Control synthesis for dynamic contact manipulation

Siddhartha Srinivasa

CMU-RI-TR-05-33

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

September 5, 2005

Thesis Committee:
Michael Erdmann, Chair
Matthew Mason, Chair
Alfred Rizzi
Yan-Bin Jia, Iowa State University

This research was supported in part by NSF grants IIS-9820180, IIS-9900322, IIS-0082339 and IIS-0222875.

Contents

List of Figures

List of Tables

Abstract

Manipulation is the art of moving things. At the heart of the problem, an object needs to be moved from start to goal by a robot that is in contact with the object. The contacts serve two purposes: they transmit forces and impose motion constraints on the object. Even if a robot can precisely control its own motion, it is constrained by the interactions at the contacts for control of the motion of the object. Contact interactions are governed by the laws of Coulomb friction and are nonlinear and non-Newtonian.

Current solutions to the manipulation problem decompose the problem into first solving for the forces required to produce a desired object motion and then commanding the robot to apply the requisite force. This imposed decomposition assumes that the robot is capable of producing the commanded forces and velocities required for manipulation. However, this assumption is broken in dynamic manipulation, where the robot operates close to actuator saturation.

In this thesis, we explore the planning and control of dynamic manipulation subject to actuator constraints. We describe a mapping of the Coulomb friction constraints and actuator constraints into a common space, obtaining a unified contact acceleration constraint. We propose two techniques for using this constraint to generate analytical trajectories for the dynamic manipulation problem. In the first technique of time-scaling, we decouple the problem into computing a feasible path followed by selecting the speed of motion along the path that satisfies the constraint. In the second technique of task and shape decomposition, we recognize that the constraint resides in a low dimensional subspace of the system state space and project the system dynamics onto, and orthogonal to that subspace. We use a feedback controller in the constraint space and plan for the orthogonal unconstrained freedoms. Finally, we demonstrate our techniques on two dynamic manipulation tasks and a constrained, nonholonomic system.

Acknowledgments

I would like to thank my advisors, Matt Mason and Mike Erdmann, for their guidance and their friendship. Thank you for the fun meetings and the crazy ideas (a robot that moves paper with its wheels?!). Most of all, thank you for treating me as a peer and for being patient and allowing me to explore the field. I hope that I can be half as good an advisor as you have been to me. Thank you.

I would like to thank my committee members Al Rizzi and Yan-Bin Jia for their invaluable comments. I would also like to thank my MLab brothers Devin Balkcom and Ravi Balasubramanian for the many discussions on manipulation, music and Indian politics. Also thanks to Garth Zeglin, Mark Moll, Jonathan Hurst, Nancy Pollard, Steve LaValle, James Kuffner and Kevin Lynch for many helpful comments and suggestions.

Thanks to Suzanne Muth and Jean Harpley for all the help and the last minute visas.

Some of the happiest times of my life were spent in the squash courts. I am indebted to Matt Mason for teaching me how to play and to Frank Pfenning for teaching me how to *really* play. Thank you, both, for your time and patience.

Many thanks to all my classmates. I said at the beginning that I was here to win and I hope I did win your friendship and respect.

My thanks to Curt Bererton for badminton, CounterStrike and teaching me how to drive, to Fernando Alfaro for being so cool and seeing the humor in every situation, and to Rob Zlot for the flipping out ninjas, HOYAB, and rekindling my love for Rajnikanth. Thanks to Cristian Dima, Carl Wellington, Kiran Bhat, Ranjith Unnikrishnan, Mamta Puri, Sapna Puri, Raghu Donamukkala, the PC Baayas, Aaron Courville, Sowmya Mahadevan and all my office mates through the years for making these years fun and exciting.

Thanks to my father for instilling in me a love for science and for always demanding the best from me. I still remember that day in 3rd grade when you showed me how to derive the equations of motion from first principles. Thanks to my mom for her quiet sacrifices and to Sham for being the most wonderful little sister anyone could have.

Thank you, Supriya, for your love and support. You are my life.

Finally, I would like the RI for paying us a stipend for something that each one of us would, in a heartbeat, do for free. Robots are cool!

Chapter 1

Introduction

We want robots to interact with the world. We want them not just to move around the world effectively, avoiding obstacles, but to change the world by moving the obstacles. To achieve this, we need an accurate model of the robot's interaction with the world. We also need tools that use this complex model of interaction to plan and control the robot's motions.

Manipulation is the art of moving things. At the heart of the manipulation problem, an object needs to be moved by a robot from a start to a goal. During the course of manipulation, contacts can be made, maintained or broken, both between the object and the robot, and between the object and the environment. Contacts that are maintained either *roll* (where there is no relative motion between the contacting bodies) or *slide* (where there is relative motion between the contacting bodies). An additional specification to the manipulation problem is the requirement that certain contacts roll, or slide, or never break. This specification is called the *contact mode* of the system.

Contacts are important because they act as the coupling between the object, the robot and the environment. They transmit forces and impose motion constraints. However, controlling the motion at the contacts is hard because their evolution is governed by the laws of Coulomb friction which are nonlinear, non-Newtonian and impose constraints that reside in two different spaces.

A solution to the manipulation problem is a set of robot controls that move the object from a start to a desired goal while satisfying the specified contact mode.

Versions of the manipulation problem have been well studied in the literature. One of the earliest problems studied was the insertion of a peg into a hole using a robot while preventing wedging or jamming of the peg [Whitney, 1982]. In the dexterous manipulation problem, an object is grasped and manipulated by a robot hand with multiple fingers [Cole et al., 1989, Cole et al., 1992]. The problems of pushing, orienting, fixturing and assembling parts for factory automation have also been well studied.

The analysis of objects with multiple frictional contacts poses two interesting problems.

The first is the *forward problem*, namely predicting the motion of an object given the applied force. Solving this problem is essential for simulation. There exist powerful analytical [Erdmann, 1994] and numerical techniques [Lotstedt, 1984, Trinkle et al., 1997] that solve this problem.

The second is the *inverse problem*, namely predicting the applied force that produces a desired object motion, or the set of applied forces that produce a desired contact mode. Solving this problem is essential for planning and control. Analytical solutions for the planar problem are presented in [Erdmann, 1994, Balkcom and Trinkle, 2002, Brost and Mason, 1989] and sampling-based techniques are explored in [Trinkle and Zeng, 1995].

In the context of a robot manipulating an object, the above solutions make an important assumption that *the robot is capable of producing the forces and velocities that are required for manipulation*. There is an imposed decomposition of the problem into first finding the force to move the object followed by commanding the robot to apply the requisite force. Admittedly, this is a useful plan of attack if the required forces and velocities are within the robot's limits. However, there exist manipulation problems in which this is not the case.

In *dynamic manipulation*, the Coriolis and centrifugal forces generated during motion are significant and cannot be neglected. Intuitively, these are motions where the object and the robot move fast. Evidently, one of the main reasons for being dynamic is to accomplish the task quickly, a requirement for many factory automation tasks, or running races. Often, being dynamic pushes actuators closer to their limits.

Consider the problem of a waiter manipulating a wineglass. His goal is to move the wineglass from start to goal as fast as possible. We shall consider two strategies for manipulating the wineglass. Note that the third and possibly the fastest strategy would be the waiter hurling the wineglass as hard as he can at the patron. We shall ignore that strategy for the purpose of our discussion.

In the first strategy, the waiter has the wineglass *grasped* in his hand. Once grasped, the wineglass is unaffected by external forces. Thus, the only constraints on the waiter's motion are how fast his limbs can move. This problem of generating *time-optimal* (fastest) trajectories for robots subject to actuator torque and velocity limits was introduced by Kahn and Roth [Kahn, 1969, Kahn and Roth, 1971]. There has been significant work on this problem since [Hollerbach, 1984, Sahar and Hollerbach, 1985, Shin and McKay, 1985, Bobrow et al., 1985] and the state of the art solutions are fast, optimal and analytical.

In the second strategy, the waiter has the wineglass resting on a tray. Depending on the friction between the wineglass and the tray, if the waiter repeats the same motion of his arms as in the first case, it is possible that the wineglass tips or slides off the tray or loses contact. Thus, there are two sets of constraints on the waiter's motions – constraints due to how fast his limbs can move and constraints imposed by friction at the contact to

maintain a contact mode. This problem of generating trajectories for a robot manipulating an object subject to both actuator limits and constraints imposed by friction was explored by [Lynch and Mason, 1999] in the study of a robot with a single revolute joint. This thesis expands their result to a class of dynamic manipulation tasks.

This thesis explores the planning and control of dynamic manipulation subject to actuator constraints. We combine Coulomb friction and actuator constraints in a common constraint space. Furthermore, we propose a new paradigm of task and shape decomposition for the planning and control of constrained dynamical systems and demonstrate its applicability to dynamic manipulation and nonholonomic control.

1.1 Contributions

In this section, we discuss the main themes of the thesis. In each item, we provide the motivation for the theme, an outline of our solution, and a brief discussion of its limitations.

1. *Actuator limits are important for dynamic manipulation. In many cases, they are the bottleneck for performing dynamic tasks.*

   It is important to map Coulomb friction constraints into a space into which the actuator constraints can be mapped as well. We use a series of transformations to map contact velocity constraints and contact force constraints into a common contact acceleration space. In this space, we describe a technique for computing analytical solutions for the set of feasible robot joint accelerations that produce a desired contact mode, for problems involving planar contact.

   There exist solutions to the inverse problem for an object with multiple frictional contacts that do not consider robot constraints. There also exist solutions to the trajectory generation problem for a constrained robot arm which do not consider its interaction with an object. We provide a unified constraint that combines both actuator limits and Coulomb friction constraints.

   The contact acceleration constraint is a complete description of the system only in the absence of internal forces. This is because internal forces are, by definition, forces that produce no acceleration and reside in the nullspace of our mapping from contact forces to contact accelerations. Intuitively, these forces are generated due to stresses and strains internal to the robot and object structure and cannot be measured by an acceleration sensor. Hence controlling internal forces requires modeling or sensorial information that is unavailable to us. However, if the system is augmented with a force-torque sensor, or a model of the deformation of the system, the new information can be added to the contact acceleration constraint, obtaining a full description of the system.

2. *Planning and control of constrained dynamic systems in the full-dimensional space is hard.*

   It is satisfying the constraints that makes the problem hard. There exist numerous solutions for the planning and control of unconstrained dynamical systems.

   Our key observation is that for many problems the constraints of the system reside in a lower dimensional subspace of the system state space. We exploit that by decomposing the system into two subsystems – the *shape subsystem* which is a projection of the system dynamics onto the constraints, and *task subsystem* which is a projection of the system dynamics that is orthogonal to the constraints. We write a controller for the shape subsystem that works exclusively to satisfy the constraints. The task subsystem is unconstrained and its planning and control is easy.

   The task and shape decomposition paradigm relies on the fact that the constraints do reside in a low dimensional space. For problems where the constraint space is a high or full dimensional subspace of the system state space, our paradigm is of little use. However, we show that there exists a class of manipulation systems — systems with *kinematic indeterminacy* — for whom the contact acceleration constraint does reside in a low dimensional space.

3. *Use natural forces.*

   Our third contribution is more of an observation.

   As mentioned earlier, actuator limits can be a bottleneck in performing dynamic manipulation tasks. One strategy to reduce the burden on the actuators is to use the load-bearing ability of contacts with the environment. Human manipulation strategies exploit this ability — when we move large objects, we tend to pivot them about the ground rather than lift them altogether. However, contacts with the environment are hard to control because the robot does not have direct access to them. The robot has to exert a force on the object which is then transmitted through the object to the remote contacts. In Ch.3, we describe how we can use the contact acceleration constraint to control the contact mode at the remote contacts.

   Another strategy to reduce actuator load is to exploit the natural dynamics of objects rather than fight them. Wineglasses tend to topple when pushed and coins tend to precess when spun. We will show in Ch.6 how it is possible to use the toppling motion of the wineglass to manipulate it. Using natural dynamics also provides us with a richer set of actions that can be performed on the object — it turns a kinematically indeterminate system into a dynamically controllable one.

1.2 Three problems

In this section, we outline the three problems we will be studying in detail in this thesis. The solutions to the problems capture the ideas of incorporating actuator constraints in dynamic manipulation and the use of the task and shape decomposition paradigm.

The block standing problem

The motivation for the block standing problem arises from a mobile robot using its wheels to slide a block along a step (Fig.1.1(i)). A kinematic skeleton of the problem is shown in Fig.1.1(ii). The desired contact mode is a fixed contact at C and sliding contacts at A and B. The goal is to stand the block up on the step ($\theta = \frac{\pi}{2}$).

The block standing problem is interesting for two reasons. First, we show that the problem does not have a quasi-static solution. This means that the robot has to use the centrifugal and Coriolis forces generated by accelerating the block to solve the problem. Second, the robot needs to use its contact at C to control the remote contacts A and B.

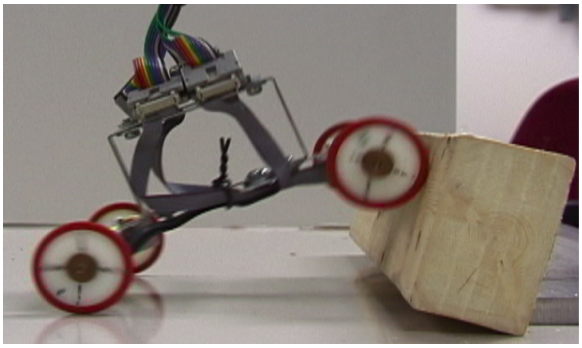
In Ch.3, we derive analytical dynamic solutions to the problem by solving the contact acceleration constraint using the technique of time-scaling. One of the resulting trajectories is shown in Fig.1.1(iii). Here, the robot pulls the block down a certain distance and then pushes it forward to stand it up. We show how the trajectories are affected by changes in the actuator limits, robot and object configuration, and the coefficients of friction.

The Mobipulator problem

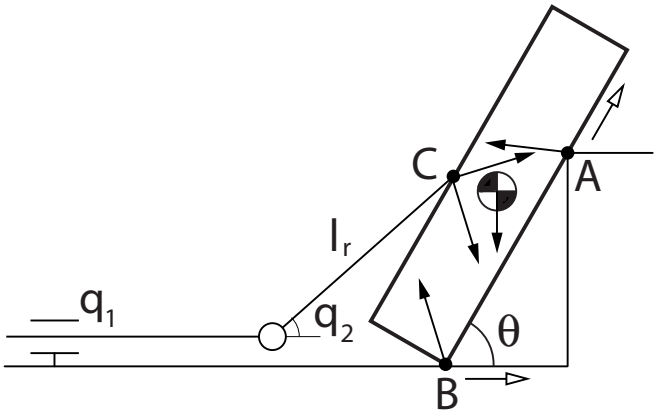
The motivation for the Mobipulator problem arises from a mobile manipulator (the *Mobipulator*, shown in Fig.1.2(i)) manipulating a sheet of paper using its wheels. The goal is to move the paper from a start to a desired goal while maintaining two wheels on the paper and the other two wheels on the desktop. For a given orientation of the robot and the paper, the constraint restricts the possible placement of the robot, as shown by the shaded polygon in Fig.1.2(ii).

The Mobipulator is a constrained kinematic nonholonomic system residing in a 6D configuration space. There do not exist any analytical solutions for trajectory planning in the full dimensional space.

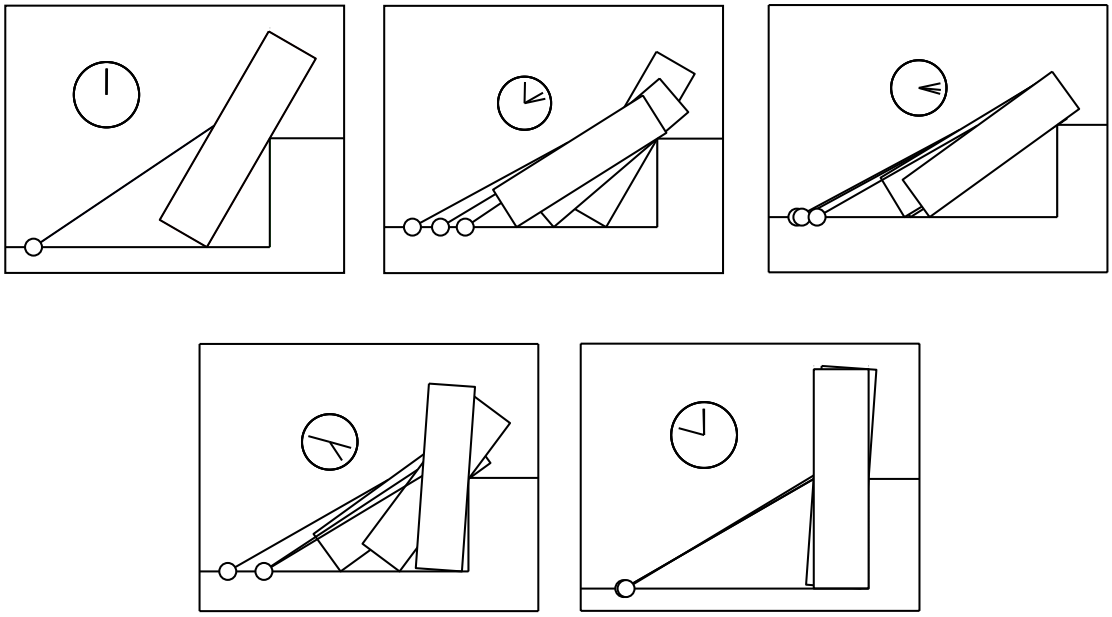
In Ch.5, we use the task and shape decomposition paradigm to generate trajectories for the Mobipulator problem. We reduce the problem to a nonholonomic planning problem in a tractable lower dimensional shape space. A snapshot of a trajectory that produces a rotation of the paper about its geometric center is shown in Fig.1.2(iii). The motion is akin to that of a tractor pushing a trailer with a moveable hitch placed optimally at each instant.



(i)

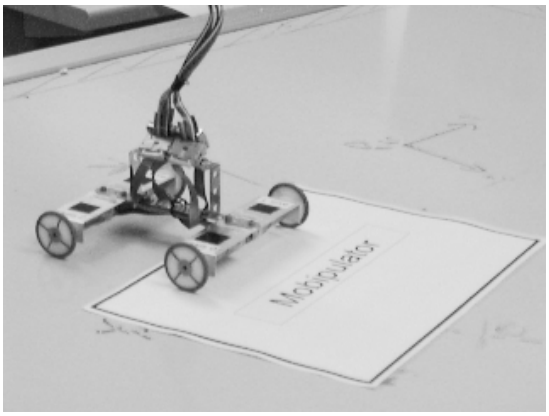


(ii)

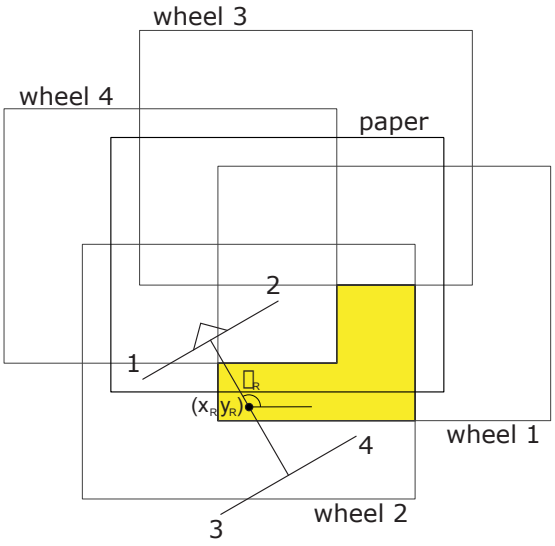


(iii)

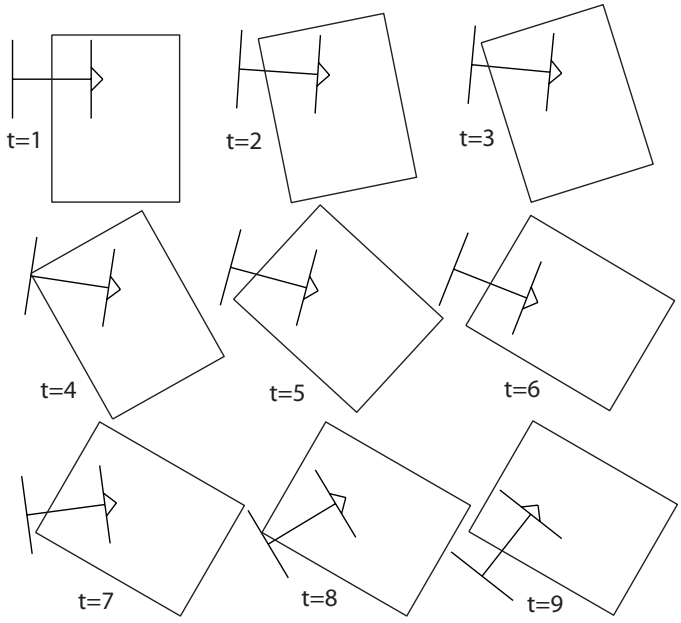
Figure 1.1: (i): The block standing problem. (ii) The kinematic skeleton of the problem. (iii) Snapshots of a feasible trajectory.



(i)



(ii)



(iii)

Figure 1.2: (i): The Mobipulator problem. (ii) The constraint on the shape freedoms. (iii) Snapshots of a feasible trajectory.

The waiter's problem

The motivation for the waiter's problem arises from a problem described earlier in this chapter, that of a waiter manipulating a wineglass (Fig.1.3(i)). A kinematic skeleton of the planar problem is shown in Fig.1.3(ii) with the wineglass represented by a block and the waiter's hand represented by a robot arm with two prismatic joints. The goal is to tip the block from $\theta = 0$ to $\theta = \frac{\pi}{2}$ while maintaining a fixed contact between the robot and the block.

Just like the block standing problem, there exist no quasi-static solutions for the waiter's problem. Furthermore, this problem addresses a shortcoming of the time-scaling technique used to solve the block standing problem — there are very few paths that can be successfully time-scaled.

In Ch.6, we use both the contact acceleration constraint and the task and shape decomposition paradigm to compute the space of all feasible trajectories for the system. For example, Fig.1.3(iii) shows a sampling of the space of all time-optimal trajectories for the tipping of the block. In all of these trajectories, the angular motion of the block is exactly the same.

1.3 Background

In this section, we provide background on the planning and control of manipulation. Each chapter will also carry background material specific to the techniques used in that chapter.

The forward dynamics problem

Central to any model of contact manipulation is a model of friction. The friction model tells us everything we need to know about the interactions at the contacts — the forces that can be transmitted and their relationship to the motion at the contacts. The most commonly used model of friction is the *dry Coulomb friction* model. The model was empirically derived by Coulomb [Coulomb, 1781] and mirrors reality to a great extent.

The dry friction model, however, is non-Newtonian, *i.e.* a force exerted on a block resting on a surface with dry Coulomb friction might not produce a unique acceleration. This can lead to inconsistencies in the solutions to the forward dynamics problem. There are situations where there exist no feasible solutions. There are also situations where there exist multiple feasible solutions. Painlevé [Painlevé, 1895] was the first to notice these inconsistencies and hence the situation is known as *Painlevé's paradox*. An everyday occurrence of this phenomenon is when a chalk is pushed over a blackboard, often resulting in a dotted line. In this case, the dry friction model will show two consistent solutions — separation and sliding.

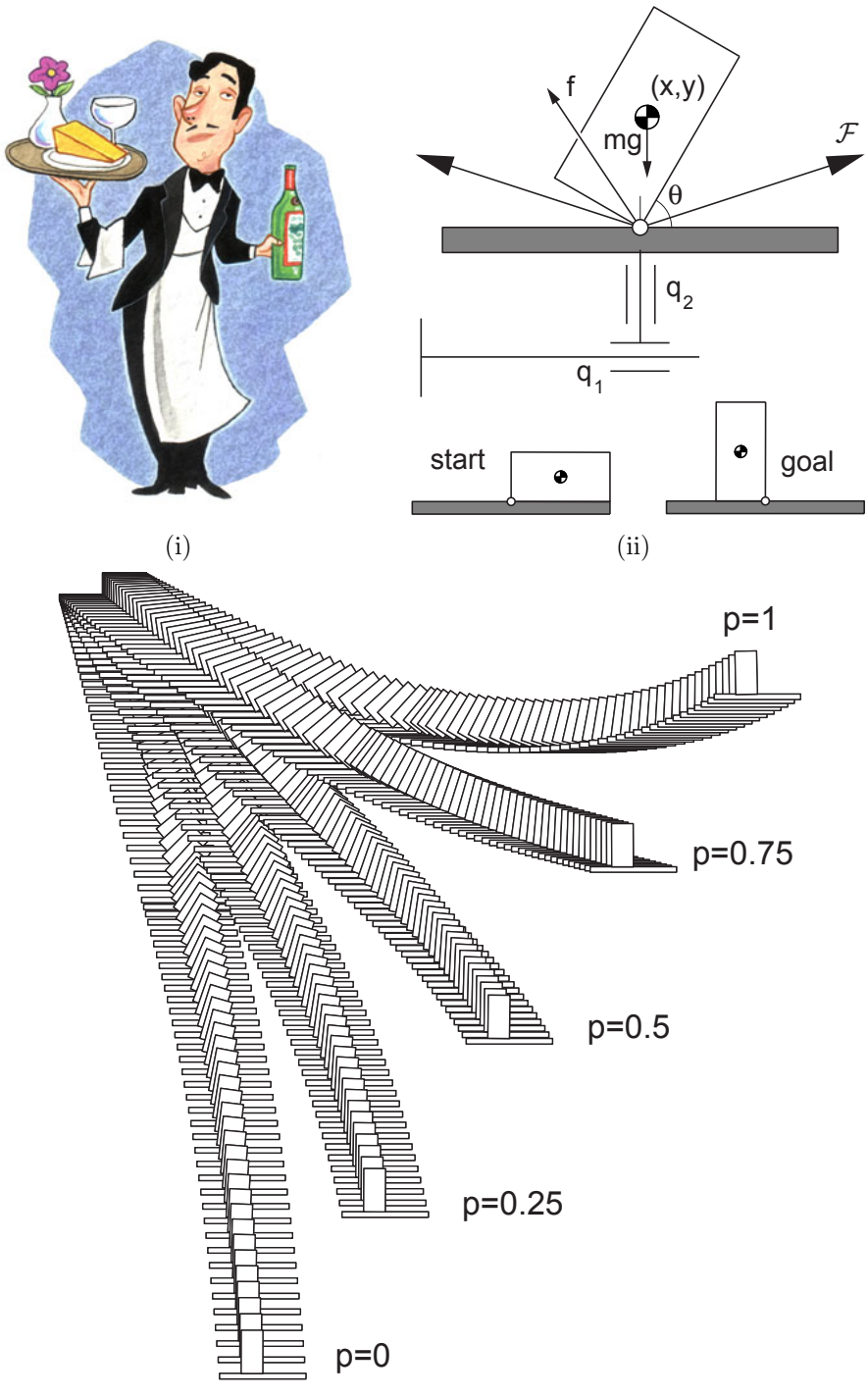


Figure 1.3: (i): The waiter's problem. (ii) A kinematic skeleton of the problem. (iii) A sampling of all feasible trajectories.

Table 1.1: A Survey of Contact Manipulation

	Rolling	Sliding	Coulomb friction	Dynamic	3D	Environmental contact	General
[Whitney, 1982]	Y	Y	Y	N	N	Y	N
[Kerr and Roth, 1986]	Y	N	Y	N	N	N	Y
[Fearing, 1986]	Y	N	Y	N	N	N	N
[Erdmann and Mason, 1988]	Y	Y	Y	N	N	Y	Y
[Brock, 1988]	Y	Y	Y	N	Y	N	N
[Cole et al., 1989]	Y	N	Y	Y	Y	N	Y
[Brost and Mason, 1989]	Y	Y	Y	N	N	Y	Y
[Trinkle and Paul, 1990]	Y	Y	Y	N	N	Y	Y
[Trinkle and Hunter, 1991]	Y	Y	N	Y	N	Y	Y
[Cole et al., 1992]	Y	Y	Y	Y	N	N	Y
[Paljug et al., 1994]	Y	N	Y	Y	N	N	N
[Rus, 1997]	Y	Y	N	Y	N	N	Y
[Sarkar et al., 1997b]	Y	N	N	Y	Y	N	Y
[Erdmann, 1998]	Y	Y	Y	N	N	Y	Y
[Lynch, 1999]	Y	N	Y	Y	N	N	N
[Cherif and Gupta, 1999]	Y	Y	Y	N	Y	N	Y
[Harada et al., 2000]	Y	N	N	Y	Y	Y	N
[Zheng et al., 2000]	Y	Y	Y	Y	Y	N	N
[Liu and Li, 2001]	Y	N	N	Y	Y	N	Y
[Maeda et al., 2001]	Y	N	Y	N	N	Y	N
[Blind et al., 2001]	Y	Y	N	Y	N	N	N
[Bicchi and Marigo, 2002]	Y	N	N	Y	Y	N	N
[Harada and Kaneko, 2002]	Y	Y	Y	N	Y	Y	Y
[Balkcom and Trinkle, 2002]	Y	Y	Y	N	N	Y	Y
[Srinivasa et al., 2005a]	Y	Y	Y	Y	N	Y	Y
[Srinivasa et al., 2005b]	Y	Y	Y	Y	N	N	N

The dry friction model was originally applicable to point contacts. Erdmann [Erdmann, 1994] generalized the model to include bodies with extent and constructed friction cones in the configuration space of rigid bodies. Using the construct, he was able to solve for the accelerations of rigid bodies in frictional contact. Erdmann also discussed situations where the solutions became inconsistent. [Lotstedt, 1981, Trinkle et al., 1997, Anitescu and Potra, 1997] used a more algebraic approach and showed that the forward dynamics problem belonged to a class of problems known as *linear complementarity problems* (LCP). They were then able to solve the problem using results from the well researched field of LCPs. Lotstedt [Lotstedt, 1981, Lotstedt, 1982, Lotstedt, 1984] applied the solutions to simulate multi-body dynamics with frictional contacts. He was also able to provide some theoretical results on the inconsistencies of the solutions based on existing results on LCPs.

Manipulation with hands

Dexterous manipulation refers to the manipulation of objects using the fingers of a robot hand. Research in dexterous manipulation has focussed both on the development of tactile sensors to better sense contact, slip and the forces exerted at the fingertips, and on the development of control algorithms to manipulate objects. We refer the reader to Okamura *et al.*[Okamura et al., 2002] for a review of the state of the art in tactile sensing. We shall focus on the latter development.

Kerr and Roth [Kerr and Roth, 1986] discussed three problems related to dexterous manipulation. The first problem was the selection of internal grasp forces — contact forces that cause no motion of the object. These forces lie in the nullspace of the grasp matrix. They framed the problem of finding the contact forces required to balance a given external wrench as a linear program. The constraints of the problem were Coulumb friction at each contact and actuator torque limits. The objective function they maximized was the Euclidean distance of the internal force from the constraints. The second problem was the kinematics of the manipulating hand for a given object motion. They derived the equations of motion of the robot hand given the motion of the object, for the case of rolling contacts. The third problem was the derivation of hand workspaces based on the observation that in the case of fixed point contacts, the object and the hand could together be represented as a closed chain manipulator.

Fearing [Fearing, 1986] explored the reorientation of grasped objects using a manipulation strategy called a *twirling rotation*. The motion is akin to twirling a pencil with three fingers, albeit quasi-static. He demonstrated the twirling of a baton using the three-fingered Stanford/JPL hand. The focus was more on developing a robust force control strategy that could be demonstrated on a robot than motion planning. Fearing studied the failure of the strategy due to slip and the effect of rolling at the cylindrical fingertips.

Brock [Brock, 1988] described the use of *controlled slip* — allowing fingers to slide on the object without the requirement of a stable grasp — as a manipulation strategy. For a particular contact mode, he used a quasi-static graphical analysis to compute the range of finger torques that causes a desired contact to slip. He demonstrated how the analysis could be used to plan the reorientation of a cylinder using controlled slip.

Cole, Hauser, and Sastry [Cole et al., 1989, Cole et al., 1992] considered the problem of dynamic control of a multi-fingered hand manipulating a planar object, where some of the fingers were designated to slide on the object's surface while the others were designated to roll. They suggested that these motions might be used by the hand to regrasp the object, if needed. The authors provided a dynamic control law that achieved simultaneous tracking of a pre-planned object trajectory together with the desired sliding motion at the fingertips. The authors assumed that the location of the contact points, and the relative velocity at the contact points were available for feedback.

