamics bounds* if (a) its dynamics equations cannot be separated into the dynamics equations of d independent one-dimensional systems or (b) or there is no fixed coordinate transformation such that the velocity and acceleration bounds in each axial direction are independent of the bounds in all other axial directions. (See figure 1.4.)

We describe provably good approximation algorithms for the optimal (safe) kinodynamic planning problems for two classes of robots with coupled dynamics

---

[3] Another analysis suggests the use a quadratic function of speed, or more generally, one that contains a quadratic form $\mathbf{v}^T\mathbf{A}(\mathbf{x})\mathbf{v}$, where $\mathbf{A}(\mathbf{x})$ is symmetric and positive definite. Assuming bounded derivitives, we can use vector calculus to derive an ϵ-safe tracking tolerance that is $\Omega(\epsilon)$ in both configuration (position) and velocity, similar to that in the Safe Tracking Lemma 2.2.4.

bounds: (a) Cartesian Robots with L_2 dynamics bounds and (b) open-chain manipulators with revolute and prismatic joints. Somewhat surprisingly, the algorithms for these two different problems are based on similar underlying principles. Given a problem instance and an approximation parameter ϵ, these algorithms will find an ϵ-optimal solution if a $\delta_v(c_0, c_1)$-safe solution exists.

These algorithms run in time polynomial in the geometric complexity N of the configuration space obstacles and in the resolution $(\frac{1}{\epsilon})$. The constant in the c_D^d (c_E^d) factor of the overall complexity (see Table 1.5.1) depends on the kinodynamic specifications of the robot, and thus this factor is constant for a given robot. Our algorithms are ϵ-approximation schemes that are *fully polynomial* in the combinatorial complexity of the geometry and *pseudo-polynomial* in the kinodynamic specifications.

Formally stated, we show the following:

Theorem 4.1.1 *Let $\overline{a}$ and $\overline{v}$ be velocity and acceleration bounds, respectively. Let $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ be an optimal kinodynamic planning problem for a Cartesian robot obeying L_2 dynamics bounds. Let $0 < \epsilon < 1$.*

Suppose there is a $\delta_v(c_0, c_1)$-safe trajectory from $\mathbf{S}$ to $\mathbf{G}$ taking time T_{opt}. Then the algorithms in Sections 4.2.1.1 and 4.2.1.2 each find a $(1-\epsilon)\delta_v(c_0, c_1)$-safe trajectory taking time at most $T_{opt}(1+\epsilon)$ from some $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*)$ to some $\mathbf{G}^* = (\mathbf{g}^*, \dot{\mathbf{g}}^*)$ such that $\mathbf{S}^*$ and $\mathbf{G}^*$ are ϵ-close to $\mathbf{S}$ and $\mathbf{G}$, respectively.

The asymptotic running time of the algorithms is $O\left(c_D^d p(N, \epsilon, d) \left(\frac{1}{\epsilon}\right)^{6d-1}\right)$, where N is the geometric complexity of the c-space obstacles, c_D is a constant dependent on the kinodynamic specifications, and $p(N, \epsilon, d)$ is the time-complexity of checking the $(1-\epsilon)\delta_v(c_0, c_1)$-safety of one of the "primitive" trajectory segments the algorithms consider.

Theorem 4.1.2 *Let $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{\mathbf{f}}, \overline{\mathbf{v}}, \mathcal{M}, l, c_0, c_1)$ be an optimal kinodynamic planning problem for an open-chain manipulator with revolute and/or prismatic joints. Let $0 < \epsilon < 1$.*

Suppose there is a $\delta_v(c_0, c_1)$-safe trajectory from $\mathbf{S}$ to $\mathbf{G}$ taking time T_{opt}. Then the algorithms in Sections 4.2.1.1 and 4.2.1.2 each find a $(1-\epsilon)\delta_v(c_0, c_1)$-safe trajectory taking time at most $T_{opt}(1+\epsilon)$ from some $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*)$ to some $\mathbf{G}^* = (\mathbf{g}^*, \dot{\mathbf{g}}^*)$ such that $\mathbf{S}^*$ and $\mathbf{G}^*$ are ϵ-close to $\mathbf{S}$ and $\mathbf{G}$, respectively.

The asymptotic running time of the algorithms is $O\left(c_E^d p(N, \epsilon, d) \left(\frac{1}{\epsilon}\right)^{6d-1}\right)$ where N is the geometric complexity of the c-space obstacles, c_D is a constant *dependent on the kinodynamic specifications, and* $p(N, \epsilon, d)$ *is the time-complexity of checking the* $(1-\epsilon)\delta_v(c_0, c_1)$*-safety of one of the "primitive" trajectory segments the algorithms consider.*

We have omitted the complexity factors containing kinodynamic specifications because they are constant for a given robot and because the terms can be very complicated. Our claim that the complexities are pseudopolynomial in these parameters is substantiated in in Sections 4.3.1 through 4.3.3, with further detail given in Appendix C. We express their overall contribution to the complexity bounds as a factor of c_D^d (or c_E^d), for constants derived from the kinodynamic specifications.

In Section 4.3.4 we argue that $p(N, \epsilon, d)$ is roughly $O(N(d + \log N))$. Substitution yields the complexities in Table 1.5.1. As we discuss in that section, for non-Cartesian open-chain manipulators amidst polyhedral real-space obstacles, exact collision detection for quadratic paths requires the solution of trignometric equations that cannot be transformed into algebraic equations using the usual substitution methods,[4] such as [Can86]. Furthermore, for these manipulators, trajectory segments corresponding to constant extremal controls are solutions to systems of trignometric differential equations, and the trajectories found by the corresponding algorithm are extremal to the accuracy of the representation.[5] In both of these cases, numerical techniques would be used to do safety-checking approximately.

For a choice of c-space coordinates in which the path is algebraic in time and in which the obstacles are represented as semi-algebraic sets *in configuration space*, $p(N, \epsilon, d)$ would be $O(N \log N)$ for the True Extremal Algorithm (applied to Cartesian robots) and to the Near-Extremal algorithm (in general). In the subcase where the configuration space obstacles are the union of convex polyhedra with geometric complexity N, $p(N, \epsilon, d) = O(dN)$.

[4]Specifically, if the *workspace obstacles* are semi-algebraic sets and the robot has at least one rotational degree of freedom, then surfaces of some c-space obstacles might be non-algebraic, *e.g.*, trignometric. If path $\mathbf{p}(t)$ is quadratic in time, then substitutions (such as $u_i = \tan \theta_i$) to get semi-algebraic c-space obstacle representations result in path representations that are no longer algebraic.

[5]See section 4.2.1.1.

4.1.4 Generalizing the Basic Algorithms to Robots with Coupled Dynamics Bounds

Intuitively, we would hope to extend the approach from previous chapters to robots with coupled dynamics by choosing appropriate analogues of $(\overline{a}, \tau)$-bangs and proving an applicable tracking lemma. Lemma 4.2.1, which we call *The Coupled Tracking Lemma*, relates a desired tracking tolerance and a dynamics advantage to a sufficiently small timestep for a class of *sets* of control primitives. Given a trajectory Γ_r, a tracking tolerance (η_x, η_v), and a set of control primitives that has a given dynamics advantage over Γ_r, we can find a timestep τ guaranteeing that some trajectory Γ_q generated by the control primitives and τ will track Γ_r to tolerance (η_x, η_v). The class of control primitives includes (a) near-extremal, piecewise-constant accelerations and (b) extremal (possibly time-varying) accelerations that result from applying piecewise-constant extremal controls. By applying Lemma 4.2.1 we obtain a provably-good kinodynamic planning algorithm that generates trajectories obeying near-extremal accelerations, and hence, near-extremal controls.

In robotics and control theory there is a family of results on the feasibility of planning and approximating optimal trajectories using only piecewise-extremal controls.[6] (These results are often called "bang-bang" theorems.) We can use Lemma 4.2.1 to find a timestep τ sufficiently small so that there exists some piecewise-extremal, ϵ-optimal trajectory having control switchings only at multiples of τ. Unfortunately, the lemma does not yield a polynomial-time approximation scheme using these trajectories, however, because the number of vertices in the reachability graph might be exponential in the path length. Fortunately, we can prove a more robust Lemma 4.2.2, which provides conditions that allow us to prune the search to a sufficiently small subset of the graph. We can thus show how to plan provably near-optimal kinodynamic solutions that only employ extremal controls with no additional cost in asymptotic complexity for a given robot.

[6]See Section 1.4 for references.

4.2 Robots with Coupled Dynamics Bounds

4.2.1 The Algorithms

We describe two general algorithms for finding near-optimal kinodynamic trajectories for Cartesian robots with L_2-norm dynamics bounds and for open-chain manipulators. The first algorithm searches a reachability graph corresponding to piecewise-constant, extremal forces and torques, and we will refer to it as "the True-Extremal Algorithm". The second uses piecewise-constant, near-extremal accelerations, and we call it "the Near-Extremal Algorithm". We present both algorithms in outline form. Derivations of key parameters are described in Sections 4.3.2 and 4.3.3.

Since we will use the notion of a "slowed-down" trajectory often in this section we now formalize two definitions.

Definition 4.2.1 *Let $\Gamma_r : [0, T_f] \to TC$, and suppose that $\epsilon \geq 0$. Then we define the ϵ-time-rescaled trajectory $\Gamma'_r : [0, (1+\epsilon)T_f] \to TC$ as follows:*

$$\Gamma'_r(t) = \left(\mathbf{p}_r\left(\frac{t}{1+\epsilon}\right), \left(\frac{1}{1+\epsilon}\right)\dot{\mathbf{p}}_r\left(\frac{t}{1+\epsilon}\right)\right). \tag{4.5}$$

We will be extensively comparing dynamical properties associated with states in rescaled trajectories.

Definition 4.2.2 *Let $\mathbf{X} = (\mathbf{x}, \dot{\mathbf{x}}) \in TC$, and suppose that $\epsilon \geq 0$. Then* ϵ-time-rescaled state $\mathbf{X}' = (\mathbf{x}', \dot{\mathbf{x}}')$ *is defined by:*

$$\begin{aligned} \mathbf{x}' &= \mathbf{x} \\ \dot{\mathbf{x}}' &= \left(\tfrac{1}{1+\epsilon}\right)\dot{\mathbf{x}}. \end{aligned} \tag{4.6}$$

4.2.1.1 The True-Extremal Algorithm

Although the correctness of the True-Extremal Algorithm is harder to prove than the correctness of the Near-Extremal Algorithm, the algorithm itself is easier to to describe.

We first describe the control primitives used to generate the reachability graph. Let $\mathcal{U}$ be the set of generalized (control) forces that are extremal with respect to given bounds $\overline{\mathbf{f}}$. (We can define similar sets for L_p bounds $\overline{f}$ and $\overline{a}$.) Given a scalar *discretization parameter* $\overline{\mu} > 0$, we say that $\mathcal{U}_{\overline{\mu}} \subset \mathcal{U}$ is a *a $\overline{\mu}$-discretization* *of the control extremals* if for every $\mathbf{u}_a \in \mathcal{U}$ there is some $\mathbf{u}_b \in \mathcal{U}_{\overline{\mu}}$ such that $\frac{1}{2}\|\mathbf{u}_a - \mathbf{u}_b\|_\infty \leq \overline{\mu}$. (See figure 4.1.) As along as $\overline{\mu}$ is small enough, our algorithm may choose any such $\mathcal{U}_{\overline{\mu}}$, so we henceforth refer to $\mathcal{U}_{\overline{\mu}}$ as *the* $\overline{\mu}$-discretization of control extremals.

The application of a member of $\mathcal{U}_{\overline{\mu}}$ for duration τ is called a *true* $(\overline{\mu}, \tau)$*-bang*[7]. Recall that the dynamics equation maps states and generalized forces to accelerations. From equation (4.1), the acceleration is given by

$$\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f}) = \mathbf{M}^{-1}(\mathbf{p}) \left(\mathbf{f} - [\dot{\mathbf{p}}^T \mathbf{C}(\mathbf{p})\dot{\mathbf{p}}] - \mathbf{G}(\mathbf{p})\right). \tag{4.7}$$

Thus, a true $(\overline{\mu}, \tau)$-bang and a state $\mathbf{X}$ together determine a trajectory segment of duration τ starting at $\mathbf{X}$, *i.e.*, the solution to (4.7) with $(\mathbf{p}(0), \dot{\mathbf{p}}(0)) = \mathbf{X}$, and $\mathbf{f}$ given by the $(\overline{\mu}, \tau)$-bang. If this trajectory ends at state $\mathbf{Y}$, then we say that there is *a true* $(\overline{\mu}, \tau)$*-bang from* $\mathbf{X}$ *to* $\mathbf{Y}$. We will sometimes use "true $(\overline{\mu}, \tau)$-bang" to refer to the trajectory segment generated by a bang when the meaning is clear.

For an L_2 Cartesian robot or open-chain manipulator, the number of states reachable from a root vertex $\mathbf{S}^*$ via a sequence of m $(1 - \epsilon)\delta_v$-safe $(\overline{\mu}, \tau)$-bangs can, in general, be exponential in the path length m; the total number states reachable via an infinite number of such bangs can be unbounded. Therefore, we will apply (b_x, b_v)*-bucket pruning*, a spatial hashing technique, in the search. (See figure 4.2.) In (b_x, b_v)-bucket pruning, TC is divided into voxels of diameter b_x in spatial dimensions and b_v in velocity dimension. When the graph search finds a vertex in a voxel in which another vertex has previously been found, the edges from the new vertex are simply not explored, and we say they are *pruned* from the search. (Compare this to the search in [Don87], figure 31 on page 315.) Since the state space is bounded and the reachability graph has a $O((\overline{\mu})^{-d})$ branching factor, the size of the subgraph searched is $O((\overline{\mu} b_x b_v)^{-d})$.

True-Extremal Algorithm Outline

Given a kinodynamic planning problem instance $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{\mathbf{f}}, \overline{\mathbf{v}}, \mathcal{M}, c_0, c_1)$ for an L_2 Cartesian robot or an open-chain manipulator, the True-Extremal Algorithm does the following.

1. Chooses the root vertex $\mathbf{S}^*$ by ϵ-rescaling $\mathbf{S}$, *i.e*, $\mathbf{s}^* = \mathbf{s}$ and $\dot{\mathbf{s}}^* = \left(\frac{1}{1+\epsilon}\right) \dot{\mathbf{s}}$.

2. It chooses a timestep τ and a control discretization $\overline{\mu}$ that are both $\Omega(\epsilon)$ and chooses a $\overline{\mu}$-discretion of the control extremals.

[7] The "true" distinguishes it from uses of "bang" which refer to controls that are not truly extremal, such the "$(\boldsymbol{\mu}, \tau)$-bangs" in later sections of this thesis and in [DX90a,DX90b]

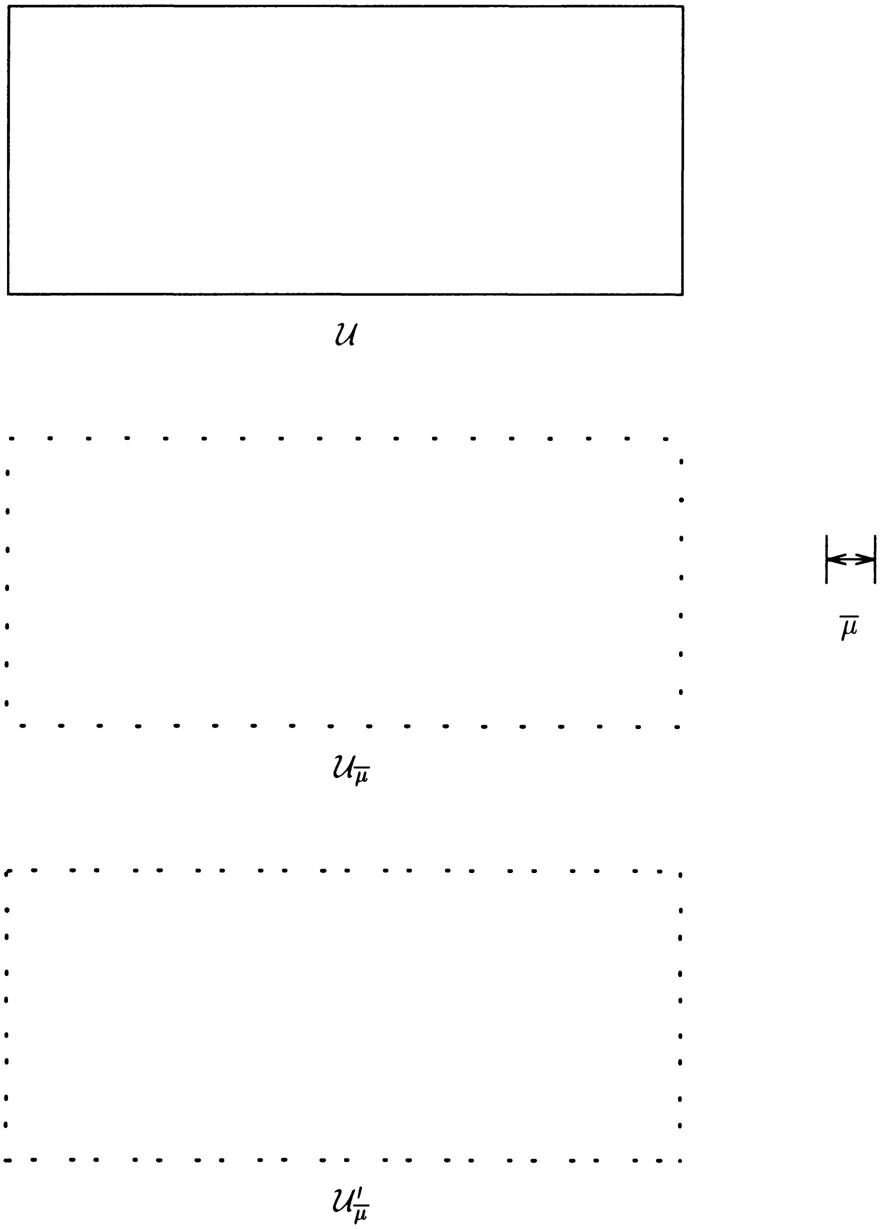


Figure 4.1: An extremal set of controls $\mathcal{U}$ and two $\overline{\mu}$-discretizations of $\mathcal{U}$: $\mathcal{U}_{\overline{\mu}}$ and $\mathcal{U}'_{\overline{\mu}}$.

3. Let $\mathcal{G}$ be the reachability graph rooted at $\mathbf{S}^*$ whose edges are $(1-\epsilon)\delta_v(c_0, c_1)$-safe $(\overline{\mu}, \tau)$-bangs. The algorithm searches for the shortest path from $\mathbf{S}^*$ to any vertex that is within (η_x, η_v) of $(\mathbf{g}, \frac{1}{1+\epsilon}\dot{\mathbf{g}})$. η_x, and η_v are both $\Omega(\epsilon)$. The search is done breadth-first with (b_x, b_v)-bucket pruning in which b_x is $\Omega(\epsilon^3)$, and b_v is $\Omega(\epsilon^2)$. The algorithm only constructs the part of the graph that it actually searches.

4. The algorithm returns the trajectory that corresponds the shortest path in the reachability graph (from step 3) if a $\delta_v(c_0, c_1)$-safe solution exits.

The *acceleration functions* associated with true $(\overline{\mu}, \tau)$-bangs beginning at a given state $\mathbf{X}$ will be important to our discussion. We call these functions the *true* $(\overline{\mu}, \tau)$*-bangs at* $\mathbf{X}$, and their set will be denoted by $\overline{\mathcal{H}}(\mathbf{X}, \tau)$. A *true* $(\overline{\mu}, \tau)$*-bang trajectory* is a concatenation of true $(\overline{\mu}, \tau)$-bangs. If Γ_r is a true $(\overline{\mu}, \tau)$-bang trajectory, then during each open time interval $(n\tau, (n+1)\tau)$, $\mathbf{a}_r(t - n\tau) \in \overline{\mathcal{H}}(\Gamma_r(n\tau), \tau)$.

Thus, the controls of a trajectory found by the True-Extremal Algorithm are constant and extremal over each individiual timestep, while the acceleration will be extremal but may vary with time, according to the integration of equation (4.7). This contrasts with the trajectories found by the Near-Extremal Algorithm, whose *accelerations* are constant and near-extremal over each timestep, but whose controls may be time-varying, as given by equation (4.1).

Note that for robots with revolute joints there is generally no closed form for trajectory segments corresponding to true $(\overline{\mu}, \tau)$-bangs. However, using an r^{th}-order numerical integration procedure will yield trajectory segments corresponding to controls within $O(\tau^r)$ (and therefore within $O(\epsilon^r)$) of being constant and extremal. Thus, as long as ϵ is sufficiently small, we can use, say $r = 4$ and Runge-Kutta numerical integration.

4.2.1.2 The Near-Extremal Algorithm

The Near-Extremal Algorithm builds a reachability graph using trajectory segments corresponding to constant near-extremal accelerations each lasting one timestep. These accelerations and a root vertex are chosen so that the size of the reachability is automatically bounded. We now describe these accelerations and outline the algorithm.

A positive vector $\boldsymbol{\mu} \in \Re^d$ determines a grid that discretizes the acceleration space $\Re^d$. Precisely, let μ_i be the grid-spacing in dimension i, and let the grid be

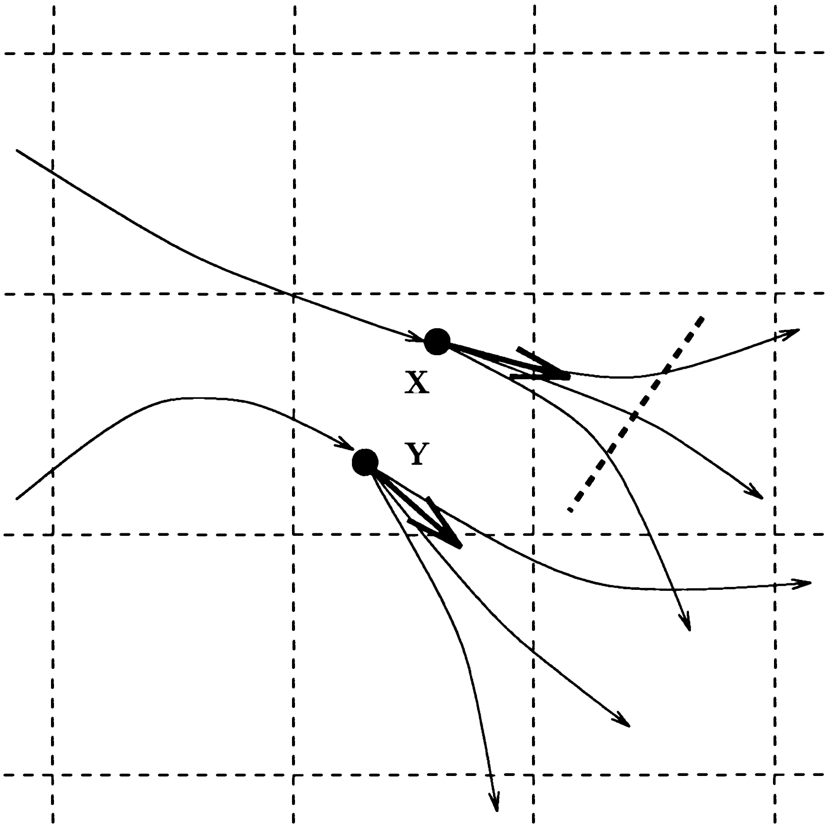


Figure 4.2: Search pruning. During the search, an edge (trajectory segment) to vertex (state) **X** is found, but a previously found vertex **Y** is in the same bucket. Consequently, edges from **X** are pruned from the search. (Compare this to the search in [Don87], figure 31 on page 315.)

aligned with the coordinate axes so that the origin $\mathbf{0}$ is a gridpoint, *i.e.*, it lies at one of the interstices of the grid. We call the set of gridpoints the $\boldsymbol{\mu}$-grid. For now, suppose timestep τ has been chosen, and suppose that the set of accelerations has been discretized. Given a state $\mathbf{X} \in TC$, let $\mathcal{A}_\mathbf{X}$ denote the set of instantaneous accelerations possible at $\mathbf{X}$ under the dynamics constraints.

Let $\hat{\mathcal{A}}_\mathbf{X} \subset \mathcal{A}_\mathbf{X}$ denote the set of constant accelerations[8] possible for trajectories of duration τ beginning at state $\mathbf{X}$:

$$\hat{\mathcal{A}}_\mathbf{X} = \{\mathbf{a} \in \mathcal{A}_X \mid \mathbf{a} \in \bigcap_{t\in[0,\tau]} \mathcal{A}_{\Gamma(\mathbf{X},\mathbf{a},t)}\},$$

where $\Gamma(\mathbf{X}, \mathbf{a}, \cdot)$ is the trajectory that results in applying constant acceleration $\mathbf{a}$ at state $\mathbf{X}$. Then, a "constant acceleration" analogue of the set of true $(\overline{\mu}, \tau)$-bangs would be the set of $\boldsymbol{\mu}$-gridpoints that lie within a $\boldsymbol{\mu}$-grid-spacing of the boundary $\partial\hat{\mathcal{A}}_\mathbf{X}$ of $\hat{\mathcal{A}}_\mathbf{X}$. Unfortunately, it appears that this set might not be easy to enumerate in general, because of non-linear dynamics and the apparent "fixed-point" nature[9] of $\hat{\mathcal{A}}_\mathbf{X}$. For the algorithm we will therefore define sets of near-extremal constant accelerations using "conservative approximations" of the sets $\hat{\mathcal{A}}_\mathbf{X}$.

Let $\mathcal{D}_\mathbf{X} \subset TC$ denote the set of states that can be reached from $\mathbf{X}$ within time τ without violating the dynamics constraints. Then one conservative approximation is

$$\tilde{\mathcal{A}}_\mathbf{X} = \bigcap_{\mathbf{Y}\in\mathcal{D}_\mathbf{X}} \mathcal{A}_\mathbf{Y} \quad \subset \quad \hat{\mathcal{A}}_\mathbf{X}. \tag{4.8}$$

This set still might be difficult to compute when the robot dynamics are non-linear, since the intersection is taken over a set of states.

The dynamics and dynamics bounds determine a global bounds on the magnitudes Δx and Δv of the possible position and velocity changes during a duration of length τ. Using these bounds, we can compute an analogous "acceleration space" bound E_A, which has the following property: if $\mathbf{a}^\mathbf{Y} \in \mathcal{A}_\mathbf{Y}$ for some $\mathbf{Y}$ within $(\Delta x, \Delta v)$ of $\mathbf{X}$, then there is some $\mathbf{a}^X \in \mathcal{A}_\mathbf{X}$ such that $\|\mathbf{a}^\mathbf{Y} - \mathbf{a}^\mathbf{X}\| \leq E_A$. (See Section 4.3.3.1 and Appendix C.)

We now define:

$$\hat{\mathcal{A}}^*_\mathbf{X} = \bigcap_{\Delta\mathbf{a}\in\mathcal{B}_{E_A}(\mathbf{0})} \{\mathbf{a} \in \mathcal{A}_\mathbf{X} | \exists \mathbf{a}' \in \mathcal{A}_\mathbf{X}, \mathbf{a} = \mathbf{a}' + \Delta\mathbf{a}\}, \tag{4.9}$$

[8] In this section (4.2.1.2) we will treat constant accelerations as the vectors that correspond to constant, vector-valued functions.

[9] Define $Frob_\mathbf{X}(\mathcal{A}) = \bigcap_{\mathbf{a}\in\mathcal{A}} \bigcap_{t\in[0,\tau]} \mathcal{A}_{\Gamma(\mathbf{X},\mathbf{a},t)}$. Then $\hat{\mathcal{A}}_\mathbf{X}$ is the solution of $Frob_\mathbf{X}(\mathcal{A}) = \mathcal{A}$.

where $B_\delta(\mathbf{y})$ denotes the δ-ball about $\mathbf{y}$. Since $\mathcal{A}_\mathbf{X}$ is a parallelpiped or a closed L_p-norm d-ball, this set has a simple geometry. Thus,

$$\widehat{\mathcal{A}}^*_\mathbf{X} \subset \widetilde{\mathcal{A}}_\mathbf{X} \subset \widehat{\mathcal{A}}_\mathbf{X} \subset \mathcal{A}_\mathbf{X},$$

with equality for Cartesian robots. (See figure 4.3.)

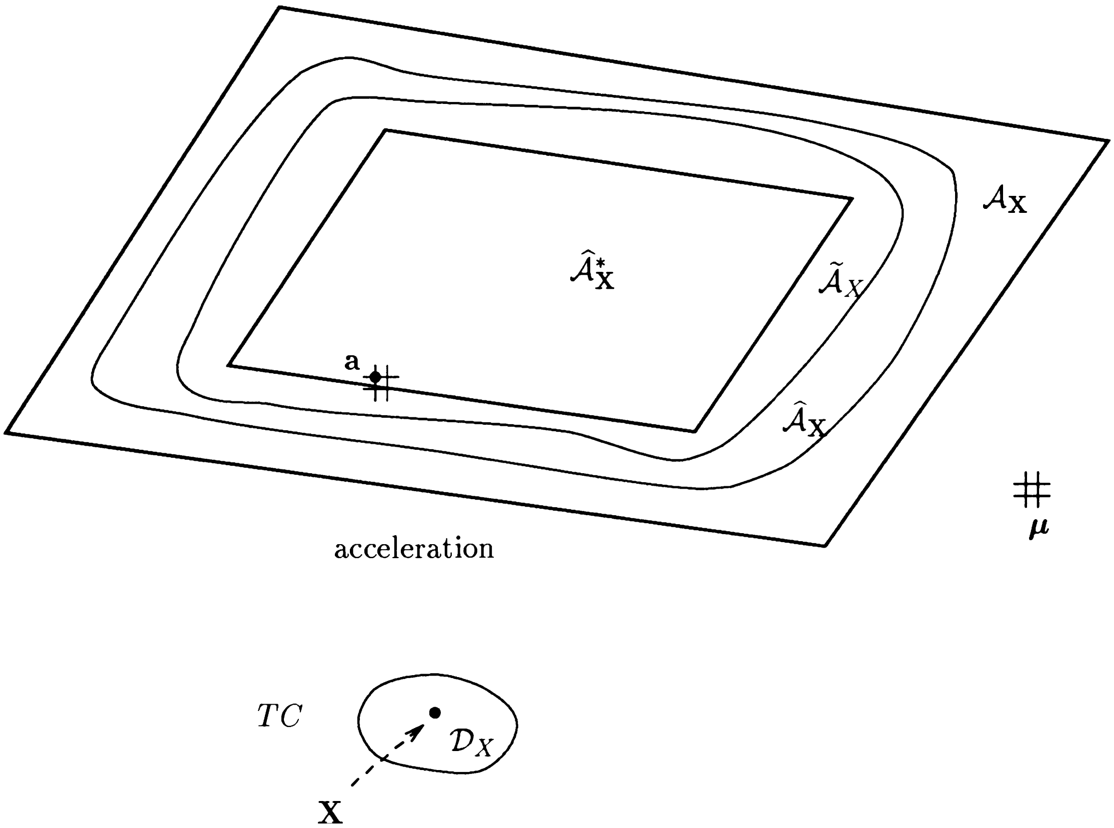


Figure 4.3: Sets of accelerations. $\mathcal{A}_\mathbf{X}$ is the set of possible accelerations at state $\mathbf{X}$. $\widehat{\mathcal{A}}_\mathbf{X}$ is the set of accelerations that can be maintained for duration τ. $\widetilde{\mathcal{A}}_\mathbf{X} = \bigcap \mathcal{A}_\mathbf{Y}$, where the intersection is taken over all $\mathbf{Y} \in \mathcal{D}_\mathbf{X}$, is a conservative estimate of this set. $\widehat{\mathcal{A}}^*_\mathbf{X}$, used by the Near-Extremal to obtain the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations $\mathcal{H}(\mathbf{X}, \tau)$, is "geometrically computable". $\mathbf{a}$ is a member of $\mathcal{H}(\mathbf{X}, \tau)$. This figure only shows set relationships; the actual corners of $\widehat{\mathcal{A}}^*_\mathbf{X}$ would be beveled.

We will call $\widehat{\mathcal{A}}^*_\mathbf{X}$ the set of *conservatively τ-feasible constant accelerations at* $\mathbf{X}$.

Our algorithms will treat[10]

$$\overline{\mathcal{A}}^*_{\mathbf{X}} = \partial \hat{\mathcal{A}}^*_{\mathbf{X}}, \tag{4.10}$$

as the set of "most-extremal" constant accelerations feasible for a trajectory of duration τ starting at state $\mathbf{X}$.

We now define the set of $(\boldsymbol{\mu}, \tau)$-*extremal shell of accelerations at* $\mathbf{X}$ to be given by

$$\mathcal{H}(\mathbf{X}, \tau) = \{\mathbf{a} \in \boldsymbol{\mu}\text{-grid} \quad | \, \exists \mathbf{a}' \in \overline{\mathcal{A}}^*_{\mathbf{X}} \forall i, |a_i - a_i'| \leq \mu_i\}. \tag{4.11}$$

This notation and this term will also refer to the corresponding set of constant acceleration functions.

The application of a $(\boldsymbol{\mu}, \tau)$-extremal acceleration for duration τ is called a $(\boldsymbol{\mu}, \tau)$-*bang.*[11] Terminology concerning $(\overline{a}, \tau)$-bangs and true $(\overline{\mu}, \tau)$-bangs will be analogously extended to $(\boldsymbol{\mu}, \tau)$-bangs. For example, if a $(\boldsymbol{\mu}, \tau)$-bang results in the transition from state, $\mathbf{X}$ to state $\mathbf{Y}$, we say there is $(\boldsymbol{\mu}, \tau)$-bang from $\mathbf{X}$ to $\mathbf{Y}$.

Suppose that for a state $\mathbf{S}^*$ and in each dimension i, the acceleration discretization μ_i divides the state velocity $\dot{s}_i^*$. Then all states reachable from $\mathbf{S}^*$ via a sequence of $(\boldsymbol{\mu}, \tau)$-bangs lie at the interstices of an underlying grid covering TC, with a grid spacing $\frac{\mu_i \tau^2}{2}$ in position dimension i and $\mu_i \tau$ in velocity dimension i. Thus, a $\boldsymbol{\mu}$-grid, a timestep τ, and a choice of origin in C *induce* an underlying regular TC-*grid.* If $\mathbf{S}^*$ is the root vertex of a reachability graph, then the number of its vertices is bounded above by the number of TC-gridpoints.

Near-Extremal Algorithm Outline

Given a kinodynamic planning problem $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{\mathbf{f}}, \overline{\mathbf{v}}, l, \mathcal{M}, c_0, c_1)$ for an L_2 Cartesian robot or an open-chain manipulator, the Near-Extremal Algorithm does the following.

1. Chooses a timestep τ and an acceleration space discretization $\boldsymbol{\mu}$. τ and all μ_i are $\Omega(\epsilon)$. If the i^{th} joint has configuration space S^1 (as revolute joints without limits do), then μ_i must be chosen so that $K_i \mu_i \tau^2 = 4\pi$ for some integer K_i.

2. It then chooses the root vertex state $\mathbf{S}^*$ such that $\mathbf{s}^* = \mathbf{s}$ and $\|\dot{\mathbf{s}}^* - \frac{1}{1+\epsilon}\dot{\mathbf{s}}\| \leq \min_i \mu_i \tau$.

[10] Recall that "∂" denotes "the boundary of".

[11] For historical reasons. See [DX90a,DX90b]. Strictly speaking, they are not bangs and are merely close to being extremal.

3. Let $\mathcal{G}$ be the $(\boldsymbol{\mu}, \tau)$-reachability graph rooted at $\mathbf{S}^*$ whose edges are $(1-\epsilon)\delta_v(c_0, c_1)$-safe $(\boldsymbol{\mu}, \tau)$-bangs. The algorithm searches for the shortest path from $\mathbf{S}^*$ to any vertex that is within (η_x, η_v) of $(\mathbf{g}, \frac{1}{1+\epsilon}\dot{\mathbf{g}})$. η_x, and η_v are both $\Omega(\epsilon)$. The search is done breadth-first.

4. If a $\delta_v(c_0, c_1)$-safe solution exists, then the algorithm returns the trajectory that corresponds to this path in the reachability graph.

4.2.2 Key Concepts

4.2.2.1 Overview

The *Safe Tracking Lemma* (2.2.3) from Section 2.2.3 relates c_0, c_1, and ϵ to a family of tracking tolerances. Specifically, given c_0, c_1 and ϵ, the lemma provides a set of tracking tolerances (η_x, η_v) such that if a trajectory Γ_a is $\delta_v(c_0, c_1)$-safe and Γ_b tracks Γ_a to tolerance (η_x, η_v), then Γ_b will be $(1-\epsilon)\delta_v(c_0, c_1)$-safe. This lemma is briefly presented again in Section 4.2.2.2, and it ensures that the algorithms can choose sufficiently small ϵ-safe tracking tolerances η_x and η_v that are $\Omega(\epsilon)$.

Recall that for a trajectory Γ, Γ' denotes Γ ϵ-time-rescaled, as in (4.5). To show the correctness of each algorithm we show that if a given problem has an optimal solution Γ_{opt} taking time T_{opt}, then the algorithm's choice of root vertex state $\mathbf{S}^*$, timestep τ, and discretization $\boldsymbol{\mu}$ (or $\overline{\mu}$) guarantee it will at worst find a graph trajectory Γ_q that tracks Γ'_{opt} to tolerance (η_x, η_v). Γ_q will therefore be $(1-\epsilon)\delta_v(c_0, c_1)$-safe and approximate $\mathbf{G}'$ to within $O(\epsilon)$ at time $(1+\epsilon)T_{opt}$. Since $\epsilon < 1$, $\mathbf{S}'$ and $\mathbf{G}'$ are within $O(\epsilon)$ of $\mathbf{S}$ and $\mathbf{G}$. It follows that $\Gamma_q(0)$ and $\Gamma_q((1+\epsilon)T_{opt})$ will also be within $O(\epsilon)$ of $\mathbf{S}$ and $\mathbf{G}$.

Following a general analysis similar to that found in the earlier chapters, we first observe that if a Cartesian robot trajectory Γ_r respects L_2 dynamics bounds $\overline{a}$ and $\overline{v}$, then the ϵ-time-rescaled trajectory Γ'_r will respect bounds $\overline{a}'$ and $\overline{v}'$ that are smaller than $\overline{a}$ and $\overline{v}$ by an $\Omega(\epsilon)$ factor. This is also true for open-chain manipulators as long as the force bound $\overline{\mathbf{f}}$ is great enough to overcome gravity in all configurations when the robot is stationary. In Sections 4.2.2.3 and 4.2.2.4 we generalize from feasible instantaneous accelerations to *state-dependent sets of acceleration functions* and formalize an appropriate notion of *acceleration advantage* over a time interval.

The *Coupled Tracking Lemma* (4.2.1) presented in Section 4.2.2.5 relates such an acceleration advantage κ_l, a maximum acceleration, and a desired tracking tolerance to a timestep τ. Using this lemma we can choose $\Omega(\epsilon)$ parameters μ_i, $\overline{\mu}$, and τ that guarantee the existence of graph trajectories that will track Γ'_{opt} to the $\Omega(\epsilon)$ tolerances obtained via the Safe Tracking Lemma. For the Near Extremal Algorithm, it is adequate to find conditions that guarantee that *some* graph trajectory will track an ϵ-time-rescaled optimal trajectory Γ'_{opt} closely enough. In Section 4.2.2.6 we present the *Robust Coupled Tracking Lemma* (4.2.2), which gives conditions ensuring that (b_x, b_v)-pruning during a breadth-first search of the reachability graph will not eliminate trajectories the True-Extremal Algorithm needs to find.

Proofs for Lemmas 4.2.1 and 4.2.2 are presented in Section 4.3.1. Details of how the algorithms choose the reachability graph parameters are given in Sections 4.3.2 and 4.3.3.