Paljug *et al.*[Paljug et al., 1994] considered a cooperative manipulation task where two 3DOF robots controlled a smooth planar object. While the motion was dynamic, the two contacts on the object were positioned to guarantee force closure at all times. The authors proposed a nonlinear feedback scheme for motion control as well as an algorithm for the placement and motion of the contacts which considered both limits on the magnitude of contact forces (to prevent fragile objects from shattering) and robustness to external disturbance wrenches.

Li, Canny and Sastry [Li et al., 1989] provided a differential geometric framework for dexterous manipulation. They decomposed the problem into various *manipulation modes* — coordinated manipulation, rolling motion, sliding motion, and finger relocation — and described the force and motion constraints for each mode. This was Part I, with the promise that Part II would solve the motion planning problem for dexterous manipulation. Sadly, there was no Part II.

Trinkle and Hunter [Trinkle and Hunter, 1991] also provided a framework for dexterous manipulation planning. They defined a contact formation (CF) as a qualitative description of a grasp based on contacts between the vertices, edges and surfaces of the robot and the object. To plan a motion from a start CF to a goal CF, they constructed CF-trees with the start and goal CFs as the parent nodes, the child nodes being the CFs the parents could transition to, and the arcs being joint angles and input torques that caused the transitions. A plan was returned when the the start and goal CF-trees had a common node.

Li and Sastry [Li and Sastry, 1988] provided a grasp metric that was based on the task that had to be accomplished. Based on the force and torque requirements of the task, they constructed a *task ellipsoid* in the object's wrench space. The shape of the ellipsoid was chosen based on the expected load along the different axes of the object's wrench space due to external forces. The metric was the largest radius of the ellipsoid that could be embedded

in the range of the grasp map. This metric is somewhat equivalent to the smallest eigenvalue of the grasp map with the eigenvalues weighted by the importance of their corresponding eigenvectors to the task.

Bicchi [Bicchi, 1994] studied the problem of immobilizing an object which is in contact both with the robot and the environment. By including the kinematics of the robot in the grasp equations of the object, he was able to decompose the wrench space of the object into three subspaces — a subspace where a force applied by the robot produced a wrench on the object, a subspace where a force applied by the robot produced no wrench on the object (called the space of *active internal forces*) and a subspace where a force applied by the environment produced no wrench on the object (called the space of *preload internal forces*). Using a metric based on the three subspaces, he was able to construct optimal grasps for the object.

Rus [Rus, 1997] addressed the reorientation of 3D polyhedra with four frictionless point fingertips. The manipulation strategy, called *finger tracking*, consisted of moving one fingertip compliantly along the object while keeping the other fingers stationary. Rus showed that, using this strategy, the dynamics of the object was reduced to an instantaneously linear problem and constructed a planner and a controller to reorient 3D polyhedra. Gupta [Gupta, 1995] used the linearity result to construct a planner for reorienting polyhedra by searching a tree of landmarks placed randomly in the state space of the object.

Cherif and Gupta [Cherif and Gupta, 1999] studied the manipulation of a smooth 3D object by a hand whose fingers were allowed to both roll and slide on the object, but not make or break contact. They decomposed the motion planning problem into a discrete grid-based global search in the object's configuration space coupled with a local search for finger motions that would cause the desired object motion from one grid point to another. The local search was performed using a quasi-static analysis and by choosing feasible contact modes at random. The authors used a pyramidal approximation for the contact friction cone and discretized the number of sliding directions at each contact point.

Zheng *et al.*[Zheng et al., 2000] studied the manipulation of an object in 3D by a three-fingered hand. They allowed one of the fingers to slide compliantly on the object while the other two fingers maintained a fixed contact with the object. For a given system state and a desired object motion, they derived a condition for the allowable sliding directions for the sliding finger. Planning was done by sampling the space of allowable sliding directions and choosing a feasible direction.

Manipulation without hands

In this section, we discuss background work on flavors of manipulation that do not involve a robot hand interacting with an object. These include the use of fixtures, palms, pins

and even other objects as manipulators. Many of these works are motivated by the idea of *minimalism*, finding the simplest actuator that can perform the specific task, as opposed to the idea of *general purpose actuators*, hands that could be used to to perform several tasks, that motivated a lot of the research in dexterous manipulation.

The peg-in-hole problem —inserting a rectangular peg into a sightly larger hole — was perhaps the first manipulation problem that generated a huge amount of interest and publications. We refer the reader to Simunovic's thesis [Simunovic, 1975] and Whitney's paper [Whitney, 1982] for a detailed review. Whitney and his colleagues at the Draper Labs analyzed the effects of friction on manipulation and proposed a controller (a physical support known as the *Remote Center of Compliance*) to make peg-in-hole insertion robust to failures due to jamming and wedging.

Balkcom and Trinkle [Balkcom and Trinkle, 2002] analyzed the quasi-dynamic manipulation of planar parts subject to contacts with the environment, in the context of part-fixturing. Given a part and a set of frictional fixture locations, they planned for insertion force required to ensure that the part fit securely in the fixtures. Furthermore, they solved the problem of finding the set of fixture locations for a given line of action of the insertion force.

Blind *et al.*[Blind et al., 2001] proposed a novel technique for manipulation using a grid of retractable pins mounted on a vertical plate, much like a *Pachinko machine*. They computed pin configurations such that any part that was dropped into the machine would be trapped by the pins. To manipulate the part from one pin configuration to another, they simply retracted the former and pushed out the latter set of pins. Their technique is unique because it is *insensitive* to environmental dynamics. The exact motion of the part during its dynamic phase is irrelevant, and it always ends up being captured by the pins.

In an earlier paper, Erdmann and Mason [Erdmann and Mason, 1988] explored the similar idea of using environmental contacts to reduce uncertainty in the pose of the manipuland. They dynamically manipulated a polygonal part placed inside a tray by tilting the tray. Their planner generated tilt sequences that resulted in a unique final part pose independent of its starting pose. No matter what pose the part started in, at the end of a tilt it ended up in one of a finite set of statically stable configurations. Each successive tilt operation further reduced the pose uncertainty until, eventually, the set of possible poses was reduced to a singleton set, signifying the success of the plan.

Bicchi and Marigo [Bicchi and Marigo, 2002] manipulated objects by rolling them between the jaws of a parallel jaw gripper. They ensured rolling contacts by coating the gripper with a high friction material. They explored both the problem of shape recovery from tactile information and the problem of planning jaw motions for the desired manipulation of the object.

Harada *et al.*[Harada et al., 2000] analyzed the dynamic manipulation and grasping of

multiple objects using a robot hand. They provided constraints on the geometry of the objects and on the friction that would result in a unique contact mode, namely rolling at each and every one of the remote contacts. Under the conditions of rolling, they developed a control scheme that robustly manipulated the multiple objects.

Sarkar, Yun, and Kumar [Sarkar et al., 1997a] explored the problem where a robot had to maintain rolling contact with an object, the motion of which was given to the robot. They motivated the problem with the example of cooperative manipulation, a task where two or more robots synergistically manipulated an object. They likened their problem to the problem of a slave robot in the cooperative manipulation task, ordered to compliantly follow the motion of the object. They designed a nonlinear feedback controller that successfully maintained rolling contact.

Lynch and Mason [Lynch and Mason, 1999] used a robot arm with a single revolute degree of freedom to demonstrated dynamic tasks such as snatching, rolling, throwing and catching an object. This work was the inspiration for the waiter's problem and will be described in more detail in Ch.6.

Erdmann [Erdmann, 1998] explored the nonprehensile manipulation of planar convex objects using two flat palms, each of which had three degrees of freedom. The aim was to construct palm motions that manipulated the object from a starting (x, y, θ) pose to a desired pose. Erdmann restricted the actions that could be performed by the palms to four primitives — TILT, ROTATE (REGRIP)[1], SLIDE and RELEASE — each corresponding to a different contact mode. Based on a static analysis, Erdmann decomposed the configuration space into regions of invariant contact mode, and searched for plans in this simplified space.

The idea of decomposing configuration space into regions of invariance was first proposed by Schwartz and Sharir [Schwartz and Sharir, 1983a, Schwartz and Sharir, 1983b, Schwartz and Sharir, 1983c, Schwartz and Sharir, 1984] to solve the, now famous, *Piano mover's problem*. The problem involves computing a continuous obstacle-avoiding motion of the piano (3D polyhedra) in a cluttered room (3D obstacle field). To obtain a discrete representation of the free configuration space of the object, they decomposed the space into *path-connected components* — a collision-free path existed between two configurations if and only if they were both in the same connected component. They then constructed a *connectivity graph* whose nodes represented the components and whose edges connected portions that were adjacent to each other in the free configuration space. By doing so, they reduced the continuous motion planning problem into a discrete problem of searching for a path in the connectivity graph.

To compute an algebraic representation of the path connected components, Schwarz and

[1]ROTATE and REGRIP share the same contact mode, namely rolling at each contact, but perform fundamentally different actions. In ROTATE, one palm rotates about the *other* palm's contact point, while in REGRIP the palm rotates about *its own* contact point.

Sharir [Schwartz and Sharir, 1983b] used a variant of Collins' *cylindrical algebraic decomposition* technique [Collins, 1975] — a recursive technique whose input is a set of collision constraints (or *critical events*) which formed the boundaries of the components and whose output is a decomposition of the free configuration space into components. It must be noted that the cylindrical algebraic decomposition provides no guarantees that the generated decomposition will possess the least number of components. A similar technique was used by Erdmann [Erdmann, 1998] to compute regions of invariant contact mode.

Table 1.1 summarizes the background work detailed in the last two sections by classifying contact manipulation based on seven attributes — the treatment of rolling and sliding contacts, the use of the Coulomb friction contact model, a dynamic model of manipulation, motion in 3D as opposed to planar motion, the presence of contact with the environment, and the generality of the result. The last attribute distinguishes a result that can be applied to a class of problems (for example, the results of Erdmann [Erdmann, 1998] can be applied to the class of convex polygons) as opposed to a result for a specific instance of a problem (for example, the results of Fearing [Fearing, 1986] are specific to a three-fingered hand performing a twirling operation).

1.4 Roadmap

Ch.2 describes the technique for combining actuator constraints and Coulomb friction constraints into a common contact acceleration space. Ch.3 describes and solves the block-standing problem. Ch.4 describes the task and shape decomposition paradigm for solving constrained dynamical systems. Ch.5 and Ch.6 use the paradigm to solve the Mobipulator problem and the waiter's problem, respectively. Ch.7 discusses the contributions of the thesis and future work.

1.5 Dependencies

The following information is useful for readers interested in reading individual chapters.

- Ch.1, Ch.2, and Ch.4 are self-contained.
- Ch.3 requires Ch.2.
- Ch.5 requires Ch.4.
- Ch.6 requires Ch.2 and Ch.4.

1.6 Publication note

Most of Ch.2 and Ch.3 appeared in [Srinivasa et al., 2005a]. Parts of Ch.5 appeared in [Srinivasa et al., 2002] and [Srinivasa et al., 2003]. Parts of Ch.4 and Ch.6 appeared in [Srinivasa et al., 2005b]. The design and construction of the Mobipulator described in Ch.5 is joint work with Cristopher Baker and Greg Reshko. The configuration space planner described in Ch.5 is joint work with Elisha Sacks. Research on walking in Ch.7 is joint work with Jonathan Hurst. Research on motion retargeting in Ch.7 is joint work with Nancy Pollard.

Chapter 2

Combining robot and object constraints

In this chapter, we discuss the combination of the constraints on the manipulator and the constraints imposed by the laws of Coulomb friction for the maintenance of a desired contact mode. In §2.1 we introduce the nomenclature used in this thesis and provide a definition of the problem. In subsequent sections, we combine the constraints using algebraic matrix operations and provide a geometric intuition for the algebra for the case of a single point contact.

2.1 Nomenclature

We denote the configuration of the object by its pose $\boldsymbol{q_o}$ and the configuration of the robot by its joint variables $\boldsymbol{q_r}$. We denote the configuration of the system by $\boldsymbol{q} = [\boldsymbol{q_o}, \boldsymbol{q_r}]^T$, the velocity of the system by $\dot{\boldsymbol{q}} = [\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}]^T$ and the state of the system by the pair $[\boldsymbol{q}, \dot{\boldsymbol{q}}]^T$.

Motion at the contacts between the object and the robot gives rise to constraints on the relative velocity between the object and the robot. These can be written as:

$$\mathsf{G}_\mathsf{r}^\mathsf{T} \dot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{J}_\mathsf{r} \dot{\boldsymbol{q}}_{\boldsymbol{r}} \tag{2.1}$$

where $\mathsf{G}_\mathsf{r}^\mathsf{T}$ is the mapping from the velocity of the object $\dot{\boldsymbol{q}}_{\boldsymbol{o}}$ to the velocity of the object at the points of contact between the object and the robot. Likewise, J_r maps the velocity of the joint variables of the robot to the velocity of the robot at the points of contact between the robot and the object. In case of a rolling contact, the relative velocity at the contact is zero while in the case of a sliding contact, the relative velocity at the contact normal to the contacting bodies is zero.

Constraints on the motion at the contacts between the object and the environment can

be written as:

$$\mathsf{G}_\mathsf{e}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} = 0 \tag{2.2}$$

where $\mathsf{G}_\mathsf{e}^\mathsf{T}$ is the mapping from the velocity of the object $\dot{\boldsymbol{q}}_{\boldsymbol{o}}$ to the velocity of the object at the points of contact between the object and the environment. Since the environment is static, in the case of a rolling contact, the velocity of the object at the contact point is zero while in the case of a sliding contact, the velocity of the object at the contact point which is normal to the environment is zero.

We can combine the above two constraints as:

$$\mathsf{G}_\mathsf{c}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{J}_\mathsf{c}\dot{\boldsymbol{q}}_{\boldsymbol{r}} \tag{2.3}$$

Constraints on the contact force $\boldsymbol{f_{ci}}$ at the ith contact are:

$$\boldsymbol{f_{ci}} \in \mathcal{F}_i$$

where $\mathcal{F}_i$ is either the contact friction cone in the case of a rolling contact or the edge of the friction cone that opposes motion in the case of a sliding contact, $i = [1, \ldots, n]$ and n is the number of contacts.

We can combine the force constraints at all contacts as:

$$\boldsymbol{f_c} \in \mathcal{F} \tag{2.4}$$

where $\boldsymbol{f_c} = [\boldsymbol{f_{c1}}, \ldots, \boldsymbol{f_{cn}}]^T$ and $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \mathcal{F}_n$.

The equations of motion of the object can be written as:

$$\mathsf{M}\ddot{\boldsymbol{q}}_{\boldsymbol{o}} + \boldsymbol{C}(\boldsymbol{q_o}, \dot{\boldsymbol{q}}_{\boldsymbol{o}}) = \mathsf{G}\boldsymbol{f_c} + \boldsymbol{n_o} \tag{2.5}$$

where M is the inertia matrix of the object, $\boldsymbol{C}(\boldsymbol{q_o}, \dot{\boldsymbol{q}}_{\boldsymbol{o}})$ is a collection of the centrifugal and Coriolis terms, G is the grasp map which maps the forces acting on the object at the contact points to wrenches about the object's center of gravity, and $\boldsymbol{n_o}$ is the wrench on the object due to gravity.

Actuator constraints appear in the form of constraints on joint acceleration $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}$. These constraints can be state dependent.

2.2 A simple example

We will now present a simple example to familiarize the reader with the various mappings introduced in the previous section. Consider a planar rigid body which is in contact both with a vertical wall and with a robot with a prismatic joint. As shown in Fig.2.1, the rod

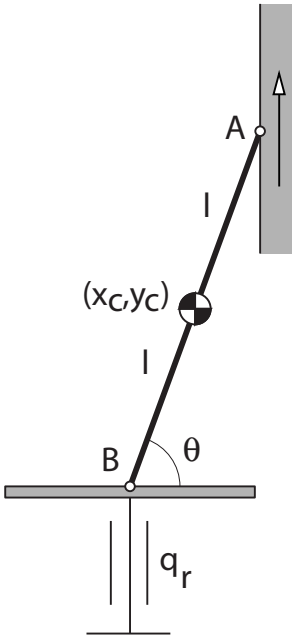


Figure 2.1: A simple example

is in contact with the wall at A and the robot at B. The center of mass of the rod is located at its geometric center and is a distance l away from both A and B. We denote the configuration of the rod by the location of its center of mass (x, y) and its orientation θ.

We desire a fixed contact at B and a sliding contact at A.

In this section, we will worry only about defining the maps and not about whether the desired motion is possible or not. As an aside, if the system is initially at rest and the line AB is enclosed by the friction cones at A and B, then the only way the desired motion can be satisfied is if the rod free falls, *i.e.*if the arm moves downward with an acceleration of g (this geometric condition for two point contact is called *Nguyen's condition* [Nguyen, 1988]).

Since contact B is fixed, the velocity of the robot at B must equal the velocity of the rod at B. This gives us:

$$\mathsf{J}_r = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.6}$$

$$\mathsf{G}_r^\mathsf{T} = \begin{bmatrix} 1 & 0 & l\sin(\theta) \\ 0 & 1 & -l\cos(\theta) \end{bmatrix} \tag{2.7}$$

Since contact A is sliding, the velocity of the rod at A normal to the surface must be

zero to prevent separation or penetration. This gives us:

$$\mathsf{G}_\mathsf{e}^\mathsf{T} = \begin{bmatrix} 1 & 0 & -l\sin(\theta) \end{bmatrix} \tag{2.8}$$

The combined maps are given by:

$$\mathsf{G}_\mathsf{c}^\mathsf{T} = \begin{bmatrix} 1 & 0 & l\sin(\theta) \\ 0 & 1 & -l\cos(\theta) \\ 1 & 0 & -l\sin(\theta) \end{bmatrix} \tag{2.9}$$

$$\mathsf{J}_\mathsf{c} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.10}$$

The grasp map is given by:

$$\mathsf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ l\sin(\theta) & -l\cos(\theta) & -l\sin(\theta) & l\cos(\theta) \end{bmatrix} \tag{2.11}$$

In the next section, we map the velocity and force constraints on the system into a common contact acceleration space. In the subsequent sections, we will show how actuator constraints can be incorporated in this space.

2.3 Combining constraints

Theorem 1. *For a given configuration* $\boldsymbol{q} = [\boldsymbol{q_o}, \boldsymbol{q_r}]^T$ *and velocity* $\dot{\boldsymbol{q}} = [\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}]^T$*, the allowable joint acceleration of the robot* $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}$ *that satisfies both the velocity and the force constraints is constrained to lie within the cone:*

$$\boxed{\mathsf{J}_\mathsf{c}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} + \mathsf{V}(\boldsymbol{q_o}, \boldsymbol{q_r}, \dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) \in \mathcal{A}} \tag{2.12}$$

$$\mathsf{V}(\boldsymbol{q_o}, \boldsymbol{q_r}, \dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) = \dot{\mathsf{J}}_\mathsf{c}\dot{\boldsymbol{q}}_{\boldsymbol{r}} - \dot{\mathsf{G}}_\mathsf{c}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} - \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}(\boldsymbol{n_o} - \boldsymbol{C}(\boldsymbol{q_o}, \dot{\boldsymbol{q}}_{\boldsymbol{o}}))$$

and

$$\mathcal{A} = \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}(\mathcal{F})$$

is the contact acceleration cone.

Proof. The constraints on the system are:

$$\mathsf{G}_\mathsf{c}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} - \mathsf{J}_\mathsf{c}\dot{\boldsymbol{q}}_{\boldsymbol{r}} = 0 \tag{2.13}$$

$$\boldsymbol{f_c} \in \mathcal{F} \tag{2.14}$$

Since the velocity constraint is valid over an interval of time, its derivative *wrt.* time is also valid over the interval of time. We can, thus, differentiate Eqn.2.13 :

$$\mathsf{G}_\mathsf{c}^\mathsf{T}\ddot{\boldsymbol{q}}_{\boldsymbol{o}} + \dot{\mathsf{G}}_\mathsf{c}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} - \mathsf{J}_\mathsf{c}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} - \dot{\mathsf{J}}_\mathsf{c}\dot{\boldsymbol{q}}_{\boldsymbol{r}} = 0 \tag{2.15}$$

We can rewrite the dynamics of the object as:

$$\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{M}^{-1}\mathsf{G}\boldsymbol{f_c} + \mathsf{M}^{-1}(\boldsymbol{n_o} - \boldsymbol{C}(\boldsymbol{q_o}, \dot{\boldsymbol{q}}_{\boldsymbol{o}})) \tag{2.16}$$

Multiplying the equation by $\mathsf{G}_\mathsf{c}^\mathsf{T}$ and substituting for $\mathsf{G}_\mathsf{c}^\mathsf{T}\ddot{\boldsymbol{q}}_{\boldsymbol{o}}$ from Eqn.2.15, we get:

$$\begin{gathered}
\mathsf{G}_\mathsf{c}^\mathsf{T}\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}\boldsymbol{f_c} + \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}(\boldsymbol{n_o} - \boldsymbol{C}(\boldsymbol{q_o}, \dot{\boldsymbol{q}}_{\boldsymbol{o}})) \\
\mathsf{J}_\mathsf{c}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} - \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}(\boldsymbol{n_o} - \boldsymbol{C}(\boldsymbol{q_o}, \dot{\boldsymbol{q}}_{\boldsymbol{o}})) - \dot{\mathsf{G}}_\mathsf{c}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} + \dot{\mathsf{J}}_\mathsf{c}\dot{\boldsymbol{q}}_{\boldsymbol{r}} = \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}\boldsymbol{f_c} \\
\mathsf{J}_\mathsf{c}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} + \mathsf{V}(\boldsymbol{q_o}, \boldsymbol{q_r}.\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) = \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}\boldsymbol{f_c}
\end{gathered}$$

Incorporating the force constraint from Eqn.2.14, we can write the above equation as:

$$\mathsf{J}_\mathsf{c}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} + \mathsf{V}(\boldsymbol{q_o}, \boldsymbol{q_r}, \dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) \in \mathsf{G}_\mathsf{c}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}(\mathcal{F}) \tag{2.17}$$

where V is a collection of terms comprising of the system velocity and the weight of the object. □

Eqn.2.12 represents a balance of accelerations in contact space. The LHS of Eqn.2.12 represents contact accelerations that can be produced by the robot. As per the laws of Coulomb friction, the acceleration must lie within the cone of contact accelerations that can be produced by forces belonging to the contact friction cone. A sketch of the various mappings is shown in Fig.2.2. The RHS of Eqn.2.12 maps the contact friction cone into the space of contact accelerations:

$$\underbrace{\underbrace{\underbrace{\mathsf{G}_\mathsf{c}^\mathsf{T}\,\mathsf{M}^{-1}\,\mathsf{G}\qquad\qquad\overbrace{(\mathcal{F})}}^{\text{Contact space friction cone}}}_{\text{Object space friction cone}}}_{\text{Object space acceleration cone}}}_{\text{Contact space acceleration cone}} \tag{2.18}$$

We shall, henceforth, refer to this map as the *contact map*.

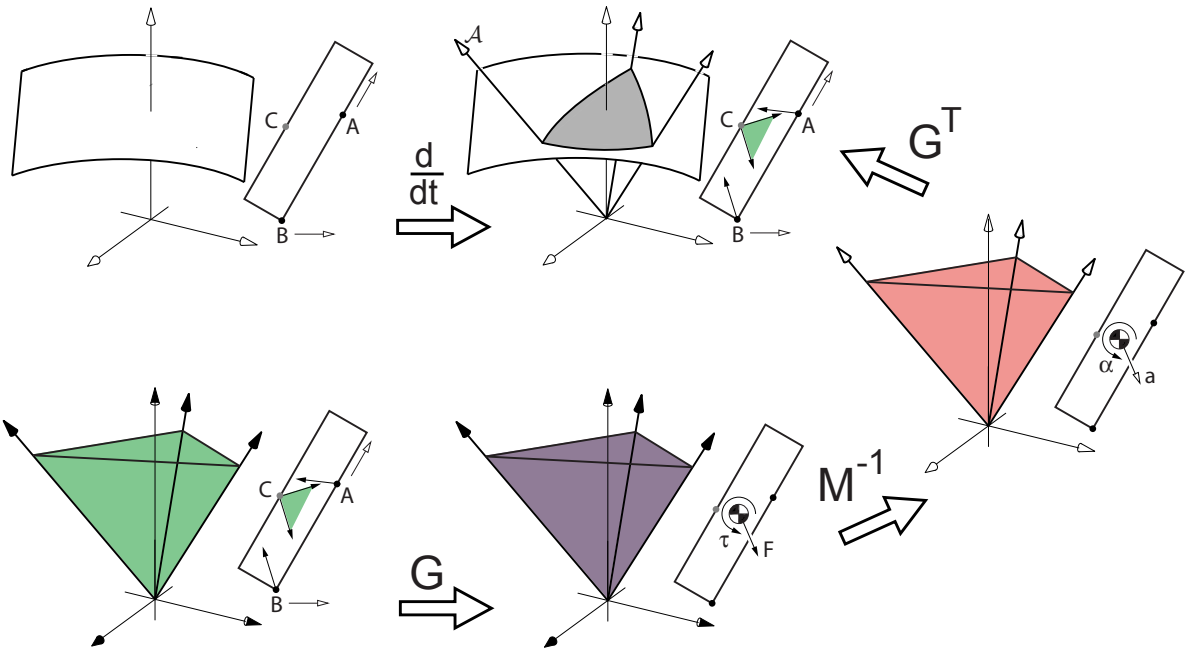


Figure 2.2: A series of transformations map the contact force constraints and the contact velocity constraints into the contact acceleration space.

It is important to observe that the contact map depends only on the configuration of the object and *not* on its velocity. This is indeed desirable because we know that the contact friction cone is unaffected by the velocity of the object and we would want the contact acceleration cone to share the same property. With Eqn.2.12, we achieve a decoupling of the dynamics of the object (given in the LHS) and the Coulomb friction constraints on the object (given in the RHS). An advantage of this decoupling for planning is that we can reuse the contact acceleration cone for different velocities of the object with the same configuration, saving computation.

2.4 Internal forces

Eqn.2.12 is a necessary but not a sufficient condition for the desired contact mode to be guaranteed if the system has *internal forces* — forces that produce no acceleration of the system.

Internal forces can result if the object is in force closure. For example, in the system shown in Fig.2.3, contact forces pointing inwards do not accelerate the object.

When we apply the contact acceleration constraint for the system initially at rest and

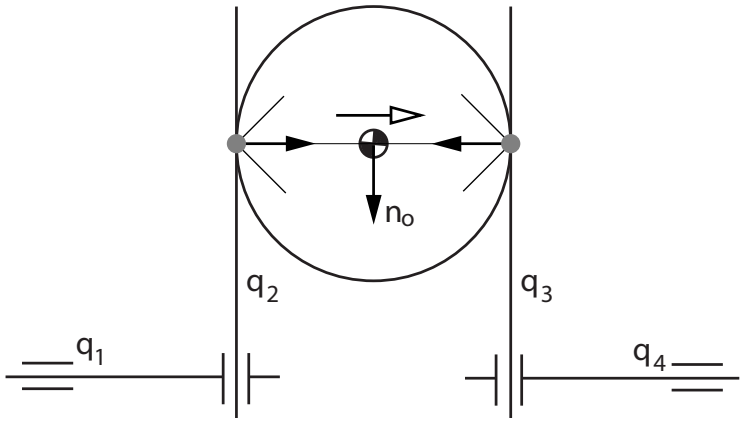


Figure 2.3: An example of the insufficiency of the contact acceleration constraints

moving to the right, we get:

$$\ddot{\boldsymbol{q}}_1 = \ddot{\boldsymbol{q}}_4 \tag{2.19}$$

Eqn.2.19 prohibits the contacts from separating or penetrating into the object. However Eqn.2.19 does not tell us how much we should squeeze the object in order to balance its weight $\boldsymbol{n_o}$. This is because the squeezing force does not change the acceleration of the robot and hence does not appear in the contact acceleration constraint.

To obtain information about the internal forces, we need to augment the contact acceleration constraint with either a model of the stresses and strains of the system or a sensor that can help us compute the internal forces. For example, if in the above example, the robot arm was augmented with a force/torque sensor that measured the internal squeezing force, we can guarantee that the object will not fall through by commanding the arms to squeeze until an internal force threshold was reached.

2.5 Single point contact

In this section, we will analyze another example problem — a planar rigid body in single point frictional contact. This particular problem has been analyzed both by Erdmann [Erdmann, 1994] and by Lynch [Lynch, 1996, pp.89]. We will show how our theorem relates to both works.

The contact map

The example problem is illustrated in Fig.2.4. We denote the pose of the object by the location of its center of mass (x_c, y_c) in a world frame and its orientation θ. We denote the

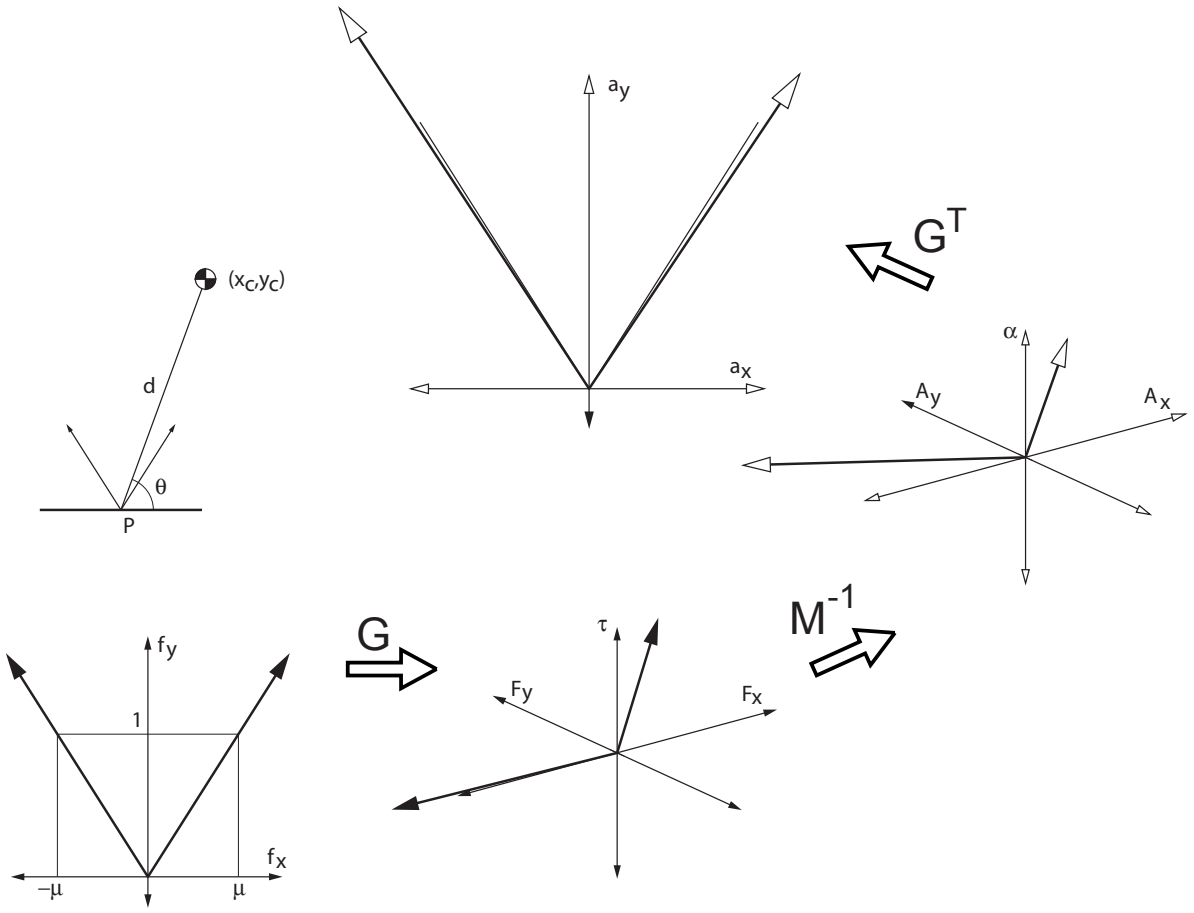


Figure 2.4: The mapping from contact forces to contact accelerations for a planar rigid body with a single point contact. Solid arrows are used to denote forces and wrenches while unfilled arrows are used to denote accelerations and twists.

mass of the object by m and its radius of gyration by ρ. We denote the distance from the center of mass to the contact point P by d. We desire a fixed contact at P.