4.2.2.2 Safe Tracking

Assuming that a optimal trajectory Γ_{opt} exists, we must determine a tracking tolerance that guarantees that any trajectory tracking Γ'_{opt} to that tolerance will be $(1-\epsilon)\delta_v(c_0, c_1)$-safe. Recall the definition of a δ_v-tube from section 4.1.2, and recall the Safe Tracking Lemma:

Lemma 2.2.3 (The Safe Tracking Lemma) *Suppose that δ_v is specified by c_0 and c_1 and that Γ_r is a δ_v-safe trajectory. Let $0 < \epsilon < 1$, and let $\delta_v{}' = (1-\epsilon)\delta_v$. Then there exists a tolerance (η_x, η_v) such that for any trajectory Γ_q, the following hold:*

1. *If Γ_q tracks Γ_r to tolerance (η_x, η_v), then Γ_q is δ'_v-safe.*
2. *Furthermore, for any positive β, the following choices suffice:*

$$\begin{aligned} \eta_v &\leq \tfrac{c_0 \epsilon}{c_1(1-\epsilon)+\beta} \\ \eta_x &\leq \beta \eta_v. \end{aligned} \tag{2.10}$$

It should be noted again that the lemma is independent of norm, as long as a particular norm is consistently used, and that distances and velocities are in TC. If we wish the safety margin to correspond to the distance between the physical robot and the obstacles in real space,[12] then η_x can be divided by the maximum modulus of the forward kinematic map. Similarly, if we wish the velocity to correspond to the velocity of point on the physical robot, then η_v can be divided

[12]This is distinguished from the margin by which c-space obstacles are avoided, which is more natural to use in our discussion.

by $\max_{\mathbf{p},\dot{\mathbf{p}}} \|\mathbf{J}(\mathbf{p})\dot{\mathbf{p}}\|$, where $\mathbf{J}(\mathbf{p})$ is the manipulator Jacobian. Let us denote these divisors by P_{max} and J_{max}, respectively. Then we would (conservatively) require:

$$\begin{aligned} \eta_v &\leq \tfrac{c_0\epsilon}{J_{max}(c_1(1-\epsilon)+\beta)} \\ \eta_x &\leq \tfrac{\beta c_0\epsilon}{P_{max}(c_1(1-\epsilon)+\beta)}. \end{aligned} \tag{4.12}$$

By choosing $\beta = 1$ we see that η_x and η_v can both be $O(\epsilon)$. The algorithm chooses β to maximize the timestep and thus minimize the overall complexity, as in Section 3.3.4.1.

Given c_0, c_1, and ϵ, we will call a tolerance (η_x, η_v) that obeys (4.12) a physical *ϵ-safe-tracking tolerance.*

4.2.2.3 Time-Rescaling and Instantaneous Acceleration Advantages

For an L_∞ Cartesian robot, if trajectory Γ_r obeys dynamics bounds $\overline{a}$ and $\overline{v}$, then the ϵ-time-rescaled trajectory Γ'_r obeys dynamics bounds $\frac{\overline{a}}{(1+\epsilon)^2}$ and $\frac{\overline{v}}{1+\epsilon}$. For any d-vector σ of 1's and –1's there is a d-vector $\hat{\sigma}$ of 1's and –1's such that if $\|\mathbf{a}(t)\|_\infty \leq \frac{\overline{a}}{(1+\epsilon)^2}$, then for each dimension i

$$\sigma_i(\hat{\sigma}_i\overline{a} - a_i(t)) \geq \frac{\overline{a}(2\epsilon + \epsilon^2)}{(1+\epsilon)^2}.$$

Therefore, a trajectory obeying $(\overline{a}, \tau)$-bang control can always out-accelerate the ϵ-time-rescaled trajectory Γ'_r by an $\frac{\overline{a}(2\epsilon+\epsilon^2)}{(1+\epsilon)^2}$ margin in *any direction over the entirety of a timestep*, as long as this would not violate the velocity bound. We can immediately reduce the analysis for L_∞ Cartesian robots to the one-dimensional case, where we exploit this *acceleration advantage* to relate the timestep τ to how closely some $(\overline{a}, \tau)$-bang trajectory is guaranteed to track Γ'_r (in the absence of obstacles). We will use the acceleration advantage together with the dynamics bounds and Lemma 2.2.3 to choose a timestep τ.

Before we introduce a notion of acceleration advantage appropriate for sets of *acceleration functions*, as used in the Coupled Tracking Lemma, we will describe a generalization of "instantaneous acceleration advantage" to both L_2 Cartesian robots and open-chain manipulators.

Recall that the robot motion obeys the following equation

$$\mathbf{f}(t) = \mathbf{M}(\mathbf{p}(t))\mathbf{a}(t) + [\dot{\mathbf{p}}^T(t)\mathbf{C}(\mathbf{p}(t))\dot{\mathbf{p}}(t)] + \mathbf{G}(\mathbf{p}(t)). \tag{4.1}$$

Suppose a trajectory Γ_r obeys the bounds $\overline{\mathbf{f}}$. Recall that Γ'_r is Γ_r ϵ-time-rescaled (4.5). We find that at time $(1+\epsilon)t$ [Hol83]:

$$\mathbf{f}'_r((1+\epsilon)t) = \frac{\mathbf{f}_r(t)}{(1+\epsilon)^2} + \frac{2\epsilon+\epsilon^2}{(1+\epsilon)^2}\mathbf{G}(\mathbf{p}_r(t)) \tag{4.13}$$

This immediately means that Γ'_r obeys some tighter bounds $\overline{\mathbf{f}}'$ if the robot is Cartesian or if there is no gravity. For an open-chain manipulator, Γ'_r obeys tighter bounds $\overline{\mathbf{f}}'$ as long as the bounds $\overline{\mathbf{f}}$ are large enough for the robot to be held stationary in the presence of gravity in all configurations. In particular, suppose that the forces necessary to balance gravity obey the bound $(1-\kappa_f)\overline{\mathbf{f}}$, so that the scalar κ_f is the "advantage over gravity". Then Γ'_r obeys the bounds

$$\overline{\mathbf{f}}' = \frac{1+(1-\kappa_f)(2\epsilon+\epsilon^2)}{(1+\epsilon)^2}\overline{\mathbf{f}}, \quad \text{in addition to} \tag{4.14}$$

$$\overline{\mathbf{v}}' = \left(\frac{1}{1+\epsilon}\right)\overline{\mathbf{v}}. \tag{4.15}$$

Then, for every degree of freedom i,

$$\overline{f}_i - \overline{f}'_i \geq \frac{\kappa_f(2\epsilon+\epsilon^2)}{(1+\epsilon)^2}\overline{f}_i > \frac{\kappa_f\epsilon}{2}\overline{f}_i, \tag{4.16}$$

Recall that $\mathcal{A}_{\mathbf{X}}$ denotes the sets of feasible instantaneous accelerations at the state $\mathbf{X}$ under the force bound $\overline{\mathbf{f}}$. Let $\mathcal{A}'_{\mathbf{X}}$ denote the corresponding set under the force bounds $\overline{\mathbf{f}}'$, and let $f_{min} = \min_i \overline{f}_i$. Recall that for any $\mathbf{y} \in \Re^d$

$$\|\mathbf{y}\|_\infty \leq \|\mathbf{y}\|_2 \leq \sqrt{d}\|\mathbf{y}\|_\infty. \tag{4.17}$$

Now, because $\mathbf{M}(\mathbf{x})$ is orthogonal (4.1), (4.16) implies that for all $\mathbf{a} \in \mathcal{A}_{\mathbf{X}}$ and $\mathbf{a}' \in \mathcal{A}'_{\mathbf{X}}$,

$$\|\mathbf{a}-\mathbf{a}'\|_2 \geq \frac{\kappa_f f_{min}\epsilon}{2\lambda_{max}}, \tag{4.18}$$

where λ_{max} is the maximum eigenvalue of $\mathbf{M}(\mathbf{x})$ taken over all of C. (4.18) is also true for a Cartesian manipulator obeying L_2-norm dynamics bounds; in this case it follows from (4.3) that $\kappa_f = 1$ and $f_{min} = \overline{f}$. The L_2 acceleration bound case is trivial, since it is equivalent to setting $\mathbf{M}$ to the identity matrix.

In all three cases, $\mathcal{A}'_{\mathbf{X}}$ lies entirely inside $\mathcal{A}_{\mathbf{X}}$ by some $\Omega(\epsilon)$ margin. In particular, we can always find a scalar $\kappa_l > 0$ such that for any $\mathbf{a} \in \mathcal{A}'_{\mathbf{X}}$ and any d-vector σ of 1's and -1's there is an $\hat{\mathbf{a}} \in \partial\mathcal{A}_{\mathbf{X}}$ such that for all degrees of freedom i,

$$\sigma_i(\hat{a}_i - a_i) \geq \kappa_l.$$

Hence, we say that $\mathcal{A}_{\mathbf{X}}$ has a *κ_l-instantaneous acceleration advantage* over $\mathcal{A}'_{\mathbf{X}}$.

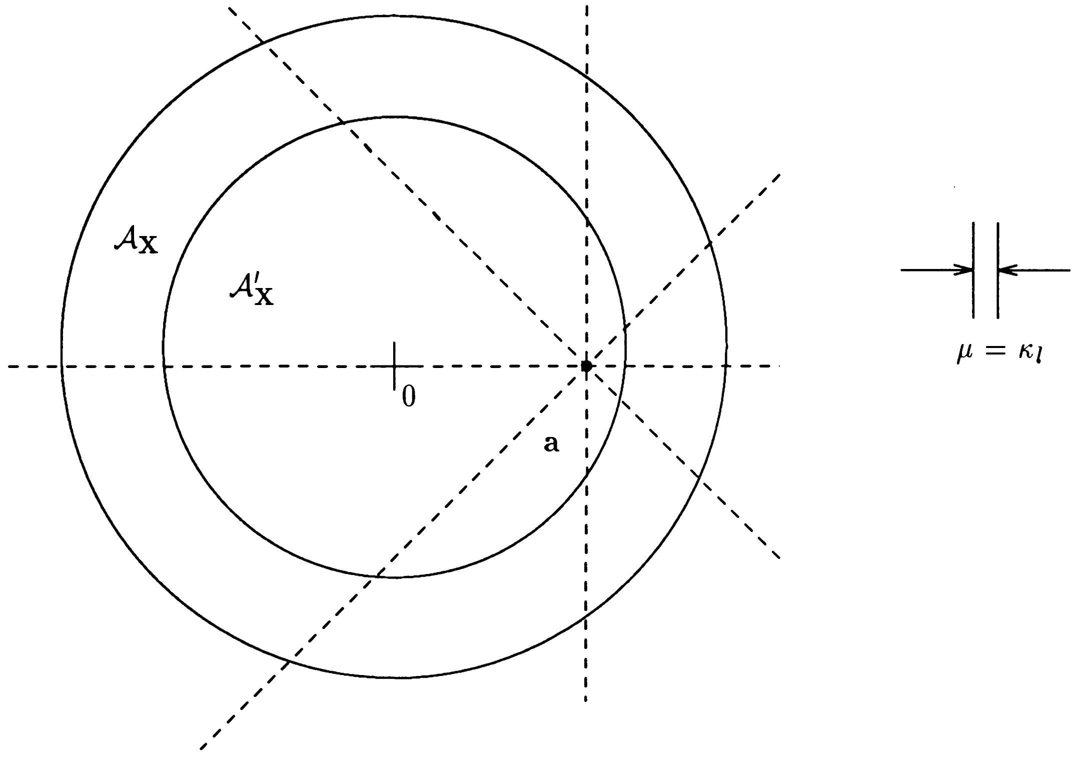


Figure 4.4: Allowable accelerations for the L_2 Cartesian robot at any state $\mathbf{X}$. $\mathcal{A}_{\mathbf{X}}$ obeys $\bar{a}$, $\mathcal{A}'_{\mathbf{X}}$ obeys $\frac{\bar{a}}{(1+\epsilon)^2}$. For any $\mathbf{a} \in \mathcal{A}'_{\mathbf{X}}$, in each direction given by a vector of 1's and -1's, there is some $\hat{\mathbf{a}} \in \mathcal{A}_{\mathbf{X}}$ that has a κ_l advantage over $\mathbf{a}$; the intersections of the dotted lines and the outer circle ($\partial\mathcal{A}_{\mathbf{X}}$) represent "witnesses". Thus, $\mathcal{A}_{\mathbf{X}}$ has a uniform κ_l advantage over $\mathcal{A}'_{\mathbf{X}}$ for any duration τ (as long as the velocity bound is not violated). If the actual uniform advantage is large enough and all μ_i are small enough, then the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations $\mathcal{H}(\mathbf{X}, \tau)$ (see equation (4.11)) has a uniform κ_l advantage over $\mathcal{A}'_{\mathbf{X}}$.

4.2.2.4 Acceleration Functions and Uniform Advantages

We now introduce a form of acceleration advantage useful for comparing sets of acceleration functions. We will say that a *trajectory* Γ_r *respects a set* $\mathcal{Y}$ *of acceleration functions in interval* $[t_1, t_2]$ if there is an acceleration function $\mathbf{a} \in \mathcal{Y}$ such that $\mathbf{a}_r(t) = \mathbf{a}(t - t_1)$ for all $t \in [t_1, t_2]$. Suppose that $\widehat{\mathcal{Y}}$ is also a set of acceleration functions, and suppose that for every $\mathbf{a} \in \mathcal{Y}$ and each d-vector σ of 1's and −1's, there is some $\hat{\mathbf{a}} \in \widehat{\mathcal{Y}}$ such that for all $t \in [0, \tau]$

$$\sigma_i \left(\hat{a}_i(t) - \frac{\int_0^\tau a_i(s)ds}{\tau} \right) \geq \kappa_l. \tag{4.19}$$

Then we say that $\widehat{\mathcal{Y}}$ *has a uniform* κ_l *advantage over* $\mathcal{Y}$ *for duration* τ. (When "for duration τ" is omitted, it is implied.)

Because the set of feasible accelerations for a Cartesian robot is state-invariant, we can find and acceleration advantage κ_l and discretizations μ_i, $\overline{\mu}$ that are $\Omega(\epsilon)$ and such that both the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations $\mathcal{H}(\mathbf{X}, \tau)$ and the set of true $(\overline{\mu}, \tau)$-bangs $\overline{\mathcal{H}}(\mathbf{X}, \tau)$ at every state $\mathbf{X}$ have a uniform κ_l-advantage over the acceleration of $\Gamma'_{opt}(t)$ at every time t for any duration τ (as long as the velocity bound is not violated).

We now generalize the notion of a set of instantaneous accelerations $\mathcal{A}_\mathbf{X}$ to a mapping

$$\mathcal{A} : TC \mapsto \{\mathcal{Y} \mid \mathcal{Y} \textit{ is a set of acceleration functions}\}$$

from TC to sets of acceleration functions for a given robot. $\mathcal{A}(\mathbf{X})$ is defined to be the set of acceleration functions obeying the dynamics bounds $\overline{\mathbf{f}}$ and $\overline{\mathbf{v}}$ for a robot motion (in the absence of obstacles) beginning at state $\mathbf{X}$. Thus, $\mathbf{a}_r \in \mathcal{A}(\mathbf{X})$ if and only if there is a trajectory Γ_r that at state $\mathbf{X}$ and that obeys the dynamics bounds. Furthermore, instantneous accleration[13] $\mathbf{a}^\mathbf{X} \in \mathcal{A}_\mathbf{X}$ if and only if there is some acceleration function $\mathbf{a} \in \mathcal{A}(\mathbf{X})$ such that $\mathbf{a}^\mathbf{X} = \mathbf{a}(0)$. Strictly speaking, $\mathcal{H}(\mathbf{X}, \tau) \subset \mathrm{A}(\mathbf{X})$, and for each $\mathbf{a} \in \mathcal{H}(\mathbf{X}, \tau)$ there is a constant $\mathbf{a}^\mathbf{X} \in \mathcal{A}_X$ such that $\mathbf{a}(t) = \mathbf{a}^\mathbf{X}$ for all times $t \in [0, \tau]$.

We will say that Γ_r *respects* $\mathcal{A}$ if for all intervals $[t_1, t_2]$, $\Gamma_r(t_1) = \mathbf{X}$ implies that Γ_r respects $\mathcal{A}(\mathbf{X})$ in $[t_1, t_2]$. $\mathcal{A}'$ will denote the map from TC to all acceleration functions feasible under the bounds $\overline{\mathbf{f}}'$ and $\overline{\mathbf{v}}'$ given in (4.14) and (4.15).

Since the set of feasible instantaneous accelerations for an open-chain manipulator is state-dependent, to choose the discretization parameter $\overline{\mu}$ or $\boldsymbol{\mu}$, and a

[13] Recall that an instantaneous acceleration is a vector, not a function.

timestep τ, we consider the relationship between feasible acceleration functions (such as the members of $\mathcal{A}(\mathbf{X})$) and feasible sets of instantaneous accelerations (such as $\mathcal{A}_X$). Suppose Γ_r respects $\mathcal{A}'$. Then, the average acceleration of Γ_r over the interval $[n\tau, (n+1)\tau]$

$$\frac{\mathbf{v}_r((n+1)\tau) - \mathbf{v}_r(n\tau)}{\tau} \in \bigcup_{t \in [n\tau,(n+1)\tau]} \mathcal{A}'_{\Gamma_r(t)}. \tag{4.20}$$

Suppose that Γ_q is a $(\boldsymbol{\mu}, \tau)$-bang trajectory. Then for all $t \in [n\tau, (n+1)\tau]$, $\mathbf{a}_q(t) = \mathbf{a}^{(n)}$ for some constant $\mathbf{a}^{(n)}$ such that for all i, μ_i divides $a_i^{(n)}$, and

$$\mathbf{a}^{(n)} \in \bigcap_{t \in [n\tau,(n+1)\tau]} \mathcal{A}_{\Gamma_q(t)}. \tag{4.21}$$

If $\Gamma_q(n\tau)$ and $\Gamma_r(n\tau)$ are close enough and τ is small enough, then

$$\bigcup_{t \in [n\tau,(n+1)\tau]} \mathcal{A}'_{\Gamma_r(t)} \subset \bigcap_{t \in [n\tau,(n+1)\tau]} \mathcal{A}_{\Gamma_q(t)} \subset \widehat{\mathcal{A}}^*_{\Gamma_q(0)}, \tag{4.22}$$

and furthermore, the boundaries of these two sets are separated by some distance. (Recall equation (4.9).)

If this separation is great enough compared to $\|\boldsymbol{\mu}\|_\infty$ and $\overline{\mu}$, then both the $(\boldsymbol{\mu}, \tau)$-extremal shell at $\Gamma_q(n\tau)$ and the set of true $(\overline{\mu}, \tau)$-bangs at $\Gamma_q(n\tau)$ will have a uniform κ_l advantages over $\mathcal{A}'(\Gamma_r(n\tau))$. We will later use this relationship between sets of instantaneous accelerations as a criterion for comparing maps from TC to sets of acceleration functions.

Since we can calculate the minimal distance between members of $\partial\mathcal{A}_\mathbf{X}$ and members of $\partial\mathcal{A}'_\mathbf{X}$, we now consider how points on $\partial\mathcal{A}_\mathbf{X}$ and $\partial\mathcal{A}'_\mathbf{X}$ "move" in response to small, continuously realizable changes, or *perturbations*, to state $\mathbf{X}$. (See figure 4.5. It is easily shown that (physically possible) state perturbations $(\Delta\mathbf{x}, \Delta\mathbf{v})$ cause points on $\partial\mathcal{A}_\mathbf{X}$ and $\partial\mathcal{A}'_\mathbf{X}$ to move by distances that are $O(\Delta\mathbf{x})$ and $O(\Delta\mathbf{v})$. Therefore, because we assume global dynamics bounds, over duration Δt, points on $\partial\mathcal{A}_{\Gamma_r(t)}$ move by $O(\Delta t)$. Recalling (4.14), this means that we can find $\Omega(\epsilon)$ uniform acceleration advantage κ_l, discretization $\boldsymbol{\mu}$, fundamental timestep[14] τ_0, and maximal tracking tolerances η_{x0} and η_{v0} such that if $\mathbf{X}$ is within (η_{x0}, η_{v0}) of $\mathbf{Y}$, then the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations at $\mathbf{X}$, $\mathcal{H}(\mathbf{X}, \tau)$, has a uniform κ_l advantage over $\mathcal{A}(\mathbf{Y})$ for duration $\min(\tau, \tau_0)$. Following similar steps, we can choose $\overline{\mu}$ to guarantee that the set of true $(\overline{\mu}, \tau)$-bangs at $\mathbf{X}$, $\overline{\mathcal{H}}(\mathbf{X}, \tau)$, also has a uniform κ_l advantage.

---

[14]This is the maximum timestep that guarantees a *some* $(\boldsymbol{\mu}, \tau)$-bang trajectory will be able to stay within *some* constant tracking tolerance of an ϵ-time-rescaled trajectory.

4.2.2.5 The Coupled Tracking Lemma

The Coupled Tracking Lemma is the key to choosing, for a given tracking tolerance, a timestep τ and other paramenters that guarantees the following: for any ϵ-time-rescaled trajectory Γ'_r beginning at $\mathbf{S}'$, *some* $(\boldsymbol{\mu}, \tau)$-bang trajectory and *some* true $(\overline{\mu}, \tau)$-bang trajectory will track Γ'_r to that tolerance. Note that the set of acceleration functions associated with a given state can be arbitrary, so long as it has the necessary uniform acceleration advantage. This allows us to apply the lemma to sets of true $(\overline{\mu}, \tau)$-bangs without knowing their forms as path functions.[15] More generally, the lemma can be applied to state-dependent sets of bounded acceleration functions whose exact form is unknown but whose time-derivatives are also bounded.

Lemma 4.2.1 (The Coupled Tracking Lemma) *Let A_{max} be the global L_∞ acceleration bound. Consider functions $\mathcal{P}, \mathcal{Q} : TC \to \{\mathcal{Y} : \mathcal{Y}$ is a set of acceleration functions$\}$. Let $\mathcal{P}$ have the property that if for some state $\mathbf{X}$, $\mathbf{a} \in \mathcal{P}(\mathbf{X})$, then for all times t, $|\mathbf{a}(t)|_\infty \leq A_{max} - \kappa_l$. Let $\mathcal{Q}$ have the property that if acceleration function $\mathbf{a} \in \mathcal{Q}(\mathbf{X})$, then for all times t, $|\mathbf{a}(t)|_\infty \leq A_{max}$. Let $\Gamma_r(t)$ respect $\mathcal{P}$.*

Suppose that there exist a maximal tracking tolerance (η_{x0}, η_{v0}) and a fundamental timestep τ_0 such that for all $\tau \leq \tau_0$ the following is true: if state $\mathbf{X}$ is within (η_{x0}, η_{v0}) of state $\mathbf{Y}$, then the set of acceleration functions $\mathcal{Q}(\mathbf{X})$ has a uniform κ_l advantage over $\mathcal{P}(\mathbf{Y})$ for duration τ.

Then, for any L_∞ tracking tolerance (η_x, η_v), there exist a timestep τ and a velocity $\mathbf{v}_0$ such that if trajectory Γ_r respects $\mathcal{P}$ and $\|\dot{\mathbf{p}}_r(0) - \mathbf{v}_0\|_\infty \leq 2A_{max}\tau$, then there exists a trajectory Γ_q such that

1. $\mathbf{v}_q(0) = \mathbf{v}_0$;
2. $\mathbf{p}_r(0) = \mathbf{p}_q(0)$;
3. *Γ_q tracks Γ_r to tolerance (η_x, η_y); and*
4. *For each n, Γ_q respects $\mathcal{Q}(\Gamma_q(n\tau))$ in $[n\tau, (n+1)\tau]$.*

Moreover, it is sufficient that

$$\begin{aligned} \tau &\leq \tau_0; \\ \tau &\leq \sqrt{\tfrac{\min(\eta_x, \eta_{x0})\kappa_l}{A_{max}(8A_{max}+6\kappa_l)}}; \quad \textit{and} \\ \tau &\leq \tfrac{\min(\eta_v, \eta_{v0})}{4A_{max}}. \end{aligned} \tag{4.23}$$

[15] *I.e.*, we don't need a series or closed form representation.

Proof: Presented in Section 4.3.1.

Recall from Section 4.2.2.2 that there is a ϵ-safe tracking tolerance (η_x, η_v) where both η_x and η_v are $\Omega(\epsilon)$. The Near Extremal algorithm uses the Coupled Tracking Lemma (above) to choose a timestep τ that is $\Omega(\epsilon)$ and that guarantees that if a solution Γ_{opt} exists, there will be a $(\boldsymbol{\mu}, \tau)$-graph trajectory that tracks Γ'_{opt} to tolerance (η_x, η_v). The True Extremal Algorithm can choose τ similarly, but because of the search pruning, this alone will not guarantee it will *find* the required trajectory.

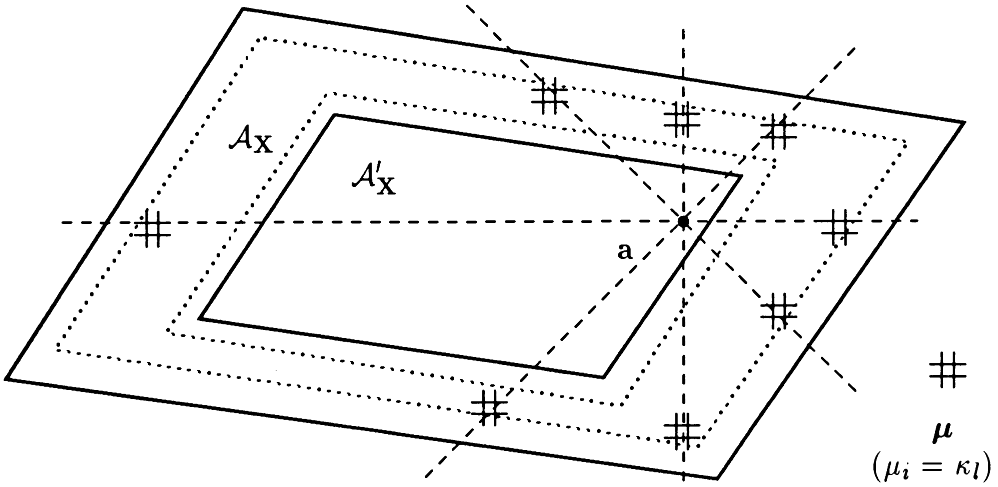


Figure 4.5: Sets in acceleration space. $\mathcal{A}_{\mathbf{X}}$ obeys bound $\overline{\mathbf{f}}$; $\mathcal{A}'_{\mathbf{X}}$ obeys $\overline{\mathbf{f}}'$. The outer dotted parallelogram represents the boundary of the set obtained by considering all "perturbations" of $\mathcal{A}_{\mathbf{X}}$ by up to $h_{q1}(\eta_x, \eta_v) + h_{q0}(\tau)$ and taking the intersection. The intersection of this set with the $\boldsymbol{\mu}$-grid has a κ_l advantage over the set (represented by the inner dotted parallelogram) obtained by considering all "perturbations" of $\mathcal{A}'_{\mathbf{X}}$ by up to $h_r(\tau)$ and taking the union. The eight μ-grid patches crossed by dashed lines show this κ_l advantage with respect to an acceleration (vector) **a**.

4.2.2.6 Coupled Tracking and (b_x, b_v)-Bucket Pruning

In the True-Extremal Algorithm the kinodynamic constraints, ϵ, and the choices of discretization parameter $\overline{\mu}$, timestep τ, and root vertex $\mathbf{S}^*$ determine a reachability graph $\mathcal{G}$ whose vertices are states and whose edges correspond to $(1-\epsilon)\delta_v(c_0, c_1)$-safe $(\overline{\mu}, \tau)$-bangs.

Recall that in (b_x, b_v)-bucket pruning, TC is divided into voxels with diameter b_x in spatial dimensions and b_v in velocity dimensions. A breadth-first search with (b_x, b_v)-bucket pruning procedes as a normal breadth-first search does, except that when the search finds a vertex in a voxel in which another vertex has previously been found[16], the edges out of the newly found vertex will be pruned from the search, *i.e.*, no edges out of that vertex will be explored.

Consider any breadth-first search of $\mathcal{G}$ with (b_x, b_v)-bucket pruning. We call removing all edges and vertices in $\mathcal{G}$ that are not explored during this search *a breadth-first* (b_x, b_v)*-bucket pruning of* $\mathcal{G}$. We say that a graph-trajectory Γ *remains* after that breadth-first (b_x, b_v)-bucket pruning if the path in $\mathcal{G}$ corresponding to it is not affected the pruning.

Like the Coupled Tracking Lemma, the Robust Coupled Tracking Lemma applies to sets of acceleration functions more general than $(\overline{\mu}, \tau)$-bangs and $(\boldsymbol{\mu}, \tau)$-bangs. We introduce three new terms used in its statement. Let $\mathcal{Q}$ be a mapping $\mathcal{Q} : TC \rightarrow \{\mathcal{Y} : \mathcal{Y}$ *is a finite set of acceleration functions*$\}$. We say that a trajectory segment from state $\mathbf{X}$ to state $\mathbf{Y}$ *is a* $(\mathcal{Q}, \tau)$*-trajectory segment* if there is a trajectory Γ_r such that $\Gamma_r(0) = \mathbf{X}$, $\Gamma_r(\tau) = \mathbf{Y}$, and $\mathbf{a}_r \in \mathcal{Q}(\mathbf{X})$. A vertex $\mathbf{S}^*$ and the set of $(\mathcal{Q}, \tau)$-trajectory segments determine the $(\mathcal{Q}, \tau)$*-reachability graph* $\mathcal{G}$ rooted at $\mathbf{S}^*$. If $\mathcal{R}$ is a subset of TC, then the *maximal subgraph of* $\mathcal{G}$ *lying in* $\mathcal{R}$ is the maximal subgraph $\mathcal{G}'$ of $\mathcal{G}$ such that all vertices and all edges (as trajectory segments) in $\mathcal{G}'$ lie in $\mathcal{R}$.

Lemma 4.2.2 (The Robust Coupled Tracking Lemma) *Consider functions* $\mathcal{P}, \mathcal{Q} : TC \rightarrow \{\mathcal{Y} : \mathcal{Y}$ is a set of acceleration functions$\}$. *Let* $\mathcal{P}$ *have the property that if* $\mathbf{a} \in \mathcal{P}(\mathbf{X})$ *for some state* $\mathbf{X}$*, then for all times* t, $|\mathbf{a}(t)|_\infty \leq A_{max} - \kappa_l$. *Let* $\mathcal{Q}$ *have the properties that* $\mathcal{Q}(\mathbf{X})$ *is finite for all* $\mathbf{X} \in TC$ *and that if* $\mathbf{a} \in \mathcal{Q}(\mathbf{X})$, *then for all times* t $\|\mathbf{a}(t)\|_\infty \leq A_{max}$.

Suppose that there exist a maximal tracking tolerance (η_{x0}, η_{v0}) *and a fundamental timestep* τ_0 *such that for all* $\tau \leq \tau_0$ *the following is true: if state* $\mathbf{X}$ *is within* (η_{x0}, η_{v0}) *of state* $\mathbf{Y}$*, then the set of acceleration functions* $\mathcal{Q}(X)$ *has a uniform* κ_l *advantage*[17] *over* $\mathcal{P}(Y)$ *for duration* τ.

[16]By previously, we mean in a previous generation or earlier in the current generation.
[17]Recall equation (4.19).

Let (η_x, η_v) be an L_∞ tracking tolerance, and let timestep τ obey the following inequalities:

$$\begin{aligned} \tau &\leq \tau_0; \\ \tau &\leq \sqrt{\frac{4\min(\eta_{x0},\eta_x)\kappa_l}{24A_{max}\kappa_l + 2\kappa_l^2 + (8A_{max}+\kappa_l)^2}}; \quad \text{and} \\ \tau &\leq \frac{2\min(\eta_{v0},\eta_v)}{8A_{max}+\kappa_l}. \end{aligned} \tag{4.24}$$

Let be $\mathcal{G}$ the $(\mathcal{Q}, \tau)$-reachability graph rooted at some $\mathbf{S}^*$.

Now, suppose that Γ_r respects $\mathcal{P}$, $\Gamma_r(0)$ is within $2A_{max}\tau$ of $\dot{\mathbf{s}}^*$, and $\mathbf{p}_r(0) = \mathbf{s}^*$. Suppose that $\mathcal{R}$ is a subset of TC containing the (η_x, η_v)-tube induced by Γ_r, and that $\mathcal{G}'$ results from some breadth-first $(\frac{\kappa_l \tau^2}{4}, \frac{\kappa_l \tau}{2})$-pruning of the maximal subgraph of $\mathcal{G}$ that lies in $\mathcal{R}$.

Then there is a Γ_q in $\mathcal{G}'$ such that $T_q \leq T_r$, $\Gamma_q(0) = \mathbf{S}^*$, and $\Gamma_q(T_q)$ approximates $\Gamma_r(T_r)$ to tolerance (η_x, η_v).

Proof: Presented in Section 4.3.1.

Having chosen $\overline{\mu}$ as described in Section 4.2.2.4 (and in detail in 4.3.2 and 4.3.3), the True Extremal algorithm uses Lemma 4.2.2 to choose a timestep τ that is $\Omega(\epsilon)$ and pruning bucket dimensions b_x and b_v that are $\Omega(\epsilon^3)$ and $\Omega(\epsilon^2)$. The parameters chosen will ensure that if a δ_v-safe solution Γ_{opt} exists, there exist a $(\overline{\mu}, \tau)$-true-bang graph trajectory that tracks Γ'_{opt} to the ϵ-safe tracking tolerance (η_x, η_v).

4.2.2.7 Asymptotic Bounds

In the preceding sections we have described the key concepts towards asymptotic bounds for choosing parameters that guarantee the algorithms will find an ϵ-optimal solution when a solution exists. Given a problem and an approximation parameter ϵ, the Safe Tracking Lemma shows how to find a family of ϵ-safe-tracking tolerances (η_x, η_v) such that η_x and η_v are $\Omega(\epsilon)$. Sections 4.2.2.3 and 4.2.2.4 sketch why there will be sufficiently small discretization parameters $\overline{\mu}$ and μ_i and a uniform acceleration advantage κ_l that are $\Omega(\epsilon)$. For open chain manipulators we find η_{x0}, η_{v0}, and τ_0 that are $\Omega(\epsilon)$. The lemmas in Sections 4.2.1 and 4.2.2 indicate that there is a correct choice of τ that is $\Omega(\epsilon)$. Section 4.2.2 implies that since κ_l and τ are both $\Omega(\epsilon)$, we can choose (b_x, b_v)-pruning bucket dimensions that are $\Omega(\epsilon^3)$ and $\Omega(\epsilon^2)$ respectively.

It follows that for a given problem, the number of (b_x, b_v)-pruning buckets is $O((\frac{1}{\epsilon})^5)$. Since $\overline{\mu}$ and the μ_i are $\Omega(\epsilon)$, the out-degrees of the graphs are $O((\frac{1}{\epsilon})^{d-1})$.

The complexity each algorithm, then, is $O(p(N, \epsilon, d)(\frac{1}{\epsilon})^{6d-1})$, where $p(N, \epsilon, d)$ is the cost of checking the $(1-\epsilon)\delta_v(c_0, c_1)$-safety of a $(\overline{\mu}, \tau)$-bang or a $(\boldsymbol{\mu}, \tau)$-bang, depending on the algorithm. In Section 4.3.4 we review techniques for safety checking that that will have a cost of $O(N(d + \log N)$ per bang in the asymptotic case. Our algorithms therefore have an overall asymptotic complexity bound of $O(N(d + \log N)(\frac{1}{\epsilon})^{6d-1})$.

4.3 Deriving Bounds

We first present proofs of the Coupled Tracking Lemma (4.2.1) and the Robust Coupled Tracking Lemma (4.2.2). These two lemmas are fundamental to obtaining our results. We use them to show how the algorithm can choose the discretization parameters $\boldsymbol{\mu}$ and $\overline{\mu}$ and the timestep τ, and how we can calculate the uniform acceleration advantage κ_l. We then briefly review the algorithmic complexity of appropriate techniques that can be used for checking whether a $(\boldsymbol{\mu}, \tau)$-bang or $(\overline{\mu}, \tau)$-bang respects $(1-\epsilon)\delta_v(c_0, c_1)$ safety.

4.3.1 Proving the Key Lemmas

If a system has decoupled dynamics bounds, the acceleration or velocity chosen along any one axis never affects what accelerations or velocities are possible along other any other axis. This is why tracking the trajectory of a Cartesian robot obeying L_∞ bounds reduces trivially to the one-dimensional case. On the other hand, if a system has coupled dynamics bounds, choosing a maximal acceleration along one axis can limit the accelerations along another. Furthermore, if the dynamics equations are state dependent, then the set of instantaneous extremal accelerations is state-dependent. The definition of a uniform κ_l acceleration advantage over a timestep is thus crucial to our proofs of Lemmas 4.2.1 and 4.2.2.

To prove these two lemmas, we begin by considering a game against an adversary in a one-dimensional space. In this game certain rules simulate dynamics. When simultaneous independent games are played, the adversary can force our movements to be governed by accelerations in the $(\boldsymbol{\mu}, \tau)$-extremal shell or by true $(\overline{\mu}, \tau)$-bangs.

4.3.1.1 The Tracking Game

Consider a game in which you are trying to track an adversary in a one dimensional space. The game is a series of rounds, each of which simulates motions taking duration τ. The game begins with adversary in state (x_0, v_0) and you in state (x'_0, v'_0). A round simulates an interval of time with length τ.

Let game parameters A_{max} and κ_l such that $A_{max} > \kappa_l > 0$ be given. When discussing the game, we will use x_n, v_n, etc, to denote game variables that correspond to round n.

In each round the following takes place:

Play for Round n:

1. The adversary plays first by choosing an acceleration function $a_n(t)$ for himself such that for all $t \in [n\tau, (n+1)\tau]$, $|a_n(t)| \leq A_{max} - \kappa_l$. His state (x_{n+1}, v_{n+1}) at the end of the round (beginning of round $n+1$) is computed from his state (x_n, v_n) and this acceleration function by integration over the time interval $[n\tau, (n+1)\tau]$.

2. For your play in round n, you then choose either HIGH and LOW. This choice limits the acceleration function that the adversary can choose *for you*, and that determines your state (x'_{n+1}, v'_{n+1}) at the end of the round. Note that you play only HIGH or LOW; the adversary chooses *your* acceleration *as well as his own.* However, his choice of your acceleration is constrained by your play.

3. Let $a_n^\circ = \frac{v_{n+1} - v_n}{\tau}$. If you played HIGH, then your adversary can choose any function $a'_n(t)$ such that $a'_n(t) \geq a_n^\circ + \kappa_l$ for all $t \in [n\tau, (n+1)\tau]$. If instead you played LOW, then your adversary's choice must obey the condition $a'_n(t) \leq a_n^\circ - \kappa_l$ for all $t \in [n\tau, (n+1)\tau]$. In both cases, $a'_n(t)$ must obey the condition $|a'_n(t)| \leq A_{max}$ for all $t \in [n\tau, (n+1)\tau]$.

4. Your state (x'_{n+1}, v'_{n+1}) at the end of the round is computed using your state at the end of the previous round and integrating $a'_n(t)$ over the time interval $[n\tau, (n+1)\tau]$.

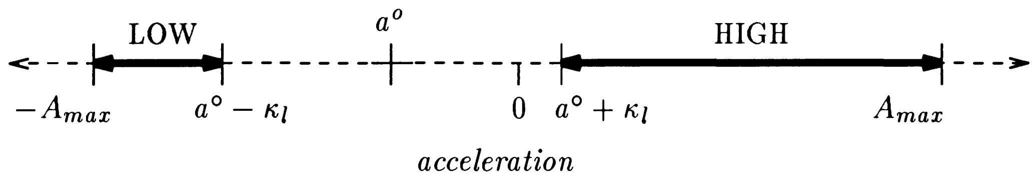


Figure 4.6: Details from a game against an adversary. Suppose in the current round the adversary's average acceleration is a_n°. Then, if you choose HIGH, the adversary assigns you an acceleration function whose range lies in $[a^\circ + \kappa_l, A_{max}]$; if you choose LOW, the adversary assigns you an acceleration function whose range lies in $[-A_{max}, a^\circ - \kappa_l]$. This function controls your motion for the current round, which covers a time interval τ.