The relevant maps for this system are given by:

$$\mathsf{G}_\mathsf{r} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ d\sin(\theta) & -d\cos(\theta) \end{bmatrix} \tag{2.20}$$

Since there is no contact with the environment,

$$\mathsf{G}_\mathsf{e} = \emptyset \tag{2.21}$$

giving us:

$$\mathsf{G_c} = \mathsf{G_r} \tag{2.22}$$

Furthermore, the grasp map G is given by:

$$\mathsf{G} = \mathsf{G_r} \tag{2.23}$$

Since the maps $\mathsf{G_r}$, $\mathsf{G_c}$ and G are identical for this example, we shall henceforth denote them all by G to reduce the subscript clutter.

The inertia matrix of the object is given by:

$$\mathsf{M} = m \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \rho^2 \end{bmatrix} \tag{2.24}$$

The inverse of the inertia matrix is given by:

$$\mathsf{M}^{-1} = \frac{1}{m\rho^2} \begin{bmatrix} \rho^2 & 0 & 0 \\ 0 & \rho^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

The contact map is given by:

$$\mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G} = \frac{1}{m\rho^2} \begin{bmatrix} \rho^2 + y^2 & -xy \\ -xy & \rho^2 + x^2 \end{bmatrix} \tag{2.26}$$

where

$$\begin{aligned} x &= -d\cos(\theta) \\ y &= -d\sin(\theta) \end{aligned}$$

i.e.(x, y) are the coordinates of the vector from the center of mass to the point of contact.

The maps are illustrated in Fig.2.4. The contact space friction cone is shown in the lower left. The grasp map G maps the friction cone to a cone in wrench space. Notice that the wrench space cone spans a two dimensional subspace of the three dimensional wrench space. The map M^{-1} transforms the wrench space cone to a cone in acceleration twist space. Since the inertia matrix is full rank, it maps the two dimensional wrench cone to a two dimensional cone in acceleration twist space. The map G^{T} then projects the cone onto the two dimensional contact acceleration space it resides in. This is shown as the topmost plot in Fig.2.4. The original two dimensional friction cone is also shown in that plot. Notice that while the contact friction cone is symmetric, the contact acceleration cone is not. This

is evident from the off-diagonal terms in the contact map in Eqn.2.26.

Geometric interpretation

The geometric interpretation of the contact map stems from Lynch's [Lynch, 1996] observation that the mapping between the applied wrench $\boldsymbol{w}$ on an object and the acceleration twist $\boldsymbol{a}$ produced can be represented by an ellipsoidal surface whose dimensions depend on the inertial properties of the rigid body. For the case of our planar rigid body, the ellipsoid is given by:

$$F(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\mathsf{M}^{-1}\boldsymbol{w} \tag{2.27}$$

It is easy to see that the acceleration twist produced is given by the gradient to the ellipsoidal surface:

$$\boldsymbol{a} = \nabla(F) = \mathsf{M}^{-1}\boldsymbol{w} \tag{2.28}$$

Now, suppose that the wrench is measured about a frame that is not the inertial frame, but one that is aligned with the inertial frame but located at (x, y) away from the inertial frame, then that wrench $\boldsymbol{w'}$ is related to $\boldsymbol{w}$ by a shear matrix S given by:

$$\mathsf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -y & x & 1 \end{bmatrix} \tag{2.29}$$

$$\boldsymbol{w} = \mathsf{S}\boldsymbol{w'} \tag{2.30}$$

Substituting for $\boldsymbol{w}$, we get:

$$F'(\boldsymbol{w'}) = \frac{1}{2}\boldsymbol{w'}^{\boldsymbol{T}}\mathsf{S}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{S}\boldsymbol{w'} \tag{2.31}$$

Furthermore, if the wrenches in the new coordinate frame are of the form $\boldsymbol{w'} = (\boldsymbol{f}, 0)$, *i.e.*if the wrenches in the new coordinate frame are pure forces that do not produce any torques, then we can further simplify Eqn.2.31 as:

$$F'(\boldsymbol{f}) = \frac{1}{2}(\boldsymbol{f}, 0)^T\mathsf{S}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{S}(\boldsymbol{f}, 0) = \frac{1}{2}\boldsymbol{f}^T\mathsf{M}'^{-1}\boldsymbol{f} \tag{2.32}$$

Substituting for S in Eqn.2.32, we get:

$$\mathsf{M}'^{-1} = \frac{1}{m\rho^2}\begin{bmatrix} \rho^2 + y^2 & -xy \\ -xy & \rho^2 + x^2 \end{bmatrix} \tag{2.33}$$

Taking the gradient of the ellipsoid $F'(\boldsymbol{f})$, as before, we get:

$$\boldsymbol{a'} = \mathsf{M'}^{-1}\boldsymbol{f} \tag{2.34}$$

where $\boldsymbol{a'}$ is a pure translational acceleration. This matches our intuition that a force passing through a point cannot produce an angular acceleration about that point.

Comparing with Eqn.2.26, we see that the two maps are identical:

$$\mathsf{M'}^{-1} = \mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G} \tag{2.35}$$

Hence, the contact map is the equivalent of an inertia matrix (or rather, its inverse) in a reference frame that is located at the contact point — mapping contact forces to contact accelerations.

Relationship to the configuration space friction cone

In [Erdmann, 1994], Erdmann described a geometric method for determining the possible motions of a part with multiple frictional contacts subjected to an applied force and torque. His motivation was the observation that for a point mass in frictional contact, the reaction force of the ground can be computed by geometrically projecting the applied force onto the friction cone. To be able to do the same for a rigid body in frictional contact, one needs to have a representation of the friction cone in wrench space.

Erdmann first constructed a generalized velocity space and, its dual, a generalized force space. The generalized velocity space was constructed such that the dot product between any two elements of the space was equivalent to the normal dot product in $\mathbb{R}^3$. This is achieved by scaling the angular velocity of the planar rigid body by its radius of gyration. The geometric consequence of this space is that the inertia ellipsoid in wrench space is mapped to a sphere in generalized force space. Since the gradient of a sphere at any point is directed along the corresponding radius vector, the generalized acceleration is always directed along the causal generalized force. Hence one can directly compare (take normal dot products of) forces and accelerations without having to go through the inertia matrix. In other words, *directions* in the the generalized force space and the generalized acceleration space are identical, with the magnitudes scaled by the mass.

Erdmann showed that the generalized friction cone was a two dimensional planar subset of the three dimensional generalized force space. Furthermore, he showed that the friction cone was orthogonal to the vector $\boldsymbol{t_r}$ which corresponded to a pure rotation about the contact point, a motion that the contact forces cannot generate by themselves.

The contact map is a projection of the contact forces to the plane perpendicular to $\boldsymbol{t_r}$. We will show this by first relating $\boldsymbol{t_r}$ to G. All motions that can be generated by the contact

forces lie in the range of G^T. Hence any motion that cannot be generated by the contact forces must lie in the nullspace of G^T. Thus,

$$\boldsymbol{t_r} \in \mathcal{N}(\mathsf{G}^\mathsf{T}) \tag{2.36}$$

We will now show that the contact map annihilates any twist that lies along $\boldsymbol{t_r}$. Now, $\boldsymbol{t_r}$ is an acceleration twist and hence has already passed through the $\mathsf{M}^{-1}\mathsf{G}$ projections of the contact map (it lies in the rightmost space in Fig.2.2). We are left to show that $\mathsf{G}^\mathsf{T}\boldsymbol{t_r} = 0$, which is evident from Eqn.2.36.

Hence, the contact map is a projection of the contact forces onto the two dimensional subspace in which the configuration space friction cone resides.

Adding actuator constraints

Our discussion until now has involved only the grasp map G and not the actuator's dynamics or constraints. Let the single point contact we have been analyzing be between the planar rigid body and the end effector of a robot arm with two prismatic joints, as shown in Fig.2.5. We denote the configuration of the arm by its joint variables (q_1, q_2). The end effector is rigidly attached to the arm and is always horizontal.

Our constraint on the robot arm is that the magnitude of the acceleration of each joint must be less than a maximum a_m.

The contact acceleration constraint on this system can be derived by using these additional maps:

$$\mathsf{J} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.37}$$

$$\mathsf{J_c} = \mathsf{J} \tag{2.38}$$

where J is the Jacobian of the arm. Using these maps and the grasp map described earlier, we can obtain the contact acceleration constraint as:

$$\begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \dot{\theta}^2 + \begin{pmatrix} 0 \\ g \end{pmatrix} \in \frac{1}{m\rho^2} \begin{bmatrix} \rho^2 + y^2 & -xy \\ -xy & \rho^2 + x^2 \end{bmatrix} \mathcal{F} \tag{2.39}$$

Notice that Eqn.2.39 relates the joint accelerations $(\ddot{q}_1, \ddot{q}_2)$ and the constraint due to the fixed contact. A graphical representation of the combination of the two sets of constraints is shown in Fig.2.5. The limit on the magnitude of the allowable acceleration constrains the feasible accelerations to a square of dimension $2a_m$. The constraint due to Coulomb friction, represented by the contact acceleration cone further restricts the set of feasible accelerations to the shaded polygon shown in Fig.2.5.

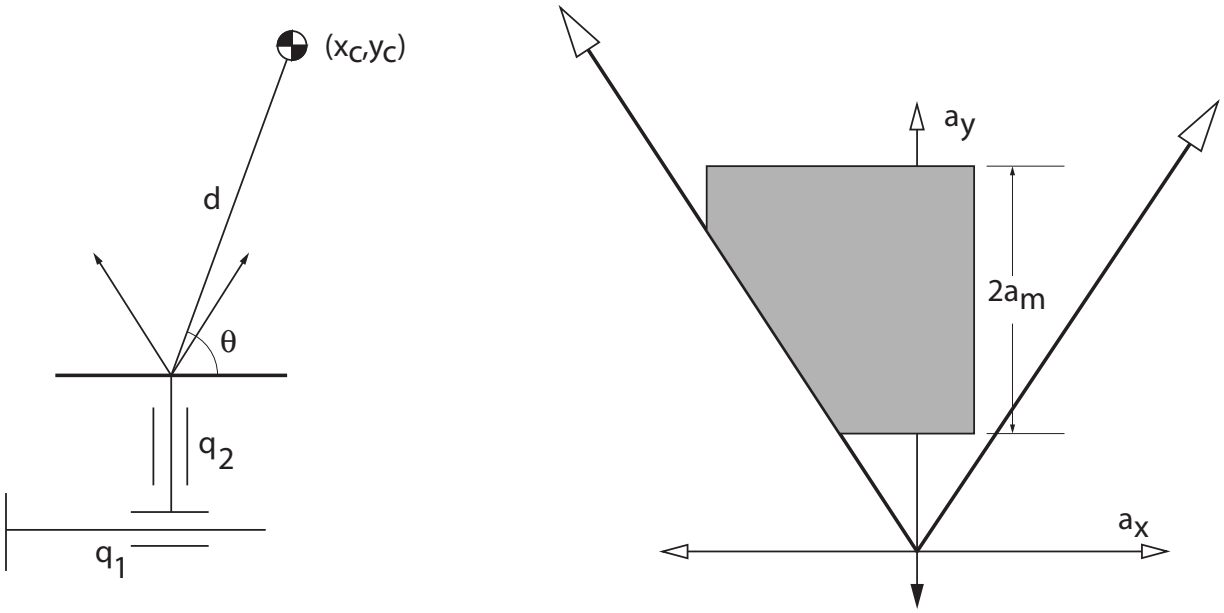


Figure 2.5: Combining actuator constraints and friction constraints for a single point contact

2.6 Summary

In this chapter, we described how we can combine constraints on the actuator and constraints due to Coulomb friction in a common space. Furthermore, we gave a geometric description of the mapping and related it to the configuration space friction cone.

What we have is an instantaneous solution to the inverse problem — given a state of the system, a desired contact mode, and actuator constraints, we can provide the set of feasible joint accelerations. Using this set to plan trajectories for the system is still a hard problem. We tackle that problem in the following chapters. We provide two techniques for planning dynamic manipulation tasks. The first is the technique of *time-scaling*, one that has been used to plan time-optimal trajectories for robot arms with actuator constraints. In §3, we describe how we can use this technique to solve for trajectories for dynamic manipulation with the contact acceleration constraint. The second is the technique of *task and shape decomposition*. This is a paradigm for motion planning that we propose in §4 and demonstrate with both a dynamic manipulation task in §6 and a nonholonomically constrained system in §5.

Chapter 3

The block standing problem

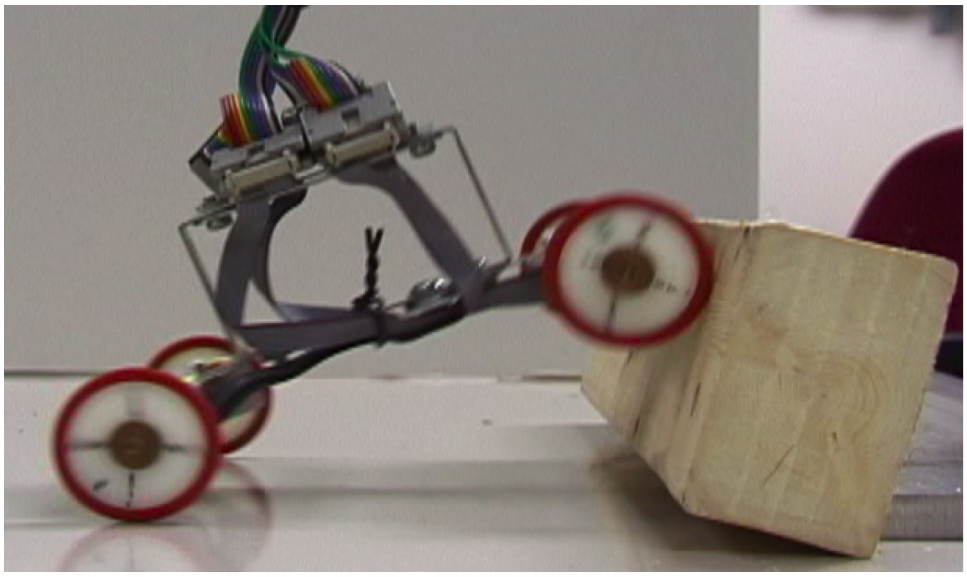
In this chapter, we explore the synthesis of controls for a robot dynamically manipulating an object in the presence of multiple frictional contacts. Contacts occur both between the object and the robot, and between the object and the environment. As detailed in §2, two sets of constraints govern the evolution of the system — contact velocity constraints that prevent separation and cause rolling, and contact force constraints that arise from Coulomb friction. We combined the constraints in the space of contact accelerations, obtaining bounds on the robot acceleration as a function of the system state.

We provide a solution to the motion planning problem by first computing a feasible path for the system and then generating the controls and the system trajectory by time-scaling the feasible path to satisfies the dynamic constraints.

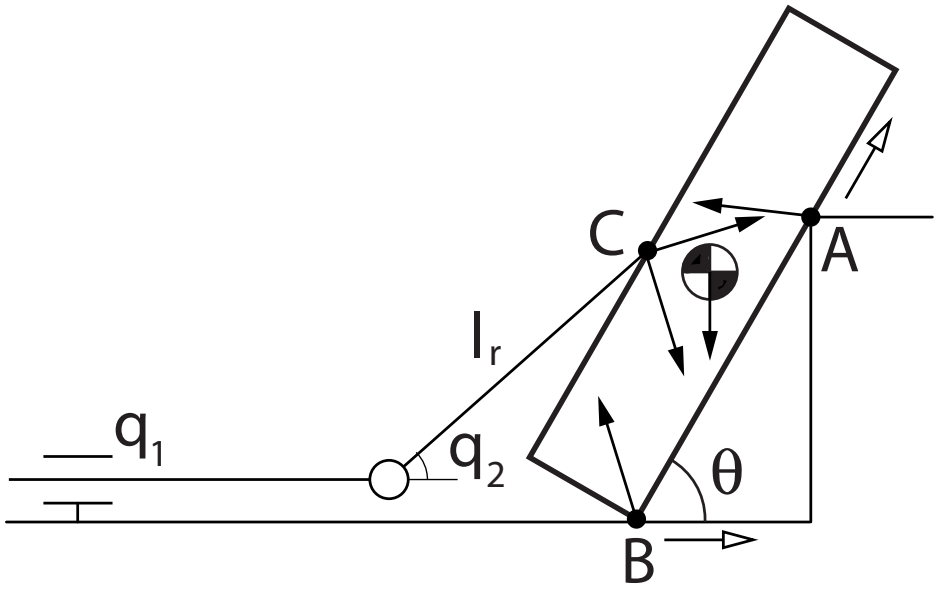
We consider an example that highlights two characteristics of the problems we are interested in solving — contact with the environment, and the use of dynamics to find a solution where a quasi-static analysis fails.

Contacts with the environment occur frequently in manipulation. They occur in peg-in-hole insertion when the peg contacts the hole, and in part fixturing where the goal is to immobilize the part against the remote fixtures. Such contacts occur when we manipulate objects too large and heavy to lift; when we push on them or topple them. Here, the environment provides a helping hand, a remote contact that bears most of the heavy load that we would not be able to bear by ourselves. Environmental contacts also occur in mobile manipulation, where a robot uses its wheels to move the manipuland. Here, the environment provides an opposing force, remote contacts that together with the wheel contact help maintain control of the manipuland and prevent it from scooting away from the robot. Hence these contacts are either inevitable, as in the case of assembly, or essential, as in the case of large object manipulation or mobile manipulation.

While contacts with the environment can be beneficial, controlling the motion of the object and the contact mode poses a challenging problem. The reason for this is that the



(i)



(ii)

Figure 3.1: The block standing problem — (i): A mobile manipulator uses its wheels to push a block against a step. The goal is to go from a starting θ to $\theta = \frac{\pi}{2}$ while maintaining sliding contacts with the step and a fixed contact between the block and the wheel. (ii): a PR arm models the kinematic skeleton of the mobile robot. The allowable forces at the contact points, as per Coulomb friction, are shown.

robot cannot directly control the forces and the motion at these contacts. It has to exert a force on its contact with the object, which then gets transmitted through the object to the remote contact.

Consider the problem shown in Fig.3.1(ii) where an arm with one prismatic and one revolute joint (henceforth referred as a PR arm) manipulates a block resting on a step. The arm contacts the block at C and the block contacts the environment at the apex(A) and base(B) of the step. As shown in the figure, the orientation of the block is denoted by θ. We control the motion of the robot via its joint acceleration. The goal is to synthesize controls to stand the block up against the step ($\theta = \frac{\pi}{2}$) while maintaining sliding contacts at A and B and a fixed contact at C.

The motivation for the problem arises from the mobile manipulation problem shown in Fig.3.1(i). Here, a mobile robot manipulates a block resting on a step, using its wheels to push on the block. During manipulation, the wheels that are in contact with the block are locked. This is to ensure that the robot does not climb up the block when the wheels on the ground push forward. This is the motivation for the fixed contact at C.

If the object starts from rest in the configuration shown in Fig.3.1, any force exerted by the robot at C causes one of three object motions. Depending on the direction of the force, the object either loses contact with B and starts tipping about A, remains immobile, or starts sliding backward. Quasi-statically there is no control that increases θ.

Fig.3.2 presents a graphical proof of the claim using the technique of *moment labeling* introduced by Mason [Mason, 1991]. This technique can be used to graphically compute the resultant of a set of forces. Forces acting on the block comprise of the force exerted by the robot at C, the contact forces at A and B and gravity. According to Newton's laws, the net force acting on the block must equal its inertia times its acceleration, also referred to as the *dynamic load*. Let us assume that the block is sliding to the right. Then, the contact forces at A and B lie on the left edge of the friction cone. Mason provided technique for computing the direction of the dynamic load on the block. The force exerted by the robot at A must then equal the difference of the dynamic load and the sum of the frictional forces and gravity. The moment labels of this difference are denoted by the two shaded regions marked "+" and "-" in Fig.3.2. For the desired motion to be produced, there must exist at least one line of action of the force at C that does not intersect either region. This is not possible since the point C lies within the region marked "+". Hence there is no force that can be exerted at C that slides the block forward.

However, we can use the dynamics of the object to accomplish the task. Intuitively, this involves sliding the block backward a distance that is sufficient to accelerate the block forward in such a way that the centripetal force generated by the motion prevents the block from tipping. Snapshots of the resulting dynamic trajectory are shown in Fig.3.7. We explore this example in detail in §3.3.

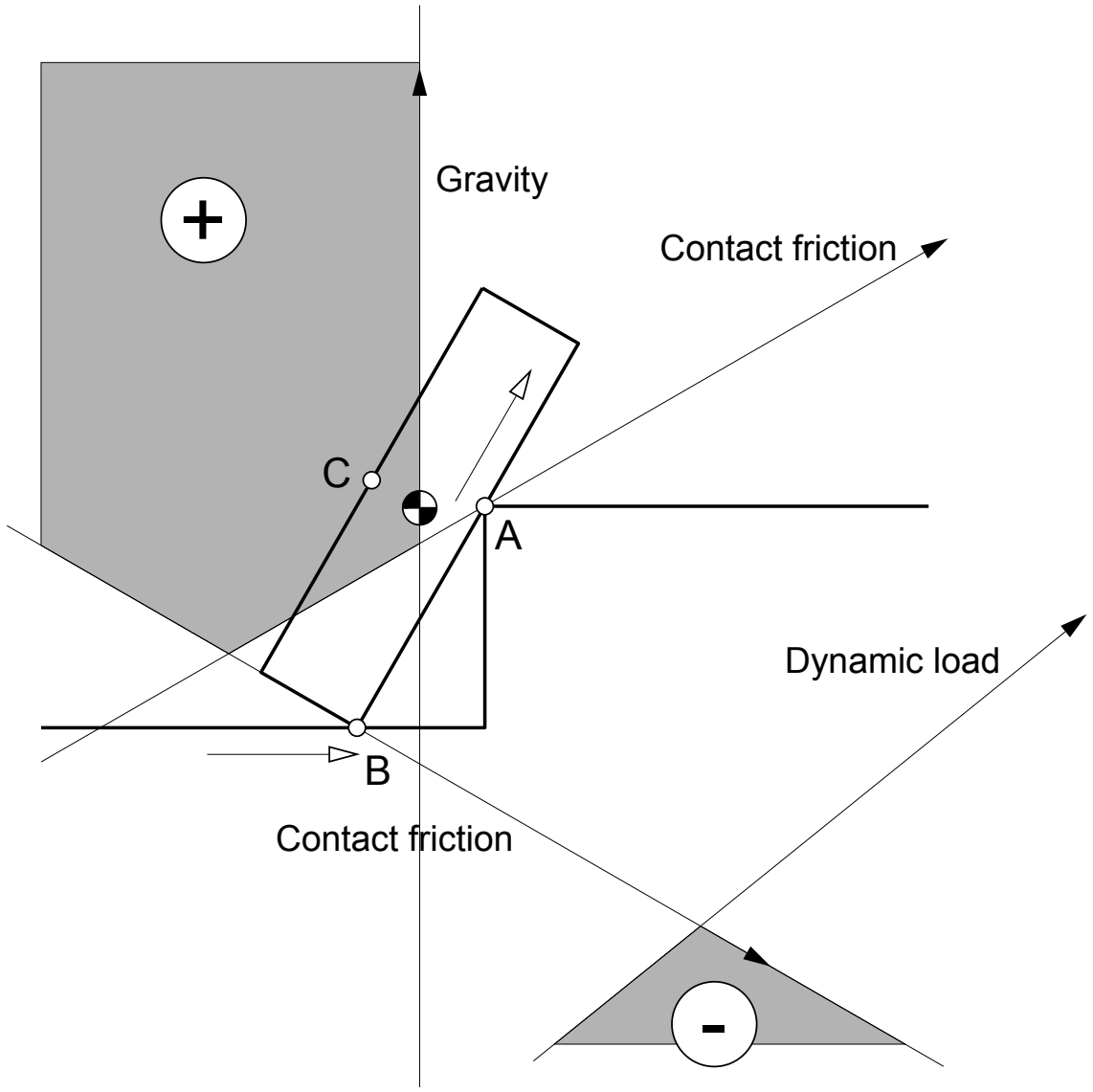


Figure 3.2: Moment labels for right sliding showing that it is not possible to apply a force at C that causes the block to slide up the step

We provide a clear definition of the block standing problem in §3.1. In §3.2, we use the technique of time-scaling to simplify the problem. We project the constraints on to a lower dimensional phase space and solve for a feasible trajectory in this space. We use the phase space trajectory to synthesize the controls and the dynamic trajectory of the system (shown in Fig.3.7). We describe two strategies for solving the problem in §3.3 and discuss the limitations of our technique in §3.4.

3.1 Problem statement

We denote the configuration of the block as $\boldsymbol{q_o} = (x, y, \theta)^T$, where (x, y) is the location of the center of mass of the block and θ is its orientation. We denote the configuration of the robot by its joint variables $\boldsymbol{q_r} = (q_1, q_2)^T$.

The Jacobian of the PR arm is given by:

$$\mathsf{J} = \begin{bmatrix} 1 & -l_r \sin(q_2) \\ 0 & l_r \cos(q_2) \end{bmatrix}$$

where l_r is th length of the second link.

Vectors from the center of mass to each contact point are:

$$\begin{aligned} \boldsymbol{r_A} &= w \begin{pmatrix} \cos(\theta - \frac{\pi}{2}) \\ \sin(\theta - \frac{\pi}{2}) \end{pmatrix} + (l - \frac{h}{\sin(\theta)}) \begin{pmatrix} -\cos(\theta) \\ -\sin(\theta) \end{pmatrix} \\ \boldsymbol{r_B} &= w \begin{pmatrix} \cos(\theta - \frac{\pi}{2}) \\ \sin(\theta - \frac{\pi}{2}) \end{pmatrix} + l \begin{pmatrix} -\cos(\theta) \\ -\sin(\theta) \end{pmatrix} \\ \boldsymbol{r_C} &= w \begin{pmatrix} \cos(\theta + \frac{\pi}{2}) \\ \sin(\theta + \frac{\pi}{2}) \end{pmatrix} + (d - l) \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \end{aligned}$$

where w is the half-width of the block, l is the half-length of the block, h is the height of C measured along the block, and d is the height of the step.

A fixed contact at C enforces the constraint that the velocity of the robot at C must equal the velocity of the block at C. This can be written as:

$$\mathsf{G}_\mathsf{r}^\mathsf{T} \dot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{J} \dot{\boldsymbol{q}}_{\boldsymbol{r}} \tag{3.1}$$

$$\mathsf{G}_\mathsf{r}^\mathsf{T} = \begin{bmatrix} 1 & 0 & -r_{Cy} \\ 0 & 1 & r_{Cx} \end{bmatrix}$$

A sliding contact at A and B enforces the constraint that the velocity of the block normal to the step at A and B must be zero, to prevent separation or penetration at the

contacts. This can be written as:

$$\mathsf{G}_\mathsf{e}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} = 0 \tag{3.2}$$

$$\mathsf{G}_\mathsf{e}^\mathsf{T} = \begin{bmatrix} \boldsymbol{n_A} & \boldsymbol{r_A} \times \boldsymbol{n_A} \\ \boldsymbol{n_B} & \boldsymbol{r_B} \times \boldsymbol{n_B} \end{bmatrix}$$

The contact normals at A and B, $\boldsymbol{n_A}$ and $\boldsymbol{n_B}$, respectively, are given by:

$$\begin{aligned} \boldsymbol{n_A} &= \begin{pmatrix} \cos(\theta + \frac{\pi}{2}) \\ \sin(\theta + \frac{\pi}{2}) \end{pmatrix} \\ \boldsymbol{n_B} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned}$$

Combining both the constraints, we get:

$$\mathsf{G}_\mathsf{c}^\mathsf{T}\dot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{J}_\mathsf{c}\dot{\boldsymbol{q}}_{\boldsymbol{r}} \tag{3.3}$$

$$\begin{aligned} \mathsf{J}_\mathsf{c} &= \begin{bmatrix} 0_2 \\ \mathsf{J} \end{bmatrix} \\ \mathsf{G}_\mathsf{c}^\mathsf{T} &= \begin{bmatrix} \mathsf{G}_\mathsf{e}^\mathsf{T} \\ \mathsf{G}_\mathsf{r}^\mathsf{T} \end{bmatrix} \end{aligned}$$

If the block is sliding to the right at contacts A and B, the contact force is constrained to lie on the left edge of the contact friction cones at A and B. The contact at C is fixed, so the contact force can range anywhere inside the contact friction cone at C. The contact force constraints can be written as:

$$\boldsymbol{f_c} \in \mathcal{F} = \mathcal{F}_A \times \mathcal{F}_B \times \mathcal{F}_C \tag{3.4}$$

$$\begin{aligned} \mathcal{F}_A &= \begin{bmatrix} -\sin(\theta + \alpha_b) \\ \cos(\theta + \alpha_b) \end{bmatrix} \boldsymbol{\lambda_A} \\ \mathcal{F}_B &= \begin{bmatrix} -\sin(\alpha_b) \\ \cos(\alpha_b) \end{bmatrix} \boldsymbol{\lambda_B} \\ \mathcal{F}_C &= \begin{bmatrix} \sin(\theta - \alpha_r) & \sin(\theta + \alpha_r) \\ -\cos(\theta - \alpha_r) & -\cos(\theta + \alpha_r) \end{bmatrix} \boldsymbol{\lambda_C} \end{aligned}$$

for $\boldsymbol{\lambda_A}, \boldsymbol{\lambda_B}, \boldsymbol{\lambda_C} \succeq \mathbf{0}$. The angle of repose between the block and the step is given by α_b while the angle of repose between the robot and the block is given by α_r. Note that the angle of repose α is related to the coefficient of friction μ by $\alpha = \arctan(\mu)$.

$\mathcal{F}_A$ and $\mathcal{F}_B$ are the left edges of the friction cones at A and B respectively and $\mathcal{F}_C$ is the friction cone at C.

The problem can be stated as:

> Given a start orientation of the block θ_s and a goal orientation of the block $\theta_s = \frac{\pi}{2}$, find the robot joint acceleration $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}(t)$ that will move the system from the start to the goal without violating the contact velocity constraint (Eqn.3.3) or the contact force constraint (Eqn.3.4).

3.2 Control synthesis

In this section, we describe a general technique for solving the control synthesis problem. We define the configuration of the system by $\boldsymbol{q} = (\boldsymbol{q_o}, \boldsymbol{q_r})^T$ and the velocity of the system by $\dot{\boldsymbol{q}}$=$(\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}})^T$. We define the state of the system by $(\boldsymbol{q}, \dot{\boldsymbol{q}})^T$. In general, for the manipulation problem, we desire a trajectory from a start state $(\boldsymbol{q_s}, \dot{\boldsymbol{q}}_{\boldsymbol{s}})^T$ to a goal state $(\boldsymbol{q_g}, \dot{\boldsymbol{q}}_{\boldsymbol{g}})^T$ which satisfies the contact acceleration constraint at each point.

The contact acceleration constraint is a constraint on the dynamics of the system, hence trajectory generation must occur in the state space of the system. Computing analytical solutions in high dimensional state spaces is usually hard. We use the technique of time-scaling to reduce the dimensionality of the space in which we need to plan.

Time-scaling

Our motivation is to search for controls in a space that has a lower dimension than the system state space (whose dimension we denote by n_s). We simplify the problem into two subproblems. The first problem is finding a feasible path $\boldsymbol{q}(s)$, where s is a parametrization of the path, from start to goal. This involves searching the configuration space of the system which has a dimension of $\frac{n_s}{2}$. Once a feasible path is found, the second problem is finding a feasible trajectory along the path. This involves searching the two dimensional phase space $[s, \dot{s}]$ for trajectories that do not violate the contact acceleration constraints. The choice of $\dot{s}$ and $\ddot{s}$ for a path is called the time-scaling of the path. The time-scaling idea is illustrated in Fig.3.3.

Definition 1. *A **feasible path** is one that does not violate the contact kinematic constraints for any time-scaling.*

Theorem 2. *Every feasible path $\boldsymbol{q}(s) = [\boldsymbol{q_o}(s), \boldsymbol{q_r}(s)]^T$ satisfies the constraint:*

$$\mathsf{G}_\mathsf{c}^\mathsf{T} \boldsymbol{q_o'}(s) = \mathsf{J}_\mathsf{c} \boldsymbol{q_r'}(s) \tag{3.5}$$

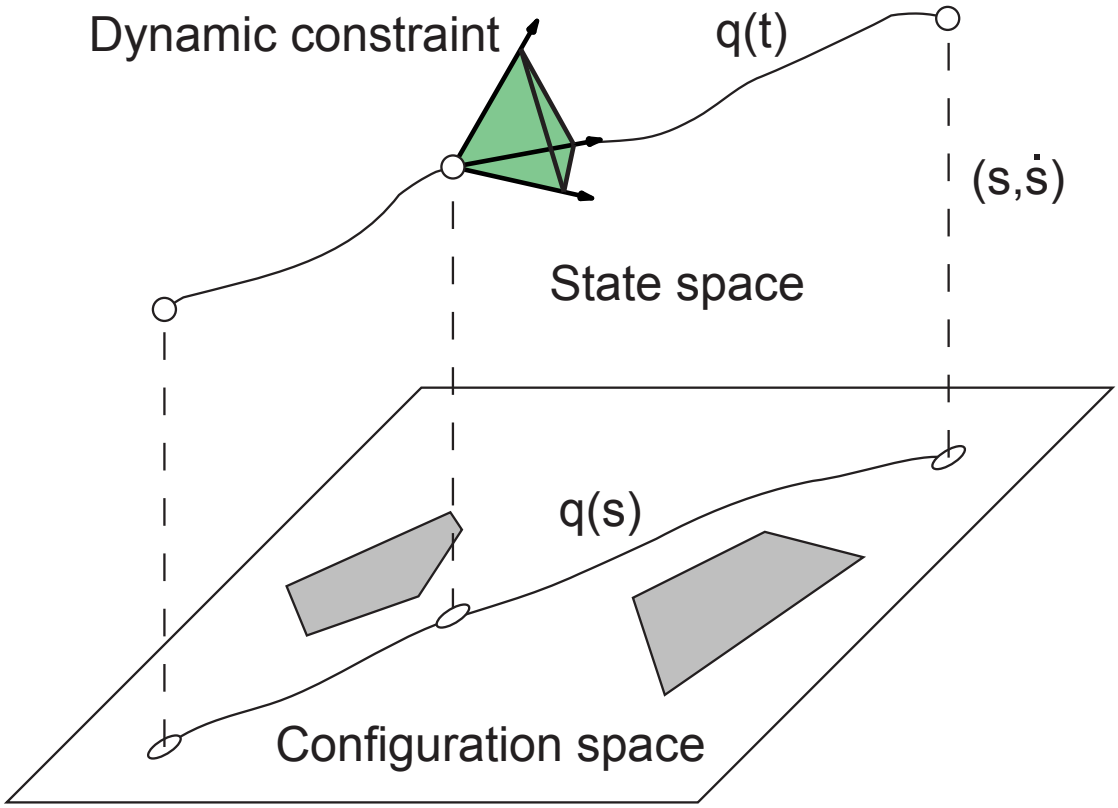


Figure 3.3: A sketch of the time-scaling concept. Trajectory generation is divided into path planning in configuration space followed by time-scaling the path to satisfy the dynamic constraints

Proof. The contact kinematic constraints are:

$$\mathsf{G}_c^T \dot{\boldsymbol{q}}_{\boldsymbol{o}} - \mathsf{J}\dot{\boldsymbol{q}}_{\boldsymbol{r}} = 0$$

Motion of the object and robot along a path is given by:

$$\dot{\boldsymbol{q}}_{\boldsymbol{o}} = \boldsymbol{q}'_{\boldsymbol{o}}(s)\dot{s} \tag{3.6}$$

$$\dot{\boldsymbol{q}}_{\boldsymbol{r}} = \boldsymbol{q}'_{\boldsymbol{r}}(s)\dot{s} \tag{3.7}$$

Since a feasible path satisfies the contact kinematic constraints,

$$\begin{aligned} \mathsf{G}_\mathsf{c}^\mathsf{T}\boldsymbol{q_o'}(s)\dot{s} - \mathsf{J}_\mathsf{c}\boldsymbol{q_r'}(s)\dot{s} &= 0 \\ \left(\mathsf{G}_\mathsf{c}^\mathsf{T}\boldsymbol{q_o'}(s) - \mathsf{J}_\mathsf{c}\boldsymbol{q_r'}(s)\right)\dot{s} &= 0 \end{aligned}$$

For the above equation to be true for all time-scalings $\dot{s}$,

$$\mathsf{G}_\mathsf{c}^\mathsf{T}\boldsymbol{q_o'}(s) - \mathsf{J}_\mathsf{c}\boldsymbol{q_r'}(s) = 0$$

□

Once a feasible path is generated, we can can apply the contact acceleration constraint to obtain constraints on the phase space trajectories. Recall from §2 that the contact acceleration constraint is given by:

$$\mathsf{J}_\mathsf{c}\ddot{\boldsymbol{q}}_\boldsymbol{r} + \mathsf{V}(\dot{\boldsymbol{q}}_\boldsymbol{o}, \dot{\boldsymbol{q}}_\boldsymbol{r}, \boldsymbol{n_o}) \in \mathcal{A} \tag{3.8}$$

Using Eqn.3.6 and Eqn.3.7 in Eqn.3.8 we obtain phase space constraints of the form:

$$\boldsymbol{v_1}(s)\ddot{s} + \boldsymbol{v_2}(s)\dot{s}^2 + \boldsymbol{v_3}(s) \in \mathcal{A}(s) \tag{3.9}$$

Here we have collected the terms that depend on $\ddot{s}$,$\dot{s}$ and s as the vectors $\boldsymbol{v_1}$, $\boldsymbol{v_2}$ and $\boldsymbol{v_3}$.

Trajectory generation

In planar contact problems we leverage the linearity of the contact friction cone to obtain a computationally efficient representation of Eqn.3.9. Since the mapping from contact force space to contact acceleration space is linear, a linear $\mathcal{F}$ maps to a linear $\mathcal{A}$.

There are two representations of linear cones — the span representation and the face normal representation. In the span representation, the cone is described as the positive linear span of a set of vectors. This can be written as:

$$\mathcal{A} = \{\boldsymbol{g} : \boldsymbol{g} = \mathsf{G}_\mathsf{A}\boldsymbol{z}, \boldsymbol{z} \succeq \boldsymbol{0}\}$$

where the matrix G_A is a collection of the vectors that make up the positive linear span.

In the face normal representation, the cone is described as the intersection of a set of half-spaces. This can be written as:

$$\mathcal{A} = \{\boldsymbol{g} : \mathsf{F}_\mathsf{A}\boldsymbol{g} \preceq \boldsymbol{0}\}$$

where the matrix F_A is a collection of the normals to the half-spaces.

Using the face normal representation in Eqn.3.9, we get:

$$\mathsf{F}_\mathsf{A}\left(\boldsymbol{v_1}(s)\ddot{s} + \boldsymbol{v_2}(s)\dot{s}^2 + \boldsymbol{v_3}(s)\right) \preceq \mathbf{0} \tag{3.10}$$

At each point $[s, \dot{s}]$ in phase space, Eqn.3.10 gives us a set of linear constraints on $\ddot{s}$. We can solve the constraints to obtain bounds on the feasible acceleration $\ddot{s} \in [\ddot{s}_{\min}, \ddot{s}_{\max}]$.

The acceleration bounds can be visualized as cones in the tangent space. At each point in phase space, the tangent to a feasible phase space path must lie within the cone. When the constraints are inconsistent, the cone disappears and no feasible path can pass through such a point. This corresponds to a physically unrealizable motion of the system for the given actuator limits and contact mode.

There is substantial literature on computing analytical solutions for trajectory generation in two dimensions [Bobrow et al., 1985, Shin and McKay, 1985, Slotine and Yang, 1989, Wen and Desrochers, 1986, Butler and Tomizuka, 1992]. For example, as a bang-bang solution, we can pick a phase space acceleration $\ddot{s}_1$ and compute the forward trajectory from the start. We can pick another phase space trajectory $\ddot{s}_2$ and compute the backward trajectory from the goal. At the point where the two trajectories intersect, we can switch from $\ddot{s}_1$ to $\ddot{s}_2$.

3.3 Trajectories for the block standing problem

In this section, we describe the application of the time-scaling concept to the block standing problem. We will see that there are two strategies for solving the problem and the location of the contact C causes a switch from one strategy to the other. We will discuss the effects of changing the coefficients of friction on the solutions to the system. We will show that it is harder to solve the problem as the coefficient of friction between the block and the ground is increased.

Time-scaling

With the imposed contact velocity constraints, the system composed of the block and the robot has one degree of freedom. The unconstrained block has three degrees of freedom and we are applying two constraints on its motion, one each to maintain contact at A and B, thereby restricting its motion to one degree of freedom. This is illustrated in Fig.3.4. The figure shows the configuration space obstacle (the step) for the block. Each face of the configuration space obstacle corresponds to the contact between an edge of the block and an edge of the step. Maintaining contact with a face imposes one constraint on the motion of the block. The feasible path for the block standing problem corresponds to the intersection of two such faces, the base and the apex of the step, and is shown by the path $AB - AB'$.

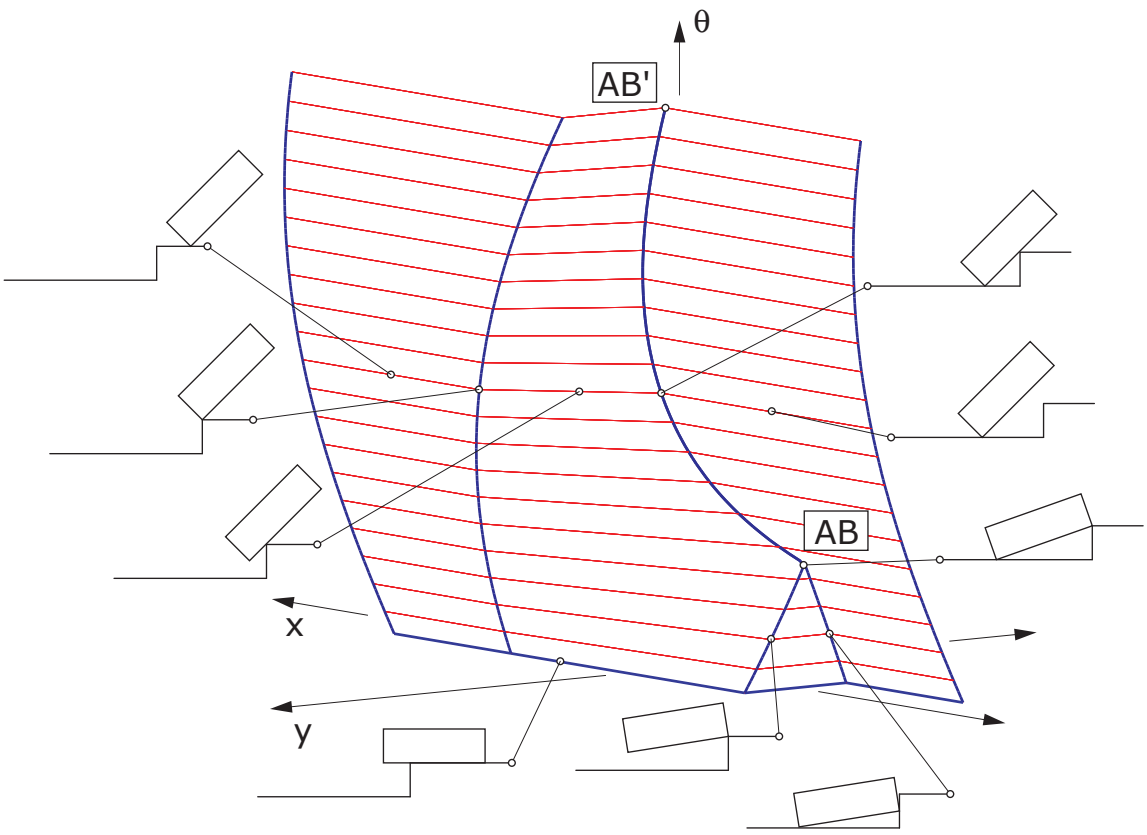


Figure 3.4: A portion of the configuration space of the block from $\theta \in [0, \frac{\pi}{2}]$.

The obstacle was constructed using the algorithm proposed in Brost's thesis [Brost, 1991]. The unconstrained PR arm has two degrees of freedom. The contact at C imposes one constraint on the motion of the arm, giving it one constrained degree of freedom.

We compute a feasible path $\boldsymbol{q}$ parametrized by the orientation of the block θ:

$$\boldsymbol{q}(\theta) \quad = \quad \begin{pmatrix} \boldsymbol{q_o}(\theta) \\ \boldsymbol{q_r}(\theta) \end{pmatrix}$$

Using Eqn.3.10, we compute the constraints on the trajectories in the phase space $[\theta, \dot{\theta}]$ of the form:

$$\mathsf{F_A}(\theta) \left(\boldsymbol{v_1}(\theta)\ddot{\theta} + \boldsymbol{v_2}(\theta)\dot{\theta}^2 + \boldsymbol{v_3}(\theta) \right) \preceq \mathbf{0}$$

Once a parametrized feasible path is chosen, we can transform the actuator constraints into constraints on the acceleration of the block in phase space. The acceleration of the

robot is related to the acceleration of the block as follows:

$$\ddot{\boldsymbol{q}}_{\boldsymbol{r}} = \dot{\boldsymbol{q}}_{\boldsymbol{r}}(\theta)\dot{\theta}^2 + \boldsymbol{q}_{\boldsymbol{r}}(\theta)\ddot{\theta} \tag{3.11}$$

At each $[\theta, \dot{\theta}]$, this gives us a linear map between $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}$ and $\ddot{\theta}$. We use this map to convert limits on the actuator joint acceleration into limits on $\ddot{\theta}$.

Results

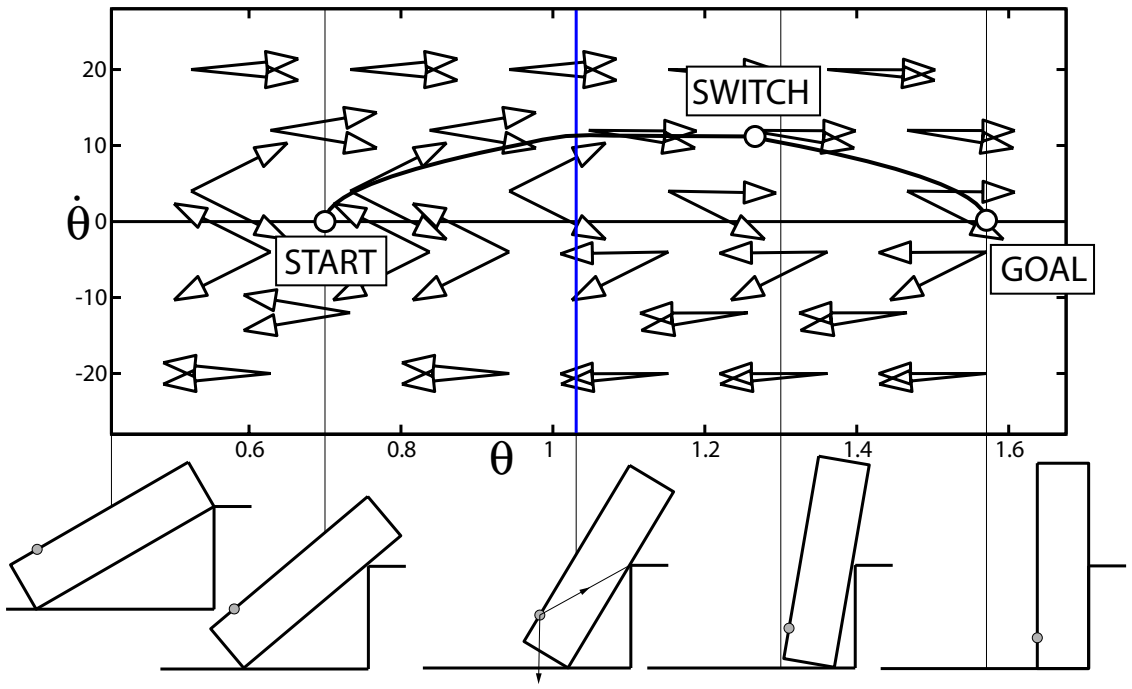


Figure 3.5: Phase plot and time-scaling trajectory for $h = 0.3l$

The phase plots and phase space trajectories for two instantiations of the problem are shown in Fig.3.5 and Fig.3.6. The arrows represent tangents. Orientations of the block that correspond to values in the horizontal axis are shown.

In Fig.3.5 contact C is at $h = 0.3l$ and the starting orientation of the block $\theta_s = 0.7$. In Fig.3.6 contact C is at $h = l$ and the starting orientation of the block is $\theta_s = \frac{\pi}{3} = 1.05$.

In Fig.3.5, the arm is able to push the block forward and make it stand. The arm accelerates the block until the switching point is reached after which the arm decelerates the block to rest. A different strategy is needed in Fig.3.6. Here the arm moves the block backward for a duration of time and then pushes the block forward and stands it up. The different strategies are a result of the phase space acceleration constraints at the start of

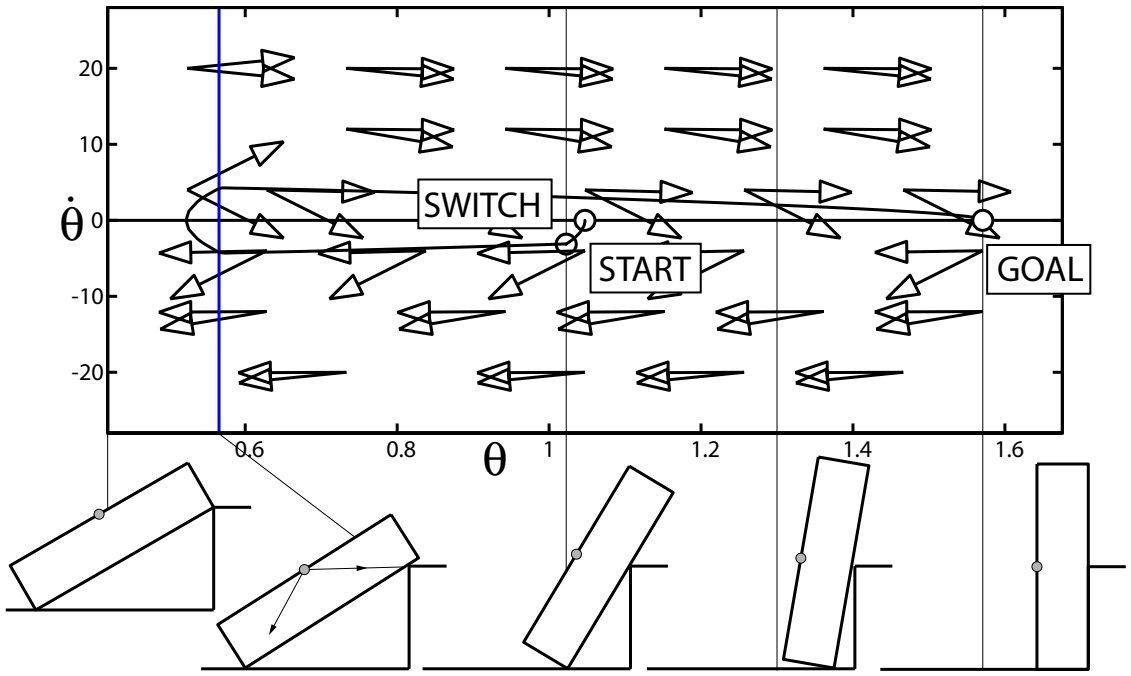


Figure 3.6: Phase plot and time-scaling trajectory for $h = 1.0l$

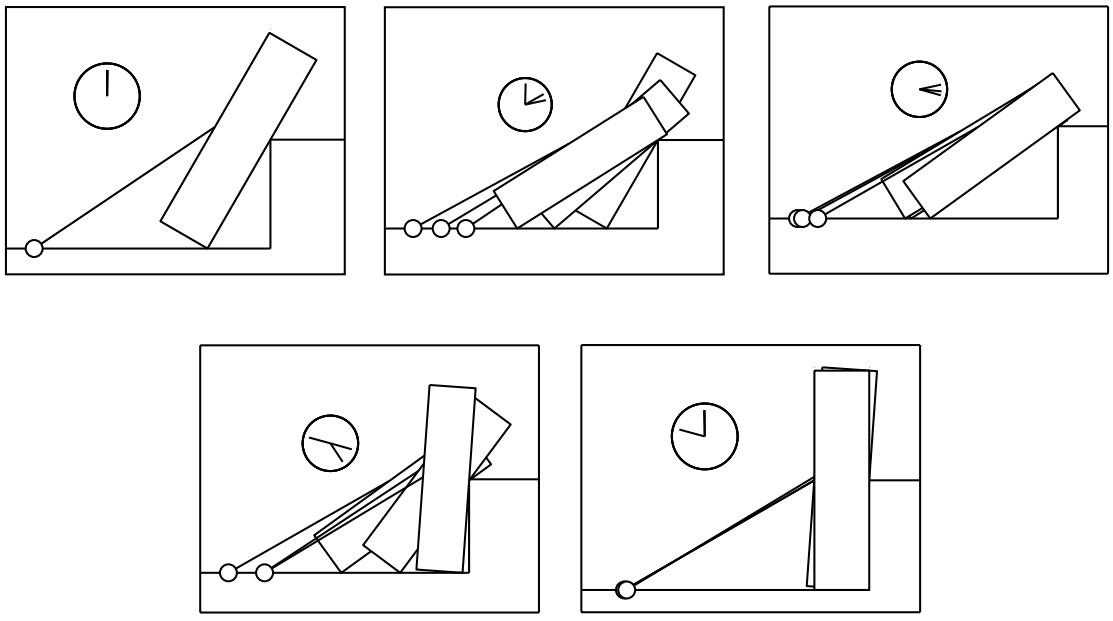


Figure 3.7: A solution trajectory for the block standing problem. The robot pulls the block back and shoots it forward to stand it up.

motion. For Fig.3.5, the phase space acceleration $\ddot{\theta}$ at the start of motion is constrained to lie in the range $[-20, 20]$. As this range includes a positive acceleration, the arm can push the block forward from rest. For Fig.3.6, the phase space acceleration $\ddot{\theta}$ at the start of motion is constrained to lie in the range $[-20, -10.83]$. Hence the only motion at the start that does not violate any of the contact constraints is one that moves the block backward. The block moves backward until the constraints allow a positive acceleration. The arm then decelerates the backward motion to rest, accelerates the block forward and stands it up. Snapshots of the trajectory are shown in Fig.3.7.

There is a critical start angle of the block θ_{crit} at which the switch from the strategy in Fig.3.5 to Fig.3.6 occurs. The critical event that causes the switch is when the left edge of the contact friction cone at C passes through the contact A. Start angles up to θ_{crit} will follow the strategy in Fig.3.5 and start angles in $[\theta_{\text{crit}}, \frac{\pi}{2}]$ will follow the strategy in Fig.3.6.

The solid vertical lines in Fig.3.5 and Fig.3.6 show the critical angles for the two problems. As we lower the contact C along the block, θ_{crit} decreases. This matches our intuition, as the higher C is along the block, the greater is the risk of breaking contact at B and tipping the block if the arm pushes forward.

Parameter variation

In this section, we study the variation of the coefficients of friction between the block and the step, and the block and the robot on the allowable time-scalings of the feasible path for the block standing problem.

We will first show the effect of varying the coefficient of friction between the block and the step on the feasible phase volume. Note that the coefficient of friction is the same at both contacts A and B. The figures Fig.3.8, Fig.3.9, Fig.3.10 and Fig.3.11 show the phase volume for $\alpha_b = 0.48, 0.70, 0.79$ and 0.88, respectively. The angle of repose between the robot and the block is held constant at $\frac{\pi}{3}$. We see that as the α_b increases, the allowable phase volume reduces. At Fig.3.9, the maximal and minimal surfaces intersect, forming an infeasible *island* (the term island for such a structure was coined by [Shin and McKay, 1985] who observed the same phenomenon for robot arm trajectories). As α_b increases to 0.88, the island has grown enough to completely cut off one section of the state space from another. At this coefficient of friction, no feasible trajectories are possible for a starting θ of 0.8 or lesser.

The intuition behind this observation is that it gets harder to slide the block on the step as the step gets stickier. With a greater coefficient of friction, there is a greater tendency for the block to jam against the step rather than slide on it.

The figures Fig.3.12 and Fig.3.13 show the effect of varying the coefficient of friction between the robot and the block on the feasible phase volume. For Fig.3.12, $\alpha_r = 0.52$ and

for Fig.3.13, $\alpha_r = 1.01$. The angle of repose between the block and the step is held constant at $\frac{\pi}{6}$. We observe the appearance of the island in Fig.3.12. We also notice that the phase volume increases as the coefficient of friction between the robot and the block increases.

The intuition behind this observation is that as the angle of repose between the robot and the block increases, the robot has a greater angular range of forces to apply on the block and can thus produce a greater range of accelerations of the block.

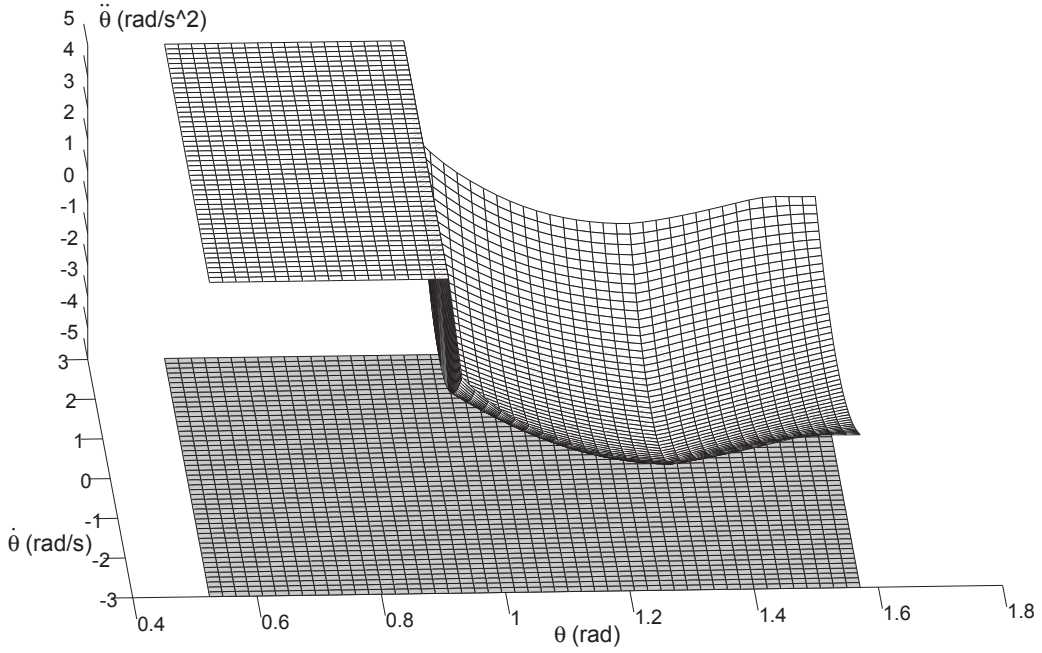


Figure 3.8: Phase volume for $\alpha_b = 0.48$ and $\alpha_r = \frac{\pi}{3}$

3.4 Summary

Issues with time-scaling

By separating path generation and trajectory generation, we have simplified a planning problem in the state space of the system into a planning problem in the 2 dimensional phase space of time-scalings. However, there can exist problems where finding paths that can be successfully time-scaled is very hard. In such problems, we might be stuck in the cycle of selecting a path, testing for a feasible time-scaling, and returning a failure, for a significant amount of time.

For the block standing problem with the desired contact mode, it turned out that the system had only one path that it could follow. Hence the planning problem of selecting a