4.3.1.2 A Winning Simple Strategy for the Tracking Game

A simple high-level strategy for the Tracking Game is: try to go faster than the adversary if you have fallen behind; try to go slower than the adversary if you have gotten ahead; but never go much faster or much slower than the adversary. The uniform κ_l advantage you have over the adversary's average acceleration during a round assures that we can follow such a strategy. (See fig. 4.7.)

We now give the detailed strategy in terms of the game parameters.

Simple Strategy

In round n choose HIGH or LOW according to the following rules.

1. If $|x_n - x'_n| \leq 2A_{max}\tau^2$, then choose HIGH if $v_{n+1} \geq v'_n$, LOW otherwise.

2. Else if $x_n - x'_n > 2A_{max}\tau^2$, then choose HIGH if $v'_n \leq v_{n+1} + 2A_{max}\tau$, LOW otherwise.

3. Else choose LOW if $v'_n \geq v_{n+1} - 2A_{max}\tau$, HIGH otherwise.

We make the following claim:

Claim 4.3.1 *Suppose that you play the Tracking Game against an adversary and follow the* Simple Strategy *above, and suppose that* $|x_0 - x'_0| \leq 2A_{max}\tau^2$ *and* $|v_0 - v'_0| \leq 2A_{max}\tau$

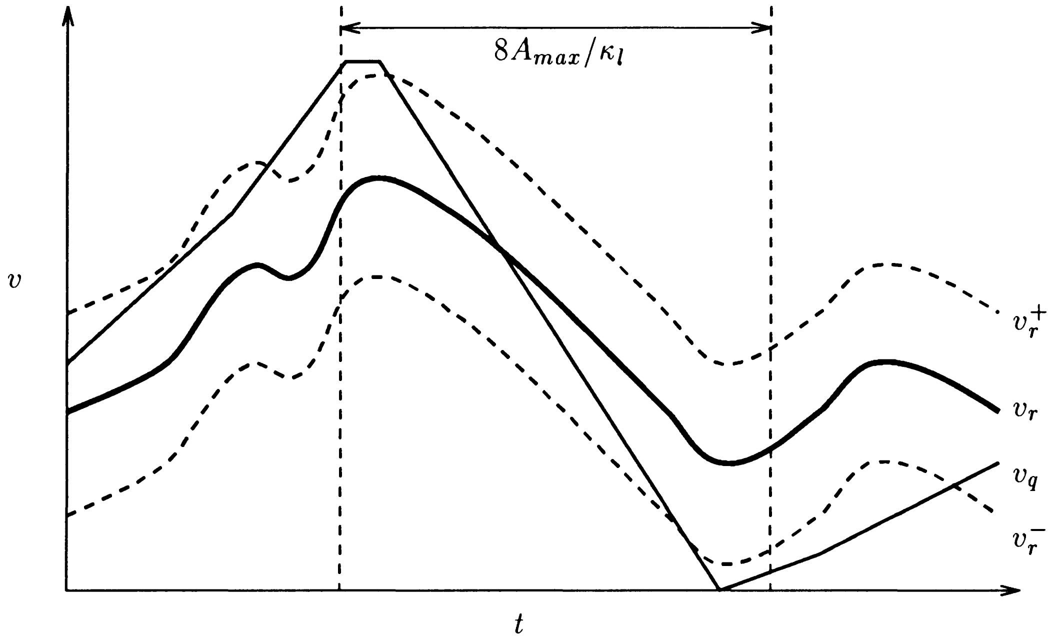


Figure 4.7: Intuition for proving Lemma 4.2.1: a game in 1D. We try to track an adversary whose acceleration a_r obeys the condition $|a_r(t)| \leq A_{max} - \kappa_l$. We generate a our velocity function v_q timestep by timestep, trying to limit $|v_q(t) - v_r(t)|$ and $|\int(v_r(t)-v_q(t))dt|$. During each round we can only choose whether our acceleration $a_q(t)$ is above or below the adversary's average acceleration over the timestep by κ_l. The adversary otherwise controls our acceleration, except for the restriction that $|a_q(t)| \leq A_{max}$. Mimicking the adversary's velocity v_r to within $2A_{max}\tau$ is straightforward, but in making up for position error, our strategy can result in a velocity error of $4A_{max}\tau$; $v_r^+ = v_r + 4A_{max}\tau$ and $v_r^- = v_r - 4A_{max}\tau$ in the figure. Following a good strategy will keep $|\int(v_r(t) - v_q(t))dt|$ within a constant bound dependent on τ. The game conditions give us a uniform κ_l acceleration advantage over each timestep.

Let two trajectories Γ_r and Γ_q be defined as follows:

$$\begin{array}{rcll} \Gamma_r(0) & = & (x_0, v_0); & \\ a_r(t) & = & a_n(t - n\tau) & \textit{for all} \quad t \in [n\tau, (n+1)\tau); \\ \Gamma_q(0) & = & (x'_0, v'_0); & \\ a_q(t) & = & a'_n(t - n\tau) & \textit{for all} \quad t \in [n\tau, (n+1)\tau). \end{array}$$

Then for all t:

$$\begin{array}{rcl} |v_r(t) - v_q(t)| & \leq & 4A_{max}\tau; \\ |x_r(t) - x_q(t)| & < & \frac{8A_{max}\tau^2}{\kappa_l} + 6A_{max}\tau^2. \end{array} \tag{4.25}$$

Proof: Suppose the hypotheses hold. (See figure 4.7.)

The velocity condition in (4.25) holds by inspection, since $|a_r(t) - a_q(t)| \leq 2A_{max}$ for all t.

Let us now define

$$\begin{array}{rcl} x_{err}(t) & = & x_r(t) - x_q(t); \\ v_{err}(t) & = & v_r(t) - v_q(t). \end{array}$$

To obtain a bound on $|x_r(t) - x_q(t)|$, we first bound the length of an interval for which x_{err} and v_{err} can have the same non-zero sign. Suppose that $|x_{err}(n\tau)| > 2A_{max}\tau^2$ and that $v_{err} > 0$. Then following the strategy ensures that $v_{err}(t_c) = 0$ at some least time $t_c > n\tau$, and in particular, $t_c \leq n\tau + \frac{4A_{max}\tau}{\kappa_l}$. Then

$$|x_{err}(t_c) - x_{err}(n\tau)| \leq \frac{8A_{max}^2\tau^2}{\kappa_l}.$$

The velocity condition ensures that if $|x_{err}(n\tau)| \leq 2A_{max}\tau^2$ but $|x_{err}((n+1)\tau)| > 2A_{max}\tau^2$ then we know that

$$|x_{err}((n+1)\tau)| < 6A_{max}\tau^2.$$

Therefore,

$$\max_t |x_{err}(t)| < \frac{8A_{max}^2\tau^2}{\kappa_l} + 6A_{max}\tau^2. \tag{4.26}$$

□

Simple arithmetic manipulation shows the following corollary.

Corollary 4.3.2 *The following conditions together guarantee that in the Tracking Game the adversary can be tracked to tolerance of* (η_x, η_v)*:*

$$\begin{aligned} |x_0 - x_0'| &\leq 2A_{max}\tau^2; && (4.27) \\ |v_0 - v_0'| &\leq 2A_{max}\tau; \quad \textit{and} && (4.28) \\ \tau &\leq \min\left(\sqrt{\frac{\eta_x \kappa_l}{A_{max}(8A_{max} + 6\kappa_l)}}, \frac{\eta_v}{4A_{max}}\right). && (4.29) \end{aligned}$$

4.3.1.3 Applying the Game to Prove Lemma 4.2.1

We now use facts proven about the Tracking Game to obtain a proof of Lemma 4.2.1.

Proof (of Lemma 4.2.1): Let $\mathcal{P}$ and $\mathcal{Q}$ be functions $\mathcal{P}, \mathcal{Q} : TC \to \{\mathcal{Y} : \mathcal{Y}$ *is a set of acceleration functions*$\}$. Let $\mathcal{P}$ have the property that if $\mathbf{X} \in TC$ and $\mathbf{a} \in \mathcal{P}(\mathbf{X})$, then for all times t, $|a(t)| \leq A_{max} - \kappa_l$. Let $\mathcal{Q}$ have the property that if $\mathbf{X} \in TC$ and $\mathbf{a} \in \mathcal{Q}(\mathbf{X})$, then for all times t, $|a(t)| \leq A_{max}$. Let $\Gamma_r(t)$ respect $\mathcal{P}$.

Suppose that there exist a maximal tracking tolerance (η_{x0}, η_{v0}) and fundamental timestep τ_0 such that for all $\tau \leq \tau_0$ the following is true: (a) if state $\mathbf{X}$ is within (η_{x0}, η_{v0}) of state $\mathbf{Y}$, then the set of acceleration functions $\mathcal{Q}(\mathbf{X})$ has a uniform κ_l advantage over $\mathcal{P}(\mathbf{Y})$ for duration τ.

Suppose that (η_x, η_y) is given, and that τ obeys the conditions

$$\begin{aligned} \tau &\leq \tau_0; \\ \tau &\leq \sqrt{\tfrac{\min(\eta_x, \eta_{x0})\kappa_l}{A_{max}(8+6\kappa_l)A_{max}}}; \quad \text{and} \qquad (4.30) \\ \tau &\leq \tfrac{\min(\eta_v, \eta_{v0})}{4A_{max}}. \end{aligned}$$

Consider d simultaneous independent playings of the Tracking Game in which your adversary's trajectory in the i^{th} game is $\Gamma_{r,i}$ and yours is $\Gamma_{q,i}$, and in which $\Gamma_{r,i}(0)$ and $\Gamma_{q,i}(0)$ meet the starting conditions ($t = 0$) of Claim 4.3.1.

We consider play during round n, assuming that you have followed the Simple Strategy and play has proceeded legally through the end of the previous round. By Claim 4.3.1, in each game i, $\Gamma_{q,i}(n\tau)$ is within (η_{x0}, η_{v0}) of $\Gamma_{r,i}(n\tau)$. Then during round n, for each of the 2^d combinations of HIGH and LOW, there is some function $\mathbf{a}^{\mathcal{Y},n} \in \mathcal{Y}_q(\Gamma_q(n\tau))$ such that the adversary can legally return[18] $a_i^{\mathcal{Y},n}$ in the

[18] That is, he can choose it as your acceleration in part 3 of the round; see Section 4.3.1.1.

i^{th} game, provided you follow the Simple Strategy. Suppose you do so. Then by the claim, in each game i, $\Gamma_{q,i}(t)$ is within (η_{x0}, η_{v0}) of $\Gamma_{r,i}(t)$ for all $t \in [n\tau, (n+1)\tau]$. By induction, we see that (4.25) holds for the duration of the game.

Therefore, if for all n, for all $t \in [n\tau, (n+1)\tau]$, $\mathbf{a}_q(t) = \mathbf{a}^{\mathcal{Y},n}(t)$, then Γ_q tracks Γ_r to L_∞ tolerance (η_x, η_v). □

4.3.1.4 Altering the Game to Prove the Robust Lemma (4.2.2)

Let us take the Tracking Game and alter the rules:

4. A temporary state $(\tilde{x}_{n+1}, \tilde{v}_{n+1})$ at the end of the round is computed using your state at the beginning of the round (end of the previous round) (x_n, v_n) and integrating $a'_n(t)$ over the time interval $[n\tau, (n+1)\tau]$ from (x_n, v_n).

5. The adversary chooses a position perturbation x_n^Δ and a velocity perturbation v_n^Δ to be applied to your position and velocity. Your position at the end of round n is
$$\begin{aligned} x'_{n+1} &= \tilde{x}_{n+1} + x_n^\Delta; \\ v'_{n+1} &= \tilde{v}_{n+1} + v_n^\Delta. \end{aligned}$$

We call the resulting game the *Tracking Game With Perturbations.*

Claim 4.3.3 *Suppose that you play the Tracking Game With Perturbations against an adversary and follow the Simple Strategy above. Suppose that at the start of the game* $|x_0 - x'_0| \le 2A_{max}\tau^2$ *and* $|v_0 - v'_0| \le 2A_{max}\tau$. *Furthermore, suppose that the adversary always chooses perturbations such that* $|x_n^\Delta| \le \frac{\kappa_l\tau^2}{2}$ *and* $|v_n^\Delta| \le \frac{\kappa_l\tau}{2}$

Let two trajectories Γ_r *and* Γ_q *be defined as follows:*

$$\begin{aligned} \Gamma_r(n\tau) &= (x_n, v_n); \\ a_r(t) &= a_n(t - n\tau) \quad \text{for all} \quad t \in (n\tau, (n+1)\tau); \\ \Gamma_q(n\tau) &= (x'_n, v'_n); \\ a_q(t) &= a'_n(t - n\tau) \quad \text{for all} \quad t \in (n\tau, (n+1)\tau). \end{aligned}$$

Then for all t:

$$\begin{aligned} |v_r(t) - v_q(t)| &\le 4A_{max}\tau + \tfrac{\kappa_l\tau}{2}; \\ |x_r(t) - x_q(t)| &< \tfrac{(8A_{max}+\kappa_l)^2\tau^2}{4\kappa_l} + 6A_{max}\tau^2 + \tfrac{\kappa_l\tau^2}{2}. \end{aligned} \tag{4.31}$$

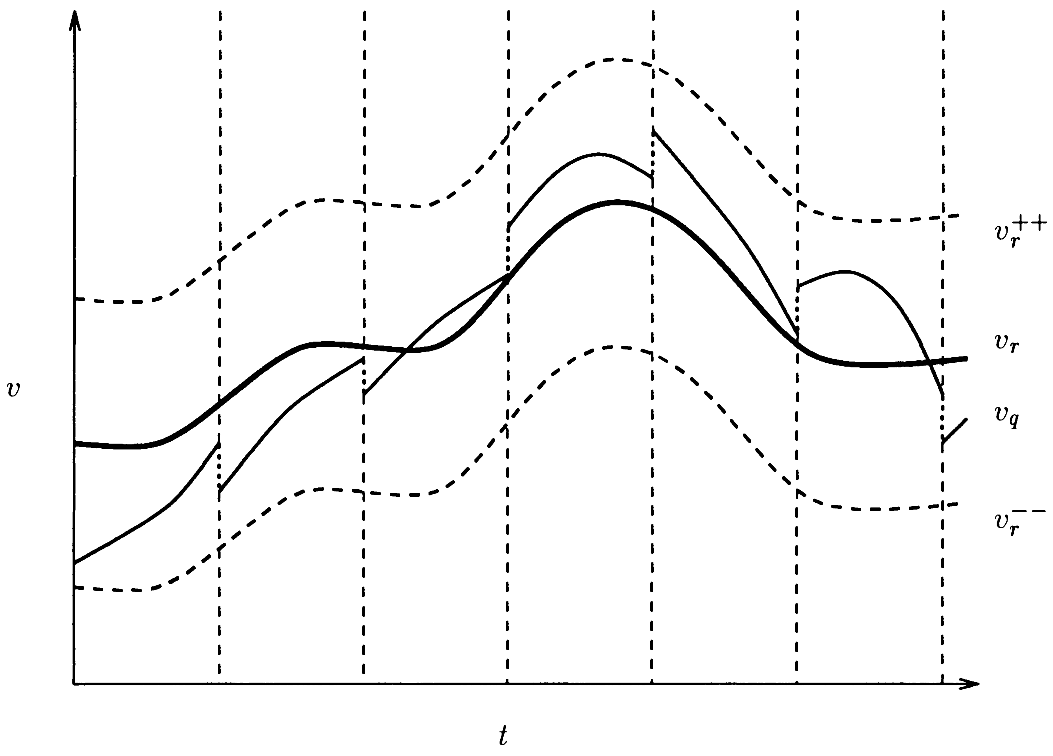


Figure 4.8: Intuition for proving Lemma 4.2.2: another tracking game in 1D. This time, the adversary can perturb our state at the end of each round, by $\frac{\kappa_l \tau^2}{2}$ in position and $\frac{\kappa_l \tau}{2}$ in velocity; the discontinuities in v_r correspond to the perturbations. However, we can still track him, though to a looser tolerance. $v_r^{++} = v_r + 4A_{max}\tau + \frac{\kappa_l \tau}{2}$, and $v_r^{--} = v_r - 4A_{max}\tau - \frac{\kappa_l \tau}{2}$.

Proof:

The analysis proceeds similarly to that for Claim 4.3.1.

Again, we let $x_{err}(t) = x_r(t) - x_q(t)$, and $v_{err}(t) = v_r(t) - v_q(t)$ and bound the length of an interval for which x_{err} and v_{err} can have the same non-zero sign. Because of the velocity perturbation, our effective acceleration advantage is only $\frac{\kappa_l}{2}$. Therefore, if $|x_{err}(n\tau)| > 2A_{max}\tau^2$, following the strategy ensures that $v_{err}(t_c) = 0$ at some later time $t_c \le n\tau + \frac{(8A_{max}+\kappa_l)\tau}{2\kappa_l}$. Then,

$$|x_{err}(t_c) - x_{err}(n\tau)| \le \frac{(8A_{max} + \kappa_l)^2\tau^2}{4\kappa_l}.$$

Because of the perturbation at the end of a round, the $\frac{\kappa_l\tau}{2}$ term must be added to the velocity-tracking tolerance and $\frac{\kappa_l\tau^2}{2}$ must be added to the position term in the final tracking accuracy.

Therefore

$$\max_t |x_{err}(t)| < \frac{(8A_{max} + \kappa_l)^2\tau^2}{4\kappa_l} + 6A_{max}\tau^2 + \frac{\kappa_l\tau^2}{2}. \tag{4.32}$$

□

We immediately get the Corollary:

Corollary 4.3.4 *The following conditions together guarantee that in the Tracking Game with Perturbations the adversary can be tracked to a tolerance of* (η_x, η_v)*:*

$$\begin{aligned} |x_0 - x_0'| &\le 2A_{max}\tau^2; \\ |v_0 - v_0'| &\le 2A_{max}\tau; \quad \textit{and} \\ \tau &\le \min\left(\sqrt{\frac{4\eta_x\kappa_l}{24A_{max}\kappa_l + 2\kappa_l^2 + (8A_{max}+\kappa_l)^2}}, \frac{2\eta_v}{8A_{max}+\kappa_l}\right). \end{aligned} \tag{4.33}$$

We can now prove Lemma 4.2.2

Proof (of Lemma 4.2.2): Let $\mathcal{P}$ and $\mathcal{Q}$ be functions $\mathcal{P}, \mathcal{Q} : TC \to \{\mathcal{Y} : \mathcal{Y}$ *is a set of accelerations*$\}$. Let $\mathcal{P}$ have the property that if $\mathbf{a} \in \mathcal{P}(\mathbf{p}, \dot{\mathbf{p}})$, then for all times t, $|\mathbf{a}(t)|_\infty \le A_{max} - \kappa_l$. Let $\mathcal{Q}$ have the properties that $\mathcal{Q}(\mathbf{X})$ is finite for all $\mathbf{X} \in TC$ and that if $\mathbf{a} \in \mathcal{Q}(\mathbf{X})$, then for all times t $|\mathbf{a}(t)|_\infty \le A_{max}$.

Suppose that there exist a maximal tracking tolerance (η_{x0}, η_{v0}) and a fundamental timestep τ_0 such that for all $\tau \le \tau_0$ the following is true: if state $\mathbf{X}$ is within (η_{x0}, η_{v0}) of state $\mathbf{Y}$, then the set of acceleration functions $\mathcal{Q}(\mathbf{X})$ has a uniform κ_l advantage over $\mathcal{P}(\mathbf{Y})$ for duration τ.

Suppose that (η_x, η_v) is an L_∞ tracking tolerance and that

$$\begin{aligned} \tau &\leq \tau_0; \\ \tau &\leq \sqrt{\frac{4\min(\eta_{x0},\eta_x)\kappa_l}{24A_{max}\kappa_l+2\kappa_l^2+(8A_{max}+\kappa_l)^2}}; \quad \text{and} \\ \tau &\leq \frac{2\min(\eta_{v0},\eta_v)}{8A_{max}+\kappa_l}. \end{aligned} \tag{4.24}$$

Let be $\mathcal{G}$ is the obstacle-free $(\mathcal{Q}, \tau)$-reachability graph rooted at some $\mathbf{S}^*$. Suppose that Γ_r respects $\mathcal{P}$, $\Gamma_r(0)$ is within $2A_{max}\tau$ of $\dot{\mathbf{s}}^*$, and $\mathbf{p}_r(0) = \mathbf{s}^*$. Suppose that $\mathcal{R}$ is a subset of TC containing the (η_x, η_v) tube induced by Γ_r, and that $\mathcal{G}'$ results from some breadth-first $(\frac{\kappa_l\tau^2}{2}, \frac{\kappa_l\tau}{2})$-pruning of the maximal subgraph of $\mathcal{G}$ that lies in $\mathcal{R}$.

Consider d simultaneous independent playings of the Tracking Game With Perturbations in which your adversary's trajectory in the i^{th} game is $\Gamma_{r,i}$ and yours is $\Gamma_{q,i}$. Suppose that all the $\Gamma_{r,i}(0)$ and $\Gamma_{q,i}(0)$ meet the starting conditions $(t = 0)$ of Claim 4.3.3. Let $(\mathbf{x}_n, \mathbf{v}_n)$ denote your state (*i.e.*, the vector of your states in the individual games) at the beginning of round n.

Now, suppose you follow the Simple Strategy. Because the breadth first pruning buckets and perturbation size are both $(\frac{\kappa_l\tau^2}{2}, \frac{\kappa_l\tau}{2})$, Claim 4.3.3 and the definition of $\mathcal{Q}$, $\mathcal{G}$, and $\mathcal{G}'$ assure that the following can legally occur in each round n of the game:

1. First, the adversary chooses a member of $\mathcal{Q}(\Gamma_q(n\tau))$ for *your* (vector of) accelerations.

2. Then, the adversary perturbs your state $(\tilde{\mathbf{x}}_{n+1}, \tilde{\mathbf{v}}_{n+1})$ if and only if it is not in $\mathcal{G}'$. In particular, if he perturbs your state, he perturbs it to the state in $\mathcal{G}'$ that caused $(\tilde{\mathbf{x}}_{n+1}, \tilde{\mathbf{v}}_{n+1})$ to be pruned from $\mathcal{G}'$. Thus, for some graph trajectory $\tilde{\Gamma}_q^{(n+1)}$ in $\mathcal{G}'$ and some $m \leq n+1$, $\tilde{\Gamma}_q^{(n+1)}(m\tau)$ is this state.

If the adversary plays in this manner, then by Corollary 4.3.4 for all n:

1. $(\mathbf{x}_n, \mathbf{v}_n)$ will be within (η_x, η_v) of $\Gamma_r(n\tau)$,

2. $(\tilde{\mathbf{x}}_n, \tilde{\mathbf{v}}_n)$ will be within (η_x, η_v) of $\Gamma_r(n\tau)$, and

3. your trajectory $\Gamma_q(t)$ will be within (η_x, η_v) of $\Gamma_r(t)$ for all $t \in [n\tau, (n+1)\tau]$.

Furthermore, for each n there will be some $\Gamma_q^{(n)}$ in $\mathcal{G}'$ such that for some $m \leq n$,

1. $\Gamma_q^{(n)}(m\tau) = \Gamma_q(n\tau)$, and
2. $\Gamma_q^{(n)}(t - (n-m)\tau) = \Gamma_q(t)$ for all $t \in [n\tau, (n+1)\tau)$.

Since this holds for all n, there must be some trajectory Γ_q^* in $\mathcal{G}'$ such that for some $T_q \leq T_r$, $\Gamma_q^*(T_q)$ approximates $\Gamma_r(T_r)$ to within (η_x, η_v). □

4.3.2 Search-Space Bounds for Cartesian Manipulators

Here we first consider an L_2 Cartesian robot with acceleration bound $\overline{a}$. We choose an acceleration discretization $\boldsymbol{\mu}$ and an $\Omega(\epsilon)$ scalar κ_l such that (a) all μ_i are $\Omega(\epsilon)$, and (b) for all $\mathbf{X}, \mathbf{Y} \in TC$, the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations at $\mathbf{X}$ has a uniform κ_l-advantage over $\mathcal{A}'_{\mathbf{Y}}$ for any τ. We then derive a similar choice of $\boldsymbol{\mu}$ for cases when an L_2-norm force bound $\overline{f}$ is given instead of an acceleration bound. We then return our focus to the acceleration bound case and examine the Near Extremal Algorithm in detail to illustrate how to derive final reachability graph parameters[19] chosen by the algorithm. We briefly discuss the derivation of the control discretization parameter $\overline{\mu}$ for the True Extremal algorithm.

4.3.2.1 Parameter Choices and Acceleration Advantages for an L_2 Cartesian Robot

The set of constant accelerations obeying the L_2-norm bound $\overline{a}$ has a uniform acceleration advantage of

$$\frac{\overline{a}(2\epsilon + \epsilon^2)}{\sqrt{d}(1+\epsilon)^2} > \frac{\overline{a}\epsilon}{2\sqrt{d}}$$

over the set of accelerations obeying the L_2-norm bound $\frac{\overline{a}}{(1+\epsilon)^2}$. Recall that $\|\mathbf{y}\|_\infty \leq \|\mathbf{y}\|_2 \leq \sqrt{d}\|\mathbf{y}\|_\infty$ for any $\mathbf{y} \in \Re^d$ (4.17). If $\boldsymbol{\mu}$ and κ_l have the property that

$$\|\boldsymbol{\mu}\|_\infty \leq \kappa_l \leq \frac{\overline{a}\epsilon}{4\sqrt{d}}, \tag{4.34}$$

then for any state $\mathbf{X}$ and any duration τ, the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations $\mathcal{H}(\mathbf{X}, \tau)$ has a uniform κ_l-advantage over the set of acceleration functions obeying the L_2-norm bound $\frac{\overline{a}}{(1+\epsilon)^2}$.

For any state $\mathbf{X}$, consider the sets of feasible instantaneous accelerations $\mathcal{A}_{\mathbf{X}}$ and $\mathcal{A}'_{\mathbf{X}}$ that correspond to force bounds $\overline{f}$ and $\frac{\overline{f}}{(1+\epsilon)^2}$, respectively. Recall that the inertia matrix $\mathbf{M}$ is symmetric and positive definite and thus orthogonal. In

[19] These parameters are $\boldsymbol{\mu}$ and τ. κ_l is used in the proof and by the True Extremal Algorithm.

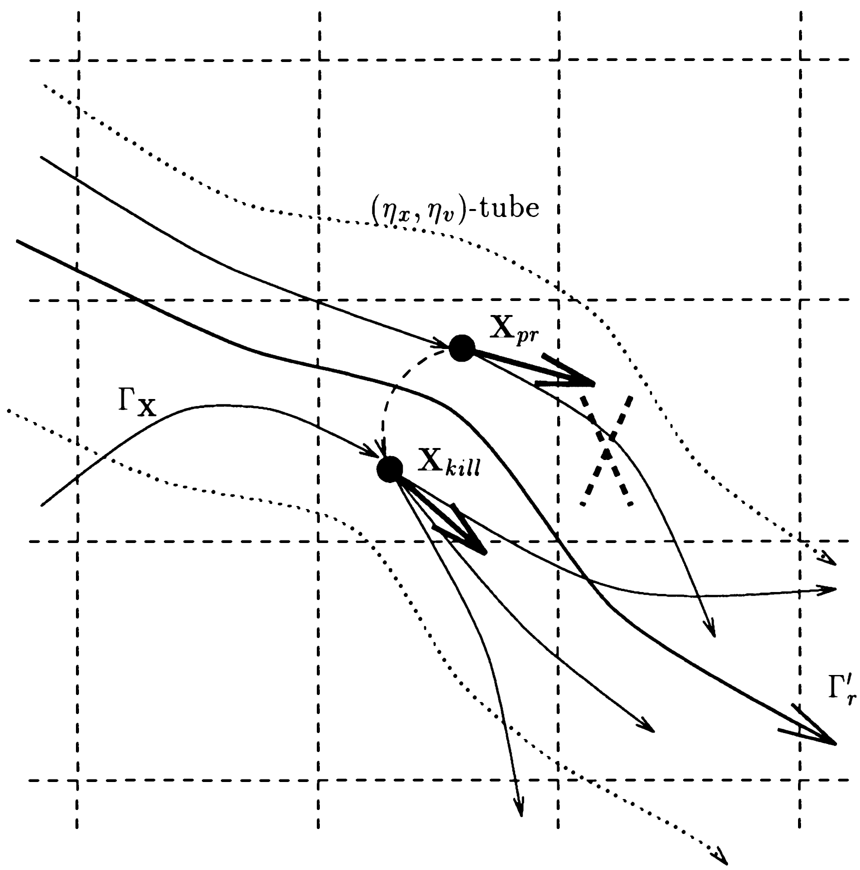


Figure 4.9: The Control Extremal Algorithm still has the desired completeness when we use single-occupancy cells to bound the search. If we choose parameters for the Tracking Game with Perturbations that correspond to the parameters chosen by the algorithm, then *some* trace of d simultaneous playings of the Tracking Game with Perturbations in which you follow the Simple Strategy corresponds to a "virtual path" in the pruned reachability graph, where "virtual edge" of zero cost goes from each pruned vertex to the vertex that caused its pruning.

addition, recall that $\mathbf{M}$ is constant for a Cartesian robot. If λ_{min} and λ_{max} are the minimum and maximum eigenvalues of $\mathbf{M}$, respectively, and $\mathbf{a} \in \partial\mathcal{A}_{\mathbf{X}}$, then $\|\mathbf{a}-\mathbf{a}'\|_2 \geq \frac{\overline{f}(2\epsilon+\epsilon^2)}{\lambda_{max}(1+\epsilon)^2}$. More compactly,

$$\|\mathbf{a}-\mathbf{a}'\|_2 > \frac{\overline{f}\epsilon}{2\lambda_{max}}. \tag{4.35}$$

It follows that if

$$\begin{aligned} \kappa_l &\leq \frac{\overline{f}\epsilon}{4\sqrt{d}\lambda_{max}}, \quad \text{and} \\ \|\boldsymbol{\mu}\|_\infty &\leq \frac{\overline{f}\epsilon}{4\sqrt{d}\lambda_{max}}. \end{aligned} \tag{4.36}$$

then $\mathcal{H}(\mathbf{X},\tau)$ has a uniform κ_l-advantage over $\mathcal{A}'(\mathbf{Y})$ for any τ, $\mathbf{X}$, and $\mathbf{Y}$.

4.3.2.2 Timestep Choice and Searchspace Bounds for the Near-Extremal Algorithm

By applying the Safe Tracking Lemma, we find a family of η_x and η_v that are $\Omega(\epsilon)$ and guarantee that any trajectory tracking a $\delta_v(c_0,c_1)$-safe trajectory to tolerance (η_x,η_v) will be $(1-\epsilon)\delta_v(c_0,c_1)$-safe. Specifically,

$$\begin{aligned} \eta_v &\leq \frac{c_0\epsilon}{c_1(1-\epsilon)+\beta} \\ \eta_x &\leq \beta\eta_v. \end{aligned} \tag{4.10}$$

It is simplest to choose $\beta = 1$, which implies $\eta_x = \eta_v$. However, by using a technique from Chapter 3, we show how to choose β to minimize the bound on the possible size of the reachability graph searched by the Near-Extremal Algorithm applied to a Cartesian robot with L_2 acceleration and velocity bounds $\overline{a}$ and $\overline{v}$.

First, we choose an underlying acceleration discretization $\boldsymbol{\mu}$ consistent with a uniform acceleration advantage κ_l (4.34):

$$\mu_i = \mu = \kappa_l = \frac{\overline{a}\epsilon}{4\sqrt{d}}. \tag{4.37}$$

We parameterize our choice of timestep τ as a function of β. We use Lemma 4.2.1 to obtain

$$\begin{aligned} \tau_x(\beta) &= \frac{\sqrt{c_0\epsilon^2\beta}}{\sqrt{2d(c_1(1-\epsilon)+\beta)(8\sqrt{2d}+3\overline{a}\epsilon)}} \\ \tau_v(\beta) &= \frac{c_0\epsilon}{4\overline{a}(c_1(1-\epsilon)+\beta)} \\ \tau(\beta) &= \min(\tau_x(\beta),\tau_v(\beta)). \end{aligned} \tag{4.38}$$

Since τ_x monotonic and β must be positive, $\tau(\beta)$ is maximized when $\tau_x(\beta)$ and $\tau_v(\beta)$ are equal. Solving for β to get a positive τ yields

$$\beta = \frac{1}{2}\left(-c_1(1-\epsilon) + \sqrt{c_1^2(1-\epsilon)^2 + \frac{c_0(16\sqrt{d}+3\epsilon)}{2\overline{a}}}\right). \tag{4.39}$$

We use the Coupled Tracking Lemma (4.2.1) to guarantee that a tracking trajectory would obey the velocity bound. Thus we choose

$$\tau = \min\left(\frac{\overline{v}\epsilon}{4\overline{a}}, \frac{c_0\epsilon}{2\overline{a}\left(c_1(1-\epsilon) + \sqrt{c_1^2(1-\epsilon)^2 + \frac{c_0(16\sqrt{d}+3\epsilon)}{2\overline{a}}}\right)}\right). \tag{4.40}$$

Given the desired start state $\mathbf{S}$, the Near-Extremal Algorithm chooses the root vertex $\mathbf{S}^*$ to meet the conditions at $t = 0$ in the Coupled Tracking Lemma, relative to ϵ-time-rescaled start state $\mathbf{S}'$. (Recall equation (4.5).) If an optimal trajectory Γ_{opt} exists, then there will be some graph trajectory Γ_q that tracks Γ'_{opt} to the tolerance (η_x, η_v) determined by the Safe Tracking Lemma and the choice of β above. (See equations (2.10) and (4.39).) Since (η_x, η_v) meets the conditions of the Safe Tracking Lemma, such a Γ_q would be guaranteed to be $(1-\epsilon)\delta_v(c_0, c_1)$-safe. Since η_x and η_v are $O(\epsilon)$, Γ_q would meet the approximation criteria at the desired start and goal states.

We can now bound the size of the reachability graph for a Cartesian robot whose maximum speed is $\overline{v}$ and whose configuration space is contained within a d-dimensional cube with diameter l. Let

$$\mu = \min_i \mu_i.$$

Then the total number of possible velocities for reachability graph vertices is bounded above by $\left(\frac{\overline{v}}{\mu\tau} + 1\right)^d$ and number of configurations by $\left(\frac{l}{\mu\tau^2} + 1\right)^d$. Thus, the total number of reachability graph vertices is

$$G_{\mathcal{V}}(\overline{a}, \mu, \tau, \overline{v}, l, d) < \left(\left(\frac{2\overline{v}}{\mu\tau} + 1\right)\left(\frac{l}{\mu\tau^2} + 1\right)\right)^d. \tag{4.41}$$

Using the choices of $\boldsymbol{\mu}$ and τ above in (4.34) (or (4.36)) and (4.40), we see that the right hand side of (4.41) is $O((\overline{v}l)^d(\frac{1}{\epsilon})^{5d})$. Recalling the definition of the $(\boldsymbol{\mu}, \tau)$-extremal shell, the out-degree of this graph is bounded above by $d\left(\frac{\overline{a}}{\mu}\right)^{d-1}$. Thus the total number of graph edges is

$$\begin{aligned} G_{\mathcal{E}}(\overline{a},\mu,\tau,\overline{v},l,d) &< d\left(\frac{\overline{a}}{\mu}\right)^{d-1}\left(\left(\frac{2\overline{v}}{\mu\tau}+1\right)\left(\frac{l}{\mu\tau^2}+1\right)\right)^{d} \\ &= O\left(\overline{a}^{d-1}(\overline{v}l)^{d}\left(\frac{1}{\epsilon}\right)^{6d-1}\right). \end{aligned} \tag{4.42}$$

We can follow a similar, straightforward development if L_2 force bounds (instead of acceleration bounds) are used. In particular, we choose $\boldsymbol{\mu}$ using (4.36), define analogues of (4.38), and choose β to maximize τ to get the necessary reachability graph parameters. We substitute $\frac{\overline{f}}{\lambda_{min}}$ for $\overline{a}$ in (4.43) to obtain similar bounds on the size of the reachability graph.

We summarize these results for a Cartesian robot obeying L_2 dynamics bounds with the following lemma.

Lemma 4.3.5 *Given a kinodynamic planning problem for a Cartesian Robot with L_2-norm dynamics bounds and given an approximation parameter ϵ, the Near-Extremal Algorithm will search a reachability graph with $O(c_D^d(\frac{1}{\epsilon})^{6d-1})$ vertices and edges; c_D is a constant dependent on the kinodynamic specifications. If an optimal (safe) solution Γ_{opt} exists, then some graph trajectory is ϵ-optimal.*

4.3.2.3 True-Extremal Algorithm Searchspace for a Cartesian Robot

Now, suppose the (Cartesian) robot obeys L_2 force bound $\overline{f}$, and again let λ_{min} and λ_{max} denote the minimum and maximum eigenvalues of its inertia matrix $\mathbf{M}$. If $\|\mathbf{f}_a\|_2, \|\mathbf{f}_b\|_2 \leq \overline{f}$ and $\|\mathbf{f}_a - \mathbf{f}_b\|_\infty \leq \overline{\mu}$, then

$$\|\mathbf{M}^{-1}\mathbf{f}_a - \mathbf{M}^{-1}\mathbf{f}_b\|_2 \leq \frac{\overline{\mu}\sqrt{d}}{\lambda_{min}}.$$

At the same time, if $\|\mathbf{f}_a\|_2 = \overline{f}$ and $\|\mathbf{f}_b\|_2 = \frac{\overline{f}}{(1+\epsilon)^2}$, then

$$\|\mathbf{M}^{-1}\mathbf{f}_a - \mathbf{M}^{-1}\mathbf{f}_b\|_2 > \frac{\overline{f}\epsilon}{2\lambda_{max}}.$$

Therefore, if we choose κ_l as in (4.36) and

$$\overline{\mu} \leq \frac{\lambda_{min}\overline{f}\epsilon}{4d\lambda_{max}}, \tag{4.43}$$

we guarantee that set of true $(\overline{\mu}, \tau)$-bangs $\overline{\mathcal{H}}(\mathbf{X}, \tau)$ (Section 4.2.1.1) has a uniform κ_l-advantage over $\mathcal{A}'(\mathbf{Y})$ for any two states $\mathbf{X}$ and $\mathbf{Y}$ over any duration τ.

Again using the Safe-Tracking Lemma (2.2.3) to find a family of sufficiently close tracking tolerances (η_x, η_v), we can now apply the Robust Coupled Tracking Lemma (4.2.2) to find a maximal timestep τ using the uniform acceleration advantage κ_l above. The algorithm's choice of $\mathbf{S}^*$ trivially satisfies the $t = 0$ condition of the lemma. Clearly, since κ_l, η_x, and η_v are $\Omega(\epsilon)$, τ will be $\Omega(\epsilon)$ also. Finally, the algorithm chooses pruning-bucket dimensions prescribed by the Lemma (4.2.2):

$$\begin{aligned} b_x &= \tfrac{\kappa_l \tau^2}{2} && \text{in configuration, and} \\ b_v &= \tfrac{\kappa_l \tau}{2} && \text{in velocity.} \end{aligned} \tag{4.44}$$

These quantities will clearly be $\Omega(\epsilon^3)$ and $\Omega(\epsilon^2)$, respectively. The number of buckets is therefore $O((\frac{1}{\epsilon})^{5d})$. Recalling the definition of true $(\overline{\mu}, \tau)$-bangs, we see that each vertex has $O((\frac{1}{\epsilon})^{d-1})$ out-edges. Since the algorithm explores the out-edges of at most one vertex from each bucket, the True Extremal Algorithm will search $O((\frac{1}{\epsilon})^{6d-1})$ vertices and edges of a reachability graph. Summarizing:

Lemma 4.3.6 *Given a kinodynamic planning problem for a Cartesian Robot with L_2-norm dynamics bounds and given a approximation parameter ϵ, the True Extremal Algorithm will search a reachability graph with $O(c_E^d(\frac{1}{\epsilon})^{6d-1})$ vertices and edges; c_E is a constant dependent on the kinodynamic specifications. If an optimal (safe) solution Γ_{opt} exists, then some graph trajectory is ϵ-approximately optimal.*

Since the complexity of each algorithm is the number of graph edges multiplied by the time it takes to check the $(1-\epsilon)\delta(c_0, c_1)$-safety of each edge (as a trajectory segment), we determine the asymptotic complexities by combining this lemma with the bounds from Section 4.3.4. This yields Theorem 4.1.1.