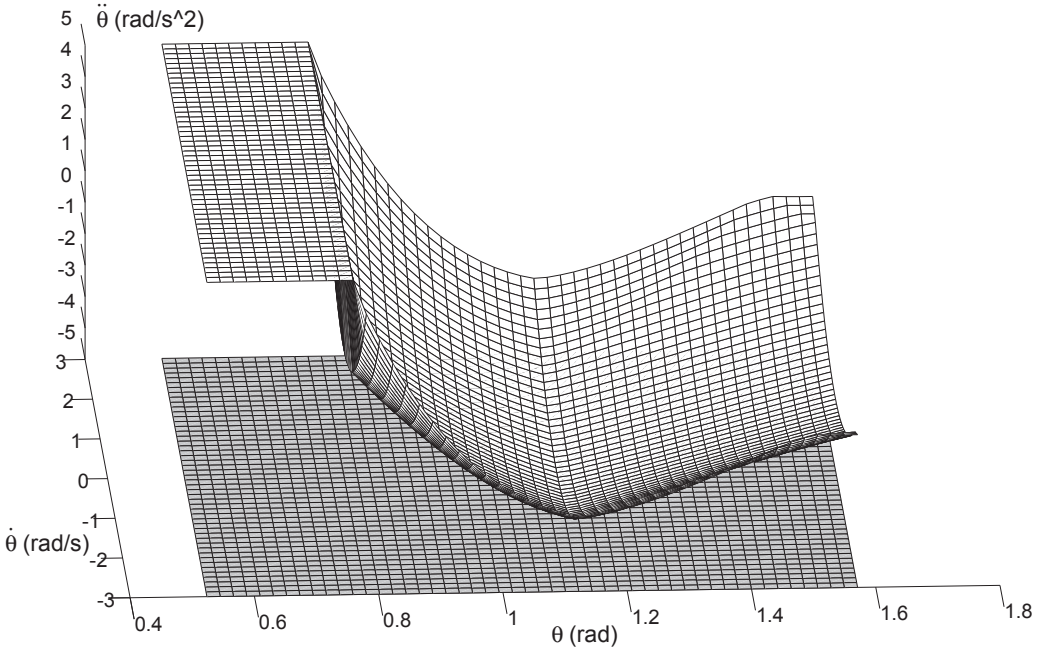


Figure 3.9: Phase volume for $\alpha_b = 0.70$ and $\alpha_r = \frac{\pi}{3}$

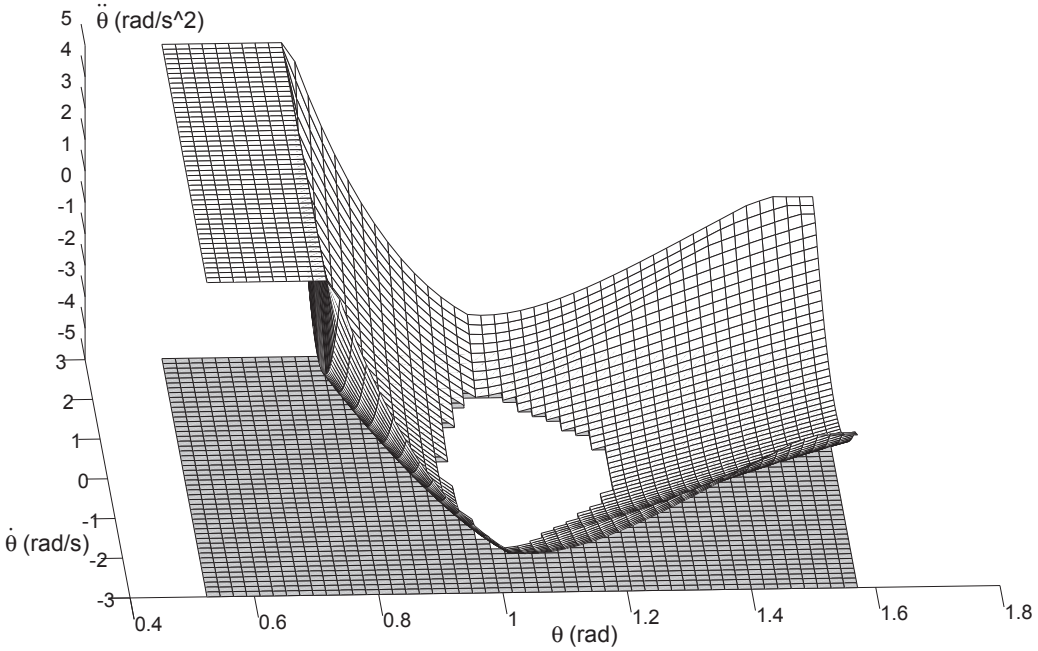


Figure 3.10: Phase volume for $\alpha_b = 0.79$ and $\alpha_r = \frac{\pi}{3}$

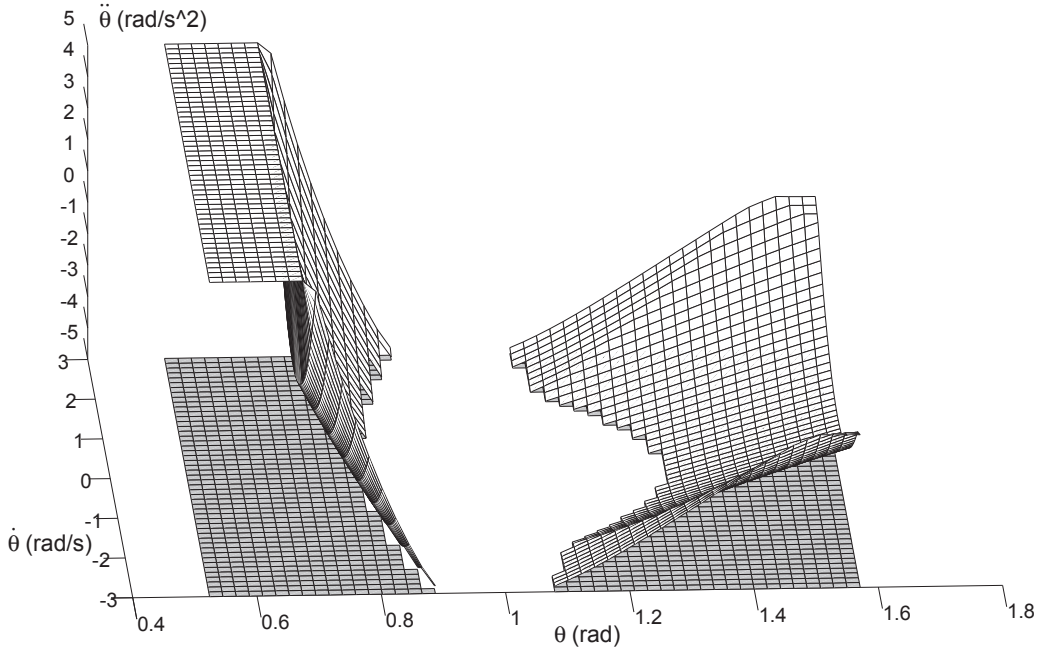


Figure 3.11: Phase volume for $\alpha_b = 0.88$ and $\alpha_r = \frac{\pi}{3}$

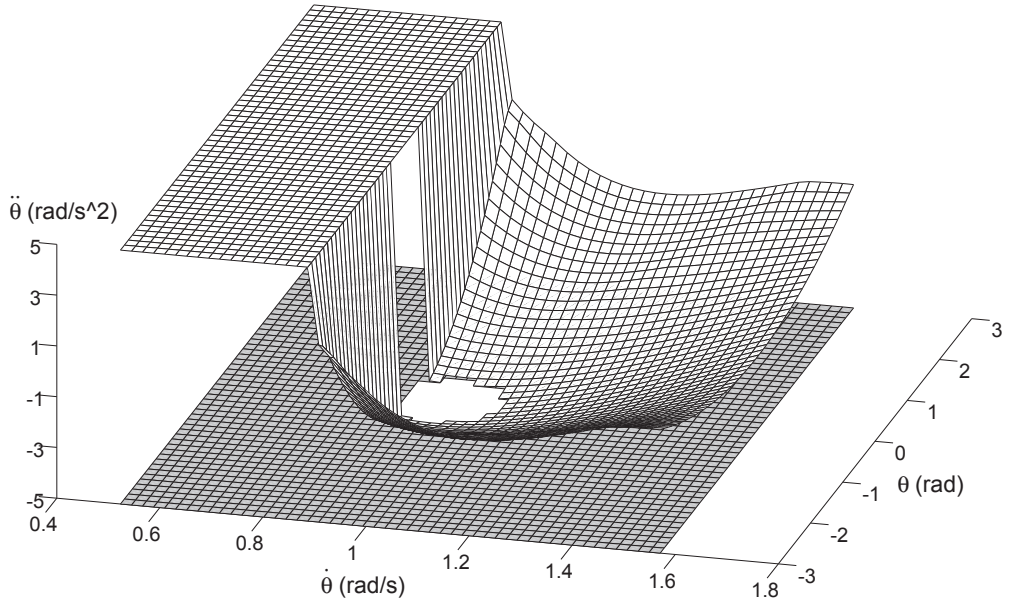


Figure 3.12: Phase volume for $\alpha_r = 0.52$ and $\alpha_b = \frac{\pi}{6}$

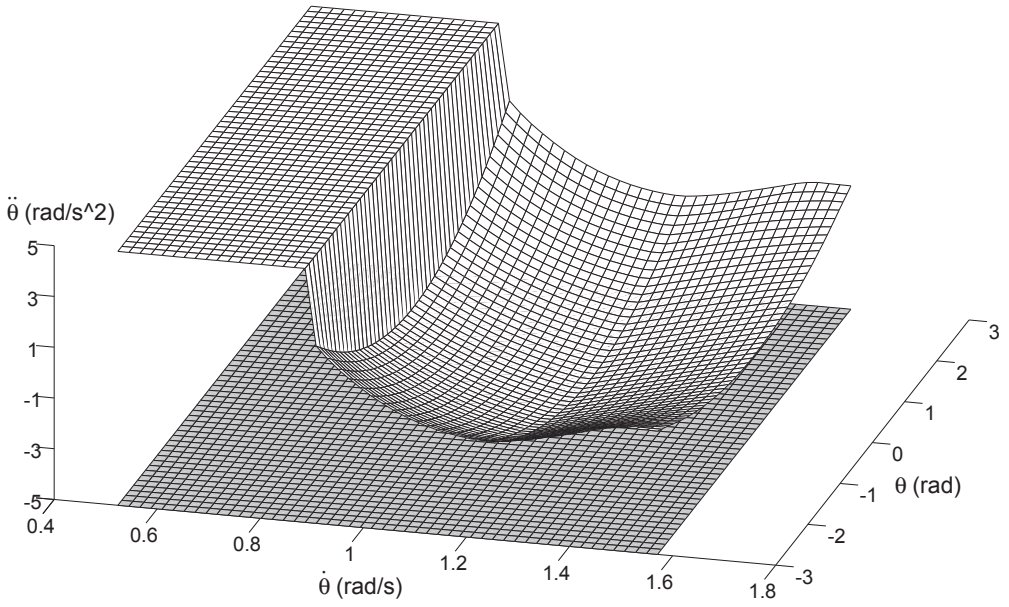


Figure 3.13: Phase volume for $\alpha_r = 1.01$ and $\alpha_b = \frac{\pi}{6}$

path was trivial. It is important to note that this is rarely the case. For example, consider the same block standing problem with a different desired contact behavior for the contact C between the block and the robot. Let the desired mode there be sliding, *i.e.*, we desire sliding at all the contacts A, B and C. With this contact mode, the system has two degrees of freedom — the angle θ that the block makes with the horizontal and the location h of the contact point A along the block. A path parametrized by θ must now also specify the change of h as a function of θ, a problem which is potentially as hard as solving the trajectory generation problem.

There are two potential solutions to this problem.

The first solution is a test which can tell us if a path can be successfully time-scaled without having to compute the phase space constraints. The contact acceleration constraint possesses local information on the tangent and the curvature of allowable paths. A future direction of research is the characterization of paths for manipulation problems that can be successfully time-scaled using this local information.

The second solution is to use our knowledge of the system dynamics to reduce the problem into a form where a large number of paths can be successfully time-scaled. In §6, we will describe a problem where only a small sliver of paths in the full configuration space can be successfully time-scaled and discuss a technique for projecting the dynamics down to a two-dimensional reduced space where the problem can be solved.

Planar vs. 3D

The contact friction cone in 3D is nonlinear. Our result for combining the constraints is general; it is valid both for planar and 3D contact problems. However, our implementation uses the linearity of the planar contact acceleration cone to efficiently compute acceleration bounds. For 3D problems, Eqn.3.9 gives us a set of nonlinear constraints on the allowable $\ddot{s}$ at each $[s, \dot{s}]$. Using the time-scaling technique, given a path, one can solve for the bounds by performing a 1D search in $\ddot{s}$.

Multiple solutions

A drawback of the Coulomb friction model is the possibility of the existence of multiple motions for a given control. Hence, to achieve a particular contact mode with certainty, one must ensure that the control is inconsistent with any other contact mode. Choosing phase space trajectories that satisfy this requirement entails searching over all feasible contact modes, which can be expensive. Given a feasible path, finding phase space trajectories where a particular contact mode is guaranteed *without* having to explicitly search over all contact modes is an interesting open problem.

Chapter 4

Task and shape decomposition

In this chapter, we discuss the task and shape decomposition paradigm for solving constrained dynamical systems. The motivation for the paradigm is the importance of satisfying constraints — the breaking of constraints either results in the system failing or in the system transitioning to a dynamic regime that is undesirable or unknown.

This chapter is arranged as follows. In §4.1, we introduce the paradigm of task and shape decomposition. In §4.2, we apply the paradigm for the specific case of manipulation systems and derive a condition for the applicability of the decomposition. In §4.3, we discuss the limitations of our technique and briefly describe the two examples that we will explore in detail in §5 and §6.

4.1 The paradigm

The systems that we are interested in are of the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state variable, $\boldsymbol{u} \in \mathbb{R}^m$ is the control variable, and $\boldsymbol{f}$ describes the evolution of the system.

The system can be subject to constraints both on the state and on the controls of the form:

$$\boldsymbol{g}(\boldsymbol{x}, t) \preceq \boldsymbol{0} \tag{4.2}$$

$$\boldsymbol{h}(\boldsymbol{u}, \boldsymbol{x}, t) \preceq \boldsymbol{0} \tag{4.3}$$

We wish to generate a set of controls $\boldsymbol{u}(t)$ that move the system Eqn.4.1 from a start state to a goal state without violating the constraints Eqn.4.2 and Eqn.4.3.

Our paradigm is has two motivations.

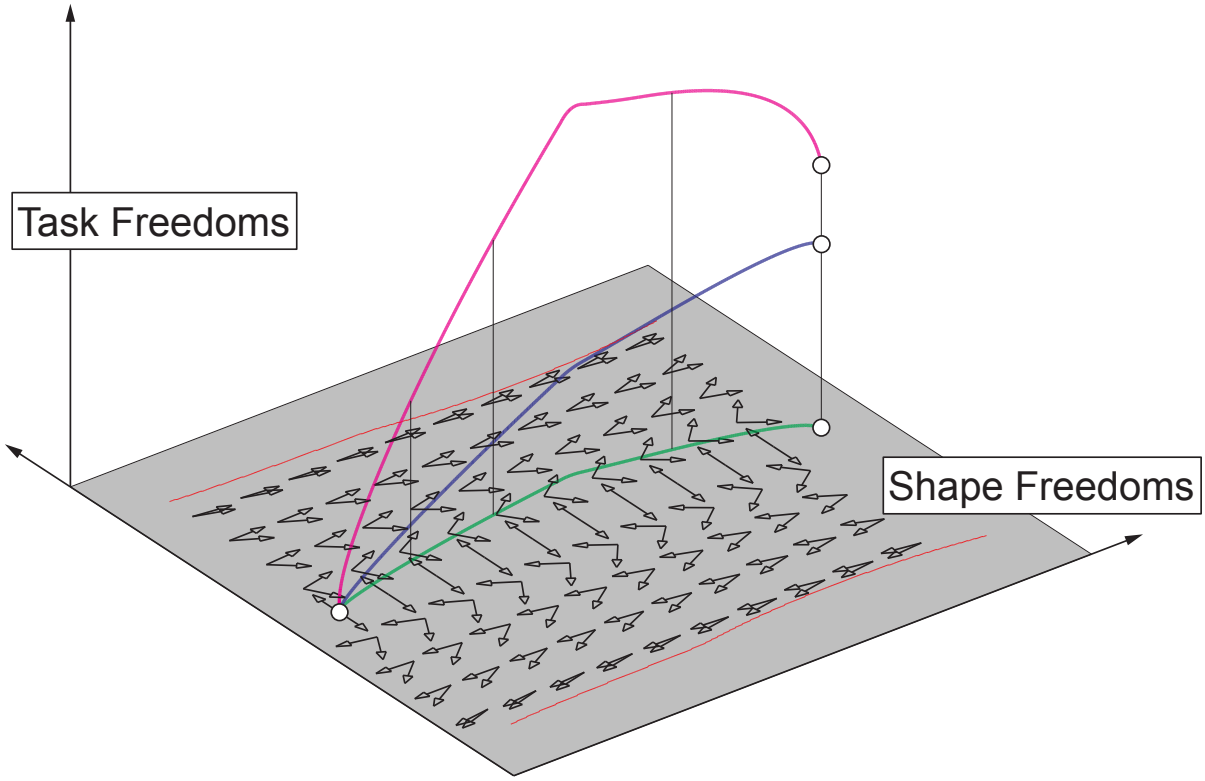


Figure 4.1: The task and shape decomposition paradigm

The first motivation is that computing trajectories for constrained dynamical systems in high dimensional state spaces is hard. There exist analytical time-optimal solutions for low dimensional spaces [Bobrow et al., 1985, Shin and McKay, 1985]. There also exist numerical techniques like dynamic programming [Bellman, 1957] which can be used, but they suffer from the curse of dimensionality as well. Hence, planning in a reduced dimensional space is important both for the analytical and numerical solution techniques. Of course, one can argue the use of a randomized technique like a rapidly-expanding random tree (RRT) [LaValle and Kuffner, 2001] for trajectory generation in such a high dimensional space. However, these techniques produce trajectories and not controllers for the system. This distinction leads us to the second motivation for our paradigm.

The second motivation is that ensuring that the constraints of the system Eqn.4.2 and Eqn.4.3 are satisfied is of primary importance. We need to ensure that the constraints are not broken even in the presence of control noise and modeling errors. As mentioned earlier, if the constraints are broken, the system is, in may case, irrecoverable. Hence, there is the need to write a robust feedback controller to ensure that the constraints are guaranteed to

be satisfied.

The key observation in the task and shape decomposition paradigm is that, for many problems, the constraints Eqn.4.2 and Eqn.4.3 reside in a space that has a *lower* dimension than the system state space. We exploit this by projecting the system equation Eqn.4.1 onto and orthogonal to the system constraints. We call the subspace in which the constraints reside the space of *shape freedoms* of the system and the subspace of the unconstrained variables the space of the *task freedoms* of the system.

The idea of decomposing a system into subsystems was studied in robotics in the control of robot arms by Hanafusa, Yoshikawa and Nakamura [Hanafusa et al., 1981, Nakamura, 1991]. In their formulation, they defined the kinematic output of a manipulator as the *manipulation variables*. These variables were task dependent and might be, for example, the pose of the end effector, the manipulability of the arm, or any other function of the joint angles. If there were more joints than manipulation variables, the arm was termed redundant. Baillieul [Baillieul, 1985] used the extended Jacobian technique to use the redundancy to satisfy other task requirements, like obstacle avoidance. In their task-priority formulation, Hanafusa *et al.* studied the satisfaction of two tasks, a primary and a secondary task, each with its own set of manipulation variables. Their idea was to exactly satisfy the first task's requirements and use the redundant freedoms to get as close to satisfying the second task as possible, without affecting the first task. The first task, in effect, had a higher priority than the second task.

We can describe the evolution of the shape freedoms as:

$$\dot{\boldsymbol{x}}_{\boldsymbol{s}} = \boldsymbol{f}_{\boldsymbol{s}}(\boldsymbol{x}_{\boldsymbol{s}}, \boldsymbol{x}_{\boldsymbol{t}}, \boldsymbol{u}_{\boldsymbol{s}}, t) \tag{4.4}$$

$$\boldsymbol{g}_{\boldsymbol{s}}(\boldsymbol{x}_{\boldsymbol{s}}, t) \preceq \mathbf{0} \tag{4.5}$$

$$\boldsymbol{h}_{\boldsymbol{s}}(\boldsymbol{u}_{\boldsymbol{s}}, \boldsymbol{x}_{\boldsymbol{s}}, t) \preceq \mathbf{0} \tag{4.6}$$

We can describe the evolution of the task freedoms as:

$$\dot{\boldsymbol{x}}_{\boldsymbol{t}} = \boldsymbol{f}_{\boldsymbol{t}}(\boldsymbol{x}_{\boldsymbol{t}}, \boldsymbol{u}_{\boldsymbol{t}}, t) \tag{4.7}$$

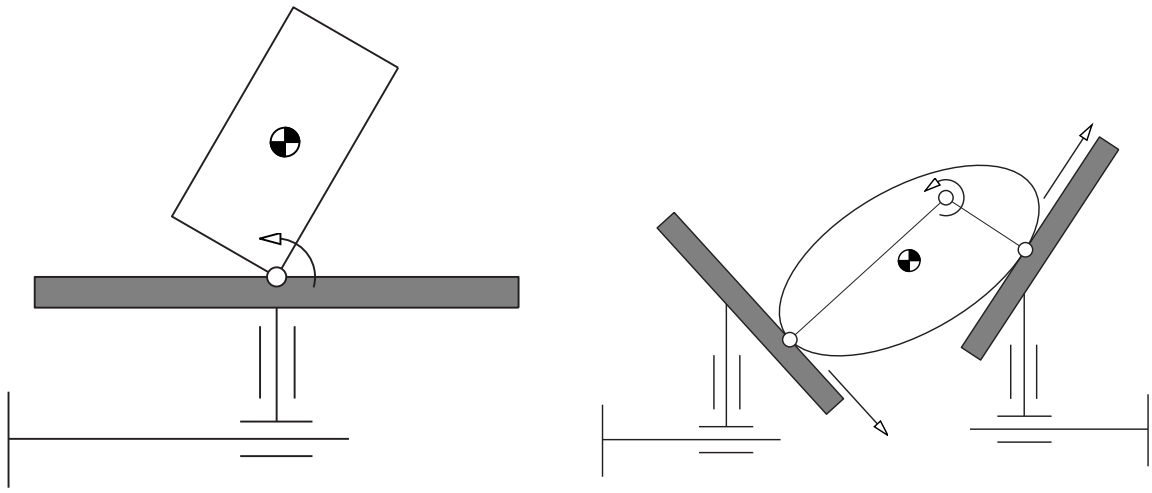
Now that we have isolated the constraints and the evolution of the constraints (which is given by the evolution of the shape freedoms Eqn.4.4), we can write a controller in this lower dimensional subspace exclusively to satisfy the constraints. This addresses both our motivations — planning is done in a low dimensional space and a controller is written to ensure robust constraint satisfaction. Since the task freedoms are unconstrained, computing their trajectories is easy. In our examples, we give the user control of the task freedoms, with the guarantee that no matter what he does, the constraints will be satisfied. A sketch

of the task and shape decomposition paradigm is shown in Eqn.4.1. The arrows denote constraints in the shape space and the solid lines indicate feasible system trajectories.

It is important to note that it is not always possible to completely decouple the two subsystems. This is manifested by the appearance of the task freedom $\boldsymbol{x_t}$ in the system equation Eqn.4.4 of the shape freedoms. In essence, the task freedoms appear as *drift* terms in the shape subsystem. In the case of an adversarial user, this drift can push the shape subsystem into a situation where the constraint will be broken, despite the best efforts of the controller. In §5, we will describe a solution to this problem using the technique of bilateral time-scaling. In §6, we will discuss a problem where the two subsystems can indeed be decoupled. In this case, planning and control are much easier, and an adversarial user can do us no harm (at least with regard to constraint satisfaction).

Another important point is that while a user can control the task freedoms, there is no guarantee that these are the freedoms that the user desires to control *i.e.*the user cannot select a set of state variables and call them his task freedoms and hope to control them in an unconstrained fashion. The decomposition is driven by the shape freedoms.

4.2 Manipulation systems



(a) (b)

Figure 4.2: Kinematic indeterminacy in manipulation systems due to (a) *Underactuation*: the rotational motion of the block about the point of contact is kinematically indeterminate (b) *Sliding contacts*: although the robot is fully actuated, the rotational motion of the ellipse about the instantaneous velocity center is kinematically indeterminate

The key observation in the task and shape decomposition paradigm is that constraints of many systems reside in a space that has a lower dimension than the system state space. In

this section, we will describe a condition for manipulation systems that will ensure that the constraint of the system, the contact acceleration constraint, resides in a low dimensional space.

Recall that the contact acceleration constraint can be written as:

$$\mathsf{J_c}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} + \mathsf{V}(\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n}_{\boldsymbol{o}}) \in \mathcal{A} \tag{4.8}$$

where

$$\mathcal{A} = \mathsf{G_c^T M^{-1} G}(\mathcal{F}) \tag{4.9}$$

is the contact acceleration cone.

Also recall that we showed that the contact acceleration cone resides in a subspace of the acceleration twist space that is orthogonal to the nullspace of $\mathsf{G_c^T}$ (denoted by $\mathcal{N}(\mathsf{G_c^T})$). Hence any motion of the system along $\mathcal{N}(\mathsf{G_c^T})$ is a motion that is orthogonal to the contact acceleration constraint Eqn.4.9.

The physical interpretation of this result for manipulation systems requires one further definition. We denote a manipulation system as *kinematically indeterminate* if, for a given contact mode, the motion of the object *cannot* completely be described by the motion of the robot. If the manipulation system is kinematically indeterminate, then $\mathcal{N}(\mathsf{G_c^T})$ exists, and the contact acceleration constraint resides in a subspace that is orthogonal to the nullspace.

There are two cases where kinematic indeterminacy can occur in manipulation systems. The first case is in the case of *underactuated manipulation* where there are fewer controls than degrees of freedom. This is illustrated in Fig.4.2(a). Here, the block with three degrees of freedom is manipulated by a robot with only two degrees of freedom. $\mathcal{N}(\mathsf{G_c^T})$ corresponds to a rotational motion of the block about the contact point. The second case can occur in the presence of sliding contacts. In Fig.4.2(b), the ellipse with three degrees of freedom is manipulated by two arms, each with two degrees of freedom. Both contacts are sliding. Here, although the arms are fully actuated, they cannot control the tangential velocity of the object at the contact point, due to the sliding constraint. As a result, the rotation of the ellipse about the instantaneous velocity center is kinematically indeterminate. The instantaneous velocity center is the point of intersection of the contact normals and, as the name suggests, the instantaneous point of zero velocity of the object and hence, the point about which the ellipse instantaneously rotates.

It is important to note that kinematic indeterminacy, although negative sounding, is not necessarily a bad thing. It can be shown that while the system shown in Fig.4.2(a) is kinematically indeterminate, it is dynamically controllable, *i.e.*the rotational motion of the block can be controlled by the motion of the robot arm if the arm moves dynamically. This means that three degrees of freedom of the block can be controlled by just two actuators, which saves us an actuator. This example of a kinematically indeterminate

but dynamically controllable system belongs to the class of *nonprehensile manipulation* [Lynch and Mason, 1999].

4.3 Summary

To summarize, in the task and shape decomposition paradigm, we project the system equations down to the subspace in which the constraints reside and write a controller to exclusively satisfy the constraints. The orthogonal freedoms are unconstrained and can be planned using standard techniques or given to a user to freely control.

The driving motivation for this paradigm is constraint satisfaction. Admittedly, the paradigm relies heavily on the constraints residing in a space that is low enough for analytical or robust numerical solutions. If the constraints depend on a large number of the state variables, or all of the state variables, then very little, or nothing is gained by the projections. However, we have shown that a class of manipulation systems exist for which the constraints do reside in a low dimensional space and for which the paradigm can be used.

In §5, we will discuss the Mobipulator problem — a kinematic nonholonomic system with an additional constraint on its state variables. We will show how this system can be decomposed into a task and a shape subsystem and use the technique of bilateral time-scaling to counter the drift that is produced in the shape subsystem due to the motion of the task freedoms. This technique ensures that the constraints are not broken by an adversarial user.

In §6, we will discuss the waiter's problem — a kinematically indeterminate manipulation system. Here, the system can be *decoupled* into the task and the shape subsystem, with the task freedoms not appearing in the system equation of the shape freedoms. We will discuss the construction of a feedback controller for the shape freedoms. We will also discuss a special case where the constraint can be deliberately broken (akin to the controlled slip strategy proposed by Brock [Brock, 1988]) to change a contact mode and how we can recover the original contact mode.

Not all kinematically indeterminate manipulation systems are dynamically controllable. In the cases where the systems are not dynamically controllable, there can be two consequences. The first, more tolerable, consequence is that the uncontrolled freedoms evolve on their own and do not affect the contact mode at the controllable freedoms. We cannot control them, but they do not cause us harm. The second, more drastic, consequence is that the dynamics of the uncontrolled freedoms results in the breaking of the contact acceleration constraint and the system transitioning into a new undesired contact mode.

There has been some work on the controllability of manipulation with frictional contact. Lynch [Lynch, 1999] studied the controllability of planar rigid bodies subjected to unilateral

thrusters. Similarities to contact manipulation are that contact forces exerted by the robot on the object are also unilateral. Each of Lynch's thrusters provided a line of force fixed in the object's frame and had unit force magnitude. Lynch proved that thruster configurations could be chosen such that with one thruster the body was small-time accessible, with two thrusters the body was controllable and with three thrusters the body was small-time locally controllable at zero velocity states. However, a solution for general manipulation systems with actuator constraints is still an open problem.

Admittedly, the success of the task and shape decomposition paradigm depends on an intelligent choice of the description of the system. It is possible to choose a system description where all the state variables appear in the constraint equation, thereby suggesting that the decomposition cannot be achieved. What we have shown in this chapter is that for kinematically indeterminate systems, the constraint does reside in a low dimensional state space, making a decomposition possible. An automated technique for transforming a description of the system into another which is more amenable to decomposition is an open problem.

Chapter 5

The Mobipulator problem

In this chapter, we will use the task and shape decomposition paradigm to find analytical solutions for a 6D kinematic system with nonholonomic constraints.

The system that we will be analyzing is called the *Mobipulator*, an abbreviation of the term "mobile manipulator". The *Mobipulator* is a wheeled robot whose domain is the desktop. The robot has four independently controlled wheels, none of them steered and uses its wheels both for locomoting itself and for manipulating common desktop items like pencils, mugs and sheets of paper. In this chapter, we will focus on the technique that the robot uses to manipulate a sheet of paper. Albeit a simple task, we will show how this simple example sheds light on the issue of motion planning for systems with nonholonomic constraints.

The mode that the Mobipulator uses to move paper is called the *dual-differential drive* mode (Fig.5.1). In this mode, the robot has two of its wheels on paper and the other two on the desktop. The robot manipulates the paper by moving the wheels on the paper and locomotes itself and the paper by moving the wheels on the desktop. The term dual-differential drive arises from the face that the behavior of the robot and the paper in this mode is akin to two differential drives, one on the desktop and the other on the paper, connected together by a rigid linkage (the robot chassis). The robot and the paper share a unique symbiotic relationship — the paper requires the robot for its motion while the robot requires the paper for turning since its wheels are not steerable.

In this chapter, we will focus on the generation of trajectories for the robot and the paper from a start to a goal while maintaining the dual-differential drive mode. We will treat the system as being purely kinematic, since the inertias of the robot and the paper are small. We control the angular velocities of the four wheels of the robot. The implication of the kinematic assumption is that the robot and the paper come to rest instantaneously when the wheels stop spinning. The system resides in a 6D configuration space of the robot and paper pose. The system has a nonholonomic constraint as well as a constraint on the

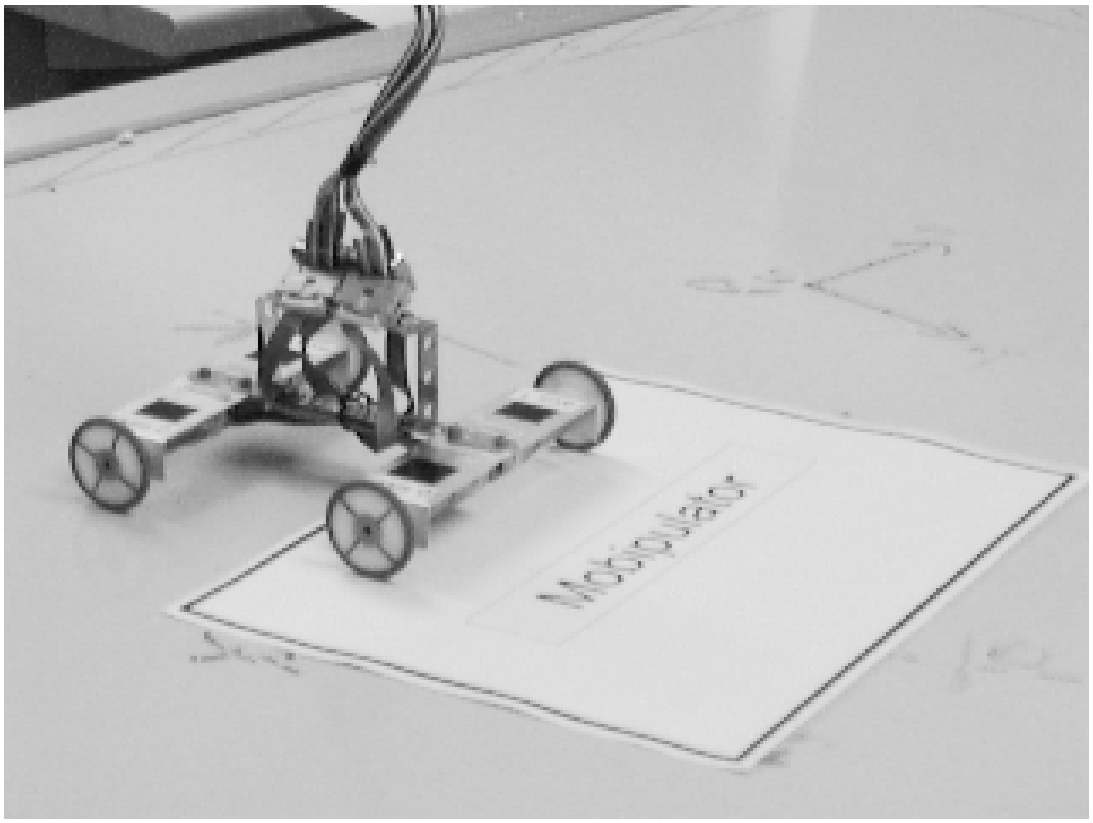


Figure 5.1: The *Mobipulator* in dual-differential drive mode

pose of the robot relative to the paper in order to maintain dual-differential drive mode.

We will solve the trajectory generation problem by first identifying the lower dimensional space of shape freedoms in which the constraints reside. We will then project the system kinematics onto this space. We will write a controller to satisfy the constraints and allow a user to control the orthogonal task freedoms. We will also consider the case of an *adversarial user*, one who forces the shape freedoms to configurations where the robot gets jammed on the paper, and discuss a solution to that problem which uses controls derived from Lie brackets.

The rest of the chapter is arranged as follows. In §5.1, we describe the hardware and the software infrastructure of the Mobipulator. In §5.2, we provide background on mobile manipulation and on the nonholonomic control problem. In §5.3, we provide the nomenclature and define the problem we wish to solve. In §5.6, we decompose the system into the task and the shape subsystems. We rewrite the shape system as a function of the task freedoms. The user controlled task freedoms appear as a drift term in the shape system. We

use bilateral time-scaling to control this drift, and the extra degree of freedom to move the shape freedoms to satisfy dual-differential drive. In §5.7, we provide a control policy that reduces the number of control switches required. §5.8 describes an implementation of the control law on the real robot and some of the problems faced. §5.9 explores a generalization for arbitrary nonholonomic systems.