4.3.3 Applying the Coupled Tracking Lemma and Robots with State-Dependent Dynamics

We now derive lower bounds for the discretization and timestep parameters used by the Near Extremal and True Extremal Algorithms for robots obeying the open-chain dynamics equation (4.1). The Near Extremal Algorithm must choose underlying acceleration discretizations μ_i and a timestep τ ensuring that if a δ_v-safe solution solution exists, then the algorithm will find an ϵ-approximately optimal solution. We describe how to find sufficiently small parameters that are all $\Omega(\epsilon)$. We also show that the True Extremal Algorithm can choose a timestep τ and a control discretization parameter $\overline{\mu}$ that are $\Omega(\epsilon)$ and that ensure the algorithm will find an ϵ-approximately optimal solution under the same conditions.

Let the problem parameters and ϵ be given. Recall from Section 4.2.2.4 that $\mathcal{A}$ and $\mathcal{A}'$ denote mappings from TC to $\{\mathcal{Y} : \mathcal{Y}$ *is a set of acceleration functions*$\}$ under bounds $\overline{\mathbf{f}}$ and $\overline{\mathbf{f}}'$ (from (4.14)), respectively. Now, let trajectory Γ_r respect $\mathcal{A}'$, and let Γ_q respect $\mathcal{A}$.

We now sketch how to find a set of *consistent parameters* $(\mu, \tau_0, \eta_{x0}, \eta_{v0}, \kappa_l)$: a maximal discretization parameter μ for either acceleration or control, a maximal timestep τ_0, a maximal tracking tolerance (η_{x0}, η_{v0}), and a uniform acceleration advantage κ_l. For any timestep $\tau \leq \tau_0$ and any trajectory Γ_r that respects $\mathcal{A}'$, the following will be true: if state $\mathbf{X}$ is within (η_{x0}, η_{v0}) of $\Gamma_r(n\tau)$, and $\overline{\mu}, \|\boldsymbol{\mu}\|_\infty \leq \mu$, then both the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations $\mathcal{H}(\mathbf{X}, \tau)$ and the set $\overline{\mathcal{H}}(\mathbf{X}, \tau)$ of true $(\overline{\mu}, \tau)$-bangs have uniform κ_l-advantages over $\mathcal{A}'(\Gamma_r(t - n\tau))$ for duration τ. We will show that sufficiently small μ, κ_l, τ_0, η_{x0}, and η_{v0} are $\Omega(\epsilon)$.

Having obtained these parameters, the algorithms can apply the Coupled Tracking Lemmas to choose τ. This leads us to bounds on the size of the reachability graphs the algorithms search.

4.3.3.1 Sufficient Conditions

Observe that the sets of feasible instantaneous accelerations $\mathcal{A}_\mathbf{X}$ and $\mathcal{A}'_\mathbf{X}$ are d-dimensional parallelpipeds for all $\mathbf{X} \in TC$. Let λ_{min} and λ_{max} be the minimum[20] and maximum eigenvalues of the inertia tensor $\mathbf{M}(\mathbf{p})$ over all positions.[21] Let $f_{min} = \min_i \overline{\mathbf{f}}_i$, and let $\overline{f} = \max_i \overline{\mathbf{f}}_i$. Let the force required to hold the robot stationary in the presence of gravity obey the bound $(1 - \kappa_f)\overline{f}$. From Section 4.2.2.3, the minimum L_2 distance between $\partial\mathcal{A}_\mathbf{X}$ and $\partial\mathcal{A}'_\mathbf{X}$ is greater than

$$\begin{aligned} \alpha &= \frac{\kappa_f f_{min}\epsilon}{2\lambda_{max}} && \text{generally, and} \\ \alpha &= \frac{f_{min}\epsilon}{2\lambda_{max}} && \text{in the absence of gravity.} \end{aligned} \tag{4.45}$$

[20] λ_{min} is the only parameter that is neither given in the problem instance nor bounded (in smallness) in the derivation; a "quick and dirty" bound is the minimum of the smallest link mass and smallest link inertia (in generalized units). A bound for λ_{max} follows from the bound on $\|\mathbf{M}(\mathbf{p})\|$ found in Appendix C.

[21] In this section and Section 4.3.3.2 we will use $(\mathbf{p}, \dot{\mathbf{p}})$ to denote both trajectories and arbitrary states. In equations such as (4.7), both types clearly make sense.

Suppose that Γ_q respects[22] the mapping $\mathcal{A}$ and tracks Γ_r to tolerance (η_{x0}, η_{v0}), and fix $\tau_0 > 0$. For any $\tau \leq \tau_0$, consider what happens over a timestep $[n\tau, (n+1)\tau]$. For any $t \in [n\tau, (n+1)\tau]$, $\Gamma_r(t) - \Gamma_r(n\tau)$ and $\Gamma_q(t) - \Gamma_q(n\tau)$ belong to sets of possible state perturbations that are functions of τ_0. That is, there exist minimal sets $\gamma_r(\tau_0), \gamma_q(\tau_0) \subset TC$ such that for all $h \leq \tau_0$:

$$\begin{aligned} \Gamma_r(t+h) - \Gamma_r(t) &\in \gamma_r(\tau_0), \quad \text{and} \\ \Gamma_q(t+h) - \Gamma_q(t) &\in \gamma_q(\tau_0). \end{aligned} \tag{4.46}$$

State perturbations about $\Gamma_r(n\tau)$ and $\Gamma_q(n\tau)$ result in perturbations of $\partial\mathcal{A}'_{\Gamma_r(t)}$ and $\partial\mathcal{A}_{\Gamma_q(t)}$ about $\partial\mathcal{A}'_{\Gamma_r(n\tau)}$ and $\partial\mathcal{A}_{\Gamma_q(n\tau)}$. We now bound the magnitudes of these perturbations as functions of η_{x0}, η_{v0}, and τ_0, respectively.

Let $\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f})$ be given by

$$\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f}) = \mathbf{M}^{-1}(\mathbf{p}) \left(\mathbf{f} - [\dot{\mathbf{p}}^T \mathbf{C}(\mathbf{p}) \dot{\mathbf{p}}] - \mathbf{G}(\mathbf{p}) \right), \tag{4.7}$$

and let $B_\delta(\mathbf{y})$ denote the δ-ball about $\mathbf{y}$. We denote global perturbation bounds

$$\begin{aligned} h_r(\tau_0) &= \max \|\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f}') - \mathbf{a}(\mathbf{p} + \Delta\mathbf{p}_{\gamma_r}, \dot{\mathbf{p}} + \Delta\dot{\mathbf{p}}_{\gamma_r}, \mathbf{f}')\|_2 \\ h_{q0}(\tau_0) &= \max \left\|\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f}) - \mathbf{a}(\mathbf{p} + \Delta\mathbf{p}_{\gamma_q}, \dot{\mathbf{p}} + \Delta\dot{\mathbf{p}}_{\gamma_q}, \mathbf{f})\right\|_2 \\ h_{q1}(\eta_{x0}, \eta_{v0}) &= \max \|\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f}) - \mathbf{a}(\mathbf{p} + \Delta\mathbf{p}_{\eta_{x0}}, \dot{\mathbf{p}} + \Delta\dot{\mathbf{p}}_{\eta_{v0}}, \mathbf{f})\|_2 \end{aligned} \tag{4.47}$$

where the maxima are taken as (a) $(\mathbf{p}, \dot{\mathbf{p}})$ ranges over TC; (b) $\mathbf{f}$ and $\mathbf{f}'$ obey the constraints $f_i \leq \overline{f}_i$ and $f'_i \leq \overline{f}'_i$ for all coordinates i; (c) $(\Delta\mathbf{p}_{\gamma_r}, \Delta\dot{\mathbf{p}}_{\gamma_r})$ ranges over $\gamma_r(\tau_0)$; (d) $(\Delta\mathbf{p}_{\gamma_q}, \Delta\dot{\mathbf{p}}_{\gamma_q})$ ranges over $\gamma_q(\tau_0)$; and (e) $(\Delta\mathbf{p}_{\eta_{x0}}, \Delta\dot{\mathbf{p}}_{\eta_{v0}})$ ranges over $B_{\eta_{x0}}(\mathbf{0}) \times B_{\eta_{v0}}(\mathbf{0})$. Note that $h_{q0}(\tau_0)$ determines the acceleration perturbation bound E_A from Section 4.2.1.2, modulo norm.

Suppose that $\|\boldsymbol{\mu}\|_\infty = \mu$ and $\tau \leq \tau_0$. Recall (4.17). If

$$\mu\sqrt{d} + \kappa_l\sqrt{d} + h_r(\tau_0) + h_{q0}(\tau_0) + h_{q1}(\eta_{x0}, \eta_{v0}) \leq \alpha, \tag{4.48}$$

then for an interval of length τ, the $(\boldsymbol{\mu}, \tau)$-extremal shell of accelerations at $\Gamma_q(n\tau)$, or $\mathcal{H}(\Gamma_q(n\tau), \tau)$, will have a uniform κ_l-advantage over the set of acceleration functions $\mathcal{A}'(\Gamma_r(n\tau))$ feasible under the bound $\overline{\mathbf{f}}'$. Similarly, if

$$\frac{\overline{\mu}\sqrt{d}}{\lambda_{min}} + \kappa_l\sqrt{d} + h_{q0}(\tau_0) + h_{q1}(\eta_{x0}, \eta_{v0}) \leq \alpha, \tag{4.49}$$

[22]Recall from Section 4.2.2.4 that Γ_r *respects* $\mathcal{A}$ if for all intervals $[t_1, t_2]$, $\Gamma_r(t_1) = \mathbf{X}$ implies that Γ_r respects $\mathcal{A}(\mathbf{X})$ in $[t_1, t_2]$.

then set of true $(\overline{\mu}, \tau)$-bangs at $\Gamma_q(n\tau)$, $\overline{\mathcal{H}}(\Gamma_q(n\tau), \tau)$ will have a uniform κ_l-advantage over $\mathcal{A}'(\Gamma_r(n\tau_0))$.

The two constraints (4.48) and (4.49) are obeyed if

$$\begin{aligned} \mu, \frac{\overline{\mu}}{\lambda_{min}}, \kappa_l &\leq \frac{\alpha}{5\sqrt{d}}; \\ h_r(\tau_0), h_{q0}(\tau_0), h_{q1}(\eta_{x0}, \eta_{v0}) &< \frac{\alpha}{5}. \end{aligned} \tag{4.50}$$

4.3.3.2 Bounds for Perturbations

We show that $h_r(\tau_0)$ and $h_{q0}(\tau_0)$ are $O(\tau_0)$, and that $h_{q1}(\eta_{x0}, \eta_{v0})$ is $O(\eta_{x0})$ and $O(\eta_{v0})$. We first note that $\|\mathbf{M}(\mathbf{p}) - \mathbf{M}(\mathbf{p} + \Delta\mathbf{p})\|$ is $O(\|\Delta\mathbf{p}\|)$. Now, define

$$\mathcal{F}(\mathbf{f}, \mathbf{p}, \dot{\mathbf{p}}) = \mathbf{f} - [\dot{\mathbf{p}}^T \mathbf{C}(\mathbf{p})\dot{\mathbf{p}}] - \mathbf{G}(\mathbf{p}). \tag{4.51}$$

Substituting and differentiating

$$\begin{aligned} h_r(\tau_0) &< \tau_0 \max \left\| \left[\frac{\partial \mathbf{M}^{-1}}{\partial \mathbf{p}} \dot{\mathbf{p}}\right] \mathcal{F}(\mathbf{f}, \mathbf{p}, \dot{\mathbf{p}}) + \mathbf{M}^{-1}(\mathbf{p}) \left[\frac{\partial \mathcal{F}}{\partial \mathbf{p}} \dot{\mathbf{p}} + \frac{\partial \mathcal{F}}{\partial \dot{\mathbf{p}}} \mathbf{a}\right] \right\|; \\ h_{q0}(\tau_0) &\leq \tau_0 \max \left\| \left[\frac{\partial \mathbf{M}^{-1}}{\partial \mathbf{p}} \dot{\mathbf{p}}\right] \mathcal{F}(\mathbf{f}, \mathbf{p}, \dot{\mathbf{p}}) + \mathbf{M}^{-1}(\mathbf{p}) \left[\frac{\partial \mathcal{F}}{\partial \mathbf{p}} \dot{\mathbf{p}} + \frac{\partial \mathcal{F}}{\partial \dot{\mathbf{p}}} \mathbf{a}\right] \right\|; \\ h_{q1}(\eta_x, \eta_v) &\leq \max \left\| \left[\frac{\partial \mathbf{M}^{-1}}{\partial \mathbf{p}} \Delta\mathbf{p}\right] \mathcal{F}(\mathbf{f}, \mathbf{p}, \dot{\mathbf{p}}) + \mathbf{M}^{-1}(\mathbf{p}) \left[\frac{\partial \mathcal{F}}{\partial \mathbf{p}} \Delta\mathbf{p} + \frac{\partial \mathcal{F}}{\partial \dot{\mathbf{p}}} \Delta\dot{\mathbf{p}}\right] \right\|. \end{aligned} \tag{4.52}$$

Now, recall a derivation of (4.1), say from [AS86] and recall that

$$\begin{aligned} [\dot{\mathbf{p}}^T(t)\mathbf{C}(\mathbf{p}(t))\dot{\mathbf{p}}(t)]_i &= \dot{\mathbf{p}}^T(t)\mathbf{C}^i(\mathbf{p}(t))\dot{\mathbf{p}}(t), \quad \text{where} \\ \mathbf{C}^i(\mathbf{p}(t))_{jk} &= \sum_{j,k} \left\{ \frac{\partial M_{jk}(\mathbf{p})}{\partial p_k} - \frac{1}{2}\frac{\partial M_{jk}(\mathbf{p})}{\partial p_i} \right\}. \end{aligned} \tag{4.2}$$

Now, $\mathbf{M}(\mathbf{p})$ is simply the inertia tensor, and

$$\mathbf{G}(\mathbf{p}) = \frac{\partial U_G(\mathbf{p})}{\partial \mathbf{p}},$$

where $U_G(\mathbf{p})$ is gravitational potential energy at state $\mathbf{p}$. Hence, each component of $\mathbf{M}(\mathbf{p})$ and $\mathbf{G}(\mathbf{p})$ is a sum of products of components of $\mathbf{p}$ and their sines and cosines (e.g, p_i, $\cos(p_j)$, etc.) This implies that $\frac{\partial \mathcal{F}(\mathbf{f},\mathbf{p},\dot{\mathbf{p}})}{\partial \mathbf{p}}$ and $\frac{\partial \mathcal{F}(\mathbf{f},\mathbf{p},\dot{\mathbf{p}})}{\partial \dot{\mathbf{p}}}$ are globally bounded. Specifically, it follows that there are C_r, C_{q0}, C_{q1}, and C'_{q1} such that:

$$\begin{aligned} h_r(\tau_0) &< C_r \tau_0; \\ h_{q0}(\tau_0) &\leq C_{q0}\tau_0; \\ h_{q1}(\eta_x, \eta_v) &\leq C_{q1}\eta_{x0} + C'_{q1}\eta_{v0}. \end{aligned} \tag{4.53}$$

C_r, C_{q0}, C_{q1} can be bounded, given the robot parameters, by bounding the norms of the tensors arising from $\frac{\partial \mathcal{F}(\mathbf{f},\mathbf{p},\dot{\mathbf{p}})}{\partial \mathbf{p}}$ and $\frac{\partial \mathcal{F}(\mathbf{f},\mathbf{p},\dot{\mathbf{p}})}{\partial \dot{\mathbf{p}}}$. This can be done by inspection because all terms are bounded. However, simple expressions bounding the tensor norms have been calculated by [HP89], and derivations of C_r, C_{q0}, C_{q1}, and C'_{q1} can be found in Appendix C. Recalling (4.45) and (4.48), we therefore choose

$$\begin{aligned} \tau_0 &= \min\left(\tfrac{\alpha}{5C_r}, \tfrac{\alpha}{5C_{q0}}\right); \\ \eta_{x0} &= \tfrac{\alpha}{10C_{q1}}; \\ \eta_{v0} &= \tfrac{\alpha}{10C_{q2}}. \end{aligned} \tag{4.54}$$

Recalling the previous section, we thus can choose consistent parameters τ_0, η_{x0}, η_{v0}, μ, and κ_l that are all $\Omega(\epsilon)$. If $\|\boldsymbol{\mu}\|_\infty \le \mu$ and $\overline{\mu} \le \boldsymbol{\mu}$, then the maps $\mathcal{H}_{\tau_0}, \overline{\mathcal{H}}_{\tau_0} : TC \to \{\mathcal{Y} : \mathcal{Y}\ \textit{is a set of acceleration functions}\}$ defined by

$$\mathcal{H}_{\tau_0}(\mathbf{X}) = \mathcal{H}(\mathbf{X}, \tau_0), \quad \text{and} \tag{4.55}$$

$$\overline{\mathcal{H}}_{\tau_0}(\mathbf{X}) = \overline{\mathcal{H}}(\mathbf{X}, \tau_0) \tag{4.56}$$

meet the hypotheses of Coupled Tracking Lemmas. Since we can use the the Safe Tracking Lemma to find sufficiently small η_x and η_v that are $\Omega(\epsilon)$, we obtain a timestep τ that is $\Omega(\epsilon)$ by applying the appropriate Coupled Tracking Lemma (4.2.1 or 4.2.2). The pruning-bucket size is again given by (4.44).

Thus, we have the following lemma:

Lemma 4.3.7 *Given a kinodynamic planning problem for an open-chain manipulator and given approximation parameter ϵ, the Near-Extremal Algorithm and the True Extremal Algorithm will search reachability graphs with $O(c^d(\frac{1}{\epsilon})^{6d-1})$ vertices and edges; c is a constant dependent on the kinodynamic parameters. If an optimal (safe) solution Γ_{opt} exists, then each algorithm will find some graph trajectory that is ϵ-approximately optimal.*

4.3.4 Safety-Checking and Final Bounds

4.3.4.1 Quadratic Paths and Polyhedral C-Space Obstacles

Observe that a kinodynamic solution Γ_q found by the True Extremal Algorithm, when applied to Cartesian robots, or by the Near Extremal Algorithm will have a piecewise-constant acceleration $\ddot{\mathbf{p}}_q$. Hence the solution trajectory Γ_q is piecewise algebraic: $\mathbf{p}_q$ is quadratic and $\dot{\mathbf{p}}_q$ is linear in time t. Therefore, when the c-space obstacles are polyhedral, we can check for safety violations exactly as in the L_∞ dynamics bounds case (Cartesian Kinodynamic Planning) in previous chapters. As described in Section 2.2.5, the c-space obstacles are "grown" affinely with trajectory speed ($\|\mathbf{v}\| \in \Re$) to obtain *expanded c-space obstacles* in $C \times \Re$, which are determined by $O(N)$ algebraic surfaces.

Safety-checking for a single $(\overline{a}, \tau)$-bang or $(\boldsymbol{\mu}, \tau)$-bang can be accomplished by intersecting the quadratic trajectory (parameterized by time) with these surfaces,[23] and performing $O(N)$ sign-tests on each intersection point. This naive approach would yield an upper bound of $O(dN^2)$ per bang, with the d factor coming from the cost of taking dot products, or $O(N^2)$ for fixed d. In Section 2.2.5 we sketched an analysis showing that the necessary operations can be done in time $O(N)$ per trajectory segment for $d \leq 3$, by observing that the total number of sign-tests is bounded by twice the number of edges. This important observation can be generalized to higher dimensions: if the c-space obstacles are convex polytopes with $O(N)$ geometric complexity, and then the cost of checking the safety of quadratic paths is $O(dN)$.

4.3.4.2 Polyhedral Workspace Obstacles

For the more general case of open chain manipulators with revolute joints and polyhedral *workspace* obstacles, safety is also checked by affinely growing the c-space obstacles and checking for intersections along the trajectory. We describe a c-space obstacle representation and a robot-obstacle collision detection method that can be generalized to a safety-checking method similar to that described in Section 2.2.5 and reviewed above. It would be convenient if we could apply an algebraic collision detection predicate such as that described by Canny [Can86, Can88a]. (We shall soon describe why we cannot.) In fact our δ_v-safety predicate uses the *structure* of his predicate and the same logical evaluation method for each pair of polyhedra that could possibly collide.

The non-overlap condition for two convex polyhedra is given by the non-overlap predicate

$$\bigwedge_i \bigvee_j \bigwedge_k \bigvee_l (C_{ijkl}(\mathbf{x}) > 0) \tag{4.57}$$

described in [CD88], with the exact form of constraint functions $C_{ijkl} : C \to \Re$

[23]Hence, we "lift" collision detection to $C \times \Re$ to so safety-checking.

given in [Can86]. The signs of the $C_{ijkl}(\mathbf{x})$ correspond exactly to spatial relationships among the vertices, edges, and faces of the possibly overlapping polyhedra. The zeros of the C_{ijkl} are surfaces that contain faces of c-space obstacles, so we call them *c-space obstacle surface functions.* To detect collisions on a path $\mathbf{p} : \textit{time} \rightarrow TC$, one substitutes $\mathbf{p}(t)$ for $\mathbf{x}$ and "merges" the sign intervals of the resulting functions in time. While the exact form of the C_{ijkl} found in [Can86] for a robot polyhedron uses quaternions to represent orientation, *we can use any other c-space representation and obstacle surface functions that have the same sign correspondence.* For an open-kinematic chain, one set of natural c-space obstacle surface functions would be mixed trignometric polynomials — the sums of products of c-space coordinates and their cosines and/or sines. In [Can88a], Canny represents these surfaces algebraically by using quarternion and half-angle substitutions.

Unfortunately, algebraic collision detection requires a c-space coordinate system in which the surfaces of the c-space obstacles are algebraic and in which the robot path is algebraic in time. In the joint coordinate system for an open kinematic chain, the position components p_i of $(\boldsymbol{\mu}, \tau)$-bangs are quadratic in time, and the velocity components v_i are linear. While it is possible to describe the c-space obstacles algebraically by using substitutions such as $u_j = \tan \frac{p_j}{2}$ for revolute joints j, for the trajectories generated by our algorithms there is no way (when the configuration space has more than one dimension) to simultaneously algebraically parameterize the *paths functions* that result from this substitution. Since our algorithm only considers motions where each p_j is a quadratic polynomial in time t, we can choose *either* (a) a coordinate system in which the c-space obstacle surfaces are the zeros of algebraic functions, and some path-position components p_j are inverse trignometric functions of time, or (b) a coordinate system in which the c-space obstacle surfaces are the zeros of trignometric polynomials, but each path-position component p_i is an algebraic function of time. Thus, there is no coordinate system in which both the c-space obstacle surface functions and $(\boldsymbol{\mu}, \tau)$-bang robot paths are algebraic, and we cannot in general perform safety-checking algebraically.

We can, however, approximately evaluate the collision predicate for $(\boldsymbol{\mu}, \tau)$-bangs by using approximating polynomials for each of the c-space obstacle surface functions. The same can be done for a corresponding δ_v-safety predicate. The degree r of the polynomials we need depends on how accurately we wish to approximate the trignometric polynomials. Intuitively, we expect that since each timestep is finite, and in practice would be very short, this polynomial approximation is reasonable. The resulting error ϵ_r in checking safety can be bounded and made arbitrarily small by increasing the degree of the polynomial. Then, one could implement either a conservative algorithm, which only finds solutions that are $(\delta'_v(\mathbf{v}) + \epsilon_r)$-safe, or an "optimistic" algorithm that finds solutions that are $(\delta'_v(\mathbf{v}) - \epsilon_r)$-safe.

We now argue that for the purpose of deriving an asymptotic complexity we can fix the degree of the approximating polynomials to a constant. Simply put, if we truncate the Taylor expansions of expanded c-space obstacle surface functions to r^{th} order polynomials, ϵ_r will be $O(\epsilon^r)$ because the timestep τ is $\Omega(\epsilon)$. Thus, if we set $r = 2$, ϵ_r will be $O(\epsilon^2)$, and thus the error of the safety-checking approximation will be asymptotically smaller than ϵ.

Since the structure of our δ_v-safety violation predicate is similar to that of [Can86], it contains $O(N)$ polynomials in t and requires $O(N \log N)$ time to evaluate, once the sign-intervals of the polynomials are known. If we restrict ourselves to polynomials of degree r_c, the total number of terms in the polynomials will be $O(r_c N)$, and forming the expressions for expanded c-space obstacle functions will take will take time $O(r_c d N)$. If finding the sign-intervals of a polynomial of degree r_c has cost $S(r_c)$, then checking the $(1-\epsilon)\delta_v(c_0, c_1)$-safety of a single trajectory segment could be done in time

$$O(N r_c(\log N + d + \log r_c) + N S(r_c)).$$

By the truncated series approximation error argument above, we can choose a constant r_c as long as ϵ is guaranteed to be sufficiently small. For this reason we say that safety-checking takes time $O(N(d + \log N))$. This completes the discussion of safety-checking for the Near Extremal Algorithm. We now turn to the True-Extremal case.

For the general open chain manipulator, true $(\overline{\mu}, \tau)$-bangs are the solutions to the set of ordinary differential equations

$$\begin{pmatrix} \dot{\mathbf{p}}(t) \\ \dot{\mathbf{v}}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{v}(t) \\ \mathbf{M}^{-1}(\mathbf{p}(t))\left\{\mathbf{f} - [\mathbf{v}(t)^T \mathbf{C}(\mathbf{p}(t))\mathbf{v}(t)] - \mathbf{G}(\mathbf{p}(t))\right\} \end{pmatrix}. \tag{4.58}$$

Since right-hand side of (4.58) is C^∞ and its time-derivative can be bounded globally, standard integration techniques (*e.g.*, Runge-Kutta) can be used to approximate the image of a $(\overline{\mu}, \tau)$-bang time $t \leq \tau$ to arbitrary precision, with computation time growing sub-linearly with the accuracy [PFTV88]. Similar techniques can be used to approximate the trajectory segment with a set of polynomials of fixed order (*e.g.*, 4^{th} order Runge-Kutta) that approximate it in respective subintervals of the timestep; the closeness of the trajectories to true extremal will be exponential in the degree of the polynomials. To do safety-checking of the True Extremal Algorithm, then, we can again use approximating polynomials in our δ_v-safety predicate.

To summarize, under a combinatorial model that charges unit cost for finding the roots of a univariate trignometric polynomial of fixed degree and size, safety checking for the Near-Extremal algorithm can be done in time $O(N(d+\log N))$. In practice, numerical methods would be used to find sign-intervals for approximating polynomials in t. The error in these methods can be bounded and incorporated into the safety margin. Approximating polynomials can also be used to check the safety of general open-chain manipulator $(\overline{\mu}, \tau)$-bangs. Thus, for a fixed number of degrees of freedom the algorithms have a time-complexity of $O(N \log N)$ in the asymptotic case.

4.3.4.3 Asymptotic Bounds

For each of our algorithms, the final complexity is the complexity of the number of graph edges explored multiplied by the cost of checking the safety of each edge as a trajectory segment. By combining the graph-size bounds of Lemmas 4.3.5, 4.3.6, and 4.3.7 with the safety checking costs described above, we obtain a final approximate cost of $O(N(d + \log N)\frac{1}{\epsilon})^{6d-1})$, assuming sufficiently small $\epsilon > 0$. Letting $p(N, \epsilon, d)$ express the cost of safety checking yields Theorems 4.1.1 and 4.1.2.

4.4 Summary

In this chapter we obtained provably good approximation algorithms for kinodynamic planning that extend the approach presented in earlier chapters to open kinematic chains and to Cartesian robots obeying L_2-norm dynamics bounds. These algorithms find trajectories that are approximately optimal with respect to a possibly speed-dependent safety margin. We presented algorithms that find near-extremal trajectories and algorithms that find truly extremal trajectories. The True Extremal algorithm is the first such provably good algorithm that uses a state density condition to prune an otherwise exponential search to polynomial size. We believe that both algorithms apply to a more general class of robots, as discussed below.

To obtain our results we proved two crucial lemmas by using a game-theoretic proof technique. By using the first lemma (4.2.1), given robot dynamics bounds, a safety margin, and ϵ we can find parameters that determine a state reachability graph *such that for every optimal trajectory whose initial state is approximated by the root of the graph, there is a graph trajectory that is within* ϵ *of being optimal.* A second lemma (4.2.2) allows us to simultaneously derive the state density condition above (4.44).

Although our results directly apply to two classes of robots, they can be easily extended to a larger class. We conjecture this class is the class of robots that have finite degrees of freedom, bounded configuration, convex generalized force and velocity bounds, and acceleration maps that obey the following constraints: (a) the dimension of the set of feasible accelerations is equal to the dimension of the configuration space, (b) the set of feasible accelerations is convex at each state, and (c) state perturbations $(\Delta\mathbf{p}, \Delta\dot{\mathbf{p}})$ result in perturbations to the acceleration map that are $O(\|\Delta\mathbf{p}\| + \|\Delta\dot{\mathbf{p}}\|)$.

Chapter 5

A "Bang-Bang" Planner and Time-Safety Trade-Offs

We now extend the main results to obtain ϵ-approximation algorithms that do not discretize accelerations or controls (applied forces) and to extend our approximation model to consider time- and safety-optimality separately.

First, in Section 5.1 we consider finding ϵ-optimal trajectories that are bang-bang, *i.e.*, for which all controls are piecewise-saturated.[1] We demonstrate polynomial-time ϵ-approximation algorithms for this kind of optimal kinodynamic planning. These "bang-bang" algorithms search reachability graphs with a 2^d branching-factor. For open kinematic chains with d degrees of freedom, such an algorithm is a natural extension of the algorithms in Chapters 2 and 3.

In Section 5.2, we consider a finer complexity analysis of the kinodynamic planning algorithms from Chapters 3 and 4. In the previous chapters, a single parameter ϵ characterized closeness to optimality both in execution time and in observance of the safety margin. We introduce separate parameters ϵ_T and ϵ_S to describe time- and safety-optimality. Analyses using these parameters show how the algorithms from previous chapters allow one to trade off planning time, execution time, and observance of the safety margin.

We illustrate this first step towards practical provably good kinodynamic planning with (a) graphical representations of the theoretical trade-offs and (b) trajectories found by our planar Cartesian implementation.

[1]A (single) one-dimensional control is said to be *saturated* when it takes one of its two extreme values. We say control is piecewise-saturated if it is saturated almost everywhere and has a finite number of switchings over every finite interval of time.

5.1 ϵ-Optimal Bang-Bang Trajectories

The algorithms for Cartesian Kinodynamic Planning in Chapters 2 and 3 find trajectories generated by a set of control primitives, $(\overline{a}, \tau)$-bangs. While the components of these acceleration functions assume values from the set $\{-\overline{a}, 0, \overline{a}\}$, it is possible to find ϵ-optimal trajectories having acceleration functions that are piecewise-constant and whose components only take values from the set $\{-\overline{a}, \overline{a}\}$. Thus, these trajectories belong to a class and of trajectories that obey piecewise constant, component-wise-fully-saturated controls, also known as bang-bang controls. (See figure 5.1.)

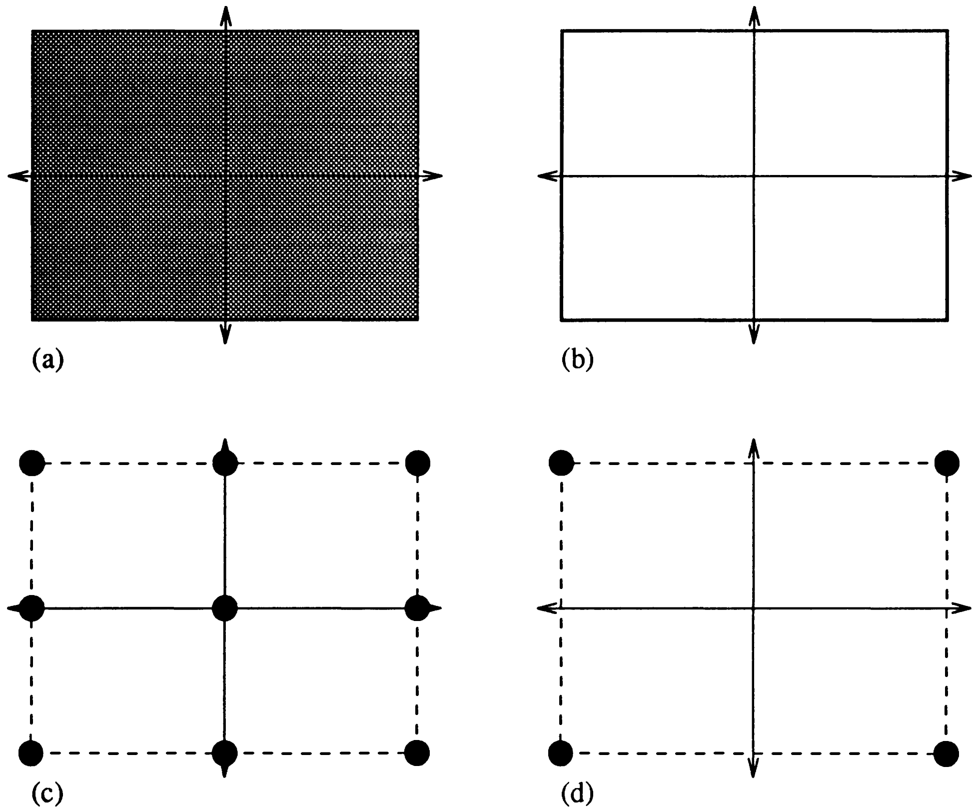


Figure 5.1: Ranges control functions from four classes: (a) simple bounds, (b) extremals, (c) $(\overline{\mathbf{f}}, \tau)$-bang, and (d) bang-bang.

Since a large class of minimal-time trajectory problems for time-invariant linear systems have bang-bang solutions, we might hope to find a provably-good approximation algorithm for ϵ-Optimal Cartesian Kinodynamic Planning that only considers bang-bang controls. However, the characterization of time-optimal controls for open-kinematic chains is still generally an open problem, even in the absence of state constraints. On the other hand, the Pontryagin Maximum Principle can be used to show that a time-optimal trajectory is extremal when not "on" a state constraint [PBGM62,Lei66].

The main result of this section is an algorithm for ϵ-Optimal Open-Chain Kinodynamic Planning that directly finds ϵ-optimal trajectories obeying bang-bang control. The algorithm searches a reachability graph whose edges are generated by constant fully-saturated control forces lasting for a problem-dependent $\Omega(\epsilon^2)$ timestep τ_B. Thus, we can analytically relate ϵ to a sufficiently short switching time[2]. Although the algorithm is the most "natural" generalization of our algorithms for Cartesian Kinodynamic Planning to open-chain robots, we are only able to show its correctness and complexity using the advanced "adversary game with perturbations" proof technique introduced in Section 4.3.1.4.

5.1.1 "Minimalist" Control

5.1.1.1 Bang-Bang Control for Simple Dynamics

Recall that for a given instance of the Cartesian Kinodynamic Planning problem $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ and approximation parameter $\epsilon > 0$, the algorithms from Chapters 2 and 3 generate $(\overline{a}, \tau)$-bang trajectories, where τ is the problem-dependent timestep chosen by the algorithm. By definition, the (vector) components of the acceleration function of such a trajectory are piecewise-constant, taking values from the set $\{-\overline{a}, 0, \overline{a}\}$ and switching only at multiples of τ.

Suppose that each component of the acceleration function for a trajectory is piecewise-constant and only takes values from the set $\{-\overline{a}, \overline{a}\}$. Then we call that trajectory and its control *completely bang-bang*. (In this case, the control is the acceleration function.) If, furthermore, switching times only occur at multiples of some timestep τ, then we call that trajectory and its control τ*-bang-bang*. Note that because a single control value corresponds to only one trajectory for a given initial state, bang-bang controls can be regarded as a *minimal set of non-trivial controls*.

[2]Intuitively, we might expect that a discrete control function having the range in figure 5.1(d) might, by "averaging", approximate the effect of a control function with the range in figure 5.1(a), provided that its switching times can be sufficiently short.

By extending the techniques used to show the approximation accuracy and complexity of Algorithm 2, we obtain a reachability-graph algorithm that directly finds a τ-bang-bang ϵ-optimal solution, where τ is problem dependent, whenever a solution exists. The algorithm has complexity $O(c^d N(\frac{1}{\epsilon})^{3d})$, for some robot-dependent constant c.

5.1.1.2 Bang-Bang Control for Open Chain Dynamics

Now, recall the kinodynamic planning problem formulation from Chapter 4. For open kinematic chain robots, the analogue of an $(\overline{a}, \tau)$-bang is the application of a constant control force $\mathbf{f}$, such that $f_i \in \{-\overline{f}_i, 0, \overline{f}_i,\}$ for all i, for a timestep τ. We call this type of control primitive and the corresponding trajectory segment an $(\overline{\mathbf{f}}, \tau)$*-bang*.

If the timestep τ is adequately small, then the "average" accelerations[3] producible by some number N_ϵ of consecutive $(\overline{\mathbf{f}}, \tau)$-bangs will have an acceleration advantage over an ϵ-time-rescaled trajectory segment of duration $N_\epsilon \tau$. By applying this observation and building on the techniques used to show the complexity and correctness of the True-Extremal Algorithm from Chapter 4, we might expect to obtain a provably-good kinodynamic planning algorithm that synthesizes solutions by searching a reachability graph generated by $(\overline{\mathbf{f}}, \tau)$-bangs. Such an algorithm would be the natural generalization of our Cartesian robot algorithms to open-kinematic chain robots.

With this in mind, and since there is an provably good approximation for ϵ-optimal Cartesian Kinodynamic Planning that considers only bang-bang trajectory segments, one might expect there to be a similar "bang-bang" kinodynamic planning algorithm for robots with open kinematic chain dynamics. This would find *bang-bang* solutions — ones obeying piecewise-constant control force functions $\mathbf{f}$ such that $f_i(t) \in \{-\overline{f}_i, \overline{f}_i,\}$ for each joint i. Here, we demonstrate such an algorithm. This algorithm searches a reachability graph whose edges are constant-control bang-bang trajectory segments lasting for time τ, the timestep chosen by the algorithm. We call these trajectory segments τ*-bangs*. Generalizing the previous definition in Section 5.1.1, a trajectory composed of τ-bangs is a τ*-bang-bang* trajectory.

The results of the previous chapter shows that we can find ϵ-optimal kinodynamic trajectories that obey piecewise extremal control with $\Omega(\epsilon)$ time between

[3] This will be defined shortly, in Section 5.1.4.1.

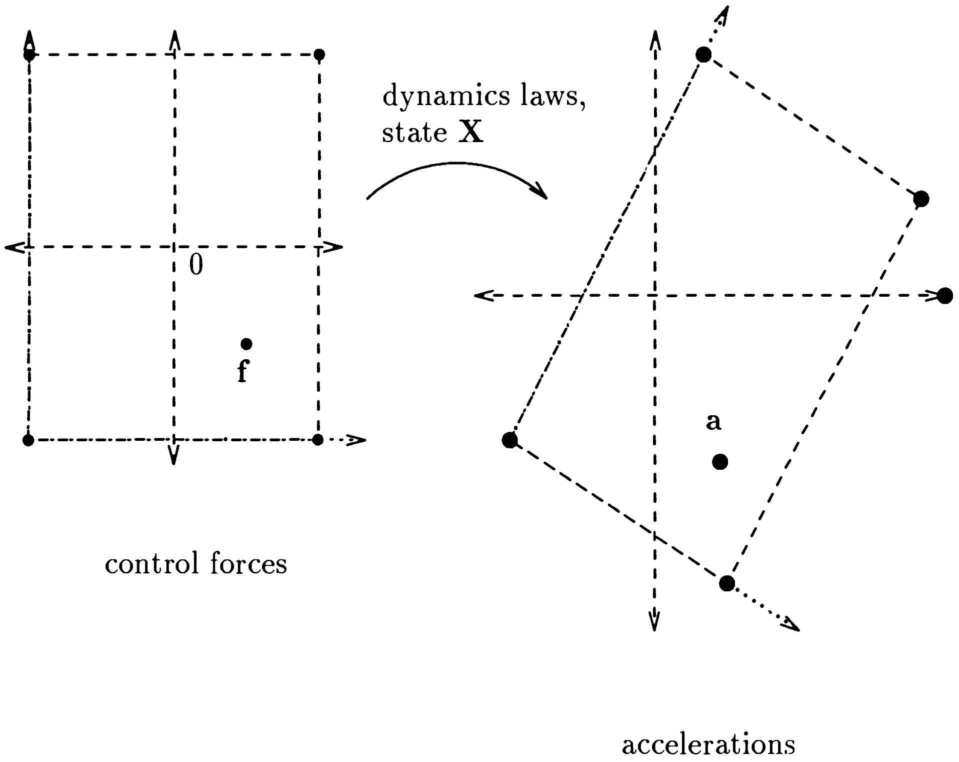


Figure 5.2: The robot dynamics laws and the robot state $\mathbf{X}$ map applied forces to accelerations. N τ-bangs determine an average force to $O(\frac{1}{N})$ accuracy. If τ were infinitesimally small, or the dynamics equation time-invariant, we could use τ-bang-bang control to achieve $O(\frac{1}{N})$ average acceleration control over a chunk of N timesteps. If $N\tau$ is small enough, we still obtain a useful accuracy.

switchings. By showing the accuracy of the bang-bang algorithm, we show that an ϵ-optimal completely bang-bang kinodynamic trajectory with an $\Omega(\epsilon^2)$ timestep will also exist. Note that like the algorithms in the previous chapters, this algorithm is only sure to find ϵ-optimal bang-bang solutions when some δ_v-safe kinodynamic solution exists.

To summarize, by extending techniques used to show the accuracy and the complexity of the True Extremal Algorithm (Chapter 4), we can show a theorem on the adequacy of bang-bang control for near-optimal kinodynamic planning. Essentially, the safety margin and start (goal) conditions of the ϵ-optimal kinodynamic planning problem *regularize* the classical optimal trajectory planning problem to guarantee that bang-bang control is sufficient. This enables us to obtain an algorithm that finds ϵ-optimal trajectories by only considering trajectories that obey bang-bang control; it is the first provably good bang-bang approximation algorithm for open-chain manipulators.

5.1.2 Bang-Bang Results

We first describe a simple modification to the timestep criteria of of Algorithm 2 that yields a probably good ϵ-approximation algorithm for Optimal Cartesian Kinodynamic Planning that finds only bang-bang trajectories. In this manner, we sketch a proof of the following:

Theorem 5.1.1 *Let* $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ *be an optimal kinodynamic planning problem for a point mass robot in d dimensions obeying* L_∞ *dynamics bounds. Let* $0 < \epsilon < 1$.

Suppose there is a $\delta_v(c_0, c_1)$*-safe trajectory from* $\mathbf{S}$ *to* $\mathbf{G}$ *taking time* T_{opt}. *Then we can find a* $(1-\epsilon)\delta_v(c_0, c_1)$*-safe trajectory going from some* $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*)$ *to some* $\mathbf{G}^* = (\mathbf{g}^*, \dot{\mathbf{g}}^*)$ *within* $O(\epsilon)$ *of* $\mathbf{S}$ *and* $\mathbf{G}$*, respectively, and taking time at most* $(1+\epsilon)T_{opt}$. *The approximation error at* $\mathbf{S}$ *and* $\mathbf{G}$ *is independent of* $\overline{v}$ *and* l.

For a given robot in a world of fixed diameter, we can do this in $O\left(c^d N \left(\frac{1}{\epsilon}\right)^{3d}\right)$ *time, where c is a robot-dependent constant and N is the geometric complexity of the obstacles.*

For robots with open-kinematic-chain dynamics we describe an algorithm, the Bang-Bang Algorithm, that attempts to find an ϵ-optimal trajectory by chosing a timestep τ_B and searching a reachability graph whose edges are τ_B-bangs.[4] Since the size of such a graph is generally exponential in path length (in terms of edges), the search uses (b_x, b_v)-bucket pruning, which we described in Chapter 4.[5]

Theorem 5.1.2 *Let $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{\mathbf{f}}, \overline{v}, \mathcal{M}, c_0, c_1)$ be an optimal kinodynamic planning problem for an open-chain manipulator with revolute and/or prismatic joints. Let $0 < \epsilon < 1$.*

Suppose there is an optimal $\delta_v(c_0, c_1)$-safe trajectory from $\mathbf{S}$ to $\mathbf{G}$ taking time T_{opt}. Then the Bang-Bang Algorithm finds a $(1-\epsilon)\delta_v(c_0, c_1)$-safe trajectory taking time at most $T_{opt}(1+\epsilon)$ from some $\mathbf{S}^* = (\mathbf{s}^*, \dot{\mathbf{s}}^*)$ to some $\mathbf{G}^* = (\mathbf{g}^*, \dot{\mathbf{g}}^*)$ such that $\mathbf{S}^*$ and $\mathbf{G}^*$ are ϵ-close to $\mathbf{S}$ and $\mathbf{G}$, respectively. This trajectory obeys bang-bang control, with switchings occurring only at multiples of an $\Omega(\epsilon)$ timestep.

The asymptotic running time of the algorithm is $O\left(c_F^d N^2 (d + \log N) \left(\frac{1}{\epsilon}\right)^{7d}\right)$ where c_F is a constant dependent on the kinodynamic specifications and N is the geometric complexity of the obstacles.

5.1.3 Minimalist Cartesian Kinodynamic Planning

5.1.3.1 Bang-Bang Approximation of an $(\overline{a}, \tau)$-bang Velocity Function

The key to seeing Theorem 5.1.1 is to recognize that any $(\overline{a}, \tau)$-bang trajectory Γ_r can be approximated by a completely bang-bang trajectory as long as state-constraints do not interfere.

Lemma 5.1.3 *Suppose that Γ_r is an $(\overline{a}, \tau)$-bang trajectory in the absence of obstacles, and that it obeys velocity bound $(\overline{v} - \frac{\overline{a}\tau}{4})$. Then there is a τ-bang-bang trajectory Γ_q that tracks γ_q to tolerance $(\frac{\overline{a}\tau^2}{16}, \frac{\overline{a}\tau}{4})$ and obeys velocity bound $\overline{v}$.*

Proof:

Since the dynamics bounds are decoupled, it is sufficient to consider the one-dimensional case. Let Γ_r satisfy the hypothesis of the lemma. Using the convention from section 3.3.2.2, we will let c_r^n describe the acceleration of Γ_r in the interval $[n\tau, (n+1)\tau]$, and c_q^n describe the acceleration of Γ_q in the interval $[\frac{n\tau}{4}, \frac{(n+1)\tau}{4}]$. Define Γ_q as follows:

[4]This extends previous notation: for L_∞-norm acceleration bound $\overline{a}$, a τ-bang is a trajectory segment having a constant acceleration whose components take values from the set $\{-\overline{a}, \overline{a}\}$.

[5]Recall that the True-Extremal Algorithm uses (b_x, b_v)-bucket pruning with a different bucket size.

1. $\Gamma_q(0) = \Gamma_r(0)$.

2. If $c_r^n = 1$, then
$$c_q^{4n} = c_q^{4n+1} = c_q^{4n+2} = c_q^{4n+3} = 1;$$

3. otherwise, if $c_r^n = -1$, then
$$c_q^{4n} = c_q^{4n+1} = c_q^{4n+2} = c_q^{4n+3} = -1;$$

4. otherwise ($c_r^n = 0$),
$$\begin{aligned} c_q^{4n} &= c_q^{4n+3} &= 1 \\ c_q^{4n+1} &= c_q^{4n+2} &= -1. \end{aligned}$$

By inspection, $|v_r(t) - v_q(t)| \leq \frac{\overline{a}\tau}{4}$ for all $t \in [0, T_r]$. Furthermore, $\Gamma_r(n\tau) = \Gamma_q(n\tau)$ for all relevant n. Position error only occurs when $c_r^n = 0$, and at most $\frac{\overline{a}\tau^2}{16}$ accumulates. □

5.1.3.2 Parameters for a "Bang-Bang" Reachability Graph

Combining the lemma above with Corollary 3.3.4 we immediately get the following.

Corollary 5.1.4 *Let τ be fixed such that $\overline{a}\tau | \overline{v}$. Let Γ_r have velocity function v_r obeying dynamics bounds $(\overline{v} - \frac{\overline{a}\tau}{4})$ and $\frac{\overline{a}}{(1+\epsilon)^2}$.*

Suppose that

$$\begin{aligned} \dot{s}^* &= \text{the multiple of } \overline{a}\tau \text{ closest to } v_r(0) \\ s^* &= v_r(0) - \tfrac{\tau}{2}(\dot{v}_r(0) + \dot{s}^*). \end{aligned}$$

Then, there exists a τ-bang-bang trajectory Γ_q such that

1. *$\Gamma_q(0) = (s^*, \dot{s}^*)$;*

2. *$\Gamma_q(\tau)$ approximates $\Gamma_r(0)$ to tolerance $(\overline{a}\tau^2, \frac{7\overline{a}\tau}{4})$;*

3. *For all $t \in [0, T_r]$, $\Gamma_q(t+\tau)$ tracks $\Gamma_r(t)$ to tolerance $(2\overline{a}\tau^2 + \frac{9(1+\epsilon)^2\overline{a}\tau^2}{4\epsilon(2+\epsilon)}, 2\overline{a}\tau)$; and*

4. *$\Gamma_q(T_r + \tau)$ approximates $\Gamma_r(T_r)$ to tolerance $(3\overline{a}\tau^2, 2\overline{a}\tau)$.*

Furthermore, $\Gamma_q(0)$ is computable from $\overline{a}$, $\overline{v}$, τ, and $\Gamma_r(0)$.

Now, suppose we are given a Cartesian Kinodynamic Planning Problem instance $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ and an approximation parameter ϵ. We can again use the Safe Tracking Lemma (2.2.3) to choose $\Omega(\epsilon)$ ϵ-safe-tracking parameters η_x and η_v, which guarantee that if a trajectory Γ_r is $\delta_v(c_0, c_1)$-safe and Γ_q tracks Γ_r to tolerance (η_x, η_v), then Γ_q will be $(1-\epsilon)\delta_v(c_0, c_1)$-safe. Recalling the Efficient Tracking Lemma 3.3.1, we can apply Corollary 5.1.4 and follow the development of Section 3.3.2 and 3.3.4.1 to choose a root vertex $\mathbf{S}^*$ and an $\Omega(\epsilon)$ timestep τ to determine a reachability graph whose edges are $(1-\epsilon)\delta_v(c_0, c_1)$-safe τ-bangs. Specifically, $\mathbf{S}^*$ and τ can be chosen so that if Γ_{opt} is an optimal solution, then some τ-bang graph trajectory Γ_q, $\Gamma_q^{4\tau}$ will track Γ'_{opt} closely enough to be ϵ-optimal.[6]

Hence, the "bang-bang" algorithm for Cartesian Kinodynamic Planning simply searches such a reachability graph to find an ϵ-optimal trajectory. This is similar to Algorithm 2.

5.1.4 A More General Bang-Bang Algorithm

For a given τ_B, recall that a τ_B-bang-bang trajectory is a bang-bang trajectory with switchings occurring only at multiples of τ_B, and that a τ_B-bang is a constant completely-saturated control of duration τ_B or a trajectory segment arising from such a control and a particular state.

For a given an optimal kinodynamic planning problem for an open-chain manipulator, together with an approximation parameter ϵ, we sketch how to choose timestep τ_B, root vertex $\mathbf{S}^*$, and (b_x, b_v)-bucket pruning dimensions such that if an optimal solution exists, then

1. some τ_B-bang-bang trajectory starting at $\mathbf{S}^*$ tracks the ϵ-time-rescaled trajectory Γ'_r to a sufficiently close tolerance to be ϵ-optimal, and

2. the reachability graph rooted at $\mathbf{S}^*$ and whose edges are $(1-\epsilon)\delta_v$-safe τ_B-bangs will, after a breadth-first (b_x, b_v)-bucket pruning, contain a path corresponding to an ϵ-optimal trajectory.

The **Bang-Bang Algorithm** is identical to the True-Extremal Algorithm (see Section 4.2.1.1), except that it searches a reachability graph generated by τ_B-bangs instead of by $(\overline{\mu}, \tau_E)$-bangs. Below, we show that τ_B and the dimensions of the

[6]Recall that Γ'_r denotes Γ_r ϵ-time-rescaled, and that $\Gamma_q^{4\tau}(t) = \Gamma_q(t + 4\tau)$. See equations (4.5) and (3.1).

pruning buckets, b_x and b_v, are polynomially small in ϵ, and that these parameters result in the algorithm complexity in Theorem 5.1.2.

5.1.4.1 Bang-Bang Trajectories and Average Acceleration Advantages

Suppose that for a given kinodynamic planning problem the True Extremal Algorithm selects an $\Omega(\epsilon)$ timestep τ_E and a control discretization parameter $\overline{\mu}_E$. Suppose that trajectory Γ_r obeys dynamics bounds $\overline{\mathbf{f}}$ and $\overline{\mathbf{v}}$, and recall that Γ'_r is this trajectory ϵ-time rescaled.

Furthermore, suppose that $\tau_B = \frac{\tau_E}{N}$ for some integer N. Over the interval $[n\tau_E, (n+1)\tau_E]$, the average applied force can be approximated to resolution $\frac{\overline{f}_i}{N}$ at each joint i by some bang-bang control $\mathbf{f}_q : [n\tau_E, (n+1)\tau_E] \rightarrow$ *force*, with switchings only occurring at multiples of τ_B. Since the largest possible change in the map from forces to accelerations over the interval is $O(\tau_E)$ is $O(\epsilon)$, it follows that average acceleration under $\mathbf{f}_q$ approximates the average of $\mathbf{a}'_r$ over the interval to resolution $\frac{2A_{max}}{N} + O(\tau_E)$ for each dimension i. (Recall that A_{max} denotes the global acceleration bound and $\mathbf{a}'$ denotes the acceleration function of the ϵ-time-rescaled trajectory Γ_r.)

To put this observation to use, we define an *average acceleration advantage.* Let $\mathcal{P}$ be a function $\mathcal{P}:$ *time* $\rightarrow \{X : X$ *is a set of accelerations*$\}$. Let $\mathcal{Y}$ be a set of acceleration functions. Suppose that for interval $\mathcal{I} = [\tau_I, \tau_I + \Delta\tau_I]$ the following is true: if $\mathbf{a} \in \mathcal{P}(t)$ for all $t \in \mathcal{I}$ and σ is a d-vector of 1's and –1's, then there is some $a^{\mathcal{Y}} \in \mathcal{Y}$ such that

$$\sigma_i \left(\int_{\tau_I}^{\tau_I + \Delta\tau_I} (a_i(t) - a_i^{\mathcal{Y}}(t))dt \right) \geq \kappa_l \Delta\tau_I \tag{5.1}$$

for each i. Then we say that $\mathcal{Y}$ *has an average* κ_l *advantage over* $\mathcal{A}$ *during* $\mathcal{I}$.

5.1.4.2 Tracking With Bang-Bang Trajectories

We now consider a game similar to the Tracking Game of Section 4.3.1, differing in the following way: in each round when you choose HIGH or LOW, the adversary is only obligated to return an acceleration (for you) that has an *average* κ_l advantage over his acceleration for that round. We let τ_G denote the amount of time simulated in one round of this game. We only need to modify step three of round play in the Tracking Game With Perturbations:[7]

[7]See Sections 4.3.1.1 and 4.3.1.4.

3. Let $a_n^\circ = \frac{v_{n+1}-v_n}{\tau_G}$. If you played HIGH, then your adversary can choose any function $a_n'(t)$ (for your acceleration) such that
$$\frac{1}{\tau_G}\int_{n\tau_g}^{(n+1)\tau_G} a_n'(t)dt \geq a_n^\circ + \kappa_l.$$
If instead you played LOW, then your adversary's choice must obey the condition
$$\frac{1}{\tau_G}\int_{n\tau_g}^{(n+1)\tau_G} a_n'(t)dt \leq a_n^\circ - \kappa_l.$$
In both cases, $a_n'(t)$ must obey the condition $|a_n'(t)| \leq A_{max}$ for all $t \in [n\tau, (n+1)\tau]$.

By following the Simple Strategy of Claim 4.3.1, we can track the adversary to a tolerance that is $O\left(\frac{A_{max}^2\tau_G^2}{\kappa_l}, A_{max}\tau_G\right)$.

Suppose we play d simultaneous games, in which the adversary chooses his motions (taken as a trajectory in d-dimension) to match Γ_τ'. Then we can track the adversary to an $O\left(\frac{A_{max}^2\tau_G^2}{\kappa_l}, A_{max}\tau_G\right)$ tolerance in each game. It follows that for a fixed A_{max}, an average κ_l-acceleration advantage, and a desired tracking tolerance tolerance (η_x, η_v) there is always a game timestep τ_G that not only guarantees we can track the adversary to tolerance (η_x, η_v). Moreover, we can find such a τ_G that is $\Omega(\epsilon)$ if κ_l, η_x, and η_v are all $\Omega(\epsilon)$.

Now, suppose the accelerations the adversary chooses for you must obey the original (not ϵ-time-rescaled) dynamics bounds, and that $N\tau_B = \tau_G$ for some integer N. If τ_G is short enough and N is large enough, then for a given round, the set of accelerations composed of N consecutive τ_B-bangs will always contain an acceleration function $\mathbf{a}$ such that in each game i the adversary can return a_i, as long our trajectory tracks Γ_τ' closely enough. Intuitively, we can think of a "chunk" of N τ_B-bangs as "averaging" to approximate the effect of $\mathbf{a}$ a time interval of length τ_G.

Using the techniques from Section 4.3.1.1, we can show that a there is a fundamental tracking tolerance (η_{x0}, η_{v0}), a fundamental timestep τ_0, and a minimal "chunk density" N_0 such that if
$$\begin{aligned} \tau_G &\leq \tau_0, \\ \eta_x &\leq \eta_{x0}, \\ \eta_v &\leq \eta_{v0}, \text{ and} \\ N &\geq N_0 \end{aligned} \tag{5.2}$$

and our trajectory tracks Γ'_r to tolerance (η_x, η_v), then the adversary can legally choose a τ_B-bang-bang acceleration function for us. Moreover, we can find η_{x0}, η_{v0}, and τ_0 that are all $\Omega(\epsilon)$, while still assuring that some $O(\frac{1}{\epsilon})$ chunk density N will be sufficient. This leads us to conclude:

Theorem 5.1.5 (Bang-Bang-Theorem) *Suppose that Γ_r obeys (4.7) and dynamics bounds $\overline{\mathbf{f}}$ and $\overline{\mathbf{v}}$, is δ_v-safe, and takes time T_r. Then for any ϵ, we can choose a timestep τ_B that is $\Omega(\epsilon^2)$ and ensures that there exists some $(1-\epsilon)\delta_v$-safe τ_B-bang-bang trajectory Γ_q such that $\Gamma_q(0) = \Gamma_r(0)$, and $\Gamma_q((1+\epsilon)T_r)$ and $\Gamma_r(T_r)$ are ϵ-close.*

The import of theorem is similar to that of the Coupled Tracking Lemma (4.2.1) with respect to $(\overline{\mu}, \tau)$-extremal trajectories. It says that we can choose a root vertex and a timestep that guarantee some graph trajectory will be ϵ-optimal if an exact solution exists. However, the size to the reachability graph might be exponential in the longest (non-cyclic) path length.

5.1.4.3 Pruning a Bang-Bang Reachability Graph

To show that we can choose reachability graph and search-pruning parameters that guarantee an ϵ-optimal bang-bang trajectory will be found in the pruned search, we again turn to a tracking game.

The Tracking Game With Multi-Perturbations

Consider a game in which you are trying to track an adversary in a one dimensional space. The game is a series of rounds, each of which simulates motions taking time τ_G. The game begins with adversary in state (x_0, v_0) and you in state (x'_0, v'_0). A round simulates an interval of time with length τ_G. When discussing the game, we will use x_n, v_n, etc, to denote your position, your velocity, etc., corresponding to round n.

Let game parameters A_{max} and κ_l such that $A_{max} > \kappa_l > 0$ be given. Let N be an arbitrary positive integer.

In each round the following takes place:

Play for Round n:

1. The adversary plays first by choosing an acceleration function $a_n(t)$ for himself such that for all $t \in [n\tau_G, (n+1)\tau_G]$, $|a_n(t)| \leq A_{max} - \kappa_l$. His state (x_{n+1}, v_{n+1}) at the end of the round (beginning of round $n+1$) is computed

from his state (x_n, v_n) and this acceleration function by integration over the time interval $[n\tau_G, (n+1)\tau_G]$.

2. For your play in round n, you then choose either HIGH and LOW. This choice limits the acceleration function that the adversary can choose *for you*, and that determines your state (x'_{n+1}, v'_{n+1}) at the end of the round. Note that you play only HIGH or LOW; the adversary chooses *your* acceleration *as well as his own*. However, his choice of your acceleration is constrained by your play.

3. Let $a_n^\circ = \frac{v_{n+1}-v_n}{\tau_G}$. If you played HIGH, then your adversary can choose any function $a'_n(t)$ such that
$$\frac{1}{\tau_G}\int_{n\tau_G}^{(n+1)\tau_G} a'_n(t)dt \geq a_n^\circ + \kappa_l.$$
If instead you played LOW, then your adversary's choice must obey the condition
$$\frac{1}{\tau_G}\int_{n\tau_G}^{(n+1)\tau_G} a'_n(t)dt \leq a_n^\circ - \kappa_l.$$
In both cases, $a'_n(t)$ must obey the condition $|a'_n(t)| \leq A_{max}$ for all $t \in [n\tau_G, (n+1)\tau_G]$.

4. Your state (x'_{n+1}, v'_{n+1}) at the end of the round is computed using your state at the end of the previous round and integrating $a'_n(t)$ over the time interval $[n\tau_G, (n+1)\tau_G]$. *However, the adversary can disturb this integration* by perturbing your state by up to $(\frac{\kappa_l \tau_G{}^2}{4N}, \frac{\kappa_l \tau_G}{2N})$ at each time that is a multiple of $\frac{\tau_G}{N}$.

By following the reasoning from the proof of Claim 4.3.3 we get the following.

Claim 5.1.6 *Suppose that you play the Tracking Game With Multi-Perturbations against an adversary and follow the Simple Strategy from Section 4.3.1.1. Suppose that at the start* $|x_0 - x'_0| \leq 2A_{max}\tau_G^2$ *and* $|v_0 - v'_0| \leq 2A_{max}\tau_G$. *Let two trajectories* Γ_r *and* Γ_q *be defined as follows:*

$$\begin{aligned}
\Gamma_r(n\tau_G) &= (x_n, v_n); \\
a_r(t) &= a_n(t - n\tau_G) \quad \textit{for all} \quad t \in (n\tau_G, (n+1)\tau_G); \\
\Gamma_q(n\tau_G) &= (x'_n, v'_n); \\
a_q(t) &= a'_n(t - n\tau_G) \quad \textit{for all} \quad t \in (n\tau_G, (n+1)\tau_G).
\end{aligned}$$

Then for all t:

$$\begin{array}{rcl} |v_r(t) - v_q(t)| & \leq & 4A_{max}\tau_G + \frac{\kappa_l \tau_G}{2}; \\ |x_r(t) - x_q(t)| & < & \frac{(8A_{max}+\kappa_l)^2 \tau_G^2}{4\kappa_l} + 6A_{max}\tau_G^2 + \frac{\kappa_l \tau_G^2}{2}. \end{array} \tag{5.3}$$

This claim and the way the game allows the adversary to perturb your movement round correspond to a generalization of the Robust Tracking Lemma. We first describe a mapping from sequences of states to acceleration functions.

Let $\mathcal{Q}$ be a function $\mathcal{Q} : TC \to \{\mathcal{Y} : \mathcal{Y} \textit{ is a set of acceleration functions}\}$, let N be a positive integer, and let T be a duration of time. Let $\mathcal{X}$ be a sequence of states $\mathbf{X}_0, \ldots, \mathbf{X}_{N-1}$. We define the function

$$\mathcal{Q}^*_{\mathcal{X},N,T} : TC \to \{\mathcal{Y} : \mathcal{Y} \textit{ is a set of acceleration functions}\}$$

as follows. Let Γ be a trajectory of length T starting at a given state X_0, and let $\mathbf{a}$ be the associated acceleration. Then $\mathbf{a} \in \mathcal{Q}^*_{\mathcal{X},N,T}(\mathbf{X}_0)$ if and only if for each $n = 0, \ldots, N-1$ there is some $\mathbf{a}^n \in \mathcal{Q}(\mathbf{X}_n)$ such that $\mathbf{a}(t) = \mathbf{a}^n(t - \frac{nT}{N})$ for all $t \in [nT, (n+1)T]$.

Lemma 5.1.7 (Another Robust Coupled Tracking Lemma) *Let $\mathcal{P}$ and $\mathcal{Q}$ be functions $\mathcal{P}, \mathcal{Q} : TC \to \{\mathcal{Y} : \mathcal{Y}$ is a set of acceleration functions$\}$. Let $\mathcal{P}$ have the property that if $\mathbf{a} \in \mathcal{P}(\mathbf{X})$ for some state $\mathbf{X}$, then for all times t, $|\mathbf{a}(t)|_\infty \leq A_{max} - \kappa_l$. Let $\mathcal{Q}$ have the properties that $\mathcal{Q}(\mathbf{X})$ is finite for all $\mathbf{X} \in TC$ and that if $\mathbf{a} \in \mathcal{Q}(\mathbf{X})$, then for all times t $\|\mathbf{a}(t)\|_\infty \leq A_{max}$.*

Suppose that there exist a maximal tracking tolerance (η_{x0}, η_{v0}), a fundamental timestep τ_0, and a chunk density N_0 such that for all $\tau \leq \tau_0$ and $N \geq N_0$ the following is true: if the states in the sequence $\mathcal{X} = \mathbf{X}_0, \ldots, \mathbf{X}_{N-1}$ are all within (η_{x0}, η_{v0}) of state $\mathbf{Y}$, then the set of acceleration functions $\mathcal{Q}^*_{\mathcal{X},N,\tau}(\mathbf{X})$ has an average κ_l acceleration advantage over $\mathcal{P}(\mathbf{Y})$ over duration τ.

Let (η_x, η_v) be an L_∞ tracking tolerance, and let timestep τ obey the following inequalities:

$$\begin{array}{rcl} \tau & \leq & \tau_0; \\ \tau & \leq & \sqrt{\frac{4\min(\eta_{x0},\eta_x)\kappa_l}{24A_{max}\kappa_l + 2\kappa_l^2 + (8A_{max}+\kappa_l)^2}}; \quad \textit{and} \\ \tau & \leq & \frac{2\min(\eta_{v0},\eta_v)}{8A_{max}+\kappa_l}. \end{array} \tag{5.4}$$

Let be $\mathcal{G}$ the $(\mathcal{Q}, \tau)$-reachability graph rooted at some $\mathbf{S}^*$.

Now, suppose that Γ_r respects $\mathcal{P}$, $\Gamma_r(0)$ is within $2A_{max}\tau$ of $\dot{\mathbf{s}}^*$, and $\mathbf{p}_r(0) = \mathbf{s}^*$. Suppose that $\mathcal{R}$ is a subset of TC containing the (η_x, η_v)-tube induced by Γ_r, and that $\mathcal{G}'$ results from some breadth-first $(\frac{\kappa_l \tau^2}{4N}, \frac{\kappa_l \tau}{2N})$-pruning of the maximal subgraph of $\mathcal{G}$ that lies in $\mathcal{R}$.

Then there is a Γ_q in $\mathcal{G}'$ such that $T_q \leq T_r + \tau$, $\Gamma_q(0) = \mathbf{S}^*$, and $\Gamma_q(T_q)$ approximates $\Gamma_r(T_r)$ to tolerance (η_x, η_v).

Proof: (Outline.) Similar to the proof of the Robust Coupled Tracking Lemma (4.2.2). We can consider d simultaneous playings of the Coupled Tracking Game With Multi-Perturbations where the timestep τ_G in the game is the same as τ in the lemma, the adversary's choice of our acceleration corresponds to $\mathcal{Q}$, and perturbations correspond to bread-first prunings. Again, *some* trace of d simultaneous playings of the Tracking Game with Multi-Perturbations in which you follow the Simple Strategy corresponds to a "virtual path" in the pruned reachability graph, where a "virtual edge" of zero cost goes from each pruned vertex to the vertex that caused its pruning. (See figure 5.3.) □

Finally, to prove Theorem 5.1.2 we consider a given problem and approximation parameter ϵ. We choose $\mathcal{Q}$ to map states to accelerations determined by constant, fully-saturated controls. Using the arguments above from the previous chapter, we can show that there exist consistent parameters η_{x0}, η_{v0}, κ_l, and τ_0 that are $\Omega(\epsilon)$, and an $O(\epsilon)$ timestep chunk density N_0 that satisfy the hypothesis of the lemma. The Safe Tracking Lemma would be used to find ϵsafe-tracking tolerances η_x and η_v that are $\Omega(\epsilon)$. Using Lemma 5.1.7, we choose an $\Omega(\epsilon)$ "chunk" of time τ_C and the timestep

$$\tau_B = \frac{\tau_C}{N},$$

where $N \geq N_0$, for the algorithm.

Then, if Γ_{opt} is an optimal kinodynamic solution, a breadth-first search of the reachability graph with $(\frac{\kappa_l \tau_B^2}{4N}, \frac{\kappa_l \tau_B}{2N})$-pruning would find a path that is ϵ-optimal. Recalling the argument outlined above, we see that some virtual path in the pruned reachability graph would correspond to a trajectory that tracks Γ'_{opt} closely enough to be $(1-\epsilon)$-safe and come "ϵ-close" tp the goal state. Since κ_l and τ_C are $\Omega(\epsilon)$ and N is $O(\frac{1}{\epsilon})$, the pruning bucket dimensions b_x and b_v are $\Omega(\epsilon^4)$ and $\Omega(\epsilon^3)$ for each degree of freedom.[8]

[8]Here N again is the chunk size and not the geometric complexity of the problem.

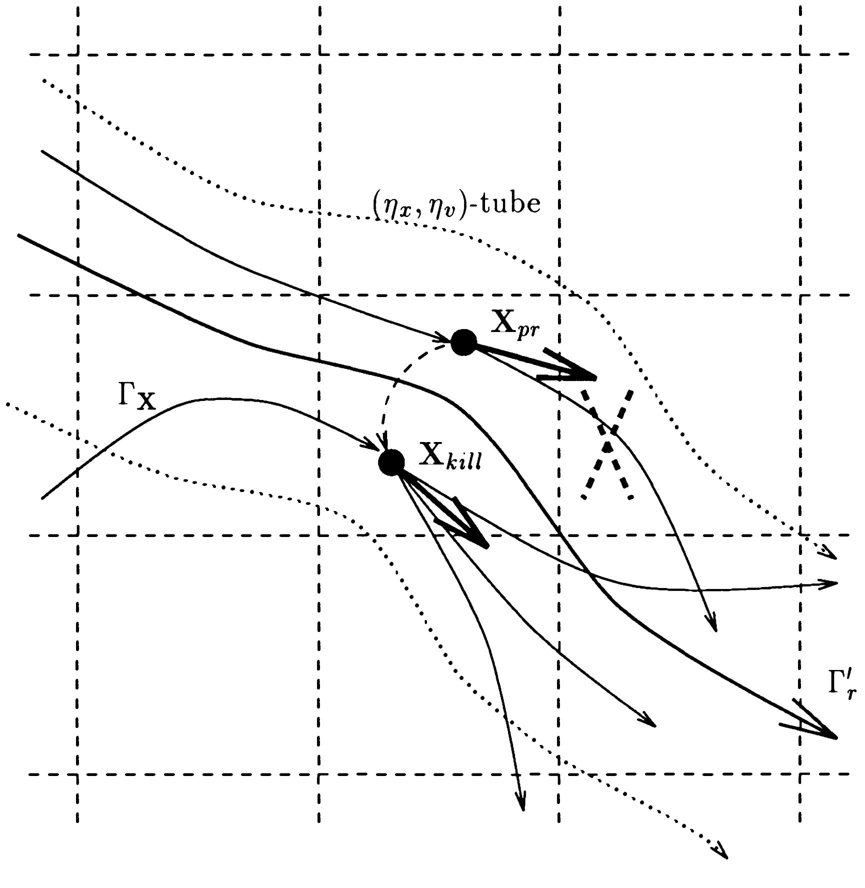


Figure 5.3: Reachability graph pruning and the Tracking Game with Multi-Perturbations. In a breadth-first reachability-graph search with (b_x, b_v)-bucket pruning, $\mathbf{X}_{kill}$ causes edges from $\mathbf{X}_{pr}$ to be pruned from the search. However, some trace of d simultaneous playings of the Tracking Game with Multi-Perturbations in which you follow the Simple Strategy corresponds to a virtual path containing a "virtual edge" of zero cost going from $\mathbf{X}_{pr}$ to $\mathbf{X}_{kill}$.

Absorbing robot-dependent terms into a constant c_F, it follows that the number of edges and vertices in the reachability graph searched by the algorithm will be $O(c_F^d(\frac{1}{\epsilon})^{7d})$.

5.2 Time-Safety Trade-Offs

5.2.1 A Finer Approximation Model

The original theoretical formulation of kinodynamic planning treats time- and safety- optimality as a single quantity and considers the asymptotic computational complexity of the algorithm as a function of the single approximation parameter ϵ. When we look towards using provably good approximation techniques in practice, however, we observe that the total time needed to plan *and* execute a trajectory is often more important than the time needed for plan execution alone. At the same time, it is often more important to guarantee that the planned trajectory will be "reasonably" fast and observe the safety margin closely than to guarantee it will be nearly time-optimal. We therefore need to understand analytically how execution time can be "traded-off" against safety and planning time while still requiring the approximation to be provably good. Thus, the hope of reaping practical rewards from theoretical developments motivates us to study how safety and time each fit into the structure of kinodynamic planning.

Our planning algorithms allow one to choose the optimum compromise between time and safety. Let ϵ_T and ϵ_S describe closeness to optimality in execution time and observance of the safety margin, respectively. A *fixed* computational complexity defines a bounding *equicomplexity* curve of (ϵ_S, ϵ_T) pairs that can be satisfied without exceeding the complexity. (See figures 5.4 through 5.7.) As the complexity varies, it parameterizes a family of such curves, which allows us to trade off ϵ_S, ϵ_T, and planning time against each other. Examples of such curves graphically illustrate the impact of trade-offs on the size of algorithm search spaces. A straightforward analysis is developed to cover the algorithms from the previous chapters.

5.2.1.1 Approximation Closeness, More Precisely

We now specify what it means for one kinodynamic solution to "approximate" another in terms of error parameters ϵ_T (speed) and ϵ_S (safety). We can regard the analysis in earlier chapters as having let

$$\epsilon = \epsilon_S = \epsilon_T \tag{5.5}$$

to simplify the problem formulation.[9]

First, we say a solution Γ_q is ϵ_T-time-optimal if the time T_q it requires required is bounded above by $(1+\epsilon_T)T_{opt}$, where T_{opt} is the time required by the optimal trajectory. Second, for a given safety function $\delta_v(c_0, c_1)$, we say Γ_q is ϵ_S-safety-optimal if the path $\mathbf{p}_q$ lies in an obstacle-free safety tube of diameter $(1-\epsilon_S)\delta_v$. Third, letting $\epsilon_{max} = \max(\epsilon_S, \epsilon_T)$, we say that a trajectory $\Gamma_q : [0, T_q] \to TC$ is "ϵ_{max}-close" at the start $\mathbf{S}$ (goal $\mathbf{G}$) if, at time 0 (time T_q), it achieves the start (goal) position to within a $O(\epsilon_{max})$ additive error, and the start (goal) velocity to within an $O(\epsilon_{max})$ multiplicative error plus a constant $O(\epsilon_{max})$ additive error. For example, if $\Gamma_q = (\mathbf{p}_q, \dot{\mathbf{p}}_q)$, then both $\|\mathbf{p}_q(0) - \mathbf{s}\|_\infty$ and $\|(1+\epsilon_{max})\dot{\mathbf{p}}_q(0) - \dot{\mathbf{s}}\|_\infty$ are[10] $O(\epsilon_{max})$. Finally, we define an (ϵ_S, ϵ_T)-*optimal solution* (relative to $\delta_v(c_0, c_1)$-safety) to be a solution Γ_q which is ϵ_S-safety-optimal, ϵ_T-time-optimal, and "ϵ_{max}-close" at the start and goal.

5.2.1.2 Alternative Approximation Formulations

Note that the notion of approximate solution to an optimal kinodynamic planning problem can be extended to observance of the dynamics bounds (4.4). For example, consider a Cartesian Robot. We would say that a trajectory obeys acceleration bound $\overline{a}$ to within ϵ_A if it obeys acceleration bound $(1+\epsilon_A)\overline{a}$. Similarly, a trajectory obeys velocity bound $\overline{v}$ to within ϵ_V if it obeys acceleration bound $(1+\epsilon_V)\overline{v}$. If a trajectory obeys the acceleration and velocity bounds to within ϵ_A and ϵ_V, we say it is (ϵ_A, ϵ_V)-*executable*, since it can be executed by a robot whose acceleration and velocity bounds are greater by factors of ϵ_A and ϵ_V, respectively. At the same time, another parameter ϵ_E could characterize how closely the solution comes to meeting the desired trajectory endpoint conditions.

The parameters ϵ_A and ϵ_V would measure how effectively the planner uses the dynamical capabilities of the robot in the following manner. Suppose that whenever an exact solution to a safe kinodynamic planning problem $(\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, \delta_v)$ taking time T *exists*, a certain planner is guaranteed to find a solution to the related

[9] To simplify things, we let ϵ_T absorb state error at the start and goal .

[10] This last choice has been made with the anticipation that discretization will determine the closeness of the approximation. It is a simple way to characterize the worst-case accuracy, and a closer look at the analysis that follows reveals that the additive errors are in fact $O(\sqrt{\epsilon_S \epsilon_T})$.

problem $(\mathcal{O}, \mathbf{S}, \mathbf{G}, (1+\epsilon_A)\overline{a}, (1+\epsilon_V)\overline{v}, (1-\epsilon_S)\delta_v)$ that also takes time T. Then that planner would "use" at least $\frac{1}{1+\epsilon_A}$ and $\frac{1}{1+\epsilon_V}$ of the robot's acceleration and velocity capabilities.

Since the planned trajectory can make use of $\epsilon_V\overline{v}$ and $\epsilon_A\overline{a}$ dynamics advantages over the bounds obeyed by the optimal trajectory, our algorithms and proof techniques extend to this problem formulation. However, we will only discuss (ϵ_S, ϵ_T)-optimal kinodynamic planning here.

5.2.2 Algorithms That Allow Compromise

To search for (ϵ_S, ϵ_T)-optimal trajectories, we us algorithms from Chapters 3 and 4 to search reachability graphs almost exactly as before. Graph edges now obey $(1-\epsilon_S)\delta_S$-safety, and other parameter choices are re-derived, using ϵ_S and ϵ_T appropriately instead of ϵ. We describe how this is done and revise our complexity analysis.

5.2.2.1 Parameter Choices For Trade-Offs

For a fixed kinodynamic planning problem and given approximation parameters ϵ_S and ϵ_T, the complexity of Algorithm 2 and 3 (Chapter 3) for Cartesian Kinodynamic planning is determined by the size of the reachability graph that the algorithm searches. This size, in turn, depends on the timestep τ chosen by the algorithm. The timestep chosen therefore shrinks with both ϵ_S and ϵ_T. For the more general Near-Extremal and True-Extremal Algorithms (Chapter 4), reachability graph degree depends on the size of the force (or acceleration) discretization parameter $\overline{\mu}$ (or $\boldsymbol{\mu}$), which shrinks only with ϵ_T.