5.1 The Mobipulator

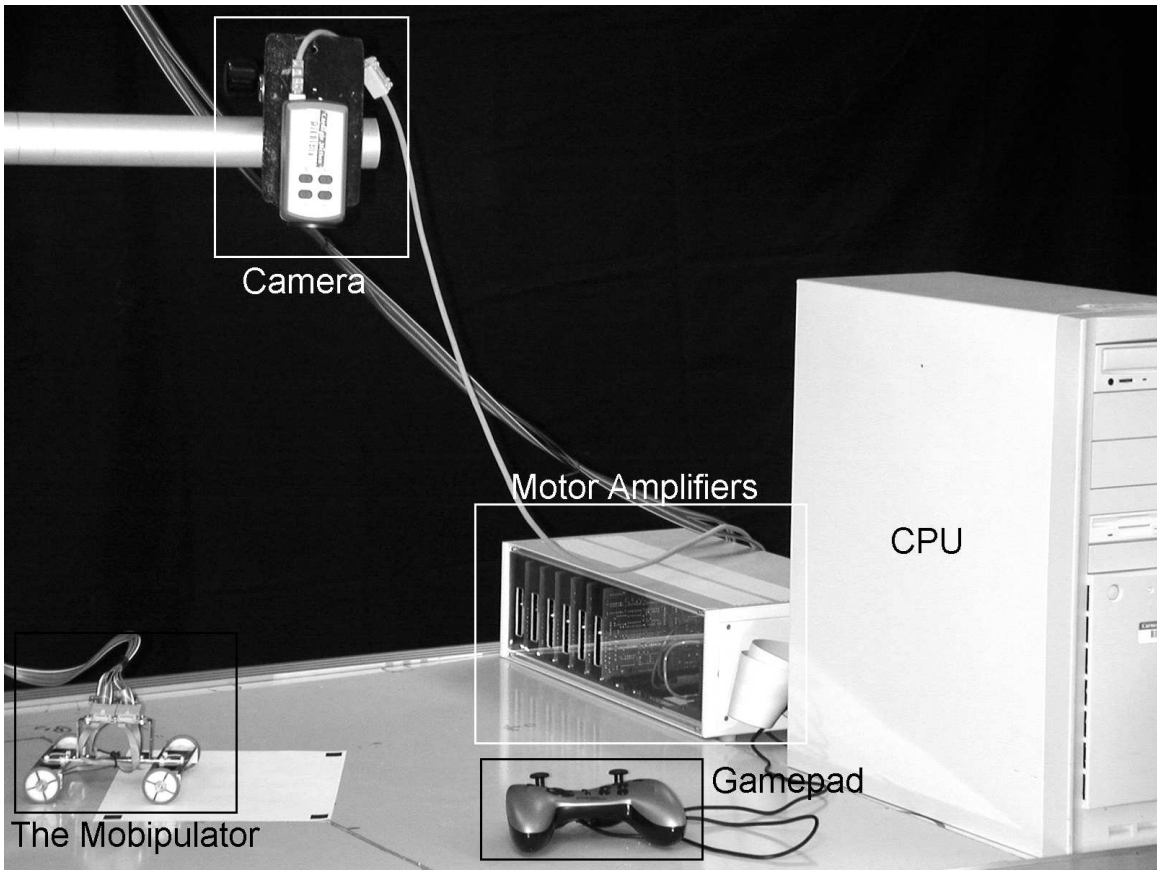


Figure 5.2: The Mobipulator system overview

The Mobipulator is a concept car. It is designed to explore different concepts of how a mobile robot can perform manipulation tasks, and in particular, the interconnectedness of locomotion and manipulation.

In this section, we provide an overview of the hardware and the software architecture of the system. As shown in Fig.5.2, the system comprises of the robot, motor amplifiers and a CPU which control the motion of the wheels, an air hockey table (with no air) which

functions as the desktop, a gamepad for joysticking the robot, and a camera that overlooks the desktop. We use the camera both for logging data and for visual servoing.

Hardware

The robot chassis is made of aluminium. The wheels are outfitted with rubber O-rings. The motors are controlled using an ISA servo control card from ServoToGo, Inc. The card provides eight independent DAC channels, each of which is connected to a LC-3002A Linear Servo Amplifier from Micro Mo Electronics, Inc. The amplifiers are housed in a custom-made control box with a PCB backplane, which makes it easy to plug and unplug all the connectors. We have also integrated a Gravis Eliminator game pad into our system to provide manual control. A Sony DFW-VL500 digital Firewire camera is mounted vertically above the desktop at a height of about 1.4 meters. The camera can grab color images at a rate of up to 30Hz at a resolution of 640x480 pixels.

Software

The software system consists of the following four components, running under Windows 2000 on a Pentium II.

The Device Driver

The Kernel-mode device driver has a top half and a bottom half. The bottom half is driven by the interrupt controller on the card (currently set at 1 kHz) to implement a PID motor controller at regular, predictable intervals. Since the driver runs in kernel mode, extended computations in an ISR can cause sluggish system behavior. Thus, all calculations are done with fixed-point values instead of floats and complex functions (such as sine) are done with lookup tables. As a result, any given ISR takes no more than 60 μs to complete on a 400 MHz Pentium II.

The Interface Library

The top half of the device driver interacts with the interface library to provide user programs a means of communicating with the driver. These are largely based on source code supplied by ServoToGo under the GNU Public License. Using the interface library, user programs may enable or disable closed-loop control and set the gains for the PID controller.

We have implemented a trajectory generator that generates velocity profiles given a demand position, velocity, and acceleration for each motor. The generator outputs various functions in real-time such as trapeziods, quadratic curves, sinusoids, and sawtooth waves

that may be concatenated to produce more interesting trajectories. The various parameters of the generator can be accessed via the interface library.

The library can also be used to retrieve data from the driver's short-term (1000 interrupt cycles) log and write directly to the DAC to affect some form of continuous control from a user program.

The Mobipulation Library

The third component, dubbed *libmobip*, combines calls to the interface library (for simplicity) with calls directly to the driver (for efficiency) to provide motion commands in physical units (meters, radians, seconds). There are four primary groups of commands.

- *Program control* : These commands to start up/shut down the library, start/stop robot motion, set up robot parameters, flush motion queues, and print error strings.
- *Four-wheel drive commands* : These commands for translating and rotating the entire robot treat the robot as a single differential drive at the center of the robot. These commands, in particular benefit from visual feedback, as described in §4.
- *Dual diff drive commands* : This command set is similar to the four-wheel drive set, except that each command also takes a parameter to indicate which half of the robot will perform the action. The two halves may be controlled either one at a time or simultaneously.
- *Logging commands* : These log the desired and actual trajectories of the robot into a simple file format for display and analysis.

Gamepad Library and GUI

We have also written a small library that allows a user to control the Mobipulator using a gamepad. The gamepad has two joysticks and eight buttons. Each joystick controls one differential drive. The buttons can be mapped to various controls like locking of individual wheels (for manipulation) and the path logger. The gamepad provides an essential tool for testing new ideas rapidly without any programming.

Perception

The motivation for using the camera is the inaccuracy in locomotion and manipulation caused by slip between the robot, the paper and the desktop. We use the camera both for measuring these errors and for correcting them by visual servoing.

The robot uses color thresholding to to locate the fiducials in each image frame. For ease of tracking, we have added four orange fiducials on the robot chasis and four green

fiducials on the corners of the paper. We have implemented a simple thresholding scheme in (R,G,B) space. At times when the robot is unable to track a feature, it uses its prior knowledge of the fiducial's geometry to reacquire them. Here is an outline of our algorithm :

1. *Initialization* : Initialize the thresholds manually.
2. *Blob Finding* : Use the thresholds to find the four largest orange and green blobs in the image. Find the centroid of each of the blobs and store them as the points tracked.
3. *Check Rectangle* : Check if the points tracked form two rectangles, one each for robot and paper, within a tolerance. Check for both orthogonality and Euclidian distance.
4. *Fit Rectangle* : If Check Rectangle fails, use the tracked points and the lengths and widths of the rectangles to fit the remaining points. Note that this can be done only if one point in a rectangle is lost.
5. *Store Old Points* : If Fit Rectangle fails, save the last set of tracked points as currently tracked.
6. *Return* : Return the position and orientation of the tracked rectangles.

The algorithm runs at 15Hz at a resolution of 640x480 pixels. The tracker runs in a separate thread from the motion commands and can thus be queried when desired. Note that the camera is first calibrated and turned until its axis is orthogonal to the desktop, prior to tracking.

The tracker has a few drawbacks. At the height at which the camera is mounted(1.4m), each pixel in the image corresponds to 3mm on the desktop. The tracker is also sometimes unable to locate the fiducials - robot fiducials are occluded by the tether and paper fiducials are occluded by the robot.

5.2 Background

Locomotion and manipulation

Several preceding systems have explored the connection between manipulation and locomotion. One of the earliest influential robots, Shakey [Nilsson, 1984], pushed boxes and other objects. More recently, the task domain of Sojourner in Mars included elements of manipulation.

A more direct approach is to attach a manipulator to a mobile platform. The JPL Cart [Thompson, 1977] provides an early example, while Romeo and Juliet [Khatib et al., 1996]

provide a current example. These robots have demonstrated effective coordination of wheels and arms in manipulation tasks.

The distributed manipulation work of Donald *et al.*[Donald et al., 1994] included a set of mobile robots pushing objects, as if each robot were a finger in a multi-fingered grasp. The Platonic Beast [Pai et al., 1994] had several limbs, each of which could be used for locomotion or manipulation.

Fiat [Mason et al., 1999], the paper-lifting robot developed by Rus and her coworkers, is also a desktop mobile manipulator. But unlike the Mobipulator, Fiat, uses an attachment with sticky adhesive tape to immobilize the paper to the robot. Once this is achieved, the robot moves to the goal location and uses another attachment to release the paper.

All of these works illustrate that there is an underlying connection between locomotion and manipulation. In our case, the robot is just one of several movable objects in the task. The function of each actuator is resolved according to the task, be it manipulation, locomotion, or something not clearly classifiable as either.

Nonholonomic systems

The general form of a control-affine system with nonholonomic constraints is given by:

$$\Sigma : \qquad \dot{\boldsymbol{x}} = u_1 \boldsymbol{f_1}(\boldsymbol{x}) + \cdots + u_m \boldsymbol{f_m}(\boldsymbol{x}) \tag{5.1}$$

where $2 \leq m < n$, $\boldsymbol{x} = (x_1 \cdots x_n)^T$ is the state vector defined in an open subset S of $\mathbb{R}^n$, $u_i \in \mathbb{R}$ are the control inputs, and $\boldsymbol{f_1}, \cdots, \boldsymbol{f_m}$ are vector fields on S. Σ is said to be *completely nonholonomic* if the rank of the controllability Lie algebra generated by $u_1 \cdots u_m$ is n. A completely nonholonomic system is completely controllable (*Chow's theorem* [Chow, 1939]).

Motion planning for nonholonomic systems is complicated by the fact that not all motions are feasible, only those which satisfy the instantaneous nonholonomic constraints. Nevertheless, the completely nonholonomic assumption guarantees that feasible motions do exist which steer an arbitrary initial state to a final state. We refer the reader to Kolmanovsky and McClamroach's paper [Kolmanovsky and McClamroch, 1995] for a detailed review of motion planning for nonholonomic systems.

The motion planning problem for nonholonomic systems can be defined as : for every pair of points $(\boldsymbol{p},\boldsymbol{q}) \in S$, generate an open-loop control $\boldsymbol{u}(t) = (u_1 \cdots u_m)^T$ that steers $\boldsymbol{p}$ to $\boldsymbol{q}$.

One approach is to consider the extended system :

$$\begin{aligned} \Sigma_e : \qquad \dot{\boldsymbol{x}} \;=\; & v_1 \boldsymbol{f_1}(\boldsymbol{x}) + \cdots + v_m \boldsymbol{f_m}(\boldsymbol{x}) + \\ & v_{m+1} \boldsymbol{f_{m+1}}(\boldsymbol{x}) + \cdots + v_r \boldsymbol{f_r}(\boldsymbol{x}) \end{aligned} \tag{5.2}$$

where $\boldsymbol{f_{m+1}}, \cdots, \boldsymbol{f_r}$ are higher-order Lie brackets of the $\boldsymbol{f_i}$ chosen so that $\boldsymbol{f_1(x)}, \cdots, \boldsymbol{f_r(x)}$ span $\mathbb{R}^n$ for all $\boldsymbol{x} \in S$. (5.2) can be solved for the control $\boldsymbol{v}(\mathrm{t})$ that steers $\boldsymbol{p}$ to $\boldsymbol{q}$. This solution is then transformed to a control $\boldsymbol{u}(\mathrm{t})$ in Σ. If Σ is nilpotent, $\boldsymbol{u}(\mathrm{t})$ is an exact solution.

The hard part is to generate Lie brackets from the control inputs. Fast switchings of piecewise constant or polynomial inputs [Lafferriere and Sussmann, 1993] and high-frequency high-amplitude periodic control inputs [Sussmann and W., 1991] are some of the techniques used to generate motions in the directions of the Lie brackets.

Motions of the Mobipulator

One motion along a higher-order Lie bracket for the Mobipulator is akin to parallel parking the robot relative to the paper. This is achieved by spinning the wheels repeatedly forwards and backwards for small time intervals. We use rubber O-rings on the wheels to help better grip the paper. This also increases friction between the feet and the desktop and thus the wheels require a minimum torque before they start spinning. As a result, fast switchings of wheel torques cause a nonsmooth sideways motion and result in the wheels slipping. This produces errors in both the pose of the robot and the paper. Hence, we would like to avoid control switches for accurate motion.

Motion planning with obstacle avoidance

Gurvits and Li [Gurvits and Li, 1993] proposed an approach for nonholonomic motion planning in the presence of obstacles. Their method first constructed a path connecting the start and the goal that avoided obstacles but was not feasible. They then used high-frequency high-amplitude periodic inputs to generate an approximate path.

Mirtich and Canny [Mirtich and Canny, 1992] used skeletons - collections of fixed (typically nonfeasible) paths which stay maximally clear of the obstacles - to steer a mobile robot from an initial to a final state. Low complexity paths were generated by making the system loosely follow the skeleton. Jacobs *et al.*. [Jacobs et al., 1991] developed a different approach, also based on approximating a nonfeasible path by feasible path segments, to plan motions for a car-like robot. This method was extended by Laumond *et al.*[Laumond et al., 1994] to car-like robots towing a trailer.

Barraquand and Latombe [Barraquand and Latombe, 1993] proposed a planner for nonholonomically constrained mobile robots in an obstacle field using a similar approach. They restricted the control space of the robot by discretizing the range of allowable steering angles. At a configuration, they applied the constant controls for a given time interval to generate child nodes. The child nodes were added to a queue if they were not 'close' enough to the parent (decided by a user given metric and threshold distance) and if the path from

parent to child did not intersect an obstacle. A path from start to goal was constructed by running a breadth first search. While the planner is resolution complete, it shares the drawbacks of other approaches that discretize state and/or controls — the generated paths tend to zigzag around the true optimal path.

5.3 Problem statement

The kinematics for the Mobipulator in dual-differential drive mode can be described as :

$$M: \quad \begin{pmatrix} \dot{\boldsymbol{x}}_{\boldsymbol{P}} \\ \dot{\boldsymbol{x}}_{\boldsymbol{R}} \end{pmatrix} = \mathsf{A} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{pmatrix} = \mathsf{A}\, \boldsymbol{\omega} \tag{5.3}$$

where $\boldsymbol{x}_{\boldsymbol{P}} = (x_P\ y_P\ \theta_P)^T$ is the pose of the paper in the world frame, $\boldsymbol{x}_{\boldsymbol{R}} = (x_R\ y_R\ \theta_R)^T$ is the pose of the robot *relative* to the paper, and the ω_i are the angular velocities of the wheels.

M requires the robot to be in dual-differential drive mode. The shaded region in Fig.5.3 describes the allowable pose of the robot relative to the paper for a given θ_R. Each light rectangle represents the locus of the center of the robot with one of the wheels touching the edge of the paper. If the center lies in the interior of a light rectangle, the corresponding wheel lies in the interior of the paper. The shaded polygon is the locus of points with the manipulating wheels (1 and 2) on the paper and the locomoting wheels (3 and 4) on the desktop.

The geometry and position of the allowable region depends only on the relative pose θ_R and the dimensions of the robot and the paper. Furthermore, the allowable region is a constraint only on the position of the robot relative to the paper and not on the pose of the paper. The constraint can thus be written as:

$$H: \quad \boldsymbol{h_1}(\theta_R) \preceq \begin{pmatrix} x_R \\ y_R \end{pmatrix} \preceq \boldsymbol{h_2}(\theta_R) \tag{5.4}$$

where "$\preceq$" denotes componentwise inequality.

The problem can be stated as:

> Given a start and a desired goal location for the paper, find the wheel angular velocities ω_i that move the robot and the paper to the desired goal while satisfying the dual-differential drive constraint constraint H? Furthermore, is it possible to construct a control policy that reduces the number of control switches required?

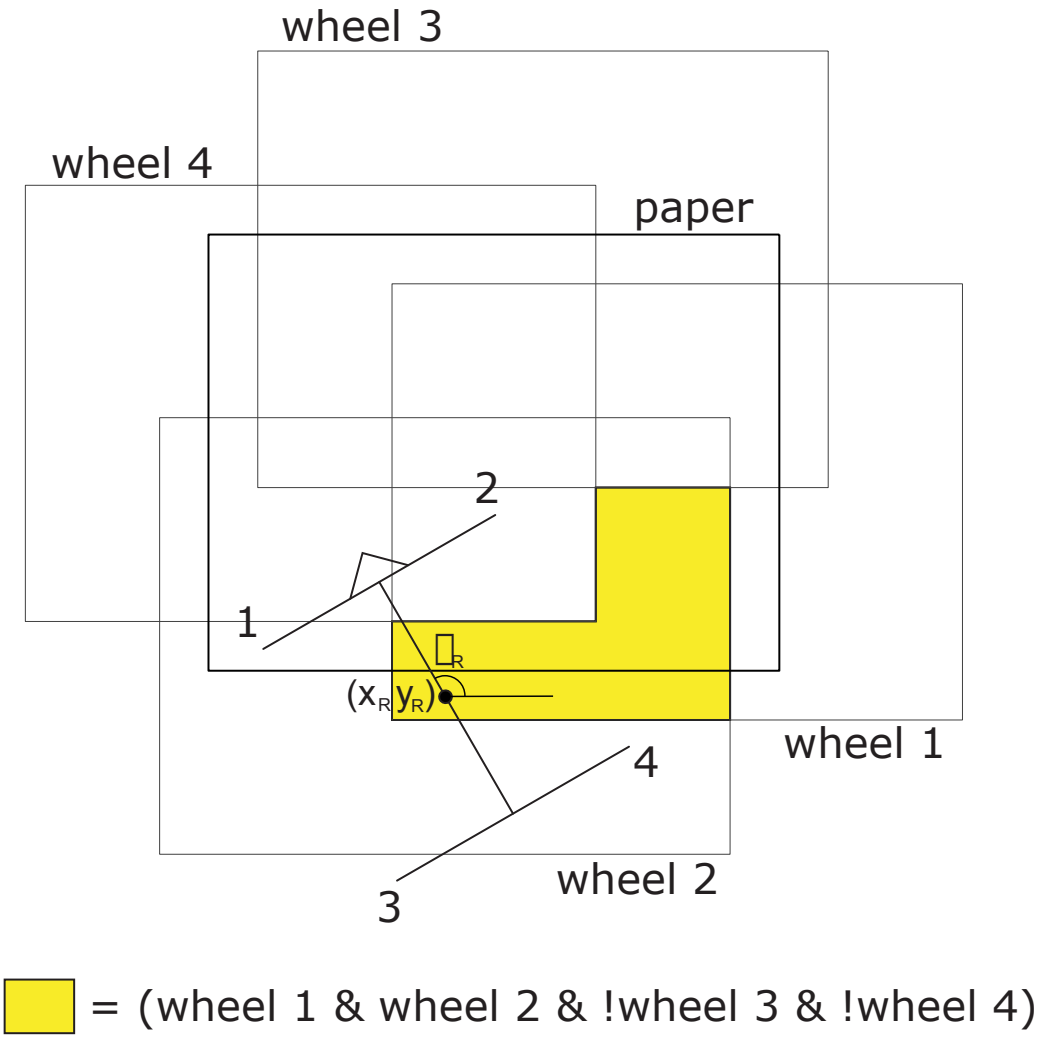


Figure 5.3: Allowable $(x_R\ y_R)$ at $\theta_R = 2\pi/3$

5.4 A configuration space planner

Motion planning for the robot and the paper is complicated due to the nonholonomic velocity constraint on the system. One approach to solving this problem is to add an additional constraint in such a way that the nonholonomic constraint becomes a holonomic constraint. In this section we will describe one such planner that we implemented on the Mobipulator and state its drawbacks.

We can envision the Mobipulator problem as a tractor (the robot) pulling a trailer (the paper) with a *moveable* hitch. At each instant, the nonholonomic constraint provides a range of allowable hitch placements. We change this constraint to a holonomic constraint by forcing the robot to virtually hitch to one of the allowable hitch points. Once hitched, the system can be treated as a two-link arm, with one link being the paper and the other link being the robot. Without the nonholonomic velocity constraints, we are allowed the luxury of planning in the system configuration space.

It is important to note that converting a nonholonomic constraint into a holonomic constraint reduces the reachable configurations of the system. As a simple example, consider a single differential drive robot. The nonholonomic constraint on the system prevents it from moving in a direction orthogonal to the orientation of the robot. We can convert this constraint into a holonomic constraint by locking the orientation of the differential drive. However, by doing so, we have restricted the reachable configuration space of the system — the robot can now only move forwards and backwards. Likewise, for the Mobipulator problem, we are sacrificing reachability for ease of planning.

Fig.5.4 shows six snapshots from an example in which the robot drags the paper from configuration 1 to 6. The Mobipulator path planner is an extension of Sacks's planner for a planar linkage with static and movable obstacles [Sacks, 2001]. The planner generates a path in two phases. The first phase searches the critical link/obstacle configuration space for a free-space path to the link goal configuration. The configuration space is represented by an exact contact space partition that is computed by a sweep plane algorithm [Sacks, 1998]. The planner performs an A* search based on a heuristic distance function. The search nodes are straight-line paths from the current configuration to the goal and contact patches (subsets of contact space where a specific link feature touches a specific obstacle feature). In our example, the paper is the critical link and the path consists of a line from 1 to 3, a curved contact path to 4, and a free path to 6.

The second phase extends the critical link path to the entire linkage. It constructs a velocity field that drives the critical link along its path and that enforces the joint constraints. It integrates the velocity field until the critical link reaches its goal. If another link hits an obstacle, the planner modifies the velocity field by subtracting the component that is normal to the obstacle. If the path ends without reaching the goal, the planner resumes

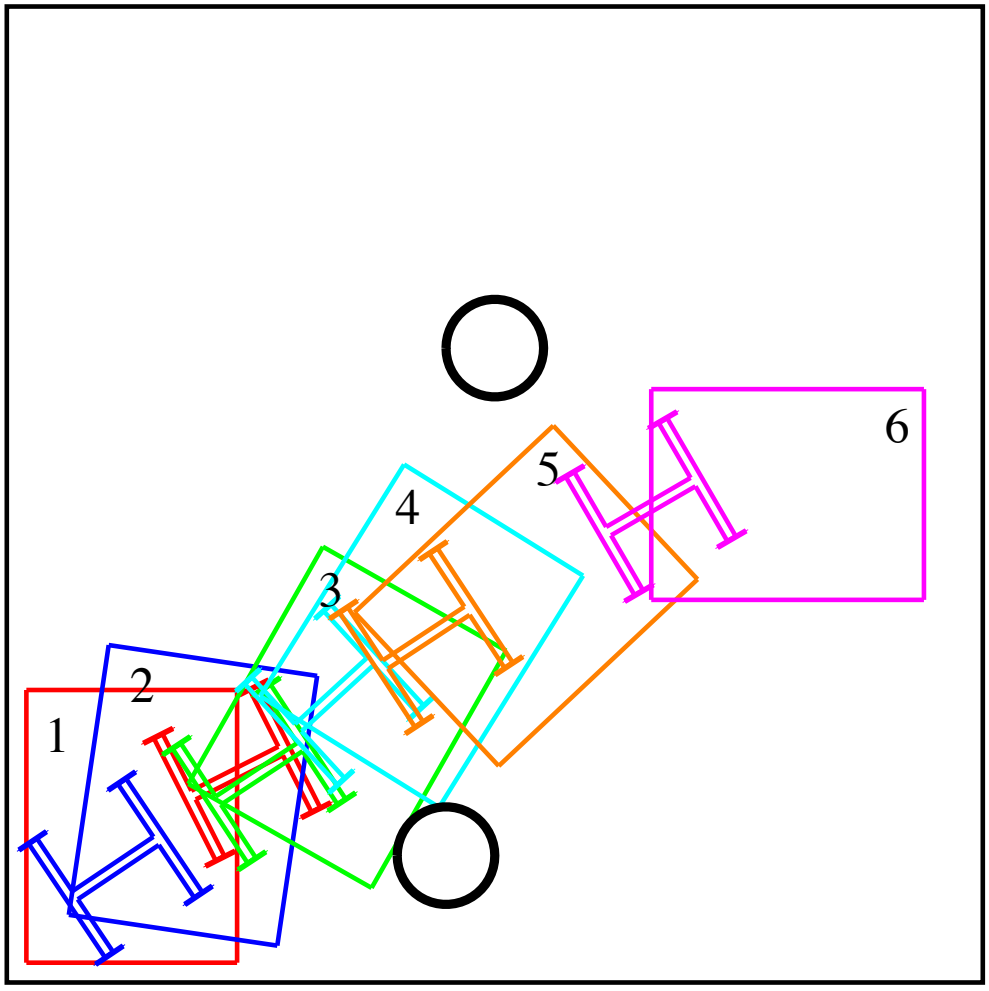


Figure 5.4: A configuration space planner for paper manipulation in dual-differential drive mode

the A* search and tries the second phase on the next best path.

The first phase plans a path for the paper to the goal configuration as before. The second phase constructs a velocity field for the four wheels that drives the paper along this path. The planner integrates the wheel velocity field until the paper reaches its goal. If dual differential-drive mode is violated, the simulation is halted.The planner then uses a series of heuristics to restore dual differential-drive mode by moving the robot without moving the paper. The simplest heuristic is to move forward or backward with four equal wheel velocities. In our example, the robot drags the paper with its back wheels from 1 until just before 3 when the left wheel comes off the paper. The robot moves backward to 3 and simulation proceeds.

The main drawback of this approach is that we are overriding the natural motion of the system and forcing it to behave in a holonomically constrained fashion. Furthermore the simple heuristic of moving across the paper when the dual-differential drive constraint is violated is not guaranteed to work if there is an obstacle across the paper which prevents the robot from re-hitching. In such situations, the robot is forced to move far away from the paper, skid steer by itself into a new orientation and try to re-hitch. A completely new plan has to be written for this motion of the robot.

In the subsequent sections, we will describe how we can use the freedom provided by the nonholonomic constraint to place the movable hitch optimally at each instant and plan for motions of the robot and the paper without the need for heuristics.

5.5 Task and shape freedoms

To choose the task and the shape freedoms of the system, we revisit the dual-differential drive constraint H:

$$H: \quad \boldsymbol{h_1}(\theta_R) \preceq \begin{pmatrix} x_R \\ y_R \end{pmatrix} \preceq \boldsymbol{h_2}(\theta_R) \tag{5.5}$$

We notice that the constraint depends only on $\boldsymbol{x_R}$ and not on $\boldsymbol{x_P}$. The absolute location of the paper is immaterial for the constraint, it is the location of the robot relative to the paper that is important. The lower dimensional space of $\boldsymbol{x_R}$ where the constraint H resides constitute the space of shape freedoms for the system. The pose of the paper $\boldsymbol{x_P}$ constitutes the space of task freedoms for the system.

In the next section, we will project the motion of the system onto the constraint space and write a controller to satisfy H. We will give the control of the task freedoms $\boldsymbol{x_P}$ to a user — the user has unrestricted freedom to move the paper on the desktop and the controller will ensure that H is satisfied.

5.6 Control of shape freedoms

In this section we will prove that we can control the Mobipulator system to follow a user-specified $\boldsymbol{x_P}(s)$ and satisfy H. We will first consider the case where the user has control of the velocity $\dot{\boldsymbol{x}}_{\boldsymbol{P}}(t)$ and show that the resulting system is *not* small time locally controllable (STLC). This negative result will provide us with insight to modify the user's control to a path $\boldsymbol{x_P}(s)$. We will show that the resulting system is STLC only if there are no constraints on $\dot{s}(t)$.

The shape and task subsystems

We decompose the Mobipulator system M into two subsystems - the task system M_t and the shape system M_s. M_t comprises of the task freedoms the user controls (the $\boldsymbol{x_P}$) and M_s comprises of the shape freedoms (the $\boldsymbol{x_R}$). We will analyze each of these subsystems separately.

$$\begin{aligned} M &: \begin{pmatrix} \dot{\boldsymbol{x}}_{\boldsymbol{P}} \\ \dot{\boldsymbol{x}}_{\boldsymbol{R}} \end{pmatrix} = \mathsf{A}\,\boldsymbol{\omega} = \begin{bmatrix} \mathsf{B} \\ \mathsf{C} \end{bmatrix} \boldsymbol{\omega} \\ M_t &: \dot{\boldsymbol{x}}_{\boldsymbol{P}} = \mathsf{B}\,\boldsymbol{\omega} \\ M_s &: \dot{\boldsymbol{x}}_{\boldsymbol{R}} = \mathsf{C}\,\boldsymbol{\omega} \end{aligned}$$

The task subsystem

We first observe that B is of rank 3. This is beacause the motion of $\boldsymbol{x_P}$ is unconstrained in M_t, i.e. if H is satisfied, any desired $\dot{\boldsymbol{x}}_{\boldsymbol{P}}$ can be attained by a correct selection of the wheel velocities. We create an augmented rank 4 system by adding the nullspace vector of B, $\boldsymbol{n_B}$, and an additional scalar α which we can control without affecting the task freedoms.

$$M_{ta}: \quad \begin{pmatrix} \dot{\boldsymbol{x}}_{\boldsymbol{P}} \\ \alpha \end{pmatrix} = \begin{bmatrix} \mathsf{B} \\ \boldsymbol{n_B} \end{bmatrix} \boldsymbol{\omega} \tag{5.6}$$

We can now invert the augmented system. Intuitively, α represents the one degree of freedom that we have in choosing $\boldsymbol{\omega}$. We will use this freedom to satisfy H.

$$\boldsymbol{\omega} = \begin{bmatrix} \mathsf{B} \\ \boldsymbol{n_B} \end{bmatrix}^{-1} \begin{pmatrix} \dot{\boldsymbol{x}}_{\boldsymbol{P}} \\ \alpha \end{pmatrix} \tag{5.7}$$

The shape subsystem

The shape freedoms can be rewritten in terms of the task freedoms using Eqn. 5.7.

$$\dot{\boldsymbol{x}}_{\boldsymbol{R}} \;=\; \mathsf{C} \begin{bmatrix} \mathsf{B} \\ \boldsymbol{n}_{\boldsymbol{B}} \end{bmatrix}^{-1} \begin{pmatrix} \dot{\boldsymbol{x}}_{\boldsymbol{P}} \\ \alpha \end{pmatrix} \tag{5.8}$$

This can be arranged in a more intuitive form :

$$M_{sa}: \qquad \dot{\boldsymbol{x}}_{\boldsymbol{R}} = \mathsf{F}\, \dot{\boldsymbol{x}}_{\boldsymbol{P}} + \boldsymbol{g}\, \alpha \tag{5.9}$$

From Eqn. 6.23 we can see that the motion of the task freedoms appears as a drift term $\mathsf{F}\, \dot{\boldsymbol{x}}_{\boldsymbol{P}}$ in the shape system and the degree of freedom α appears as the scalar control.

We can gain further insight on M_{sa} by studying $\boldsymbol{g}$:

$$\boldsymbol{g} = \frac{c}{4} \begin{pmatrix} \cos(\theta_R) \\ \sin(\theta_R) \\ 0 \end{pmatrix} \tag{5.10}$$

where c is the radius of each wheel.

This field lets the robot move forwards and backwards relative to the paper. For example, if there was no drift (i.e. if $\dot{\boldsymbol{x}}_{\boldsymbol{P}}$ were $\mathbf{0}$), then the only robot motion that does not manipulate the paper is this forwards-backwards motion.

A negative result

For H to be satisfied, we would like to be able to control the shape freedoms (described by M_{sa}) immaterial of the drift generated by the user-controlled task freedoms. If we allow the user to control $\dot{\boldsymbol{x}}_{\boldsymbol{P}}(t)$, this requires M_{sa} to be STLC for all $\dot{\boldsymbol{x}}_{\boldsymbol{P}}$ and $\boldsymbol{x}_{\boldsymbol{R}}$. We use the following theorem to prove that this is *not* true.

Theorem 3 ([Sussman, 1978])**.** *The system M_{sa} is small-time locally controllable at $\boldsymbol{x}_{\boldsymbol{R}}$ if and only if there exists some α^0 such that for all $\dot{\boldsymbol{x}}_{\boldsymbol{P}}$*

$$\mathbf{0} = \mathsf{F}\, \dot{\boldsymbol{x}}_{\boldsymbol{P}} + \boldsymbol{g}\, \alpha^0 \tag{5.11}$$

This theorem defines a condition for a nonlinear system with drift to be STLC with a single control. The intuition for the condition is that the control must be capable of maintaining the equilibrium state for all possible drifts.

Corollary 1. *The system M_{sa} is not STLC*

Proof. (5.11) is an overconstrained system with three equations and one variable. Any non-zero choice of $\dot{\boldsymbol{x}}_{\boldsymbol{P}}$ will yield no solution for α^0. □

Intuitively this means that the drift field will always dominate and can force the system towards a constraint boundary, and then violate it. Thus, if the user has control of $\dot{\boldsymbol{x}}_{\boldsymbol{P}}(t)$, it is not always possible to satisfy H.