An algorithm for finding (ϵ_S, ϵ_T)-optimal trajectories uses the Safe Tracking Lemma the same way the related algorithm for finding ϵ-optimal trajectories does. Given $\epsilon_S > 0$ and δ_v-safety parameters c_0 and c_1, the lemma specifies a one-parameter family of ϵ_S-safe-tracking tolerances (η_x, η_v):

$$\begin{aligned} \eta_v(\beta) &\leq \tfrac{c_0\epsilon_S}{c_1(1-\epsilon_S)+\beta} \\ \eta_x(\beta) &\leq \beta\eta_v. \end{aligned} \tag{5.6}$$

Next, the algorithm chooses reachability graph parameters that guarantee that if an exact solution exists, for some optimal trajectory Γ_{opt}, a graph trajectory Γ_q will track Γ'_{opt} to a safe-tracking tolerance[11] (η_x, η_v). For a given tracking toler-

[11]For a given trajectory Γ_r, we now let Γ'_r denote Γ_r ϵ_T-time-rescaled.

ance and ϵ_T, the appropriate tracking lemma(s) be used to choose an adequately small timestep,[12] with ϵ_T substituted for ϵ. However, to minimize the size of the reachability graph, the algorithm must maximize the timestep τ, which requires choosing the parameter β in equation (5.6) appropriately.

5.2.2.2 Maximizing the Timestep

The technique from Section 3.3.4.1 can be used to choose β in equation (5.6) to maximize the timestep required by the appropriate tracking lemma. Recalling that each tracking lemma (*i.e.*, Lemma 3.3.1, 3.3.9, or 4.2.1) dictates that the timestep τ to be at most the minimum of an expression dependent on η_x and another dependent on η_v, we define functions $\tau_x(\beta)$ and $\tau_v(\beta)$ and consider their values for β that equalize them.

Let us consider an algorithm for Cartesian Kinodynamic planning that is a "hybrid" of Algorithms 2 and 3. We will refer to this as the **Hybrid Algorithm for Cartesian Kinodynamic Planning**. It simply computes the timestep that would be chosen by each algorithm according to ϵ_S and ϵ_T, computes an upper bound on the size of the reachability graph that would be searched by each algorithm, and executes the one for which the bound is lower.

Corresponding to Algorithm 2, we have

$$\begin{aligned} \tau_{x2}(\beta) &\le \sqrt{\tfrac{4\epsilon_T(2+\epsilon_T)\eta_x(\beta)}{3\overline{a}(3+10\epsilon_T+5\epsilon_T^2)}}; \\ \tau_{v2} &\le \tfrac{4\eta_v(\beta)}{7\overline{a}}; \quad \text{and} \\ \tau_{max2}(\beta) &= \min_{\beta>0}(\tau_{x2}(\beta), \tau_{v2}(\beta)) \end{aligned} \tag{5.7}$$

We consider the general case for maximizing the timestep chosen by any one of our algorithms. For arbitrary constants $A_1, \ldots, A_5$, and B, and with $\eta_x(\beta)$ and $\eta_v(\beta)$ given by (5.6), let us specify:

$$\begin{aligned} \tau_x(\beta) &\le \sqrt{\tfrac{(A_1\epsilon_T^2+A_2\epsilon_T)\eta_x(\beta)}{A_3\epsilon_T^2+A_4\epsilon_T+A_5}}; \\ \tau_v &\le B\eta_v(\beta); \quad \text{and} \\ \tau_{max} &= \min_{\beta>0}(\tau_x(\beta), \tau_v(\beta)) \end{aligned} \tag{5.8}$$

[12] The appropriate tracking lemma (*i.e.*, Lemma 3.3.1, 3.3.9, or 4.2.1) matches the robot dynamics while allowing the largest timestep.

Consulting *Mathematica*, we find

$$\tau = \frac{2Bc_0\epsilon_S}{c_1(1-\epsilon_S) + \sqrt{c_1^2(1-\epsilon_S)^2 + \frac{4B^2c_0\epsilon_S(A_3\epsilon_T^2+A_4\epsilon_T+A_5)}{A_1\epsilon_T^2+A_2\epsilon_T}}} \tag{5.9}$$

It follows that if

$$\frac{A_1\epsilon_T^2 + A_2\epsilon_T}{A_3\epsilon_T^2 + A_4\epsilon_T + A_5} \leq \frac{4B^2c_0\epsilon_S}{c_1^2(1-\epsilon_S)^2}, \tag{5.10}$$

then

$$\tau_{max} = o\left(\frac{c_0\epsilon_S(A_1\epsilon_T^2 + A_2\epsilon_T)}{A_3\epsilon_T^2 + A_4\epsilon_T + A_5}\right); \tag{5.11}$$

otherwise

$$\tau_{max} = o\left(\frac{2Bc_0\epsilon_S}{c_1(1-\epsilon_S)}\right). \tag{5.12}$$

(Recall that "$o(exp)$" means "larger than $c * exp$ for some constant $c > 0$.")

Substituting in the appropriate terms, we obtain

$$\tau_{max2} = \frac{8c_0\epsilon_S}{\overline{a}\left(7c_1(1-\epsilon_S) + \sqrt{49{c_1}^2(1-\epsilon_S)^2 + \frac{48c_0\epsilon_S(3+10\epsilon_T+5\epsilon_T^2)}{\overline{a}\epsilon_T(2+\epsilon_T)}}\right)}. \tag{5.13}$$

which simplifies to (3.10) when $\epsilon = \epsilon_S = \epsilon_T$.

5.2.3 Bounds For Trade-off Algorithms

5.2.3.1 Trade-offs Basic Kinodynamic Planning

Returning to the Hybrid Algorithm for Cartesian Kinodynamic Planning, we now tighten Algorithm 3's bound on the timestep to reflect the tighter bounds obeyed by a rescaled trajectory:

$$\tau_{max3} = \frac{c_0\epsilon_S(1+\epsilon_T)}{c_1\frac{2\overline{a}(1-\epsilon_S)}{1+\epsilon_T} + 5\overline{v}}. \tag{5.14}$$

We conclude:

Theorem 5.2.1 *Let $\mathcal{K} = (\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ be an optimal kinodynamic planning problem for a polyhedral Cartesian robot obeying L_∞ dynamics bounds. Let $0 < \epsilon_S < 1$, and let $\epsilon_T \geq 0$.*

Then if $\mathcal{K}$ has a solution, the Hybrid Algorithm for Cartesian Kinodynamic Planning derived from Algorithms 2 and 3 finds an (ϵ_S, ϵ_T)-approximately optimal safe solution in time

$$O\left(c^d N \left(\frac{\overline{v} l}{\overline{a}^2 \tau^3}\right)^d\right),$$

where N is the geometric complexity of the problem, c is a robot-dependent constant, and τ is the largest timestep such that $\tau \leq \frac{\overline{v}}{\overline{a}}$ and

$$\tau \;\leq\; \max\left(\frac{8c_0\epsilon_S}{\overline{a}\left(7c_1(1-\epsilon_S)+\sqrt{49{c_1}^2(1-\epsilon_S)^2+\frac{48c_0\epsilon_S(3+10\epsilon_T+5\epsilon_T^2)}{\overline{a}\epsilon_T(2+\epsilon_T)}}\right)}, \frac{c_0\epsilon_S(1+\epsilon_T)}{\left(c_1\frac{2\overline{a}(1-\epsilon_S)}{1+\epsilon_T}+5\overline{v}\right)}\right). \tag{5.15}$$

A corollary describes the asymptotic $(\epsilon_S, \epsilon_T \to 0)$ behavior of the hybrid algorithm.

Corollary 5.2.2 *Suppose that τ_{max2} and τ_{max3} as given in (5.13) and (5.13) are both less than $\frac{\overline{v}}{\overline{a}}$. Then the following describe the asymptotic complexity of the Hybrid Algorithm for Cartesian Kinodynamic Planning:*

1. *As long as $\tau_{max2} \leq \tau_{max3}$ and*

   $$\frac{\epsilon_T(2+\epsilon_T)}{(3+10\epsilon_T+5\epsilon_T^2)} \leq \frac{48c_0\epsilon_S}{49{c_1}^2(1-\epsilon_S)^2},$$

   the complexity of the algorithim is

   $$O(c^d N (\frac{1}{\epsilon_S\epsilon_T})^{\frac{3d}{2}})$$

   as ϵ_S and ϵ_T go to zero.

2. *Otherwise, the complexity of the algorithm is*

   $$O(c^d N (\frac{1}{\epsilon_S})^{3d})$$

   as ϵ_S and ϵ_T go to zero.

In both cases, N is the geometric complexity of the obstacles, and c is a robot-dependent constant.

5.2.3.2 Trade-offs Lemmas For Robots With Coupled Dynamics Bounds

For Cartesian Robots with L_2-norm dynamics bounds, we simply alter the choice of discretization parameter μ and timestep τ in the Near-Extremal Algorithm (Chapter 4). Acceleration advantage κ_l and discretization μ are given by

$$\kappa_l = \mu = \frac{\overline{a}\epsilon_T}{4\sqrt{d}} \tag{5.16}$$

Recalling the Coupled Tracking Lemma (4.2.1) and the development in Section 4.3.2.2, we find that the choice of timestep (4.40) is replaced by

$$\tau = \min\left(\frac{\overline{v}\epsilon_T}{4\overline{a}}, \frac{c_0\epsilon_S}{2\overline{a}\left(c_1(1-\epsilon_S)+\sqrt{c_1^2(1-\epsilon_S)^2+\frac{c_0\epsilon_S(16\sqrt{d}+3\epsilon_S)}{2\overline{a}\epsilon_T}}\right)}\right). \tag{5.17}$$

The minimally modified algorithm is called the L_2 **Cartesian Algorithm with Trade-offs.** Straightforward computation yields the following lemma.

Lemma 5.2.3 *Let $\mathcal{K} = (\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{a}, \overline{v}, l, c_0, c_1)$ be an optimal kinodynamic planning problem for a Cartesian robot with L_2-norm dynamics bounds. Then the complexity of the L_2 Cartesian Algorithm With Trade-offs applied to this problem is $O(c^d N(\frac{1}{\epsilon_T})^{6d-1})$ when*

$$\frac{\overline{v}\epsilon_T}{4\overline{a}} < \frac{c_0\epsilon_S}{2\overline{a}\left(c_1(1-\epsilon_S)+\sqrt{c_1^2(1-\epsilon_S)^2+\frac{c_0\epsilon_S(16\sqrt{d}+3\epsilon_S)}{2\overline{a}\epsilon_T}}\right)}.$$

Otherwise, the complexity is $O(c^d N(\frac{1}{\epsilon_T})^{\frac{9d}{2}-1}(\frac{1}{\epsilon_S})^{\frac{3d}{2}})$ if

$$\frac{3\epsilon_T}{16\sqrt{d}+3\epsilon_T} \leq \frac{3c_0\epsilon_S}{2\overline{a}c_1^2(1-\epsilon_S)^2},$$

and $O(c^d N(\frac{1}{\epsilon_T})^{3d-1}(\frac{1}{\epsilon_S})^{3d})$ if not. In both cases, N is the geometric complexity of the obstacles, and c is a robot-dependent constant.

The three complexity expressions correspond to how the algorithm chooses its timestep. First, when the safety approximation constraints are weak enough, the need to obey the velocity bound $\overline{v}$ appears to determine the timestep; this is caused by the algorithm's exclusive use of near-extremal accelerations. Second, when the constant and speed-dependent parts of the safety function are "balanced relative to ϵ_S and ϵ_T", ϵ_S and ϵ_T both contribute significantly to the choice of timestep. Finally, when the constant part dominates, ϵ_S dominates the timestep choice.

For open-kinematic chain robot, the choice of timestep for the corresponding **Near-Extremal Algorithm With Trade-offs** would be very complicated. However, we can still characterize the behavior of the algorithm's complexity. Recalling the origin of fundamental tracking-tolerance parameters η_{x0} and η_{v0} and timestep τ_0, we see that these parameters are $\Omega(\epsilon_T)$. Similarly, acceleration advantage κ_l is also $\Omega(\epsilon_T)$. Referring to the Coupled Tracking Lemma and equations (5.6) (5.8), we obtain:

Lemma 5.2.4 *Let $\mathcal{K} = (\mathcal{O}, \mathbf{S}, \mathbf{G}, \overline{\mathbf{f}}, \overline{\mathbf{v}}, l, c_0, c_1)$ be an optimal kinodynamic planning problem for a open kinematic chain robot. Then the complexity Near-Extremal Algorithm With Trade-offs*[13] *for this problem is $O(c^d N(d+\log N)(\frac{1}{\epsilon_T})^{6d-1})$ when ϵ_T is sufficiently small compared to ϵ_S. Otherwise, there exist positive, robot-dependent constants A_1, A_2, which are independent of N, such that the complexity is*

$$O(c^d N(d+\log N)(\frac{1}{\epsilon_T})^{\frac{9d}{2}-1}(\frac{1}{\epsilon_S})^{\frac{3d}{2}})$$

if

$$\frac{\epsilon_T}{A_2+\epsilon_T} \le \frac{c_0\epsilon}{A_1 c_1^2(1-\epsilon_S)^2},$$

and $O(c^d N(\frac{1}{\epsilon_T})^{3d-1}(\frac{1}{\epsilon_S})^{3d})$ otherwise. In both cases, N is the geometric complexity of the obstacles and c is a robot-dependent constant.

The complexity describes three ranges of behavior. In the first, either (a) the timestep chosen is the fundamental timestep (see Section 4.2.2.4), which is depends on ϵ_T but not ϵ_S, or (b) the need to obey the velocity bound $\overline{\mathbf{v}}$ appears to determine the timestep. As with the L_2 Cartesian algorithm, when the speed-dependent part of the safety margin is significant, neither nor ϵ_S and ϵ_T dominates the timestep. Again, when the constant part dominates, the ϵ_S dominates the timestep choice.

5.2.4 Complexity Surfaces and Equicomplexity Curves

Figures 5.6 and 5.7 show the computational complexity of the Hybrid Cartesian Kinodynamic Planning Algorithm and the L_2 Cartesian Algorithm with Trade-Offs

[13]This is the Near-Extremal Algorithm from Chapter 4 with the modifications sketched above.

as a function of the approximation parameters ϵ_S and ϵ_T. The complexity (vertical) axis is a $\log_{10}$ scale, so the figures reveal how changing the parameters slightly can greatly affect the running time of the algorithms, especially when parameters are small. In particular, changing ϵ_S or ϵ_T by a factor of two can change the running times of the algorithms by orders of magnitude. Of course, as either ϵ_S or ϵ_T gets close to zero, the complexity grows enormously.

The contour lines in Figures 5.4 and 5.5 illustrate how a fixed discretization or, equivalently, a fixed computational complexity defines a one-parameter family of solutions to a given kinodynamic planning problem. Each solution can be classified by its safety approximation ϵ_S its time approximation ϵ_T. Hence a fixed discretization defines a curve of solutions (really equivalence classes) in the ϵ_S-ϵ_T plane. These "equicomplexity curves" sweep out a surface in complexity-ϵ_S-ϵ_T space which, for a given problem instance, completely characterizes the guaranteed behavior of the algorithm and its solutions. Hence, we can move along a contour line to "trade off" ϵ_S between ϵ_T at a fixed complexity bound, or we can move across contour lines to alter the running time of the planner.

Finally, we note that an equicomplexity curve can be the composite of two level curves corresponding to the different expressions that can determine the timestep. (Say, the two determined by the timestep choices from which the minimum is taken in (5.17).) Furthermore, if we consider equation (5.9) and fix τ, we find that the ϵ_T terms are "separable", and with algebraic manipulation we can reduce the problem of expressing ϵ_T in terms of ϵ_S and τ to a quadratic equation:

$$4B^2(A_3\epsilon_T^2 + A_4\epsilon_T + A_5) = (A_1\epsilon_T^2 + A_2\epsilon_T)(4Bc_0((B + c_1\tau)\epsilon_S - c_1\tau)). \tag{5.18}$$

5.2.5 Using Trade-offs with an Implementation

We illustrate time-safety trade-offs with trajectories found by our planner. Although the planner has been used on more interesting problems, such as the one in Figure 5.8, simple examples are sufficient to show how trajectories found by the planner change with ϵ_S and ϵ_T, and how the planner running time also changes with these parameters.

In Figure 5.9a we see a simple environment with a start state in the lower left and a goal state in the upper right. For the examples, we used the kinodynamic bounds: $\overline{a} = 0.2$, $\overline{v} = 0.3$, $c_0 = 0.2$, and $c_1 = 0.5$. With $\epsilon_S = \epsilon_T = 0.6$, the planner

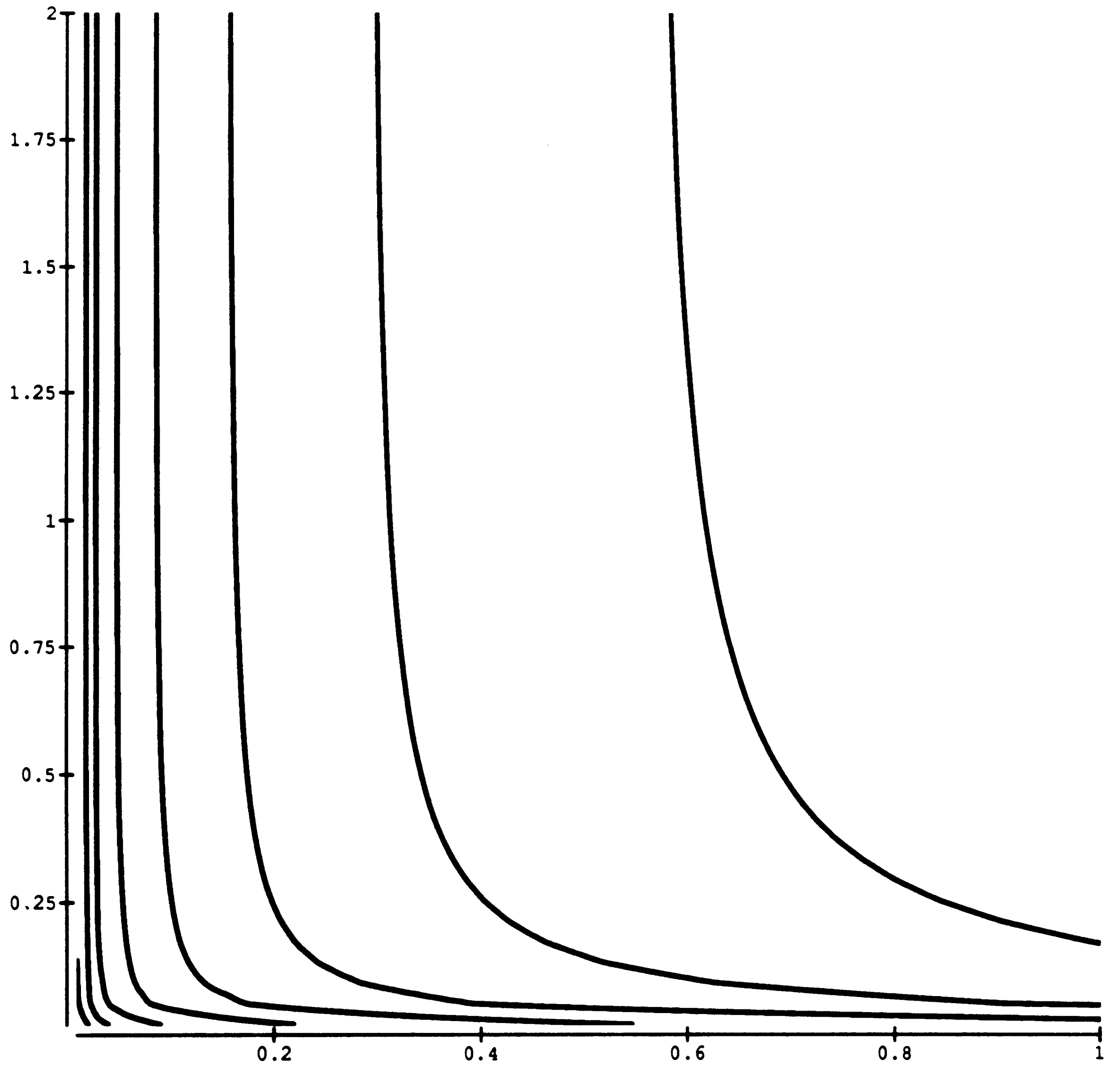


Figure 5.4: Example of a contour map of computational complexity as a function of ϵ_S and ϵ_T for the Hybrid Algorithm for 2D Cartesian Kinodynamic Planning (sketched in section 5.2.2.1). Lines represent both constant complexity and constant discretization.

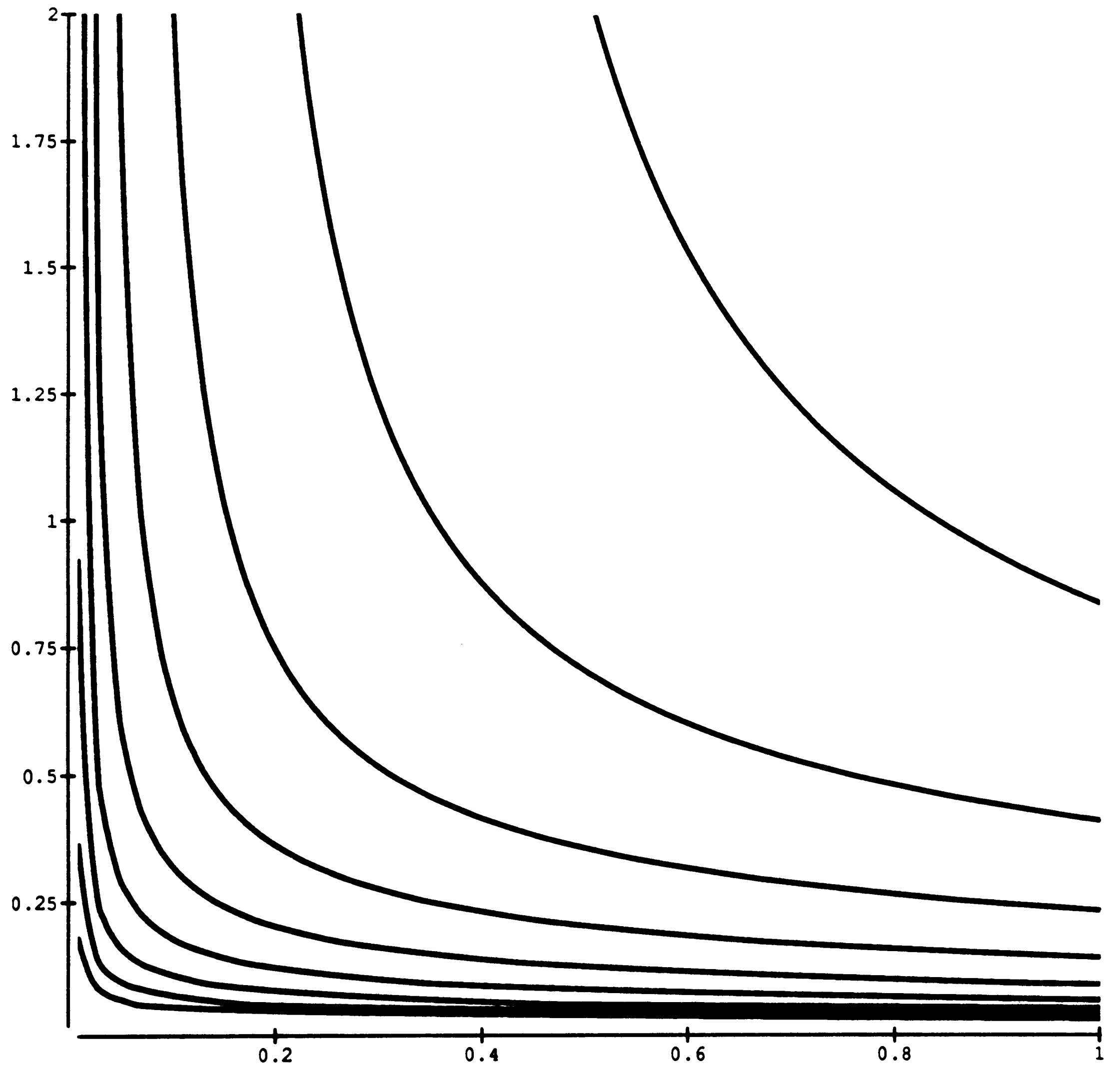


Figure 5.5: Example of a contour map of computational complexity as a function of ϵ_S and ϵ_T for the L_2 Cartesian Algorithm with Trade-offs .

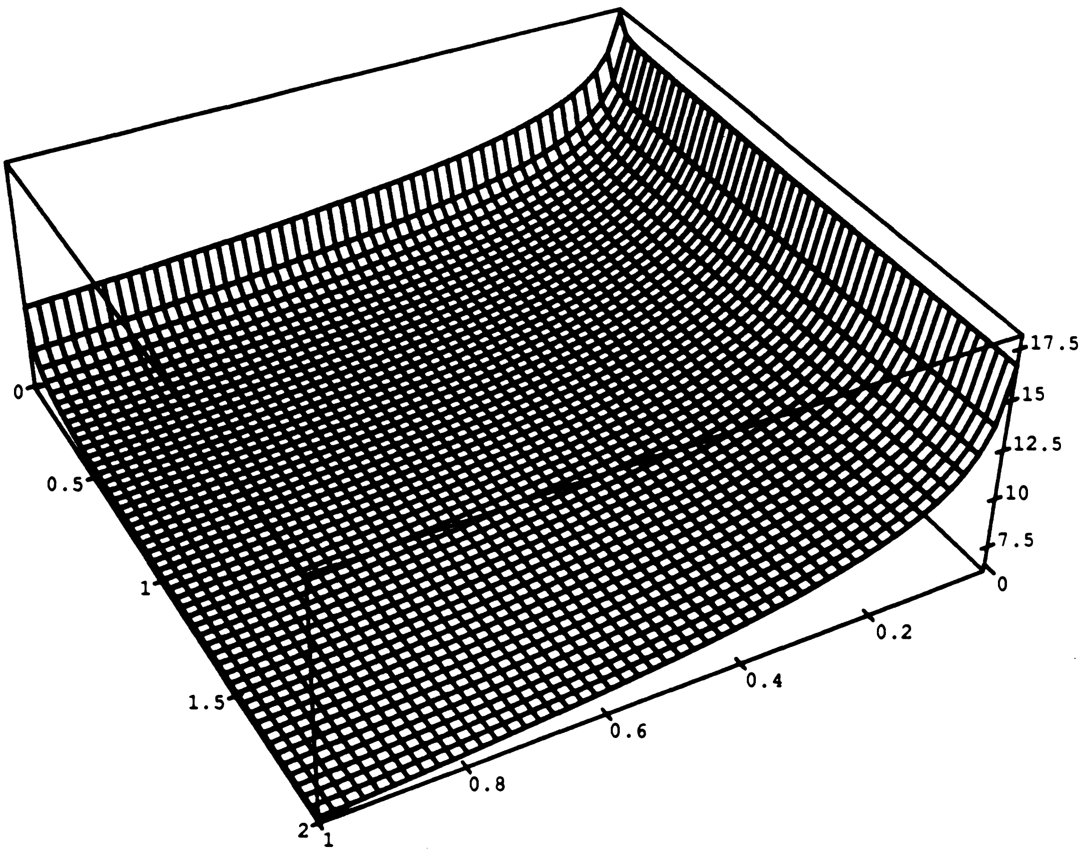


Figure 5.6: Example of a three-dimensional plot of computational complexity as a function of ϵ_S and ϵ_T for the Hybrid Algorithm for 2D Cartesian Kinodynamic Planning (sketched in Section 5.2.2.1). For a given problem instance, such a surface completely characterize the guaranteed the behavior of the algorithm and its solutions. Complexity (vertical) axis is $\log_{10}$ scale.

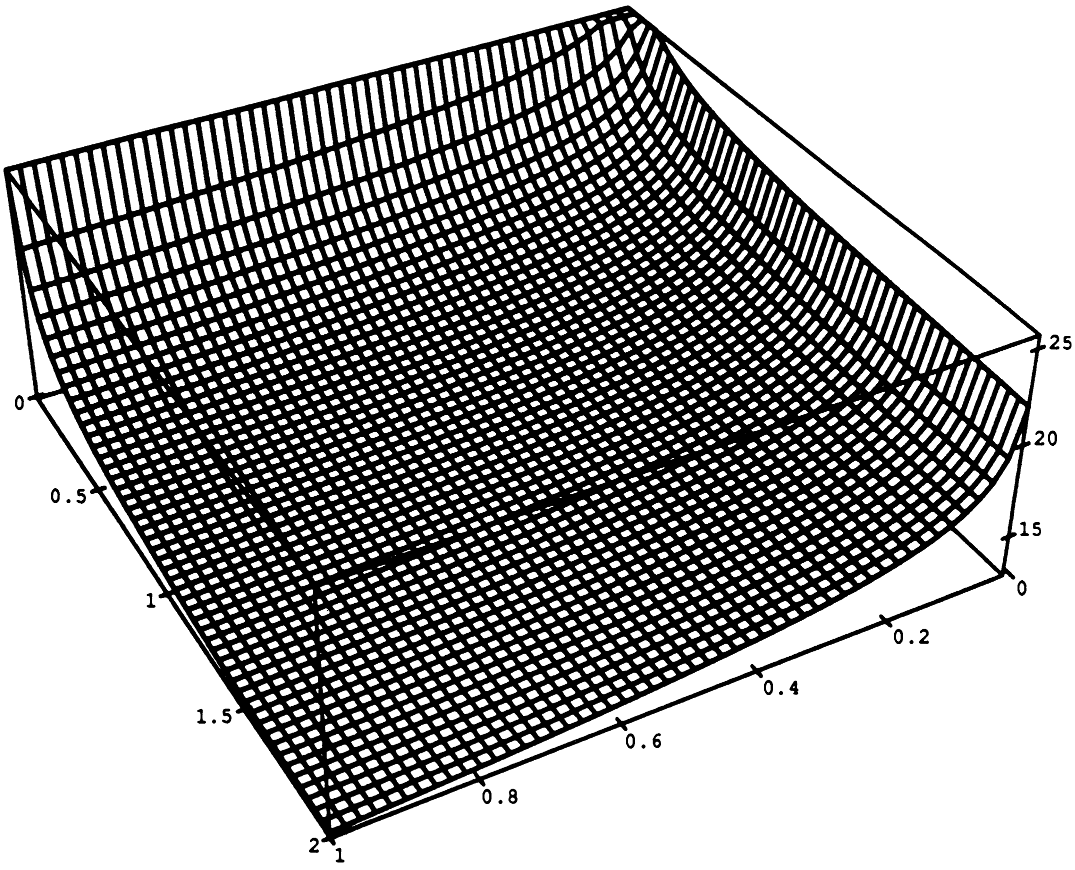


Figure 5.7: Example of three-dimensional plot of computational complexity as a function of ϵ_S and ϵ_T for the two-dimensional L_2 Cartesian Algorithm with Trade-offs. Complexity (vertical) axis is $\log_{10}$ scale.

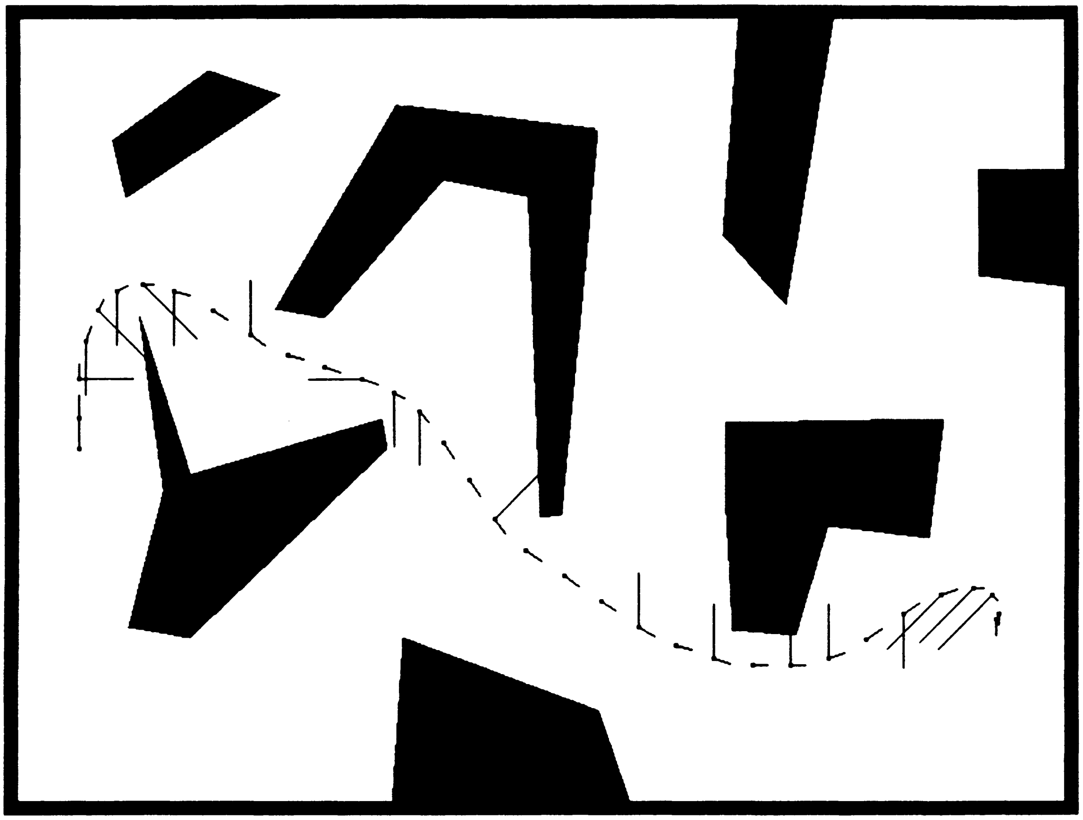


Figure 5.8: A trajectory found by our planner. $\overline{a} = 0.2$, $\overline{v} = 0.3$, $c_0 = 0.15$, $c_1 = 0.2$, $\epsilon_S = 0.6$, and $\epsilon_T = 0.8$. World dimensions are 4 by 3. The planner took 9 minutes and 23 seconds of runtime (14 minutes and 38 seconds total elapsed time) on a 32Mbyte Sparcstation2. Goal state is at lower right.

found the solution in Figure[14] 5.9b. This is the same example as in Figure 3.8, but the figure here also shows accelerations. The solution takes time 11.1 units, and required 4 minutes and 32 seconds of run time to compute on a 32Mbyte SparcStation2. For $\epsilon_S = 0.55$ and $\epsilon_T = 0.9$, and $\epsilon_S = 0.7$ and $\epsilon_T = 0.4$, the planner chose the same timestep within two percent, and found the trajectories in Figure 5.10a and 5.10b, which take time 11.1 and 10.5 respectively. Planning took took 4 minutes and 29 seconds for the former and 4 minutes and 15 seconds for the latter.

In Figure 5.11a, which shows the three trajectories together, we see how the "less safe" trajectories come closer to the obstacles to shave off time. Figure 5.11b shows the trajectory the planner found for $\epsilon_S = 0.45$ and $\epsilon_T = 0.5$, which takes time 11.8, and required 20 minutes and 57 seconds of run time to find.[15]

The trajectories in Figure 5.11 are all quite similar even though they obey different guaranteed approximation conditions. However, for a "nearby" problem, different choices of ϵ_S and ϵ_T cause the planner to find trajectories that are not in the same homotopy class. We see this in Figure 5.12, which shows two solutions our planner found for problem with a slightly different goal state, but the same kinodynamic parameters. It appears that with $\epsilon_S = 0.55$, a trajectory going between the squares has to slow its progress in the x direction enough so that is faster to go around the second square, whereas with $\epsilon_S = 0.7$, a trajectory can "cut corners" closely enough so that this would not be true. Figure 5.13 and 5.14 illustrate how the choices of ϵ_S and ϵ_T can affect the trajectories found by our planner even more dramatically.

[14]The original goal state is also shown, but the start state is not, as it would cover the solution's start state completely.

[15]Total elapsed time, which includes idle process time (*e.g.* swapping), was 52 minutes

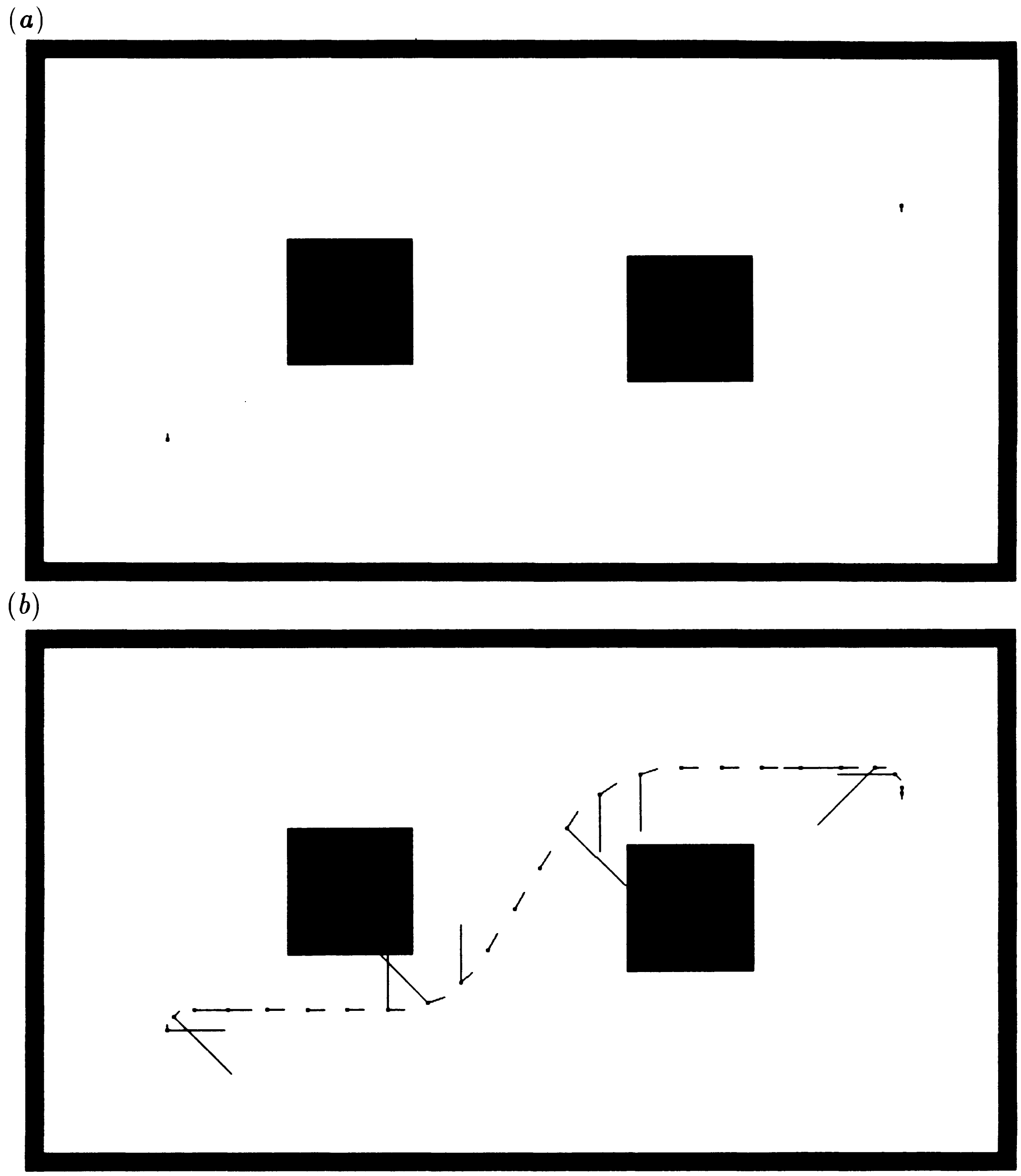
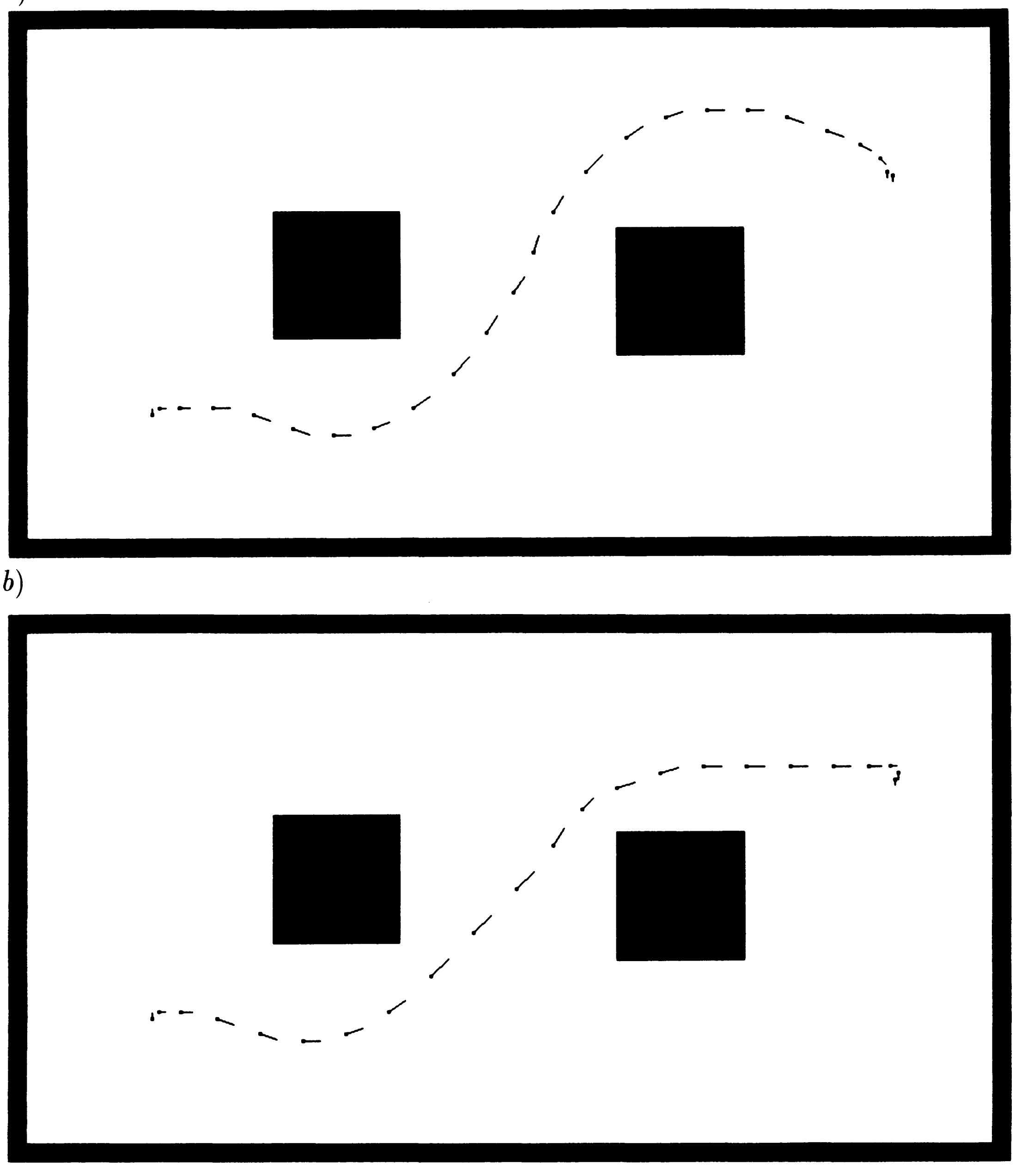


Figure 5.9: An illustrative problem and a solution found by our planner, with $\epsilon_S = \epsilon_T = 0.6$. The kinodynamic bounds of the problem were: $\overline{a} = 0.2$, $\overline{v} = 0.3$, $c_0 = 0.2$, and $c_1 = 0.5$.

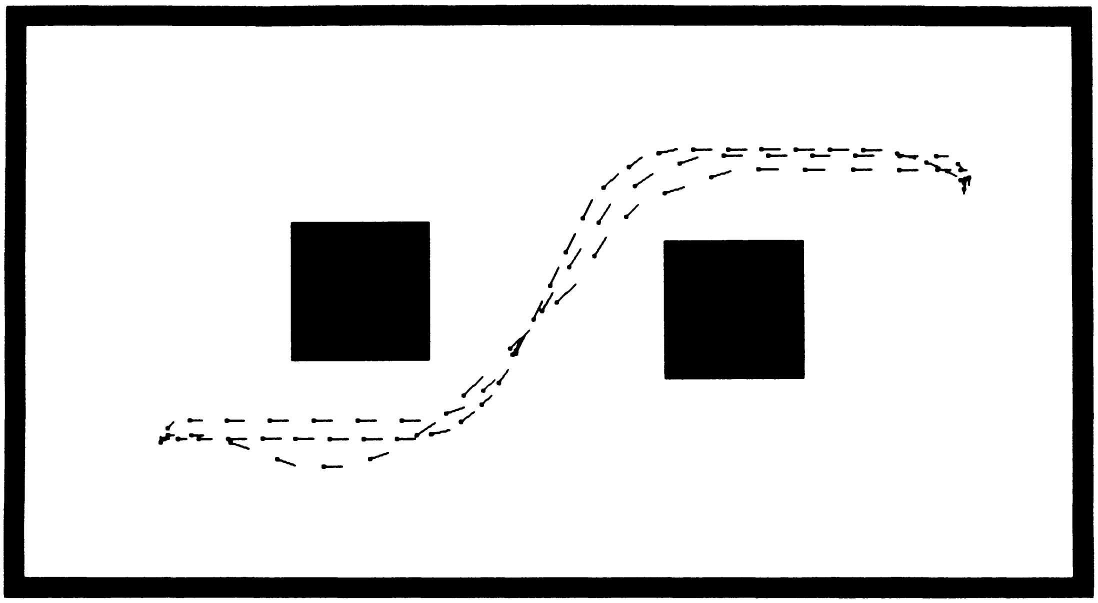
(a)



(b)

Figure 5.10: Solutions found by our planner for the same problem, with (a) $\epsilon_S = 0.55$ and $\epsilon_T = 0.9$, and (b) $\epsilon_S = 0.7$ and $\epsilon_T = 0.4$.

(a)



(b)

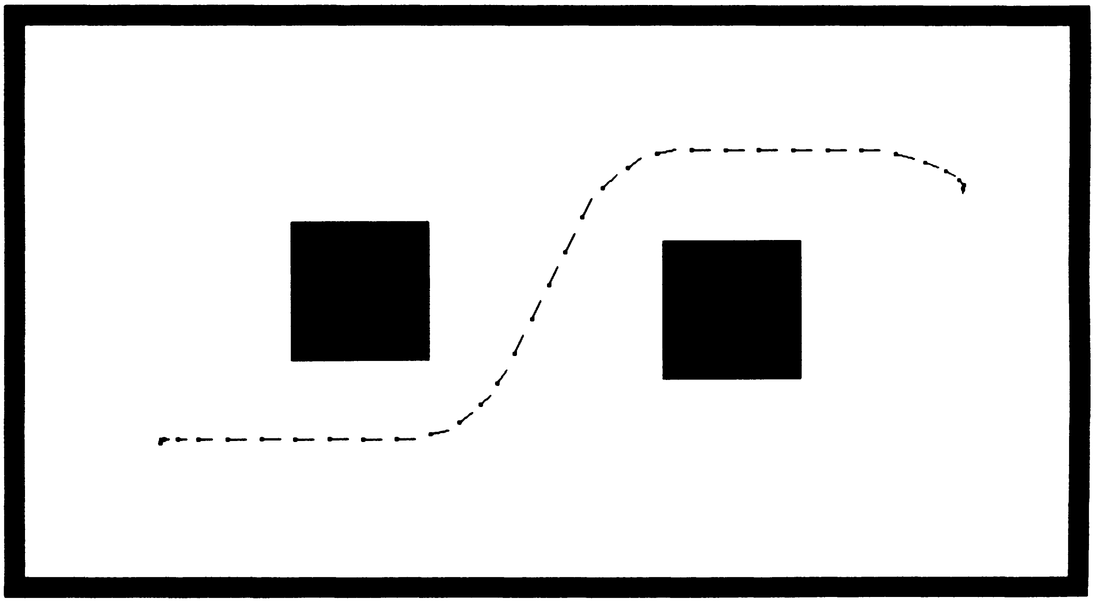


Figure 5.11: Four solutions found by our planner for the same problem. (a) shows trajectories for $\epsilon_S = 0.55$ and $\epsilon_T = 0.9$, $\epsilon_S = \epsilon_t = 0.6$, and $\epsilon_S = 0.7$ and $\epsilon_T = 0.4$. (b) shows a trajectory found with $\epsilon_S = 0.45$ and $\epsilon_T = 0.5$, which took the planner about five times as much runtime.

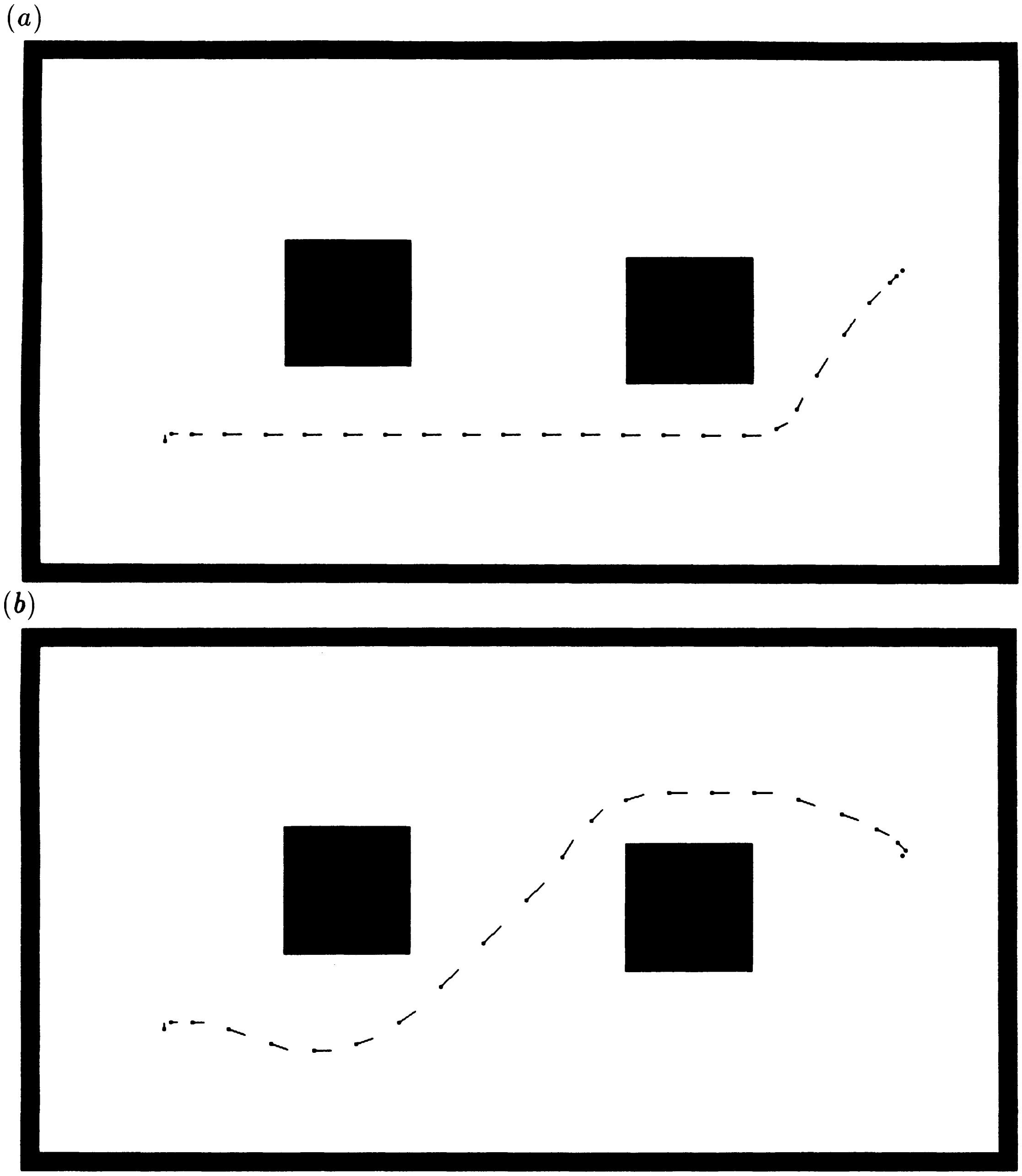


Figure 5.12: Different ϵ_S and ϵ_T choices can result in homotopically different trajectories: (a) $\epsilon_S = 0.55$ and $\epsilon_T = 0.9$; (b) $\epsilon_S = 0.7$ and $\epsilon_T = 0.4$.

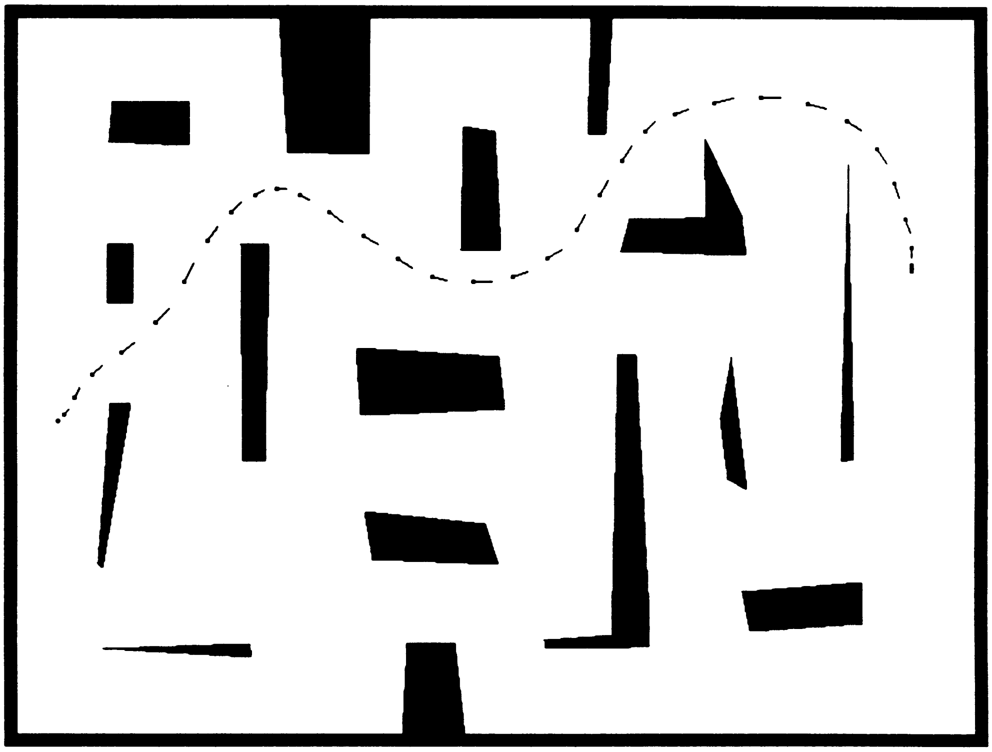


Figure 5.13: Another trajectory found by our planner. $\overline{a} = 0.2$, $\overline{v} = 0.4$, $c_0 = 0.15$, $c_1 = 0.2$, $\epsilon_S = 0.6$, and $\epsilon_T = 0.8$. World dimensions are 4 by 3. The planner took 24 minutes and 34 seconds of runtime (29 minutes and 14 seconds total elapsed time) on a 48Mbyte Sparcstation2. Goal state is at right.

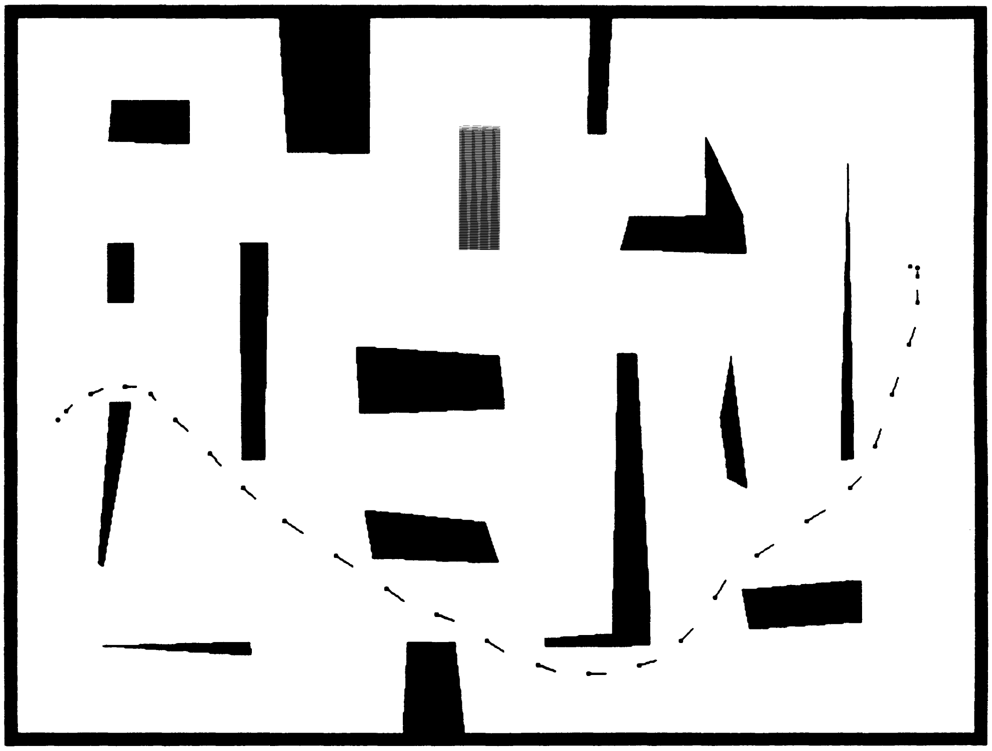


Figure 5.14: A trajectory found by our planner for the same problem, but with $\epsilon_S = \epsilon_T = 0.8$, is homotopically different. The planner took 10 minutes and 48 seconds of runtime (11 minutes and 56 seconds total elapsed time) on a 48Mbyte Sparcstation2.

5.3 Summary

We have presented two extensions of our kinodynamic results. First, we showed how to obtain a provably good polynomial-time ϵ-approximation algorithm for optimal kinodynamic planning for open-chain robots that directly finds ϵ-optimal bang-bang trajectories. Because the branching-factor of its reachability graph does not change with $\frac{1}{\epsilon}$ (it is 2^d), this algorithm is in a sense the "natural" extension of our algorithms for (L_∞) Cartesian robots. However, to show the correctness of the algorithm, we found it necessary to extend the techniques of Chapter 4. In particular, we generalized the notion of an acceleration advantage and adapted the Tracking Game with Perturbations to show that we could prune the search of the reachability graph to $O((\frac{1}{\epsilon})^{7d})$ vertices. Our results also show that an $O(\frac{1}{\epsilon})$ switching frequency is sufficient for an ϵ-optimal bang-bang trajectory.

Second, we presented a finer complexity analysis of the algorithms from previous chapters. Using two approximation parameters ϵ_T and ϵ_S to express closeness to time-optimality and observance of the safety margin, we showed how, to a certain extent, the algorithms from previous chapters allow one to trade off planning time, execution time, and observance of the safety margin. We illustrated these trade-offs with graphical representations and with empirical results obtained using our prototype planner, which executes a version of Hybrid Algorithm for Cartesian Kinodynamic Planning we described in this chapter.

Chapter 6

Conclusion

In this thesis, we have presented the kinodynamic planning framework for optimal motion planning, and we have described a family of provably-good, (pseudo-)polynomial-time ϵ-approximation algorithms for optimal kinodynamic planning.

First, we motivated a constrained optimization problem that (a) matches the requirements of finding a time-optimal planned trajectory and (b) assures that when an optimal solutions exist, non-identical ϵ-optimal solutions would exist. This problem includes a safety margin, which we allow to be speed dependent, for obstacle avoidance. We then formalized this problem for decoupled dynamics and decoupled dynamics bounds, calling it the Optimal Cartesian Kinodynamic Planning problem, and presented a provably-good polynomial-time ϵ-approximation algorithm for it. This algorithm, which reduced the problem to finding shortest path in a uniform-weight reachability graph, was the first such algorithm for time-optimal motion planning in more than one dimension. The "time-rescaling-tracking" proof technique we introduced is the basis of all known correctness and complexity proofs for provably-good approximation algorithms for optimal kinodynamic planning, except for our Algorithm 3 (from Chapter 2). In this proof structure, a tracking lemma relates a tracking tolerance and dynamics bounds to sufficient reachability graph parameters.

Using progressively more sophisticated constructive tracking-trajectory proofs and adversary game proofs, we obtained a series of tracking lemmas that we used to obtain more efficient algorithms and algorithms for kinodynamic planning for robots with less restricted dynamics laws and dynamics bounds, including fully-controllable robots obeying full ("open-chain") Lagrangian dynamics. Extending these basic results, we showed the existence of an approximation algorithm for finding ϵ-optimal bang-bang trajectories, and we presented a closer analysis that uses separate approximation parameters ϵ_S and ϵ_T to describe the quality of the approximation in terms of observance of the safety margin and closeness to time-optimality. This analysis allows one to trade of safety, execution time, and planning time to an extent when using our algorithms. We presented an implementation and some simple experiments, which illustrated these trade-offs, in a restricted domain. All the algorithms we presented are provably-good polynomial-time ϵ-approximation algorithms.

We now sketch some research directions for the future and offer a few closing words.

6.1 Future Directions

Two main goals stand out for future research provably-good near-optimal kinodynamic planning. First, while the polynomial ϵ-approximation properties of the algorithms presented here are theoretically encouraging, it would be desirable to have provably-good approximation algorithms that are fast enough to be useful in practice. Second, we have addressed a limited problem formulation, and the question of whether provably-good polynomial time approximation algorithms exist for more general problem formulations is open.

6.1.1 Towards Practical, Provably-Good Algorithms

We see several research directions towards practical, provably-good approximation algorithms for optimal kinodynamic planning.

Improving Techniques Described in This Thesis. One strategy would be to improve the current algorithms.

- Our algorithms naively search for a shortest path in a graph breadth-first. They could be improved by using A^* search. One simple *admissible heuristic cost function* would be the length of the shortest path from a vertex position to the goal position. A table of approximate L_∞- or L_1- shortest-path lengths to the goal can be computed cheaply with existing path-planning techniques.[1] Other cheaply-computable admissible heuristics might be possible.

[1]Discretized all-sources-single-sink L_∞- or L_1- shortest-path planning can be done in several seconds on a SpracStation2 for a 256x256x128 discretization of configuration space [Res92]. Also, see [LRDG90].

- A close examination of the proofs for Cartesian Robots shows that the timestep is chosen with the assumption that *all* ϵ-optimal trajectories might have to pass through a narrow channel between obstacles. Away from such channels, the current ϵ-safe-tracking-tolerance criterion appears to be unnecessarily harsh for choosing a timestep, and it seems intuitively plausible that a larger timestep should suffice for generating graph edges. Unfortunately, it does not appear that the timestep can be increased for robots with state-dependent dynamics as long as the algorithm's proof relies on that Coupled Tracking Lemma (4.2.1) or the Robust Tracking Lemma 4.2.2.

- Because of the structure and spatial characteristics of the reachability graphs they search, parallel implementations of our algorithms hold promise.

  1. The shortest path between two vertices in a uniform-cost directed-graph can be found a parallel "brushfire" or "wavefront" algorithm, which is based on a simple propagation rule: if in the current search generation a vertex is "told" for the first time that it has been found, then in the next generation it tells its neighbors that they have been found.

  2. A "natural" assignment of vertices to SIMD processors would map one processor to each family of vertices that have the same configuration component. When the robot dynamics laws are constant, the reachability graph is translationally invariant in configuration (position) dimensions if there are no obstacles, and this mapping would therefore allow uniform routing.

  3. If configuration space is divided up into sufficiently large (hyper-) cubic regions, then all neighbors of a given graph vertex whose configuration lies in a particular region will lie in that region also, or in a *spatially adjacent* region. If we map vertices to processors by region, as the side length gets larger, the ratio of computations to communications will become proportional to the ratio of the volume of each region to its area. It might be possible to keep processor utilization high by assigning multiple regions to each processor, using a repeating pattern.

Approaches that are radically different from what we have presented might be at least as productive.

Optimization Approaches. One might be able to show that certain optimization-based approximation algorithms for optimal kinodynamic planning are provably-good. For many applications, some of these algorithms might be fast enough to be practical, even if their asymptotic complexity is exponential.

- Some parameterization of trajectories might include ϵ-optimal motions, grow reasonably with the geometric complexity of the environment and $\frac{1}{\epsilon}$, and otherwise be suitable for a numerical optimization approach. Cubic-spline paths are a possibility, and they have been used in approaches not shown to be provably-good ϵ-approximation algorithms. (See Section 1.4.)

- The local minima of the parameter space would have to be understood well enough so that a provable optimization technique would be guaranteed to find one that corresponds to an ϵ-optimal kinodynamic trajectory when an optimal solution exists.

Other Notions of Provably Good. Algorithms that are provably good under reasonable, well-defined alternatives to a worst case analysis of accuracy might be faster than provably-good ϵ-approximation algorithms.

- We might define a *probablistically provably-good ϵ-approximation algorithm* to be one that, under suitable distribution assumptions about the start and goal states and the obstacles, is guaranteed to find an ϵ-optimal trajectory with some probability P_s if an optimal trajectory exists. Note that the algorithm itself does not have to use randomization.

- It is to be expected that the geometric complexity of the obstacles, their scale and the scale of the gaps between them, and P_s will all affect the complexity of such an algorithm. More generally, we might consider using an *expected case* analysis of complexity.

- Algorithms that are fast for a fixed ϵ and P_s would be useful, particularly if it could be shown that randomly perturbing the start and goal state according to some distribution would, with some known probability, yield a problem for which the algorithm would succeed.

- *Anytime algorithms* [DB88] that have these properties would be desirable.

- Algorithms for finding trajectories that are guaranteed to be $(1-\epsilon_S)\delta_v$-safe and expected to be ϵ_T-time-optimal (according to some problem distribution) would be useful. Furthermore, we could allow ϵ_T be a function of the complexity of the obstacles.

6.1.2 Theoretical Extensions

There are open questions concerning how much we can demand from provably-good approximation algorithms for optimal kinodynamic planning.

Tighter Approximation Conditions. It would be interesting to consider approximate optimal kinodynamic planning with tighter solution requirements.

- To ensure that the approximation problem is "reasonable", our problem formulation allows ϵ-optimal trajectories to be ϵ-close at the start and goal and only requires that these trajectories obey $(1-\epsilon)\delta_v$-safety. It would be useful to precisely characterize the conditions that guarantee the existence of a set of finitely-parameterizable near-optimal trajectories whose paths each intersect the path of the optimal trajectory finitely often. One would then hope to develop a provably-good approximation algorithm to find near-optimal trajectories that are exact at the start and goal and/or that are δ_v-safe when these conditions are met.

- Results in this direction might also enable us develop proof techniques besides "rescaling-tracking" for showing the correctness of ϵ-approximation algorithms for kinodynamic planning. An alternative proof technique could yield algorithms of lower time-complexity for the present problem formulation.

Generalizing the Problem. Provably-good approximation algorithms should be developed for other robot classes and for problem formulations that address questions in motion planning closely related to the kinodynamic planning problem defined in this thesis. We do not include the "single-start, multiple-goal" and "single-goal, multiple-start" versions of the problem, because our algorithms can be used to solve them with trivial modifications.[2]

[2]To solve the former, we simply check for approximation of each goal during the reachability graph search. The latter problem is the former under "time-reversal".

- For wheeled robots, the set of feasible accelerations has a lower dimension than the configuration space. While provably-good algorithms for a variety of path planning problems for this class have been given [Wil88,FW88, JC89], the corresponding minimal-time trajectory problems remain open. It is unclear whether the "rescaling-tracking" technique could be extended to show the correctness of approximation algorithms for such robots, although adversary-game proofs would seem to hold promise.

- Algorithms for robots with dynamics not described by equation (4.7) should be explored. Furthermore, the dynamics constraints in the problem formulation should be generalized. It would be particularly interesting to include power constraints, friction, and non-holonomic constraints [LC89].

- Moving obstacles could be incorporated into the kinodynamic planning problem. Fraichard and Laugier [FL92] have made progress in this direction.

- There are other quantities besides trajectory time that might be used as a cost function. A classic example from control theory is fuel consumption. Secondary cost functions, such as (3.17), and combinations of cost functions might be useful in obtaining "natural" motions with provable properties.

- Provably-good ϵ-approximation algorithms that are also anytime algorithms [DB88] would be desirable.

6.2 Closing Thoughts

Kinodynamic planning represents a new direction in algorithmic motion planning. Computational kinodynamics seems a particularly fruitful area in which to pursue provably fast, provably good approximation algorithms, since while the problems are of considerable intrinsic interest, exact solutions may well be intractable. We have presented such algorithms for optimal kinodynamic planning for a family of robot classes. While the problems have an optimization flavor, our algorithms and proof techniques draw on several branches of computer science and robotics. There is a great deal of challenging theoretical and experimental work to be done, especially in the direction of practical approximation algorithms for optimal kinodynamic planning.

Appendix A

Kinodynamic Planning Lower Bounds

This section sketches how to extend Canny's and Reif's proof [CR87] of $\mathcal{NP}$-hardness of the 3D shortest-path problem to show that Optimal Cartesian Kinodynamic Planning in 3D is also $\mathcal{NP}$-hard.[1] While the general description here should convey the flavor of the result, the technical specifics are intended to accompany or follow a close reading Section 2 of [CR87].

The idea for the extension is simple: if we could set the acceleration bound $\overline{a}$ to ∞, the reduction would be trivial because the problems would be identical. However, to do a polynomial-time reduction, we must show that for a given shortest-path problem instance we can find a "sufficiently large" $\overline{a}$ that only requires a polynomial number of bits to encode.

A.1 Canny's and Reif's Proof

A.1.1 A Reduction

To prove that the 3D shortest-path problem for a point among polyhedral obstacles is $\mathcal{NP}$-hard, [CR87] give a polynomial-time reduction from 3SAT. They describe how to take a given instance of 3SAT and construct an instance of the polyhedral path-planning problem such that a shortest path from the start position to the goal position can be used to determine whether the 3CNF formula in the instance is

[1] This claim was made in [CDRX88], but without proof.

satisfiable.[2] The proof considers a 3CNF formula F of m clauses $C_1, \ldots, C_m$ over n variables $b_1, \ldots, b_n$ and their negations. The polyhedral environment constructed by the reduction has a description of length $O(mn)$. The construction allows one to compute (in $O(mn)$ bits) a path length *lower bound* l and an accuracy δ_{ac} such that the shortest path between the start and goal positions has length greater than $l + \delta_{ac}$ if and only if the formula F is not satisfiable.[3]

A.1.2 Correspondence Between Homotopy Classes and Truth-Value

The polyhedral environment will be the interior of a square box separated into levels by flat plates, each having one, two, or three slits. The plate thickness, slit width, and plate separation will all have the same size ϵ_{CR}. In addition, internal walls will divide the spaces between certain plates into rooms having one slit on the "ceiling" and one slit on the "floor". The start position will be above the top plate, and the goal below the lowest.

The environment is constructed so that it will have special properties that we describe, avoiding details as much as possible. Recall that each clause C_i in a 3CNF formula is a disjuction of literals $l_{i1} \vee l_{i2} \vee l_{i3}$. If we separate paths that go from the start to the goal without ever visiting a level twice into their homotopy equivalence classes, then each class will encode a truth assignment for $b_1, \ldots, b_n$ and the evaluation of a conjunction of m literals, one from each clause of F, using this truth assignment. In other words, if F' is the disjunctive normal form of F obtained by distribution, then for each term (conjunction of literals) in F' and each truth assignment, there will be a corresponding homotopy class. If the truth assignment fails to satisfy the term formula, then some obstacle will "stretch" the shortest path in the class. Furthermore, this path will have a length greater than than $l + \delta_{ac}$ if and only if the truth assignment fails to satisfy the term formula.

A.1.3 Bit-Counting

To show their reduction is polynomial-time, in the end [CR87] must show that the number of bits necessary to encode the corresponding shortest-path problem

[2] "3CNF" means "three-literal conjunctive-normal form formula."

[3] We mostly follow the notation of [CR87] in this section. In particular, "l" is used as in [CR87], not as in the rest of the thesis; *i.e.*, it does *not* mean "world diameter". δ_{ac} is not found in [CR87], but it just takes the place of a more complicated term.

instance will be polynomial in n and m. The main problem is to show that ϵ_{CR} does not have to be too small. Suppose that the construction procedes for 3CNF formula F. Let $l_{unstretched}$ denote the length of the actual shortest path if $F \in 3SAT$, and $l_{stretched}$ if not.

Again, modulo some details, we can view their analysis as having three steps, showing that:

1. If the slit width and plate thickness were zero, the construction guarantees that for path length lower bound $l = 2^{3n}$ and another parameter[4] $\delta_{min} = 2^{-nm}$:

$$\begin{aligned} l_{unstretched} &= l, \quad and \\ l_{stretched} &\geq l + \frac{\delta_{min}^2}{4l}. \end{aligned}$$

   (Equation (9) in [CR87].)

2. The total number of plates is bounded from above by

$$7nm + 10n + 2m + 4.$$

   Furthermore, each plate can add no more than ϵ_{CR} to the shortest path length, so that if $F \in 3SAT$, then

$$l \leq l_{unstretched} \leq l + (7nm + 10n + 2m + 4)\epsilon_{CR}.$$

   (Equation (10).)

3. It follows that if

$$\frac{\delta_{min}^2}{4l} > (7nm + 10n + 2m + 4)\epsilon_{CR},$$

   which is guaranteed if $\epsilon_{CR} = \frac{2^{-2nm-3n-3}}{(7nm+10n+2m+4)}$, then the reduction works with $\delta_{ac} = \frac{\delta_{min}^2}{4l}$.

Thus, ϵ_{CR} can be specified in $O(nm)$ bits. It then follows from other arguments that any dimension in the environment can be specified in $O(nm)$ bits.

[4]δ_{min} is the minimum (virtual) source spacing described in [CR87].

A.2 The Extension for Kinodynamic Lower Bounds

A.2.1 The Idea

To extend the $\mathcal{NP}$-hardness result to kinodynamic planning, we can use the essentially the same polynomial-time reduction to reduce an instance of $3SAT$ to an optimal kinodynamic planning problem instance. Hence, we would use a point amongst polyhedral obstacles. We will also choose the units for time to be the same as the units for distance. For a given 3CNF formula F, the new reduction constructs the a kinodynamic planning problem instance identical to the path-planning problem instance in the [CR87] reduction except that

1. the slit width and plate thickness ϵ'_{CR} will be smaller than ϵ_{CR};
2. the start and goal positions will be lifted to the corresponding states with velocity zero;
3. the velocity must obey a unit bound (in the same L_p-norm used for distance in the shortest-path problem);
4. the acceleration will obey some bound $\overline{a}$ (in the same norm); and
5. for lower time bound l and time-accuracy δ_T, the minimal-time kinodynamic solution (obstacle-avoiding trajectory) would take more than time $l + \delta_T$ if and only if the given 3CNF formula F is not satisfiable.

The specification of δ_v-safety parameters c_0 and c_1 will be omitted from the discussion; it is easy to choose them, because they can be incorporated linearly into the choices of slit widths and plate separations.

A.2.2 Bit-Counting Again

Similarly to the above, the minimal-time solution will be "stretched" if and only if $F \notin 3SAT$. Let $T_{stretched}$ be the minimal solution time in this case, $T_{unstretched}$ be the least time otherwise. Suppose that we choose times $T_l = l$ and $\delta_T = \delta_{min}$ via unit conversion. It follows from the path-planning analysis that if the slit width and plate thickness were zero, and if the acceleration bound were infinite, then

$$T_{unstretched} \;=\; l, \quad and$$

$$T_{stretched} \geq l + \frac{\delta_{min}^2}{4l}.$$

With slit width and plate thickness ϵ'_{CR} but infinite acceleration, it follows that

$$l \leq T_{unstretched} \leq l + (7nm + 10n + 2m + 4)\epsilon'_{CR}.$$

Now, if the acceleration bound is $\overline{a}$, $T_{unstretched}$ can increase by at most $\frac{2}{\overline{a}}$ for each obstacle edge along the shortest path, since this is twice the amount of time it takes to go from zero velocity to full speed or vice-versa. Since there will be at most two edges along the shortest path for each plate,

$$l \leq T_{unstretched} \leq l + (7nm + 10n + 2m + 4)\left(\epsilon'_{CR} + \frac{4}{\overline{a}}\right).$$

By similar reasoning as above, if

$$\begin{aligned} \epsilon'_{CR} &= \frac{2^{-2nm-3n-4}}{(7nm + 10n + 2m + 4)}, \quad \text{and} \\ \overline{a} &= 2^{2nm+3n+6}(7nm + 10n + 2m + 4), \end{aligned}$$

then the reduction will work with $\delta_T = \frac{\delta_{min}^2}{4l}$. (Recall that $\delta_{min} = 2^{-nm}$.) Thus, ϵ'_{CR} and $\overline{a}$ can be specified in $O(nm)$ bits.

Appendix B

Guide to Notation

It was necessary to use a rather large number of symbols in this thesis; we have tried to keep our notation consistent and "intuitive". The following is a partial glossary of symbols.

C the configuration space of a particular robot

TC the tangent bundle of this space

d the number of dimensions of the configuration space

$\mathcal{O}$ the encoding of the obstacle set

δ_v a safety function parameterized by $c_0 > 0$ and $c_1 \geq 0$; $\delta_v(c_0, c_1)(\mathbf{v}) = c_0 + c_1 \|\mathbf{v}\|$

$\mathbf{p}(t)$ a path; $\mathbf{p} : Time \rightarrow C$

$(\mathbf{p}, \dot{\mathbf{p}})$ a trajectory state; $\mathbf{p}$ is position and $\dot{\mathbf{p}}$ is velocity

$\mathbf{x}$ a configuration (also $\mathbf{y}$)

$\mathbf{v}$ a velocity

$\mathbf{X}$ a state (also $\mathbf{Y}$)

Γ a trajectory; $\Gamma : Time \rightarrow TC$

$\mathbf{S}$ start state, usually the desired start state

$\mathbf{G}$ goal state, usually the desired goal state

(η_x, η_v) a tracking tolerance

$\mathbf{a}$ acceleration

$\overline{a}$ acceleration bound

$\overline{\mathbf{a}}$ vector of acceleration bounds

$\overline{v}$ velocity bound

$\overline{\mathbf{v}}$ vector of velocity bounds

A_{max} global (non-dimensional) acceleration bound

$\mathbf{f}$ generalized force vector

$\overline{\mathbf{f}}$ a vector of generalized force bounds; for each i, $|f_i| \leq \overline{f}_i$

$\mathcal{M}$ the encoding of a the dynamics equation obeyed by a robot

l world diameter or length of greatest translational degree of freedom

ϵ approximation parameter; $0 < \epsilon < 1$.