Time scaling

We overcome this problem by allowing the user to control the path $\boldsymbol{x}_{\boldsymbol{P}}(s)$. Note that there is no notion of time in the path specification. We control the *trajectory* of the task freedoms by specifying a *time-scaling* $s(t)$ which assigns a point on the path for each $t \in [0, T]$, where T is the time to completion of the path. The velocity at any time t is given by :

$$\dot{\boldsymbol{x}}_{\boldsymbol{P}}(\boldsymbol{s}(\boldsymbol{t})) = \boldsymbol{x}'_{\boldsymbol{P}}(\boldsymbol{s}(\boldsymbol{t}))\ \dot{s}(t) \tag{5.12}$$

We control the the time-scaling rate $\dot{s}(t)$. This in turn provides us with control of the the velocity of the task freedoms along the path.

The concept of time scaling was used by Bobrow *et al.*[Bobrow et al., 1985] to find time-optimal trajectories of a fully-actuated manipulator along a specified path, subject to limitations in actuator torque. They used a time-scaling $s(t)$ with a unilateral constraint $\dot{s} > 0$, i.e. the end-effector was not allowed to move backwards along the specified path.

We first restrict $\dot{s}(t) > 0$, i.e. we cannot reverse the motion of the paper along the path. Denoting $\dot{s}(t)$ by β, we write the time-scaled shape system as :

$$M_{su} : \begin{pmatrix} \dot{\boldsymbol{x}}_{\boldsymbol{R}} \\ \dot{s} \end{pmatrix} = \begin{pmatrix} \mathsf{F}\boldsymbol{x}'_{\boldsymbol{P}} \\ 1 \end{pmatrix} \beta + \begin{pmatrix} \boldsymbol{g} \\ 0 \end{pmatrix} \alpha \tag{5.13}$$

M_{su} is a control-affine system with one unilateral control $\beta > 0$. We can test STLC for this system :

Theorem 4 ([Goodwine and Burdick, 1996])**.** *The system M_{su} is small-time locally controllable at $\boldsymbol{x}_{\boldsymbol{R}}$ if and only if there exists some α^0 and some $\beta^0 \neq 0$ such that for all $\boldsymbol{x}_{\boldsymbol{P}}$*

$$\mathbf{0} = \begin{pmatrix} \mathsf{F}\boldsymbol{x}'_{\boldsymbol{P}} \\ 1 \end{pmatrix} \beta^0 + \begin{pmatrix} \boldsymbol{g} \\ 0 \end{pmatrix} \alpha^0 \tag{5.14}$$

Corollary 2. The system M_{su} is not STLC

Proof. The last row of (5.14) can be 0 for any arbitrary $\boldsymbol{x}_{\boldsymbol{P}}$ only if $\beta^0 = 0$. Hence the system M_{su} is *not* STLC. □

We now remove the restriction on $\dot{s}$. This results in a bilateral control β. What this means is that the paper is allowed to move forwards and backwards along the path. For example, in Fig.5.5, the user specifies the path $(ac - cb - bd)$, we specify the bilateral time-scaling $s(t)$ and the resulting trajectory is $(ac - cb - bc - cb - bd)$.

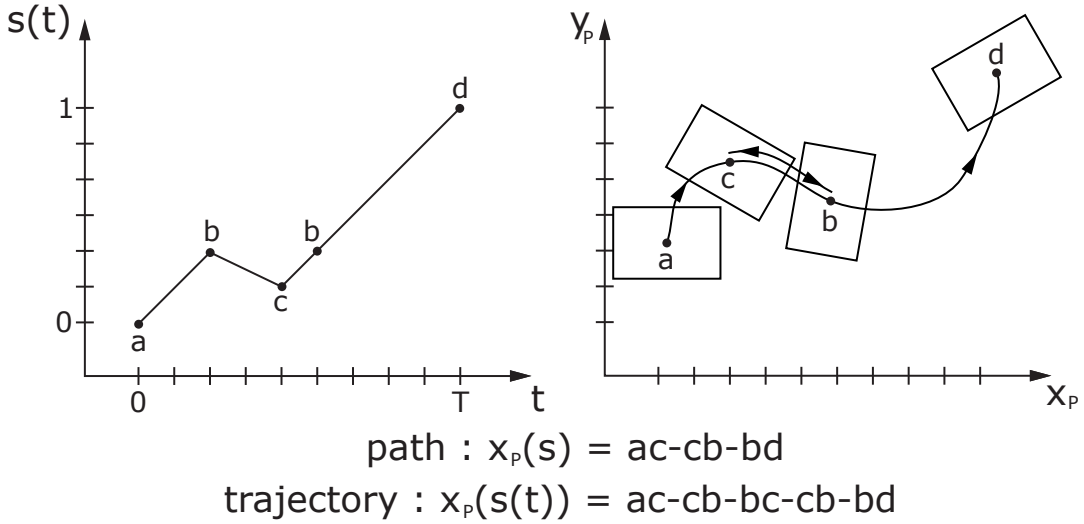


Figure 5.5: Time-scaling of user's path

The untestricted system M_{sb} looks like (5.13) but with β unrestricted. We rewrite it as :

$$M_{sb}: \quad \begin{pmatrix} \dot{\boldsymbol{x}}_{\boldsymbol{R}} \\ \dot{s} \end{pmatrix} = \boldsymbol{f}_{\boldsymbol{\beta}}\, \beta + \boldsymbol{f}_{\boldsymbol{\alpha}}\, \alpha \tag{5.15}$$

We prove this system is STLC with the following theorem:

Theorem 5 ([Chow, 1939]). *The system M_{sb} is small-time locally controllable at $\boldsymbol{x}_{\boldsymbol{R}}$ if the controllability Lie algebra generated iterated Lie brackets of $\boldsymbol{f}_{\boldsymbol{\beta}}$ and $\boldsymbol{f}_{\boldsymbol{\alpha}}$ spans the state space.*

Corollary 3. *The system M_{sb} is STLC*

Proof. We compute the brackets $[\boldsymbol{f}_{\boldsymbol{\beta}}]$, $[\boldsymbol{f}_{\boldsymbol{\alpha}}]$, $[\boldsymbol{f}_{\boldsymbol{\beta}},\boldsymbol{f}_{\boldsymbol{\alpha}}]$ and $[\boldsymbol{f}_{\boldsymbol{\beta}}, [\boldsymbol{f}_{\boldsymbol{\beta}},\boldsymbol{f}_{\boldsymbol{\alpha}}]]$. The expressions for the brackets are long and have been omitted here. They were computed automatically using MATLAB. To check if the brackets span the 4-dimensional tangent space, we compute the determinant of the matrix comprising of the four brackets. The expression for the determinant is, again, extremely large. However, by examining the terms, we can show that the determinant goes to zero only if $\boldsymbol{x}'_{\boldsymbol{P}} = \boldsymbol{0}$. Hence the brackets span the 4-dimensional tangent space at all states where $\boldsymbol{x}'_{\boldsymbol{P}} \neq \boldsymbol{0}$. □

Discussion

We have proved that we can follow any task freedom path and satisfy the constraints. To show STLC, we have had to use higher order brackets of $\boldsymbol{f_\beta}$ and $\boldsymbol{f_\alpha}$. An application of these brackets will cause control switchings and the resulting motion will not be smooth. Hence we must choose our control policy in such a way that we minimize the use of the brackets. We will describe one such policy in §5.7. Another point of concern is that since $\dot{s}$ is unrestricted, we can move forwards and backwards along our path. We will prove in §5.7 that any arbitrary path can be completed in a finite amount of time.

5.7 Control of the Mobipulator

Here we will propose control policies α and β, for the Mobipulator. We will show that control switchings can be restricted to a finite number of *switching points*.

Control policy

Recall that the control vector field $\boldsymbol{g}$ (Eqn. 5.10) allows the motion of the robot forwards and backwards relative to the paper. This is shown as the line LL' in Fig.5.6. We can use α to servo to any point on this line. LL' intersects the constraint boundaries at L_{max} where the robot is farthest from paper, and L_{min} where the robot is closest to the paper. Our policy servos the robot to the midpoint L_{mid}. This maximizes the minimum distance between the wheels and the edges of the paper and is a safe policy that is less severe to motion errors.

For β, we use the simple policy of $\beta = 1$ as long as H is satisfied. The next sub-section describes the policies chosen when H is violated.

Switching points

Note that with $\beta = 1$ we are deliberately allowing the paper to drift. We do this to avoid motions that require control switchings until imperative. H is violated when $L_{min} \rightarrow L_{max}$ and no further motion is possible along the control $\boldsymbol{g}$. This occurs either when a manipulating wheel and a locomoting wheel touch edges of the paper (Q and R in Fig.5.7) or when a manipulating wheel touches a corner of the paper (P in Fig.5.7). We term these points *switching points* as they mandate a switch in the control policy.

Control policy at switching points

At switching points, we move the paper backwards along the path for a small time interval δt by setting $\beta = -1$. Choice of α depends on whether we want net motion forwards or

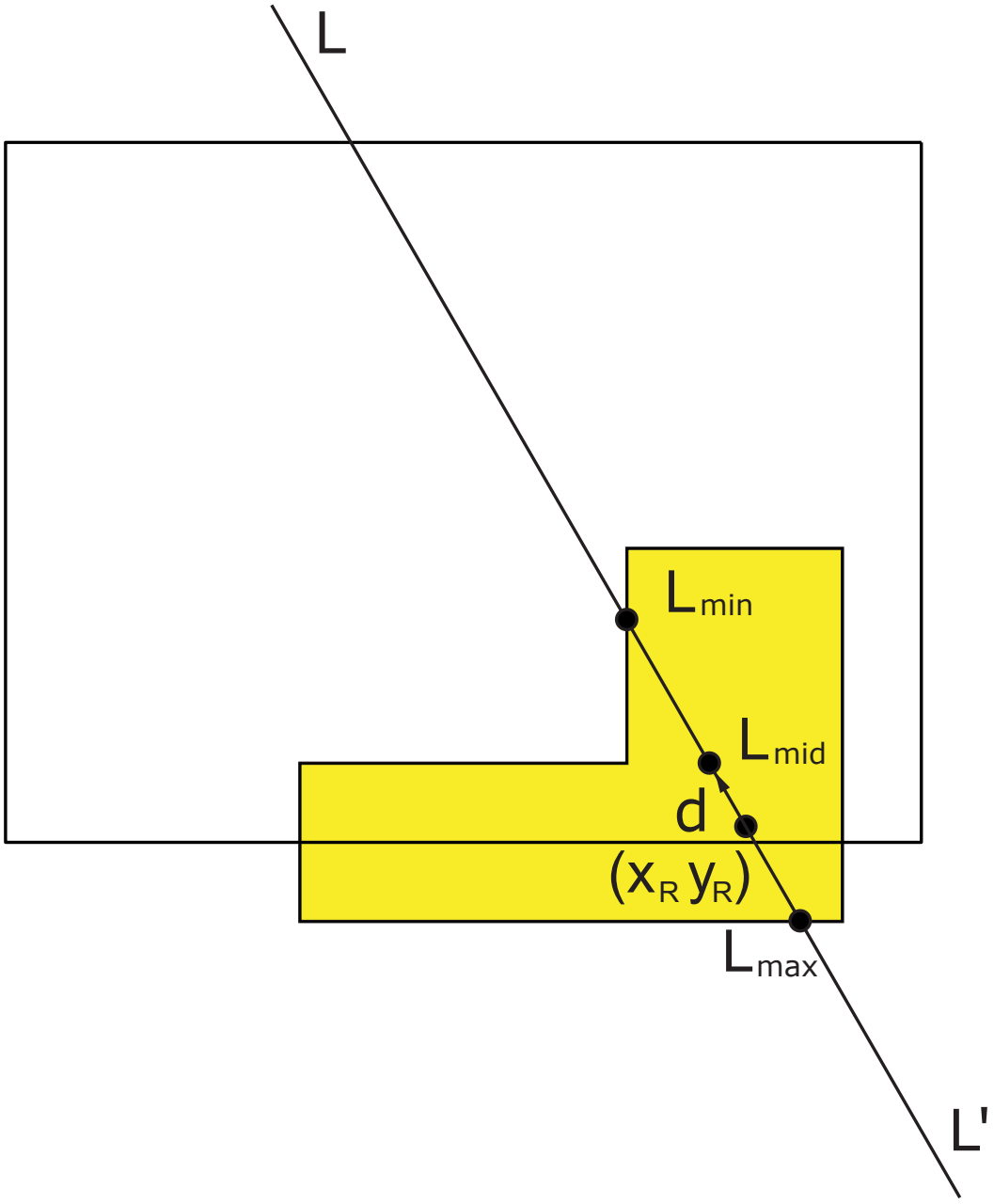


Figure 5.6: Control policy for α

backwards. To escape from P, we need to move along x_R, while to escape from Q, we need to move along $-y_R$. Consider P. The motion of the robot along x_R is given by :

$$\dot{x}_R = \boldsymbol{F(1)x'_P}\beta + \boldsymbol{g(1)}\alpha \tag{5.16}$$

$\boldsymbol{F(1)}$ and $\boldsymbol{g(1)}$ are the first row of F and $\boldsymbol{g}$ respectively.

For small δt, we can approximate $\boldsymbol{F(1)x'_P}$ and $\boldsymbol{g(1)}$ as constants k_F and k_g respectively. At switching points we first set $\beta = -1$ and α to α_1. After δt, we set $\beta = 1$ and α to α_2. The net motion along x_R is given by the sum :

$$\begin{aligned} \delta x_R \approx \dot{x}_R \delta t &= k_F \beta \delta t + k_g \alpha \delta t \qquad (5.17) \\ \delta x_{R1} &\approx -k_F \delta t + k_g \alpha_1 \delta t \\ \delta x_{R2} &\approx k_F \delta t + k_g \alpha_2 \delta t \\ \delta x_{R1} + \delta x_{R2} &\approx k_g(\alpha_1 + \alpha_1)\delta t \qquad (5.18) \end{aligned}$$

By servoing to L_{max}, we can obtain the largest positive α and get the largest motion along x_R. Conversely, by servoing to L_{min}, we can obtain the largest negative α and get the largest motion along $-x_R$. Note that the net motion obtained is independent of the motion of the paper as k_F is cancelled during motion.

We continue the motions until the wheel is sufficiently clear of the constraining edge or vertex (defined by a threshold w shown in Fig.5.7). Since each of the abovementioned motions produces the same net motion along the desired direction, w will be attained in finite time.

5.8 Implementation

The Mobipulator system has a camera that tracks colored fiducials marked on the robot and the paper, and provides pose information at 10 Hz. Our test path is a hexagon whose width is 8 times the width of the robot. The paper is rotated by 60^o about its center at each vertex. Note that, with the chosen path and the starting pose of the robot, no switching points are encountered. We will describe the effect of switching points separately.

The movable hitch

Fig.5.8 illustrates the effect of the controller. During the turn, the robot moves in and out of the paper optimally placing the virtual hitch at each instant such that it remains safely in dual-differential drive mode. At each instant, the robot maximized the minimum distance between the wheels and the paper.

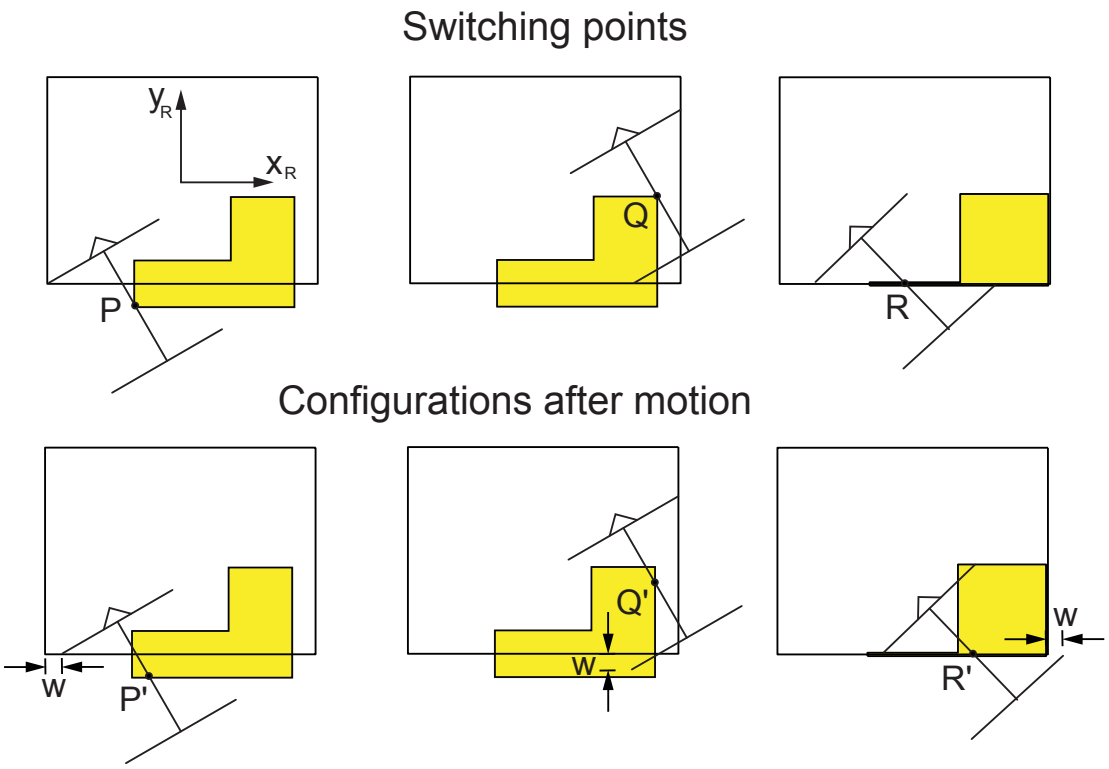


Figure 5.7: Switching points

Open loop execution

We have written a simulator that inputs a user-specified path, executes our control policies and outputs wheel angular velocities. We fed the output to the robot. The intended path (the solid hexagon in Fig.5.9(a)) and two runs of the open-loop execution are shown. The path has large error because slip between the robot and the paper causes a loss of dual-differental drive. The error builds up as there is no feedback. The maximum error between the start and the goal was 16cm. and the angular error was 27^o. The robot took about 3 minutes to complete the path.

Error correction

Since the task system M_t is holonomic, errors in the motion of the paper can be corrected by proportional control. We used the pose information from the camera to servo the paper to its intended path. At every camera update, we computed the paper velocity required to servo to the path. This velocity was fed to the simulator which provided the necessary wheel angular velocities. We applied these velocities until the camera updated the pose again. Fig.5.9(b) shows four runs of the robot. The maximum deviation from the path was

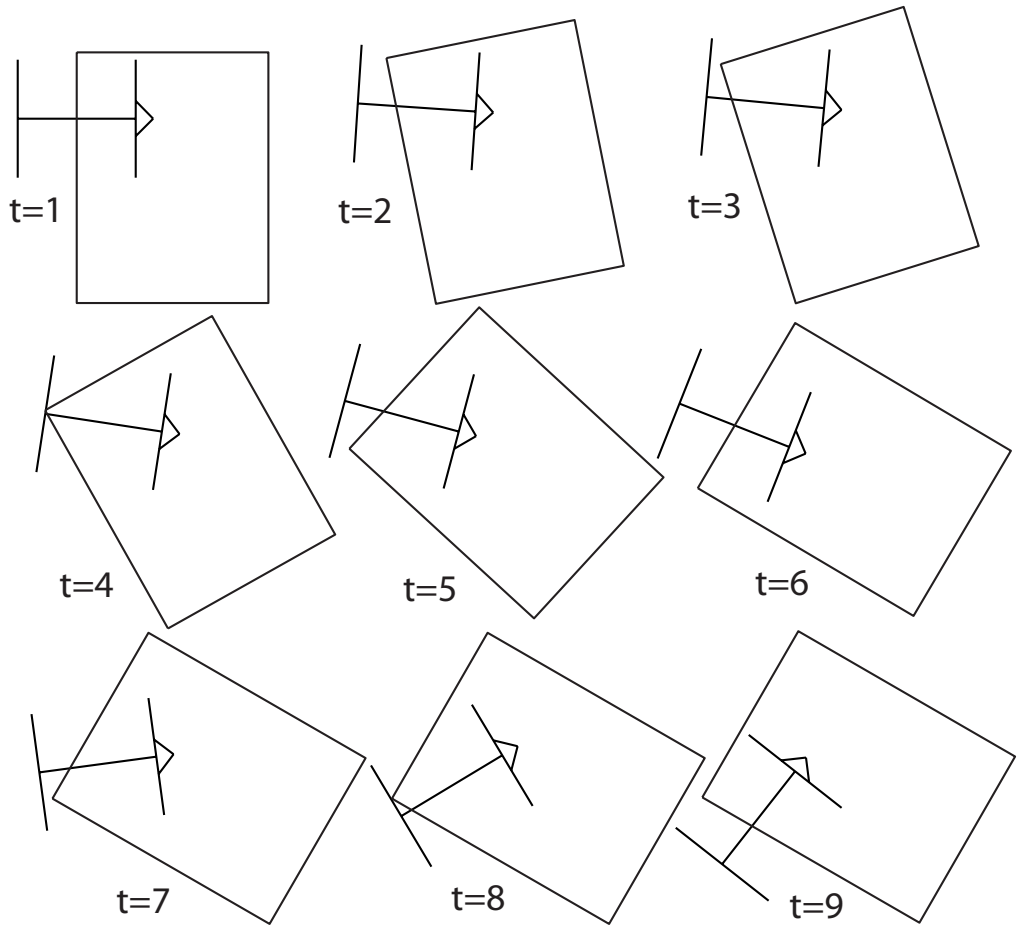
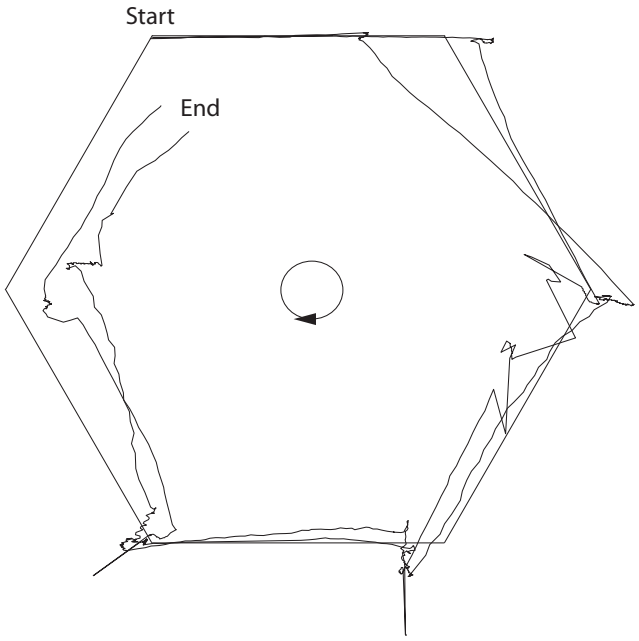
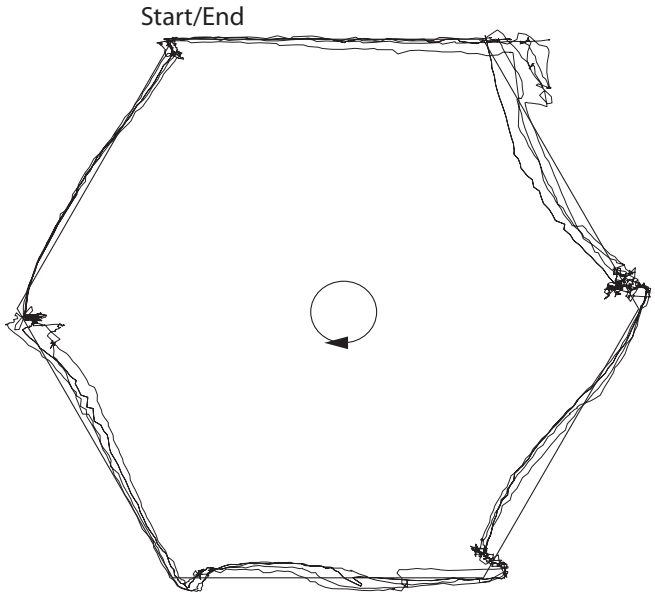


Figure 5.8: Implementation of the control policy to move the paper in a hexagon whos width is 8 times the width of the robot. The paper is rotated by 60^o at each vertex. Shown here is the motion of the robot during one such rotation of the paper.



(a)



(b)

Figure 5.9: (a) Motion of the paper pose for open-loop execution. (b) Motion of the paper with visual feedback.

4.6cm. and the maximum angular error was 4^o. The time to completion was comparable to the open-loop implementation.

Switching points

We ran separate trials to test the accuracy of motion at switching points. We ran 15 tests open-loop for the robot in switching point R. None of the tests were successful. This was because this motion required the wheels to be placed vey close to the edge of the paper. The accumilated slip caused a loss of dual-differential drive mode and the motion failed. When implemented with visual-servoing, 10 out of 15 tests were successful. However, the robot required 18 repetitions of the motion described in §5.7 which took about 1 minute to complete. The resulting error in the location of the paper was 1.4cm.

5.9 Summary

Our method decomposes the system state into task freedoms and shape freedoms and analyzes each subsystem separately. By doing so, we reduce a nonholonomic control problem to two subproblems — one of following a holonomic path in task space and another of controlling the nonholonomic shape freedoms to satisfy constraints. We thus isolate the nonholonomy to a smaller set of state variables. By designing a suitable control policy in shape space we can also ensure that the task space path is followed smoothly with reduced control switchings. Furthermore, since the task system is holonomic, error correction is easy.

The method outlined in §5.6 can be applied to any nonholonomic system. The key lies in the fact that the matrix B is full rank. Given a nonholonomic system that looks like M, we can perform row operations on the A and generate a full rank B. The corresponding state variables can be treated as the task freedoms and the remaining state variables can be treated as the shape freedoms. We can then apply the same theorems outlined in §5.6 to prove STLC and generate control laws.

The selection of policies and paths that minimize control switchings is an exciting future direction of research. Currently, we use the safe policy of using α to servo to L_{mid}. One can imagine a mixed policy that servoes the robot to L_{max} when the robot is close to a corner to increase the clearance between the robot and the paper. It is unclear as to whether there exists a single optimal policy for all possible paths. Another interesting problem arises when the controller has the freedom to choose the path — given a fixed policy, what is the path that minimizes the number of switching points?

Chapter 6

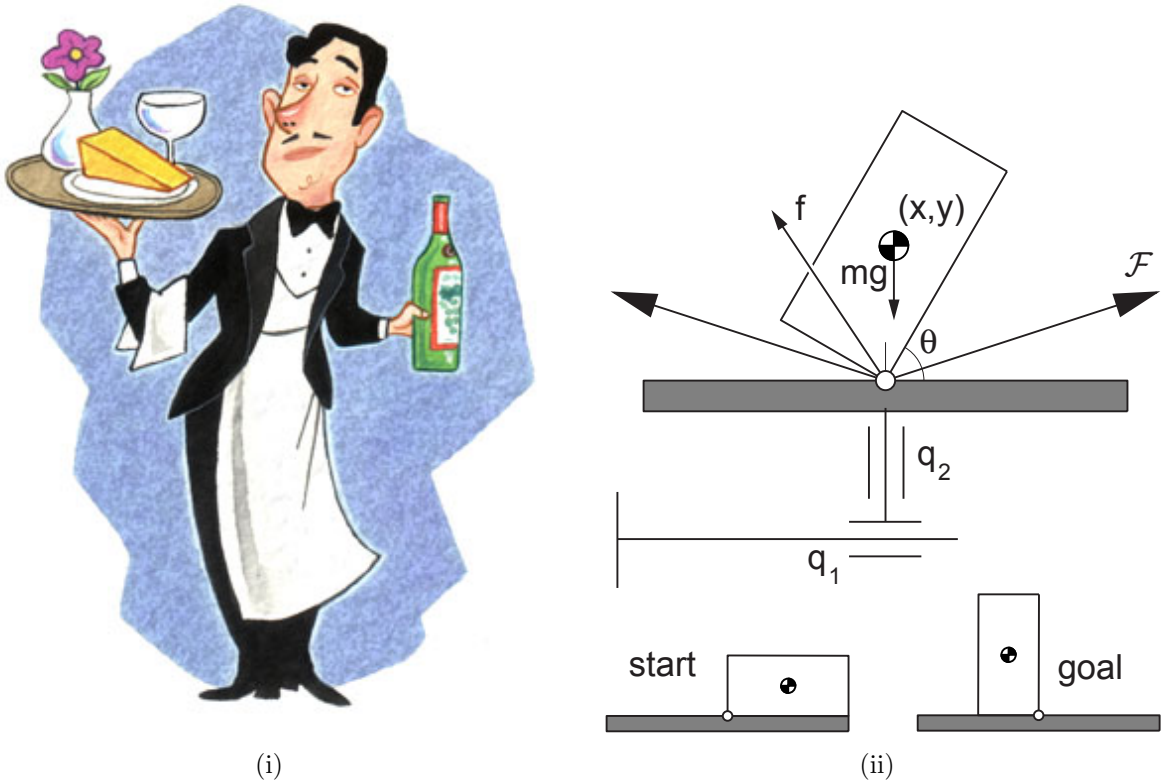
The waiter's problem

In Ch.3, we introduced the technique of decomposing the trajectory generation problem into path planning followed by time-scaling the generated path. As discussed, a drawback of this technique is that, for some problems, finding a candidate path that can be time-scaled is very hard. In this chapter, we explore the trajectory generation of one such problem. We show how we can use the paradigm of task and shape decomposition introduced in Ch.4 to project the system dynamics down to a 2D shape space which allows us to compute analytical trajectories.

Our focus will be on the problem shown in Fig.6.1(i-ii). We will first explain the physical intuition behind the problem and subsequently state the mathematical description of the problem.

Imagine a waiter carrying a wineglass on a tray (Fig.6.1(i)). The wineglass begins to tip. How does the waiter move the tray so that the wineglass is brought back to its vertical configuration? It is intuitively clear that any solution to the problem will be dynamic — the waiter will have to move the tray fast, before the wineglass falls down on its side on the tray. Consequently, limits on how fast the waiter can move his hands will decide whether he can or cannot rescue the wineglass. Furthermore, friction between the wineglass and the tray is another crucial factor which will decide whether or not the waiter's motion causes the wineglass to slide off the tray.

Fig.6.1(ii) shows the mathematical description of the problem. We restrict all motion to be in a vertical plane with gravity pointing down. We approximate the kinematic skeleton of the waiter by a robot arm with two prismatic joints, one each for horizontal and vertical motion. We specify limits on how fast the waiter can move his hands by specifying limits on the accelerations of each joint. The end-effector of the robot is a flat *palm*. We approximate the wineglass by its bounding rectangle and will henceforth refer to it as a *block*. We assume that the block's mass is uniformly distributed and that the mass and mass distribution are time invariant (there is no wine sloshing in or out of the the glass). We denote the angle



(i) (ii)

Figure 6.1: (i): The waiter's problem — a wineglass rests on its side on a flat tray. The goal is to tip the wineglass while keeping the tray horizontal (else the cheesecake will fall) and maintaining rolling contact between the wineglass and the tray, (ii): A robotic version of the problem — the waiter's hands are represented by an arm with two prismatic joints and the wineglass is represented by a block with uniform mass distribution. The angle θ denotes the orientation of the block relative to the palm. The goal is to go from $\theta = 0$ to $\theta = \frac{\pi}{2}$ while maintaining the contact force f within the friction cone $\mathcal{F}$.

made by the block with the palm by θ. The goal is to manipulate the block from $\theta = 0$ to $\theta = \dfrac{\pi}{2}$. The palm cannot rotate. We control the horizontal and vertical acceleration of the palm $(\ddot{q}_1, \ddot{q}_2)$. Furthermore, we need to maintain a rolling contact between the block and the palm. As per the laws of Coulomb friction this means that the contact friction force $\boldsymbol{f}$ must lie within the friction cone at the contact $\mathcal{F}$.

This problem was first proposed by Erdmann [Erdmann, 1998]. There is a solution to the problem in Erdmann's paper, but one that was derived by hand-tuning the controller and the system parameters. One of our goals in this paper is to automate the trajectory planning. The term *waiter's problem* was coined by Kevin Lynch (`http://lims.mech.northwestern.edu/~lynch/projects/index.html`). The problem solved in this paper is also closely related to the work of Lynch and Mason [Lynch and Mason, 1999]. Using a

robot arm with a single revolute degree of freedom, they demonstrated dynamic tasks such as snatching, rolling, throwing and catching an object.

We would also like to answer questions like:

> What is the set of all feasible palm motions that move the block from $\theta = 0$ to $\theta = \frac{\pi}{2}$ in, say, 2.5 seconds, and have the exact same angular motion $\theta(t)$ of the block?