ϵ_S safety approximation parameter; $0 < \epsilon_S < 1$

ϵ_T time approximation parameter; $\epsilon_T > 0$

δ'_v safety function for ϵ-safety; $\delta'_v = (1 - \epsilon)\delta_v$

Γ'_r the trajectory Γ, but ϵ-time-rescaled

$\mathbf{p}'$ the path (or state) $\mathbf{p}$, but ϵ-time-rescaled

$\mathbf{S}'$ ϵ-time-rescaled start state

$\mathbf{G}'$ ϵ-time-rescaled goal state

$\mathbf{S}^*$ ϵ-close approximation of the start state

$\mathbf{G}^*$ ϵ-close approximation of the goal state

$\mathcal{A}_{\mathbf{X}}$ the set of all instantaneous accelerations possible at state $\mathbf{X}$ obeying dynamics bounds

$\widehat{\mathcal{A}}_{\mathbf{X}}$ the set of possible constant accelerations of duration τ for trajectories beginning at state $\mathbf{X}$; these trajectories must obey the dynamics bounds

$\widehat{\mathcal{A}}^*_{\mathbf{X}}$ an easier-to-compute conservative approximation (subset) of $\widehat{\mathcal{A}}_{\mathbf{X}}$

$\mathcal{A}$ first, a set of acceleration functions; in Chapter 4 and afterwards, a function $\mathcal{A} : TC \to \{\mathcal{Y} : \mathcal{Y} \textit{ is a set of acceleration functions}\}$

$\mathcal{A}'$ as above, but corresponding to tighter generalized force bounds

$\mathcal{P}, \mathcal{Q}$ with single arguments, arbitrary functions $\mathcal{P}, \mathcal{Q} : TC \to \{\mathcal{Y} : \mathcal{Y} \textit{ is a set of acceleration functions}\}$

τ a timestep size

κ_l a minimal acceleration advantage

μ a grid-spacing

$\boldsymbol{\mu}$ a vector of grid-spacings; μ_i is the spacing in dimension i

$\overline{\mu}$ discretization parameter for generalized forces

$\mathcal{H}(\mathbf{X}, \tau)$ the $(\boldsymbol{\mu}, \tau)$-extremal shell at (corresponding to) state $\mathbf{X}$; this is a set of acceleration functions

$\overline{\mathcal{H}}(\mathbf{X}, \tau)$ the set of true $(\overline{\mu}, \tau)$-bangs at (corresponding to) state $\mathbf{X}$; another set of acceleration functions

$\mathcal{H}_{\tau_0}(\mathbf{X}) = \mathcal{H}(\mathbf{X}, \tau_0)$

$\overline{\mathcal{H}}_{\tau_0}(\mathbf{X}) = \overline{\mathcal{H}}(\mathbf{X}, \tau_0)$

τ_0 for a given robot and ϵ, the fundamental timestep

(η_{x0}, η_{v0}) for a given robot and ϵ, a fundamental tracking tolerance

N_0 fundamental chunk density; we think of N_0 bang-bang τ_B-bangs as approximating an average acceleration of duration $N_0\tau$

$\oplus$ Minkowski sum

Subscripts r, q, and u denote trajectories.
Subscript i usually denote the i^{th} coordinate axial direction.
Subscript n usually denotes a timestep.

Appendix C

Computing Parameters for the True-Extremal and Near-Extremal Algorithms for Open-Chain Manipulators

C.1 Introduction

We wish to show for any given problem such that the force bounds always exceed the forces necessary to hold the robot stationary and for a given ϵ, that the parameters η_{x0}, η_{v0}, τ_0 will be $\Omega(\epsilon)$. We will use $(\mathbf{p}, \dot{\mathbf{p}})$ to denote both trajectories and arbitrary states.

We first show that $\|\mathbf{M}^{-1}(\mathbf{p}) - \mathbf{M}^{-1}(\mathbf{p} - \Delta\mathbf{p})\|$ are $O(\|\Delta\mathbf{p}\|)$. We can then compute bounds for the perturbation magnitudes h_r, h_{q0}, and h_{q1}, defined in (4.47), in terms of τ_0, η_x, and η_y. Given $\mathbf{M}(\mathbf{p})$ and $\mathbf{U}_G(\mathbf{p})$ (or $\mathbf{G}(\mathbf{p})$), it is possible to bound their derivatives by inspection, because each of their components is a sum of products of components of $\mathbf{p}$ and their sines and cosines.

Following any derivation of (4.1), say from [AS86], we write

$$G_i(\mathbf{p}) = \sum_j m_j \mathbf{g}^T J_{Li}^{(j)}(\mathbf{p}), \tag{C.1}$$

where m_j is the mass of the j^{th} link, $\mathbf{g}$ is the gravitational acceleration vector, and $J_{Li}^{(j)}(\mathbf{p})$ is the i^{th} column of the linear velocity Jacobian for link j. More importantly,

$$\mathbf{M}(\mathbf{p}) = \sum_i \left(m_i \mathbf{J}_L^{(i)}(\mathbf{p})^T \mathbf{J}_L^{(i)}(\mathbf{p}) + \mathbf{J}_A^{(i)}(\mathbf{p})^T \mathbf{I}_i(\mathbf{p}) \mathbf{J}_A^{(i)}(\mathbf{p}) \right), \tag{C.2}$$

where m_i is the mass of the i^{th} link, $\mathbf{I}_i(\mathbf{p})$ is its inertia tensor, and $\mathbf{J}_L^{(i)}$ and $\mathbf{J}_A^{(i)}$ are its linear velocity and angular velocity Jacobians. Recalling equations (4.7) and (4.1.1), we see that an encoding of the Jacobians $\mathbf{J}_L^{(i)}(\mathbf{p})$ and $\mathbf{J}_A^{(i)}(\mathbf{p})$ and the constants m_i, $\mathbf{I}_i$, and $\mathbf{g}$ fully encodes the dynamics of the manipulator.

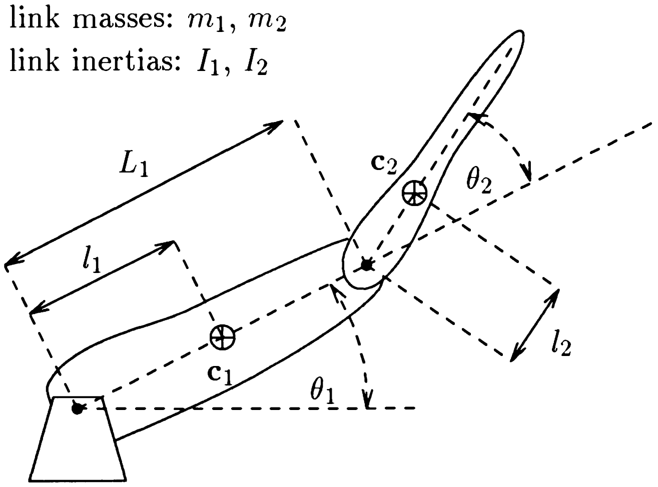


Figure C.1: A planar 2DOF arm, with constant physical parameters m_1, m_2, l_1, l_2, I_1, I_2, L_1 and configuration parameters θ_1 and θ_2. $\mathbf{c}_i$ indicates the centroid of joint i; in this simple example, the centroid of the first link lies on the line between the joint axes. In a problem instance, the constants and equations (C.3) and (C.4) are encoded in $\mathcal{M}$. Together with equations (C.2), (C.1), (4.1.1), (4.7), they provide a complete description of the dynamical laws that govern the manipulator.

Example C.1.1 Consider the planar manipulator with two revolute joints in figure C.1. For each link i, let m_i be the mass of the link, l_i be the distance from the i^{th} joint axis to the link centroid, and I_i be the inertia of the link about the axis parallel to the joint axis through the link centroid. Let L_1 be the distance from the first joint axis to the second, and (to simplify the example) let the centroid of the first link lie on the line between the joint axes. We compute linear velocity Jacobians.

$$\begin{aligned}
\mathbf{J}_L^{(1)} &= \begin{pmatrix} -l_1 \sin\theta_1 & 0 \\ l_1 \cos\theta_1 & 0 \end{pmatrix} \\
\mathbf{J}_L^{(2)} &= \begin{pmatrix} -L_1 \sin\theta_1 - l_2 \sin(\theta_1+\theta_2) & -l_2 \sin(\theta_1+\theta_2) \\ L_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) & l_2 \cos(\theta_1+\theta_2) \end{pmatrix},
\end{aligned} \tag{C.3}$$

We note that here the angular velocity Jacobians are simple because all revolute axes are parallel.

$$\begin{aligned}
\mathbf{J}_A^{(1)} &= (1 \quad 0) \\
\mathbf{J}_A^{(2)} &= (1 \quad 1)
\end{aligned} \tag{C.4}$$

Clearly, θ_1 and θ_2 will only appear in sines and cosines in the dynamics equation. Furthermore, we observe that an encoding $\mathcal{M}$ of the m_i, l_i, I_i, L_1, equations (C.3) and (C.4), and the gravitational acceleration vector $\mathbf{g} = (0 \quad g)^T$ is thus a sufficient encoding of the manipulator dynamics.

Heinzinger and Paden [HP89] exploit the relationships between $\mathbf{M}(\mathbf{p})$ and the robot kinematic map and Jacobian to bound the general derivatives of $\mathbf{M}(\mathbf{p})$ and $\mathbf{G}(\mathbf{p})$. We will use their results to show one way to derive the desired bounds.

C.2 Bounding Changes in $\mathbf{M}^{-1}(\mathbf{p})$

Observe that $\|\mathbf{M}(\mathbf{p}) - \mathbf{M}(\mathbf{p}+\Delta\mathbf{p})\|_2$ is polynomially dependent on the components of $\Delta\mathbf{p}$. Let $\mathbf{M}' = \mathbf{M}(\mathbf{p}+\Delta\mathbf{p})$, and $\Delta\mathbf{M} = \mathbf{M} - \mathbf{M}'$. We wish to bound $\|\Delta\mathbf{M}^{-1}\| = \|\mathbf{M}^{-1} - \mathbf{M}'^{-1}\|$. If $\mathbf{M}$ and $\mathbf{M}'$ are inertia matrices, then they are symmetric and positive definite. Let λ_{min} be the minimum of their minimum eigenvalues.
Now consider the solutions $\mathbf{x}$ and $\mathbf{y}$ to the systems

$$\begin{aligned}
\mathbf{M}\mathbf{x} &= \mathbf{b}; \\
\mathbf{M}'\mathbf{y} &= \mathbf{b}.
\end{aligned} \tag{C.5}$$

By substitution,

$$\mathbf{M}(\mathbf{y}-\mathbf{x}) = (\mathbf{M}-\mathbf{M}')\mathbf{y} + \mathbf{M}'\mathbf{y} - \mathbf{M}\mathbf{x} = \Delta\mathbf{M}\mathbf{y},$$

and thus $||\mathbf{M}(\mathbf{x}-\mathbf{y})|| = ||\Delta\mathbf{M}\mathbf{y}||$. We now choose to use the L_2-norm. Now,

$$||\Delta\mathbf{M}\mathbf{y}||_2 \leq ||\Delta\mathbf{M}||_2||\mathbf{y}||_2 \leq ||\Delta\mathbf{M}||_2\frac{||\mathbf{b}||_2}{\lambda_{min}}$$

Since

$$\lambda_{min}||\mathbf{x}-\mathbf{y}||_2 \leq ||\mathbf{M}(\mathbf{x}-\mathbf{y})||_2 = ||\Delta\mathbf{M}\mathbf{y}||2,$$

it follows that

$$||\mathbf{x}-\mathbf{y}||_2 \leq \frac{||\Delta\mathbf{M}||_2||\mathbf{b}||_2}{\lambda_{min}^2}. \tag{C.6}$$

Recall that

$$\Delta\mathbf{M}^{-1}\mathbf{b} = (\mathbf{M}^{-1} - \mathbf{M}'^{-1})\mathbf{b} = \mathbf{x}-\mathbf{y}. \tag{C.7}$$

Since $\Delta\mathbf{M}$ is arbitrary except for the condition that $\mathbf{M}'$ be non-singular, and $\mathbf{b}$ is arbitrary, (C.6) and (C.7) imply that

$$||\Delta\mathbf{M}^{-1}||_2 \leq \frac{||\Delta\mathbf{M}||_2}{\lambda_{min}^2} \tag{C.8}$$

Therefore, since $||\Delta\mathbf{M}||_2$ is $O(||\Delta\mathbf{p}||)$, so is $||\Delta\mathbf{M}^{-1}||_2$.

C.3 Acceleration Bounds and Perturbations

We first review a notation for tensor-valued functions, as used by [HP89]. $\mathbf{M}$ is a smooth tensor field, and $\mathbf{M}(\mathbf{p})$ is a tensor of rank 2. For $\mathbf{x} \in C$, $\mathbf{M}(\mathbf{x})(\mathbf{v}_1, \mathbf{v}_2)$ denotes the tensor acting on $(\mathbf{v}_1, \mathbf{v}_2) \in T_{\mathbf{x}}C \times T_{\mathbf{x}}C$. We have been representing $\mathbf{M}(\mathbf{p})$ as a matrix. For example, the kinetic energy of the system in state $(\mathbf{p}, \dot{\mathbf{p}})$ can be expressed as $\frac{1}{2}\dot{\mathbf{p}}^T\mathbf{M}(\mathbf{p})\dot{\mathbf{p}}$ or $\frac{1}{2}\mathbf{M}(\mathbf{p})(\dot{\mathbf{p}}, \dot{\mathbf{p}})$.

The n^{th} derivative of $\mathbf{M}(\mathbf{p})$ with respect to $\mathbf{p}$ is defined as follows:

$$D^n\mathbf{M}(\mathbf{p})(\alpha, \beta)(\gamma_1, ..., \gamma_n) = \sum_{i,j=1}^{d}\sum_{k_1,...,k_n=1}^{d}\frac{\partial^n\mathbf{M}(\mathbf{p})_{ij}}{\partial\mathbf{p}_{k_1}...\partial\mathbf{p}_{k_n}}\alpha^i\beta^j\gamma_1^{k_1}...\gamma_n^{k_n}. \tag{C.9}$$

Hence, for a given state $(\mathbf{p}, \dot{\mathbf{p}})$ and acceleration $\mathbf{a}$, (4.1) corresponds to

$$\mathbf{f}(\cdot) = \mathbf{M}(\mathbf{p})(\mathbf{a}, \cdot) + D\mathbf{M}(\mathbf{p})(\dot{\mathbf{p}}, \cdot)(\dot{\mathbf{p}}) - \frac{1}{2}D\mathbf{M}(\dot{\mathbf{p}}, \dot{\mathbf{p}})(\cdot) + G(\mathbf{p})(\cdot), \tag{C.10}$$

with $f_i = \mathbf{f}(\mathbf{e}_i)$, where $\mathbf{e}_i$ is the usual i^{th} basis vector in $\Re^d$. The second and third terms on the right side are equivalent to the $[\dot{\mathbf{p}}^T\mathbf{C}(\mathbf{p})\dot{\mathbf{p}}]$ term in (4.1)

We will use four results ((C.11) through (C.14)) from [HP89]. Let M_i denote the mass of link i, L_i its maximum length from the near joint axis, and $\overline{L_i}$ the greatest distance of its centroid from the first joint axis of the manipulator. Define $\|\mathbf{y}\|_{\hat{i}} = \sum_{j=1}^{i} |y_i|$. Then

$$\begin{aligned}
|\mathbf{M}(\mathbf{p})(\mathbf{w}_1, \mathbf{w}_2)| &\leq \sum_{i=1}^{d} M_i L_i^2 \|\mathbf{w}_1\|_{\hat{i}} \|\mathbf{w}_2\|_{\hat{i}}; && \text{(C.11)} \\
|D^n\mathbf{M}(\mathbf{p})(\mathbf{w}_1, \mathbf{w}_2)(\mathbf{w}_3, ..., \mathbf{w}_{n+2})| &\leq 2^d \sum_{i=1}^{d} M_i L_i^2 \|\mathbf{w}_1\|_{\hat{i}} ... \|\mathbf{w}_{n+2}\|_{\hat{i}}; && \text{(C.12)} \\
|\mathbf{G}(\mathbf{p})(\mathbf{w}_1)| &\leq \sum_{i=1}^{d} g M_i \overline{L}_i \|\mathbf{w}_1\|_{\hat{i}}; && \text{(C.13)} \\
|D^n\mathbf{G}(\mathbf{p})(\mathbf{w}_1, ..., \mathbf{w}_{n+1})| &\leq \sum_{i=1}^{d} g M_i \overline{L}_i \|\mathbf{w}_1\|_{\hat{i}} ... \|\mathbf{w}_{n+1}\|_{\hat{i}}. && \text{(C.14)}
\end{aligned}$$

Now, we bound the acceleration. Let us define

$$\mathcal{F}(\mathbf{f}, \mathbf{p}, \dot{\mathbf{p}}) = \mathbf{f} - [\dot{\mathbf{p}}^T\mathbf{C}(\mathbf{p})\dot{\mathbf{p}}] - \mathbf{G}(\mathbf{p}). \tag{C.15}$$

Then we can rewrite (4.7) as

$$\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f}) = \mathbf{M}^{-1}(\mathbf{p})\mathcal{F}(\mathbf{f}, \mathbf{p}, \dot{\mathbf{p}}). \tag{C.16}$$

Recalling (4.7),

$$\begin{aligned}
\|\mathbf{a}(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{f})\| &\leq \|\mathbf{M}^{-1}(\mathbf{p})\| \|\mathcal{F}(\mathbf{f}, \mathbf{p}, \dot{\mathbf{p}})\| \\
&\leq \frac{1}{\lambda_{min}^M} \left\| \mathbf{f} - [\dot{\mathbf{p}}^T\mathbf{C}(\mathbf{p})\dot{\mathbf{p}}] - \mathbf{G}(\mathbf{p}) \right\| \qquad \text{(C.17)} \\
&\leq \frac{1}{\lambda_{min}^M} \left\{ \|\mathbf{f}\| + \|[\dot{\mathbf{p}}^T\mathbf{C}(\mathbf{p})\dot{\mathbf{p}}]\| + \|\mathbf{G}(\mathbf{p})\| \right\}.
\end{aligned}$$

Using (C.12) and recalling that $\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{d}\|\mathbf{x}\|_2$ for any d-vector $\mathbf{x}$, we obtain

$$\begin{aligned}
\|[\dot{\mathbf{p}}^T\mathbf{C}(\mathbf{p})\dot{\mathbf{p}}]\| &\leq \max_{\|\mathbf{x}\|_2\leq 1}\left\{|D\mathbf{M}(\mathbf{p})(\dot{\mathbf{p}},\mathbf{x})(\dot{\mathbf{p}})| + |\tfrac{1}{2}D\mathbf{M}(\mathbf{p})(\dot{\mathbf{p}},\dot{\mathbf{p}})(\mathbf{x})|\right\} \\
&\leq 2^{d-1}3\textstyle\sum_{i=1}^{d}\sqrt{d}M_iL_i^2\|\mathbf{v}_{max}\|_i^2 \\
&\leq 2^{d-1}3d^{3/2}\|\mathbf{v}_{max}\|^2\textstyle\sum_{i=1}^{d}M_iL_i^2.
\end{aligned} \tag{C.18}$$

Using (C.13)

$$\begin{aligned}
\|\mathbf{G}(\mathbf{p})\| &\leq \max_{\|\mathbf{x}\|_2\leq 1}\textstyle\sum_{i=1}^{d} gM_i\overline{L}_i\|\mathbf{x}\|_i \\
&\leq g\sqrt{d}\textstyle\sum_{i=1}^{d}M_i\overline{L}_i.
\end{aligned} \tag{C.19}$$

Thus we have the following global bound for the acceleration:

$$A_{max} \leq \frac{1}{\lambda_{min}^M}\left\{\|\mathbf{f}_{max}\| + 3d^{3/2}\|\mathbf{v}_{max}\|^2\sum_{i=1}^{d}M_iL_i^2 + g\sqrt{d}\sum_{i=1}^{d}M_i\overline{L}_i\right\}. \tag{C.20}$$

We now bound $h_r(\tau_0)$, $h_{q0}(\tau_0)$, and $h_{q1}(\eta_x,\eta_v)$. Recalling the definitions in (4.47), we obtain

$$\begin{aligned}
h_r(\tau_0) &< \tau_0\max\left\|\left[\frac{\partial\mathbf{M}^{-1}}{\partial\mathbf{p}}\dot{\mathbf{p}}\right]\mathcal{F}(\mathbf{f},\mathbf{p},\dot{\mathbf{p}}) + \mathbf{M}^{-1}(\mathbf{p})\left[\frac{\partial\mathcal{F}}{\partial\mathbf{p}}\dot{\mathbf{p}} + \frac{\partial\mathcal{F}}{\partial\dot{\mathbf{p}}}\mathbf{a}\right]\right\|; \\
h_{q0}(\tau_0) &\leq \tau_0\max\left\|\left[\frac{\partial\mathbf{M}^{-1}}{\partial\mathbf{p}}\dot{\mathbf{p}}\right]\mathcal{F}(\mathbf{f},\mathbf{p},\dot{\mathbf{p}}) + \mathbf{M}^{-1}(\mathbf{p})\left[\frac{\partial\mathcal{F}}{\partial\mathbf{p}}\dot{\mathbf{p}} + \frac{\partial\mathcal{F}}{\partial\dot{\mathbf{p}}}\mathbf{a}\right]\right\|; \\
h_{q1}(\eta_x,\eta_v) &\leq \max\left\|\left[\frac{\partial\mathbf{M}^{-1}}{\partial\mathbf{p}}\Delta\mathbf{p}\right]\mathcal{F}(\mathbf{f},\mathbf{p},\dot{\mathbf{p}}) + \mathbf{M}^{-1}(\mathbf{p})\left[\frac{\partial\mathcal{F}}{\partial\mathbf{p}}\Delta\mathbf{p} + \frac{\partial\mathcal{F}}{\partial\dot{\mathbf{p}}}\Delta\dot{\mathbf{p}}\right]\right\|.
\end{aligned} \tag{C.21}$$

Recall (C.8), which we use to bound $\|\mathbf{M}^{-1}(\mathbf{p}) - \mathbf{M}^{-1}(\mathbf{p}+\Delta\mathbf{p})\|$, and observe that

$$\begin{aligned}
\|\mathbf{M}(\mathbf{p}) - \mathbf{M}(\mathbf{p}+\Delta\mathbf{p})\| &\leq \max_{\|x\|=\|y\|=1}|D\mathbf{M}(\mathbf{p})(\mathbf{x},\mathbf{y})(\Delta\mathbf{p})| \\
&\leq 2^d d^{3/2}\|\Delta\mathbf{p}\|\textstyle\sum_{i=1}^{d}M_iL_i^2.
\end{aligned} \tag{C.22}$$

By substitution, we obtain

$$\|\Delta\mathbf{M}^{-1}\|_2 \leq \frac{2^d d^{3/2}\|\Delta\mathbf{p}\|\sum_{i=1}^{d}M_iL_i^2}{(\lambda_{min}^M)^2}. \tag{C.23}$$

It follows that

$$\tau_0 \left\| \frac{\partial \mathbf{M}^{-1}}{\partial \mathbf{p}} \dot{\mathbf{p}} \right\| \le \frac{2^d d^{3/2} \|\dot{\mathbf{p}}\tau_0\| \sum_{i=1}^d M_i L_i^2}{(\lambda_{min}^M)^2} \quad \text{and} \tag{C.24}$$

$$\left\| \frac{\partial \mathbf{M}^{-1}}{\partial \mathbf{p}} \Delta \mathbf{p} \right\| \le \frac{2^d d^{3/2} \|\Delta \mathbf{p}\| \sum_{i=1}^d M_i L_i^2}{(\lambda_{min}^M)^2}. \tag{C.25}$$

Applying (C.12) and (C.18), we obtain

$$\begin{aligned} \left\| \tfrac{\partial [\dot{\mathbf{p}}^T \mathbf{C}(\mathbf{p})\dot{\mathbf{p}}]}{\partial \mathbf{p}} \Delta \mathbf{p} \right\| &\le \max_{\|\mathbf{x}\|\le 1} \left\{ \begin{array}{l} |D^2\mathbf{M}(\mathbf{p})(\dot{\mathbf{p}},\mathbf{x})(\dot{\mathbf{p}},\Delta\mathbf{p})| \\ \quad + |\tfrac{1}{2} D^2\mathbf{M}(\mathbf{p})(\dot{\mathbf{p}},\dot{\mathbf{p}})(\mathbf{x},\Delta\mathbf{p})| \end{array} \right\} \\ &\le 2^{d-1} \cdot 3d^2 \|\dot{\mathbf{p}}\|^2 \|\Delta\mathbf{p}\| \textstyle\sum_{i=1}^d M_i L_i^2; \end{aligned} \tag{C.26}$$

$$\begin{aligned} \left\| \tfrac{\partial [\dot{\mathbf{p}}^T \mathbf{C}(\mathbf{p})\dot{\mathbf{p}}]}{\partial \dot{\mathbf{p}}} \Delta \dot{\mathbf{p}} \right\| &\le \max_{\|\mathbf{x}\|\le 1} \left\{ \begin{array}{l} |D\mathbf{M}(\mathbf{p})(\dot{\mathbf{p}},\mathbf{x})(\Delta\dot{\mathbf{p}})| \\ \quad + |D\mathbf{M}(\mathbf{p})(\Delta\dot{\mathbf{p}},\mathbf{x})(\dot{\mathbf{p}})| \\ \quad + |D\mathbf{M}(\mathbf{p})(\Delta\dot{\mathbf{p}},\dot{\mathbf{p}})(\mathbf{x})| \end{array} \right\} \\ &\le 2^{d-1} \cdot 3d^{3/2} \|\dot{\mathbf{p}}\| \|\Delta\dot{\mathbf{p}}\| \textstyle\sum_{i=1}^d M_i L_i^2; \quad \text{and} \end{aligned} \tag{C.27}$$

$$\left\| \tfrac{\partial \mathbf{G}(\mathbf{p})}{\partial \mathbf{p}} \Delta \mathbf{p} \right\| \le g\sqrt{d} \textstyle\sum_{i=1}^d M_i \overline{L}_i. \tag{C.28}$$

We now give the final bounds for $h_r(\tau_0)$, $h_{q0}(\tau_0)$, and $h_{q1}(\eta_{x0}, \eta_{v0})$. First, we define

$$\begin{aligned} H &= \textstyle\sum_{i=1}^d M_i L_i^2; \\ \overline{H} &= \textstyle\sum_{i=1}^d g M_i \overline{L}_i; \\ F &= \|\mathbf{f}\| + 2^{d-1} \cdot 3d^{3/2} \|\mathbf{v}_{max}\|^2 H + \sqrt{d} g \overline{H}; \\ F' &= \|\mathbf{f}'\| + 2^{d-1} \cdot 3d^{3/2} \|\mathbf{v}'_{max}\|^2 H + \sqrt{d} g \overline{H}. \end{aligned} \tag{C.29}$$

In this notation, we can rewrite (C.20) as

$$A_{max} \le \frac{F}{\lambda_{min}^M}. \tag{C.30}$$

Finally,

$$
\begin{aligned}
h_r(\tau_0) &< \frac{(2^d d^{3/2}\|\mathbf{v}'_{max}\|\overline{H}F' + 2^{d-1}\cdot 3d^2\|\mathbf{v}'_{max}\|^3)\tau_0}{\lambda^M_{min}} \\
&\quad + \frac{2^{d-1}\cdot 3d^{3/2}\|\mathbf{v}'_{max}\|F'\tau_0}{(\lambda^M_{min})^2}; && \text{(C.31)} \\
h_{q0}(\tau_0) &\leq \frac{(2^d d^{3/2}\|\mathbf{v}_{max}\tau\|HF + 2^{d-1}\cdot 3d^2\|\mathbf{v}_{max}\|^3)\tau_0}{\lambda^M_{min}} \\
&\quad + \frac{2^{d-1}\cdot 3d^{3/2}\|\mathbf{v}_{max}\tau\|F\tau_0}{(\lambda^M_{min})^2}; && \text{(C.32)} \\
h_{q1}(\eta_{x0}, \eta_{v0}) &\leq \left\{ \frac{2^d d^{3/2}HF}{(\lambda^M_{min})^2} + \frac{2^{d-1}\cdot 3d^2\|\mathbf{v}_{max}\|H\sqrt{d}g\overline{H}}{\lambda^M_{min}} \right\} \eta_{x0} \\
&\quad + \frac{2^{d-1}\cdot 3d^{3/2}\|\mathbf{v}_{max}\|H}{\lambda^M_{min}}\eta_{v0}. && \text{(C.33)}
\end{aligned}
$$

We summarize these bounds by saying that $h_r(\tau_0)$ and $h_{q0}(\tau_0)$ are linear in τ_0, and that $h_{q1}(\eta_{x0}, \eta_{v0})$ is linear in η_{x0} and η_{v0}. In other words we can write, with obvious substitutions for C_r, C_{q0}, C_{q1}, and C'_{q1}:

$$
\begin{aligned}
h_r(\tau_0) &< C_r\tau_0; \\
h_{q0}(\tau_0) &\leq C_{q0}\tau_0; && \text{(C.53)} \\
h_{q1}(\eta_x, \eta_v) &\leq C_{q1}\eta_x + C'_{q1}\eta_{v0}.
\end{aligned}
$$

Bibliography

[AF66] M. Athans and P. L. Faulb. *Optimal Control: An Introduction to the Theory and Its Application.* McGraw-Hill, New York, 1966.

[AS86] H. Asada and J. J. Slotine. *Robot Analysis and Control.* John Wiley & Sons, New York, 1986.

[BBD+90] J. Baillieul, R. Brockett, B. Donald, et al. *Robotics: Symposia in Applied Mathematics*, volume 41. American Mathematical Society Press, Providence, Rhode Island, 1990.

[BD89] M. Boddy and T. Dean. Solving time-dependent planning problems. In *Proceedings IJCAI-89*, pages 979–984, 1989.

[BDG85] J. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robot manipulators along specified paths. *International Journal of Robotics Research*, **4**(3), 1985.

[BHJ+82] M. Brady, J. Hollerbach, T. Johnson, T. Lozano-Pérez, and M. Mason, editors. *Robot Motion: Planning and Control.* MIT Press, Cambridge, Massachusetts, 1982.

[Bob82] J. E. Bobrow. *Optimal Control of Robotic Manipulators.* Ph.D. dissertation, U.C.L.A., Los Angeles, California, 1982.

[Bob88] J. E. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE Journal of Robotics and Automation*, **4**(4):443–450, August 1988.

[Bra89] M. Brady, editor. *Robotics Science.* MIT Press, Cambridge, Massachusetts, 1989.

[Bro84] M. L. Brown. Optimal robot path planning via state space networks. Masters dissertation, Department of Mechanical and Aerospace Engineering, Princeton University, June 1984.

[Can86] J. Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **8**(2):200–209, 1986.

[Can88a] J. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, Massachusetts, 1988. Book version of Canny's 1986 Ph.D. thesis.

[Can88b] J. Canny. Exact solution of some robot motion planning problems. In *International Symposium on Robotics Research*, pages 181–189, 1988.

[CD88] J. Canny and B. Donald. Simplified Voronoi diagrams. *Discrete and Computational Geometry*, **3**(3):219–236, 1988.

[CDRX88] J. Canny, B. Donald, J. Reif, and P. Xavier. On the complexity of kinodynamic planning. In *Proceedings of the* 29^{th} *Annual Symposium on the Foundations of Computer Science*, pages 306–316, White Plains, New York, 1988.

[CKS81] A. C. Chandra, D. C Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association of Computing Machinery*, **28**(1):114–133, January 1981.

[Cla87] K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proceedings of the* 19^{th} *Symposium on the Theory of Computing*, pages 56–65, New York, 1987.

[CR87] J. Canny and J Reif. New lower bound techniques for robot motion planning. In *Proceedings of the* 28^{th} *Annual Symposium on the Foundations of Computer Sciende*, Los Angeles, California, 1987.

[CRR90] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, pages 271–280, Berkeley, California, 1990.

[DB88] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings AAAI-88*, pages 49–54, 1988.

[DNS86] S. Dubowsky, M. A. Norris, and Z. Shiller. Time optimal trajectory planning for robotic manipulators with obstacle avoidance: A CAD approach. In *Proceedings 1986 IEEE International Conference on Robotics and Automation*, pages 1906–1912, San Fracisco, California, 1986.

[Don87] B. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, **31**(3):295–353, 1987.

[DX89a] B. Donald and P. Xavier. Near-optimal kinodynamic planning for robots with coupled dynamics bounds. In A. C. Sanderson, A. A. Derochers, and K. Valvanis, editors, *Proceedings of the Fourth IEEE International Symposium on Intelligent Control*, pages 354–359, Albany, New York, September 1989. Invited paper.

[DX89b] B. Donald and P. Xavier. A provably good approximation algorithm for optimal-time trajectory planning. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pages 958–963, Scottsdale, Arizona, 1989.

[DX90a] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, pages 290–300, Berkeley, California, June 1990.

[DX90b] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators. Department of Computer Science Technical Report TR-1095, Cornell University Department of Computer Science, Ithaca, New York, February 1990. Supercedes TR-971.

[DX91] B. Donald and P. Xavier. Time-safety trade-offs and a bang-bang algorithm for kinodynamic planning. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, 1991.

[FL92] T. Fraichard and C. Laugier. Kinodynamic planning with moving obstacles: the case of a structured workspace. In *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Raleigh, NC (USA), July 1992.

[FW88] S. Fortune and G. Wilfong. Planning constrained motion. In *Twentieth Annual ACM Symposium on Theory of Computing*, Chicago, 1988.

[GJ85] E. G. Gilbert and D. W. Johnson. Minimum-time robot path planning in the presence of obstacles. In *Proceedings 24^{th} Conference on Decision and Control*, pages 21–30, Fort Lauderdale, Florida, 1985.

[GM87] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. In *Proceedings of the 27^{th} Annual Symposium on the Foundations of Computer Science*, pages 11–19, 1987.

[HBH88] E. J. Horvitz, J. S. Breese, and M. Henron. Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning*, 1988.

[HJCP90] G. Heinzinger, P. Jacobs, J. Canny, and B. Paden. Time-optimal trajectories for a robot manipulator: A provably good approximation algorithm. In *Proceedings ot the 1990 IEEE International Conference on Robotics and Automation*, pages 150–155, Cincinnati, Ohio, May 1990.

[Hol83] J. M. Hollerbach. Dynamic scaling of manipulator trajectories. MIT A.I. Memo 700, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1983.

[HP89] G. Heinzinger and B. Paden. Bounds on robot dynamics. In *Proceedings 1989 IEEE International Conference on Robotics and Automation*, pages 1227–1232, Scottsdale, Arizona, 1989.

[JC89] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Proceedings ot the 1989 IEEE International Conference on Robotics and Automation*, pages 2–7, Scottsdale, Arizona, 1989.

[JHCP89] P. Jacobs, G. Heinzinger, J. Canny, and B. Paden. Planning guaranteed near-time-optimal planning in a cluttered workspace. Technical Report ESRC 89-20/RAMP 89-15, Engineering Systems Research Center, University of California, Berkeley, California, October 1989.

[Kah69] M. E. Kahn. *The Near-Minimal-Time Control of Open-Loop Articulated Kinematic Chains*. Ph.D. dissertation, Stanford University, Stanford, California, December 1969. Also Stanford Artificial Intelligence Memo No. AIL-106.

[Kir70] D. E. Kirk. *Optimal Control Theory*. Prentiss-Hall, Englewood Cliffs, New Jersey, 1970.

[KR71] M. E. Kahn and B. Roth. The near-minimum-time control of open-loop articulated kinematic chains. *Journal of Dynamic Systems, Measurement, and Control*, **93**:164–172, 1971.

[Lat91] J-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[LC89] Z. X. Li and J. Canny. Robot motion planning with non-holonomic constraints. UCB ERL Technical Report Memo No. UCB/ERL M89/13, University of California, Berkeley, California, 1989.

[Lei66] Leitman. *An Introduction to Optimal Control.* McGraw-Hill, Inc., New York, 1966.

[LL81] J. Y. S. Luh and C. S. Lin. Optimum path planning for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, **102**:142–151, 1981.

[LLZ90] R. Lindberg, R. Longman, and M. Zedd. Kinematic and dynamic properties of an elbow manipulator mounted on a satellite. *The Journal of the Astronautical Sciences*, **38**(4):397–421, 1990.

[LP83] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, **C–32**(2):108–120, 1983. Also MIT A.I. Memo 605, December 1982.

[LPW79] T. Lozano-Pérez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, **22**(10):560–570, October 1979.

[LRDG90] J. Lengyel, M. Reichert, B. Donald, and D. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. In *Proceedings, SIGGRAPH '90*, Dallas, Texas, Aug 1990.

[LW77] J. Y. S. Luh and M. W. Walker. Minimum-time along the path for a mechanical arm. In *Proceedings, IEEE Conference on Decision and Control*, pages 755–759, New Orleans, Louisiana, 1977.

[LWP80] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. Online computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, **102**:69–76, 1980.

[NA84] M. Niv and D. M Auslander. Optimal control of a robot with obstacles. In *Proceedings of the 1984 American Control Conference*, pages 280–287, San Diego, California, 1984.

[Ó87] C. Ó'Dúnlaing. Motion planning with inertial constraints. *Algorithmica*, **2**(4):431–475, 1987.

[Pap85] C. H. Papadimitriou. An algorithm for shortest path motion in three dimensions. *Information Processing Letters*, 20:259–263, 1985.

[PBGM62] L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko. *The Mathematical Theory of Optimal Processes.* Interscience Publishers/John Wiley & Sons, Inc., New York, 1962.

[PFTV88] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, New York, 1988.

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, New York, 1982.

[PT86] C. H. Papadimitriou and J. Tsitsiklis. Intractable problems in control theory. *SIAM Journal of Control and Optimization*, **4**(4):639–654, 1986.

[Raj85] V. T. Rajan. Minimum time trajectory planning. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 759–764, St. Louis, Missouri, 1985.

[Rei79] J. Reif. Complexity of the mover's problem and generalizations. In *Proceedings 20^{th} IEEE Symposium on the Foundations of Computer Science*, pages 421–427, 1979.

[Res92] E. Ressler. Planning 2D solid robot motion on a discrete grid. In B. Donald and D. Huttenlocher, editors, *Course Notes for CS 661/462 "Robotics and Machine Vision"*. Cornell University Department of Computer Science, Ithaca, New York, 1992.

[RS85] J. Reif and J. Storer. Shortest paths in euclidean space with polyhedral obstacles. Technical Report CS-85-121, Brandeis University Computer Science Department, April 1985.

[RT90] J. Reif and S. Tate. Approximate kinodynamic planning using l_2-norm dynamics bounds. Technical Report CS-1990-13, Duke University Department of Computer Science, Durham, North Carolina, 1990.

[RT91] J. Reif and S. Tate. Continuous alternation. Technical report, Duke University Department of Computer Science, Durham, North Carolina, 1991.

[Sch87] H. M. Schaettler. On the optimality of bang-bang trajectories in $\Re^3$. *Bulletin of the American Mathematical Society*, **16**(1):113–116, 1987.

[SD85] Z. Shiller and S. Dubowsky. On the optimal control of robotic manipulators with actuator and end-effector constraints. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 614–620, St. Louis, Missouri, 1985.

[SD88] Z. Shiller and S. Dubowsky. Global time-optimal motions of robotic manipulators in the presence of obstacles. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 370–375, Philadelphia, Pennsylvania, 1988.

[SH85] G. Sahar and J. Hollerbach. Planning of minimum-time trajectories for robot arms. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 751–758, St. Louis, Missouri, 1985.

[SM85a] K. G. Shin and N. D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions of Automatic Control*, **30**(6):531–541, June 1985.

[SM85b] K. G. Shin and N. D. McKay. Selection of near-minimum geometric paths for robotic manipulators. *IEEE Transactions of Automatic Control*, **31**(6):531–541, June 1985.

[SR84] V. Scheinman and B. Roth. On the optimal selection and placement of manipulators. In *Proceedings 5^{th} CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, pages 25–32, Udine, Italy, 1984.

[SS82] J. T. Schwartz and M. Sharir. On the piano mover's problem: Ii. general techniques for computing topological properties of real algebraic manifolds. Technical Report 41, Computer Science Department, New York University, 1982.

[SS83] J. T. Schwartz and M. Sharir. On the piano mover's problem: I. the case of a rigid polygonal body moving amongst polygonal barriers. *Communications in Pure and Applied Mathematics*, **36**, 1983.

[SS84] M. Sharir and A. Schorr. On shortest pathes in polyhedral spaces. In *Proceedings* 16^{th} *ACM Symposium on Theory of Computing*, pages 144–153, 1984.

[SS85] E. Sontag and H. Sussmann. Remarks on the time-optimal control of two-link manipulators. In *Proceedings of the* 24^{th} *Conference on Decision and Control*, Ft. Lauderdale, Florida, 1985.

[SS86] E. Sontag and H. Sussmann. Time-optimal control of manipulators. Technical report, Department of Mathematics, Rutgers University, New Brunswick, New Jersey, 1986.

[SY89] J. J. Slotine and H. S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, **5**(1):118–124, February 1989.

[Wil88] G. Wilfong. Motion planning for an autonomous vehicle. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 529–533, Philadelphia, Pennsylvania, 1988.

[Yap86] C. Yap. Algorithmic motion planning. In J. Schwartz and C. Yap, editors, *Advances in Robotics: Volume 1*. Lawrence Erlbaum Associates, 1986.