The answer to the question is illustrated in Fig.6.9.

We propose a new technique for solving the trajectory generation problem. We first derive the contact acceleration constraint for the tipping problem using a result from [Srinivasa et al., 2005a]. We identify the subset of the state variables that the constraint depends on. We call these state variables the *shape freedoms* of the system. For the block tipping, the shape freedom is the rotation of the block relative to the palm, the angle θ. We then project the dynamics and the constraints onto the space of shape freedoms. For the block tipping, the state space comprises of the configuration of the block and its velocity, a 6 dimensional space. We project the dynamics and the constraints onto the 2 dimensional space of $(\theta, \dot{\theta})$. We plan a feasible trajectory in the lower dimensional shape freedom space and control the nullspace of the projection to control the reminder of the state variables. By varying the control in the nullspace, we are able to produce various palm motions that do not affect the task freedom trajectory [Srinivasa et al., 2005b]. The family of trajectories generated is shown in Fig.6.9.

A main theme in the task and shape decomposition paradigm is the construction of feedback controllers in the space of shape freedoms which work exclusively to satisfy the constraints on the system. For the waiter's problem, the constraints are the limits on the accelerations of the joints and the Coulomb friction constraint to maintain rolling contact between the block and the palm. In this paper, we develop a feedback controller for the shape freedoms using the numerical technique of backward dynamic programming. We discuss the implementation of the controller in §6.7.

The rest of the chapter is arranged as follows. In §6.1, we introduce the terminology used in the rest of the chapter and give a clear definition of the waiter's problem. §6.5 describes the solution of the planning problem. §6.7 describes the feedback controller. In §6.8, we discuss our contributions and future work.

6.1 Problem statement

We describe the configuration of the block by its pose $\boldsymbol{q_o} = (x, y, \theta)$, where (x, y) is the location of the center of mass and θ is the angle between the block and the palm, and the configuration of the robot by its joint variables $\boldsymbol{q_r} = (q_1, q_2)$. We choose our joint variables

in such a way that the location of the contact point on the palm is also (q_1, q_2). We denote the configuration of the system by $\boldsymbol{q} = (\boldsymbol{q_o}, \boldsymbol{q_r})$.

Since the contact between the block and the palm is a rolling contact, the velocity of the block at the contact point must be equal to the velocity of the palm at the contact point:

$$\mathsf{G}(\boldsymbol{q_o})^{\mathsf{T}}\dot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{J}(\boldsymbol{q_r})\dot{\boldsymbol{q}}_{\boldsymbol{r}} \tag{6.1}$$

where $\mathsf{G}(\boldsymbol{q_o}) \in \mathbb{R}^{3\times 2}$ is the grasp map which relates contact forces to wrenches on the object and $\mathsf{J}(\boldsymbol{q_r}) \in \mathbb{R}^{2\times 2}$ is the Jacobian of the robot.

For our problem, G is given by:

$$\mathsf{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ d\sin(\theta+\beta) & -d\cos(\theta+\beta) \end{bmatrix} \tag{6.2}$$

where $\beta = \arctan(w/l)$, w and l are the width and length of the block respectively and d is the distance from the center of mass of the block to the contact point.

Since our robot is comprised of two prismatic joints, J is merely the identity matrix given by:

$$\mathsf{J} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.3}$$

Coulomb friction imposes a constraint on the contact force $\boldsymbol{f}$:

$$\boldsymbol{f} \in \mathcal{F}(\mu) \tag{6.4}$$

where $\mathcal{F}(\mu)$, the friction cone, is a convex cone and μ is the coefficient of friction between the palm and the object, a material property. The span representation of the friction cone is given by:

$$\mathcal{F}(\mu) = \begin{bmatrix} \mu & -\mu \\ 1 & 1 \end{bmatrix} \boldsymbol{\lambda} \qquad \boldsymbol{\lambda} \geq 0 \tag{6.5}$$

The motion of the object is governed by its dynamics:

$$\mathsf{M}\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{G}\boldsymbol{f} + \boldsymbol{n_o} \tag{6.6}$$

where M is the inertia matrix of the object and $\boldsymbol{n_o}$ is the wrench on the object due to gravity:

$$\mathsf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & \frac{m}{3}d^2 \end{bmatrix} \tag{6.7}$$

$$\boldsymbol{n_o} = \begin{pmatrix} 0 \\ -mg \\ 0 \end{pmatrix} \tag{6.8}$$

The problem can be stated as:

> Given a start $\boldsymbol{q_s}$ and a goal $\boldsymbol{q_g}$ configuration for the system, find the robot joint acceleration $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}(t)$ that will move the system from the start to the goal without violating the contact velocity constraint (Eqn.6.1) or the contact force constraint (Eqn.6.4).

An industrial robot typically accepts desired joint torque as an input. The mapping from robot joint acceleration to joint torque involves the dynamics of the robot and is a well studied problem [Craig, 1989].

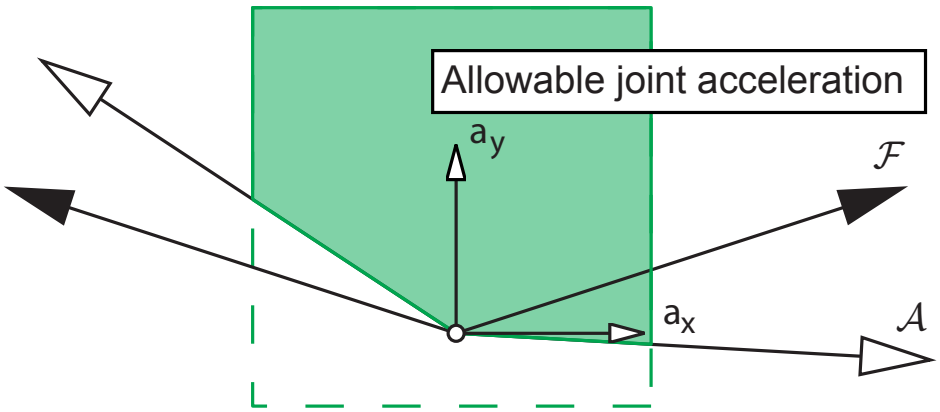


Figure 6.2: Constraints in contact acceleration space: the dark polygon denotes the set of feasible contact accelerations. The friction cone $\mathcal{F}$ is mapped to the acceleration cone $\mathcal{A}$.

6.2 The contact acceleration constraint

In §2, we derived a constraint that the robot joint acceleration has to satisfy in order to obtain both a desired motion of the object and a desired contact condition. We restate the theorem in the following paragraph for convenience.

Theorem 6. *For a given configuration* $\boldsymbol{q} = [\boldsymbol{q_o}, \boldsymbol{q_r}]^T$ *and velocity* $\dot{\boldsymbol{q}} = [\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}]^T$*, the allowable joint acceleration of the robot* $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}$ *that satisfies both the velocity and the force constraint is constrained to lie within the cone:*

$$\boxed{\mathsf{J}\ddot{\boldsymbol{q}}_{\boldsymbol{r}} + \mathsf{V}(\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) \in \mathcal{A}} \tag{6.9}$$

$$\mathsf{V}(\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}}, \boldsymbol{n_o}) = \dot{\mathsf{J}}\dot{\boldsymbol{q}}_{\boldsymbol{r}} - \dot{\mathsf{G}}^{\mathsf{T}}\dot{\boldsymbol{q}}_{\boldsymbol{o}} - \mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\boldsymbol{n_o}$$

and

$$\mathcal{A} = \mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G}(\mathcal{F})$$

is the contact acceleration cone.

The theorem gives us a series of mappings that can be used to combine the contact force constraint and the contact velocity constraint in a common space. The contact friction cone $\mathcal{F}$ is mapped to a contact acceleration cone $\mathcal{A}$ and the contact velocity constraint is mapped to a contact acceleration constraint. Limits on the accelerations of the joints can be incorporated into this mapping.

When we apply Thm.6 to the tipping problem, we obtain the following constraints on the allowable joint acceleration:

$$\begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} - d \begin{pmatrix} \cos(\theta+\beta) \\ \sin(\theta+\beta) \end{pmatrix} \dot{\theta}^2 + \begin{pmatrix} 0 \\ g \end{pmatrix} \in \mathcal{A}(\theta) \tag{6.10}$$

For a given state of the system, an illustration of Eqn.6.10 in the contact acceleration space is shown in Fig.6.2. The rays with solid arrows represent the friction cone at the contact and the rays with outlined arrows represent the acceleration cone at the contact. The bounds that we have imposed on the joint accelerations of the robot appear as a rectangle in the contact acceleration space. The dynamics of the block, as manifested by the terms with velocity produts, shift the rectangle from the origin. Feasible contact accelerations lie in the intersection of the contact acceleration cone and the rectangle of joint acceleration limits. It is important to note that the combined constraint polygon has contributing edges from both the friction constraint and the joint acceleration limits. We will discuss the influence of the acceleration limits on the space of feasible solutions for the problem in §6.4.

The origin of the contact acceleration space is the mapping of the origin of the contact force space via the contact map. It corresponds to the situation where there is no contact force, *i.e.*where the block is falling freely under the influence of gravity and the robot is tracking the motion of the contact point. In the absence of rotational velocity for the block, this corresponds to the robot moving vertically downwards with an acceleration of g.

6.3 Task and shape decomposition

Recall that in the task and shape decomposition paradigm, the important observation is the fact that constraints reside in a space which has a lower dimension compared to the system state space.

For our problem, the state space of the object is $(\boldsymbol{q_o}, \dot{\boldsymbol{q}}_{\boldsymbol{o}})$, a 6D space. We note that the constraint (Eqn.6.10) depends only on $(\theta, \dot{\theta})$, and not on (x, y) or $(\dot{x}, \dot{y})$. This suggests a natural decomposition of the state variables into the shape freedoms $(\theta, \dot{\theta})$ on whom the constraint on the system depends on, and the task freedoms $(x, y, \dot{x}, \dot{y})$ which do not play a part in constraint satisfaction.

In the subsequent paragraphs, we will find the maps that project the system dynamics onto the space of shape and task freedoms.

They dynamics of the block are given by:

$$\mathsf{M}\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{G}\boldsymbol{f} + \boldsymbol{n_o} \tag{6.11}$$

The contact map takes us from the space of contact forces to the space of contact accelerations:

$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} = (\mathsf{G}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G})f \tag{6.12}$$

where (a_x, a_y) is the contact acceleration.

In the absence of internal forces the contact map is invertible. This makes intuitive sense since an internal force is a contact force that produces no contact acceleration and by definition lies in the nullspace of the map from contact forces to contact accelerations. If no internal forces exist, the map has no nullspace and is invertible. The proof is as follows:

Theorem 7. *In the absence of internal forces, the contact map is invertible, i.e.*

$$\mathcal{N}(G) = \{\mathbf{0}\} \implies \mathcal{N}(\mathsf{G}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}) = \{\mathbf{0}\} \tag{6.13}$$

Proof. We will prove the theorem by contradiction. We claim that there exists an $\boldsymbol{x} \neq \mathbf{0}$ such that

$$\mathsf{G}^\mathsf{T}\mathsf{M}^{-1}\mathsf{G}\boldsymbol{x} = \mathbf{0}$$

Then

$$\begin{aligned} & \boldsymbol{x}^{\boldsymbol{T}}\mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G}\boldsymbol{x} && = 0 \\ \implies & |\mathsf{M}^{-\frac{1}{2}}\mathsf{G}\boldsymbol{x}| && = 0 \\ \implies & \mathsf{M}^{-\frac{1}{2}}\mathsf{G}\boldsymbol{x} && = \mathbf{0} \\ \implies & \mathsf{G}\boldsymbol{x} && = \mathbf{0} \\ \implies & \boldsymbol{x} && = \mathbf{0} \end{aligned}$$

since $\mathcal{N}(G) = \{\mathbf{0}\}$. This is a contradiction of our earlier claim. □

For the waiter's problem $\mathcal{N}(G) = \{\mathbf{0}\}$ *i.e.*the system has no internal forces. We can, hence, invert the map to obtain:

$$\boldsymbol{f} = (\mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G})^{-1}\begin{pmatrix} a_x \\ a_y \end{pmatrix} \tag{6.14}$$

We rewrite the dynamics of the block as:

$$\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{M}^{-1}(\mathsf{G}\boldsymbol{f}_{\boldsymbol{c}} + \boldsymbol{n}_{\boldsymbol{o}}) \tag{6.15}$$

We combine Eqn.6.14 and Eqn.6.15 as:

$$\ddot{\boldsymbol{q}}_{\boldsymbol{o}} = \mathsf{A}\begin{pmatrix} a_x \\ a_y \end{pmatrix} + \boldsymbol{b} \tag{6.16}$$

where

$$\mathsf{A} = \mathsf{M}^{-1}\mathsf{G}(\mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G})^{-1} \tag{6.17}$$

and

$$\boldsymbol{b} = \mathsf{M}^{-1}\boldsymbol{n}_{\boldsymbol{o}} \tag{6.18}$$

Hence the evolution of the system is given by Eqn.6.16 subject to the constraint given by Eqn.6.10.

Let us take a moment to parse Eqn.6.17. If we blindly transmit the inverse into the bracketed terms using the product rule, we will end up with:

$$\mathsf{A} = \mathsf{G}^{-\mathsf{T}} \tag{6.19}$$

However, there is no guarantee that $\mathsf{G}^{-\mathsf{T}}$ exists. In fact, for the waiter's problem, G^{T} is not invertible. The physical intuition behind this statement is as follows: G^{T} is the map from object twist (generalized velocity) to contact velocity. The fact that G^{T} is non

invertible means that it has a nullspace. In a physical sense, this means that there exists a motion of the block that does not produce any velocity at the contact point. Sure enough, a rotation of the block about the contact point does not produce any contact velocity but still produces a motion in the block.

Since we cannot invert G^T, we do the next best thing, which is generate a *pseudoinverse* — an inverse with an extra constraint placed on the nullspace to generate a unique answer. What we have in Eqn.6.17 is a *weighted pseudoinverse* of G^T with the inertia matrix M used as the weight matrix.

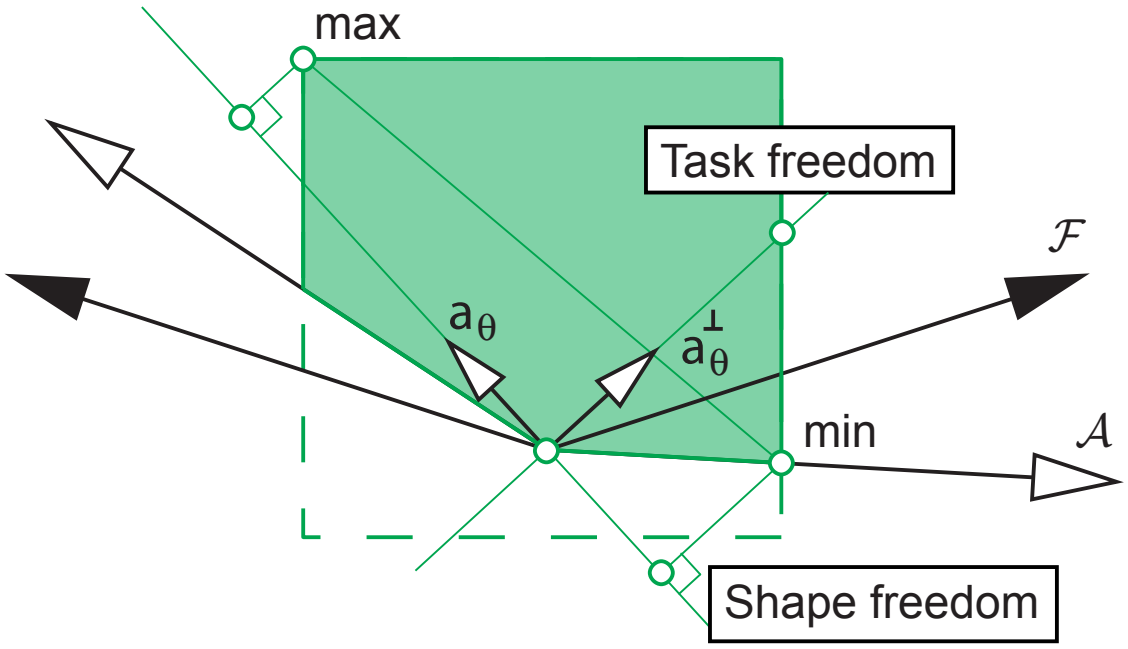


Figure 6.3: Projecting the contact acceleration constraint onto and orthogonal to the space of task freedoms

Our goal is to project the system dynamics given by Eqn.6.16 along and perpendicular to the constraint given by Eqn.6.10. We do this in two stages. In the first stage, we rotate the contact acceleration frame to a new frame $(\boldsymbol{a_\theta}, \boldsymbol{a_\theta^\perp})$ such that any control applied along $\boldsymbol{a_\theta^\perp}$ does *not* affect the motion of the shape freedoms $(\theta, \dot{\theta})$.

To obtain $\boldsymbol{a_\theta}$, we rewrite Eqn.6.16 as:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{pmatrix} = \begin{bmatrix} \mathsf{A}_{\mathsf{xy}} \\ \mathsf{A}_\theta \end{bmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{bmatrix} \boldsymbol{b_{xy}} \\ b_\theta \end{bmatrix} \tag{6.20}$$

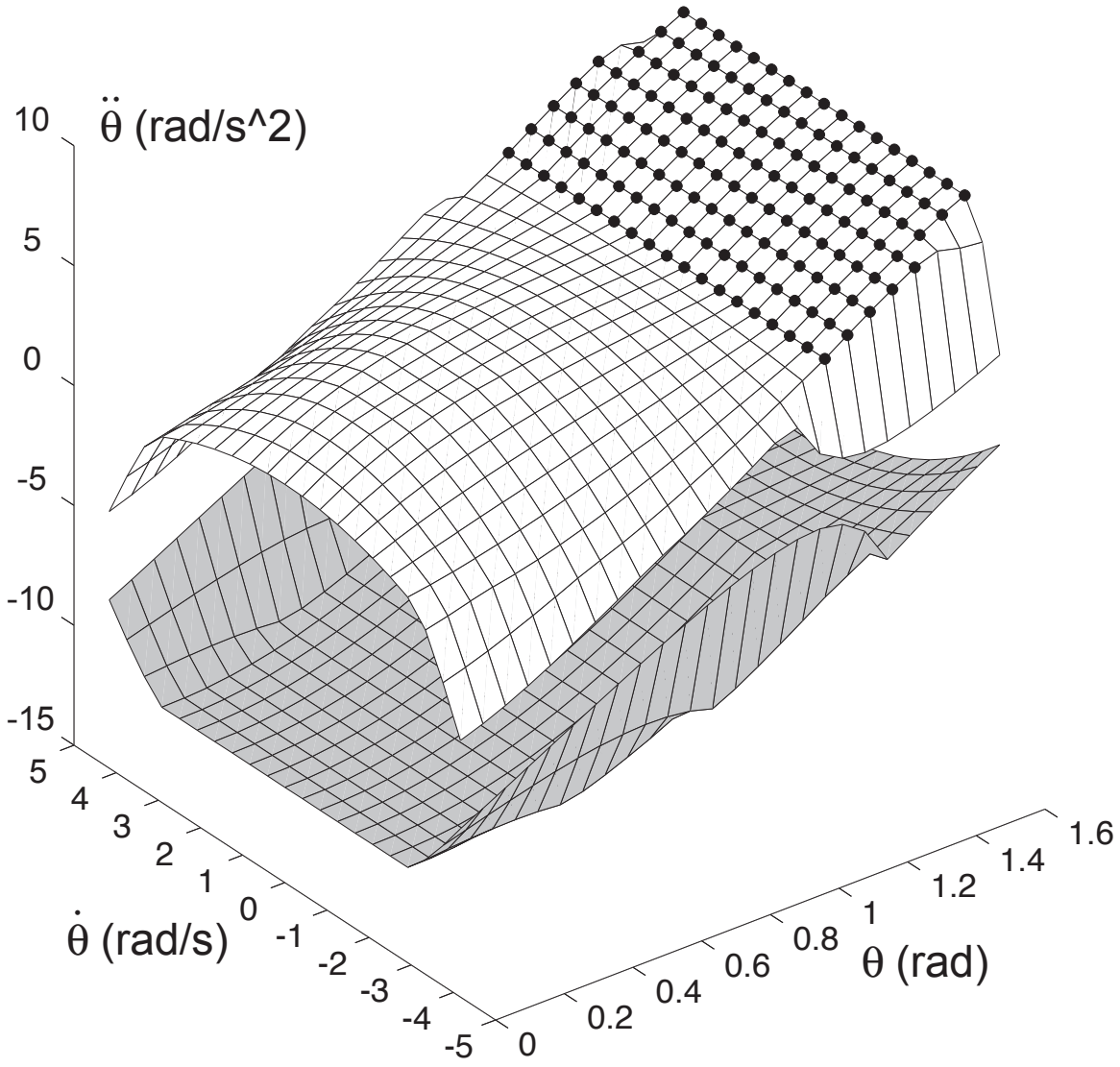


Figure 6.4: The volume of feasible angular accelerations for an actuator acceleration limit of $10m/s^2$. The constraint surfaces are generated by sampling $(\theta, \dot{\theta})$ space uniformly. The dots indicate samples where the actuator constraint is active.

We then perform a transformation of the input by rotating it along and orthogonal to $\boldsymbol{a_\theta}$:

$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} = \boldsymbol{a_\theta}\alpha + \boldsymbol{a_\theta^\perp}\beta \tag{6.21}$$

We compute the orthogonal unit vectors $\boldsymbol{a_\theta}$ and $\boldsymbol{a_\theta^\perp}$ using the additional relation that $\boldsymbol{a_\theta^\perp}$ does not affect the evolution of $\ddot{\theta}$, *i.e.*$\mathsf{A}_\theta \boldsymbol{a_\theta^\perp} = 0$.

We can now decouple the evolution of the task freedoms and the rest of the state variables as:

$$\ddot{\theta} = \mathsf{A}_\theta \boldsymbol{a_\theta^T}\alpha + b_\theta \tag{6.22}$$

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \mathsf{A}_1\alpha + \mathsf{A}_2\beta + \boldsymbol{b_{xy}} \tag{6.23}$$

We can now project the contact acceleration constraint onto the space of task freedoms. The projection is illustrated in Fig.6.3. As mentioned in the previous subsection, the dark polygon in Fig.6.3 corresponds to set the feasible contact accelerations. Each point in this set can be mapped to a feasible $\ddot{\theta}$ using Eqn.6.16. In Fig.6.3, this mapping is illustrated by a projection onto the unit vector $\boldsymbol{a_\theta}$. Limits on the feasible $\ddot{\theta}$ can be easily computed by projecting the vertices of the feasible polygon onto $\boldsymbol{a_\theta}$, as shown in the figure.

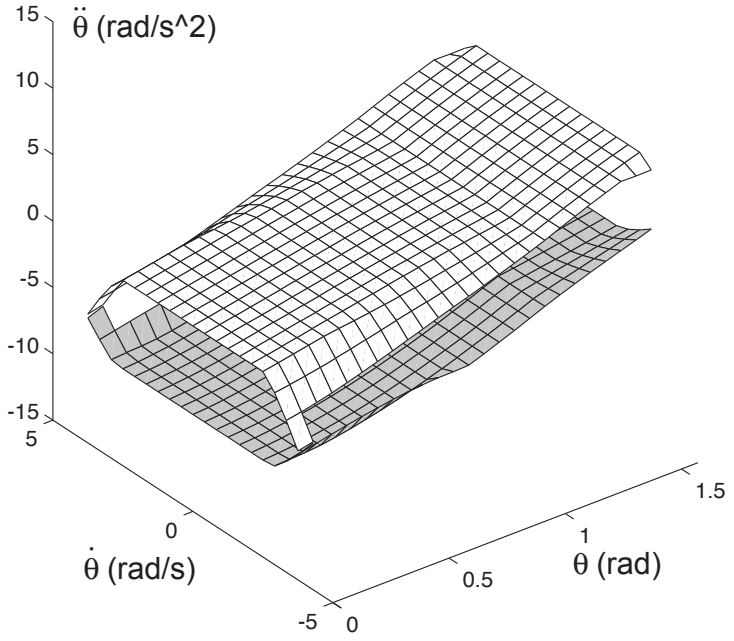
We compute limits on α by projecting the vertices of the feasible polygon onto $\boldsymbol{a_\theta}$. We use the limits on α to compute limits on $\ddot{\theta}$ using Eqn.6.22.

6.4 Dependence on acceleration limits

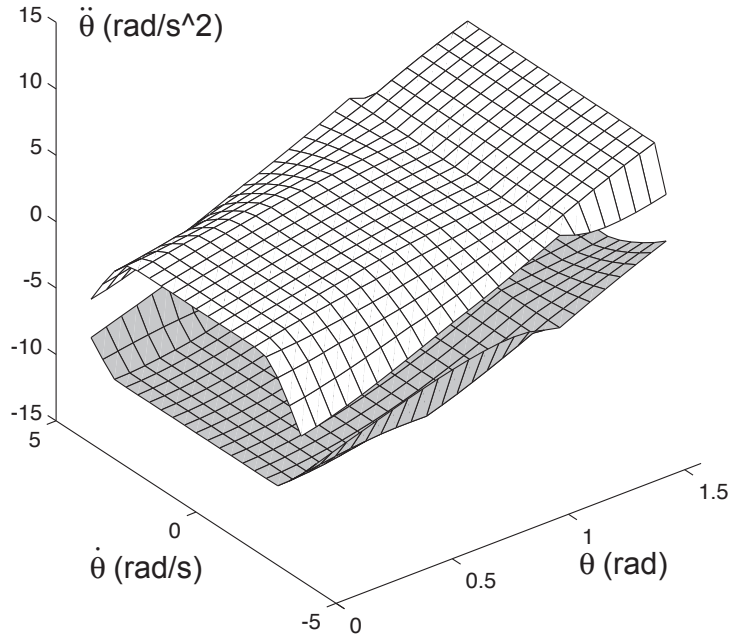
Our motivation for deriving the contact acceleration constraint was the claim that actuator constraints and Coulomb friction constraints were *both* important for dynamic manipulation. Fig.6.3 justifies our claim for the waiter's problem. In the figure, the maximum and the minimum allowable α are illustrated. Notice that the minimum depends on the Coulomb friction constraint while the maximum depends on the actuator limit.

The importance of both constraints is further illustrated in Fig.6.4. The figure plots the feasible phase volume for the shape freedoms for a uniform sampling of the $(\theta, \dot{\theta})$ space. The mesh is constructed by first computing the maximum and the minimum α for a given sample and then computing the maximum and minimum $\ddot{\theta}$ using Eqn.6.22. Black dots on the grid points indicate samples where the actuator constraint is active. It is clear that the constraint surfaces generated depend both on the actuator limit and the friction limit.

In reality, the constraint surfaces intersect at the sides forming a tube. The gaps in the sides of the surface in Fig.6.4 and in all of the following phase volume plots are an artifact of the sampling.

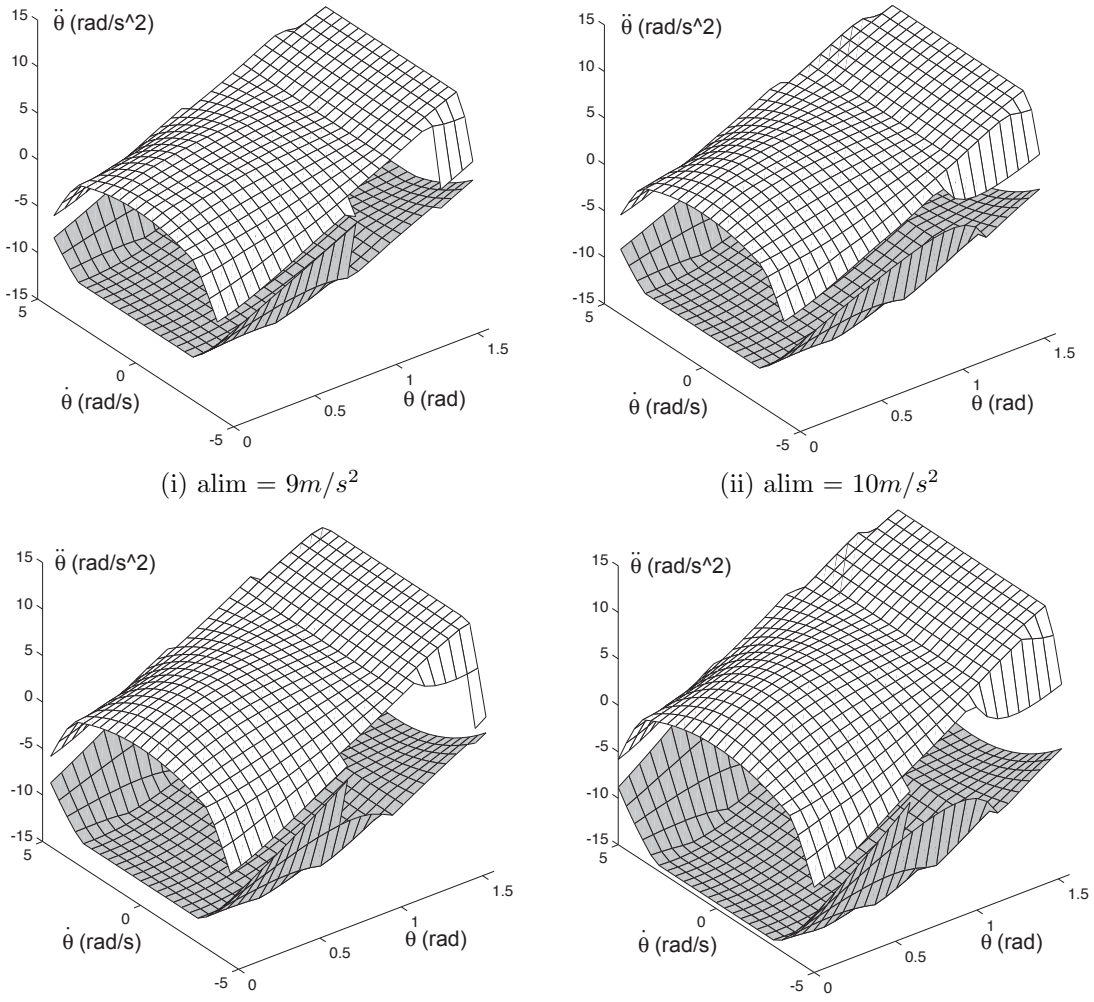


(i) alim = $5m/s^2$



(ii) alim = $7m/s^2$

Figure 6.5: The volume of feasible angular accelerations. The grid denotes the sampling of $(\theta, \dot{\theta})$ used to construct the plots. For each sample, the maximum and minimum $\ddot{\theta}$ are computed. The light and dark surfaces are tessellations of the maxima and the minima, respectively. (i) Plot for an acceleration limit of $5m/s^2$. (ii) Plot for an acceleration limit of $7m/s^2$.



(i) alim = $9m/s^2$ (ii) alim = $10m/s^2$

(iii) alim = $12m/s^2$ (iv) alim = $15m/s^2$

Figure 6.6: The volume of feasible angular accelerations for varying acceleration limits.

To further explore the role the acceleration limits play in the waiter's problem, we computed the feasible phase volumes for various acceleration limits. Fig.6.5 shows the phase volumes for an acceleration limit of $5m/s^2$ and $7m/s^2$ in Fig.6.5(i) and Fig.6.5(ii), respectively. It can be shown that in both cases the block *cannot* be manipulated from $\theta = 0$ to $\theta = \frac{\pi}{2}$ — the robot is too slow to achieve the required motion. As the actuator limit is increased, the feasible volume swells proportionally, as shown in Fig.6.6(i-iv).

6.5 Motion planning

In this section, we describe an analytical solution to the problem of generating trajectories for the waiter's problem. We exploit the fact that the constraints of the system reside in

the lower dimensional shape space by first planning a feasible trajectory in this space. At each point of this feasible trajectory, we have independent control over the orthogonal task freedoms. We use this freedom to control the motion of the palm.

Planning in shape space

We now focus our attention on the evolution of the task freedoms given by Eqn.6.22. Limits on $\ddot{\theta}$ constrain the tangent space of the task freedom $(\theta, \dot{\theta})$, as shown in Fig.6.7. At every point of a feasible trajectory in the task freedom space, the tangent must lie within the cone of allowable tangents at that point. Note that Fig.6.4 and Fig.6.7 convey the same information. The maximal and minimal constraint surfaces of Fig.6.4 appear as the extremals of the feasible tangents in Fig.6.7.

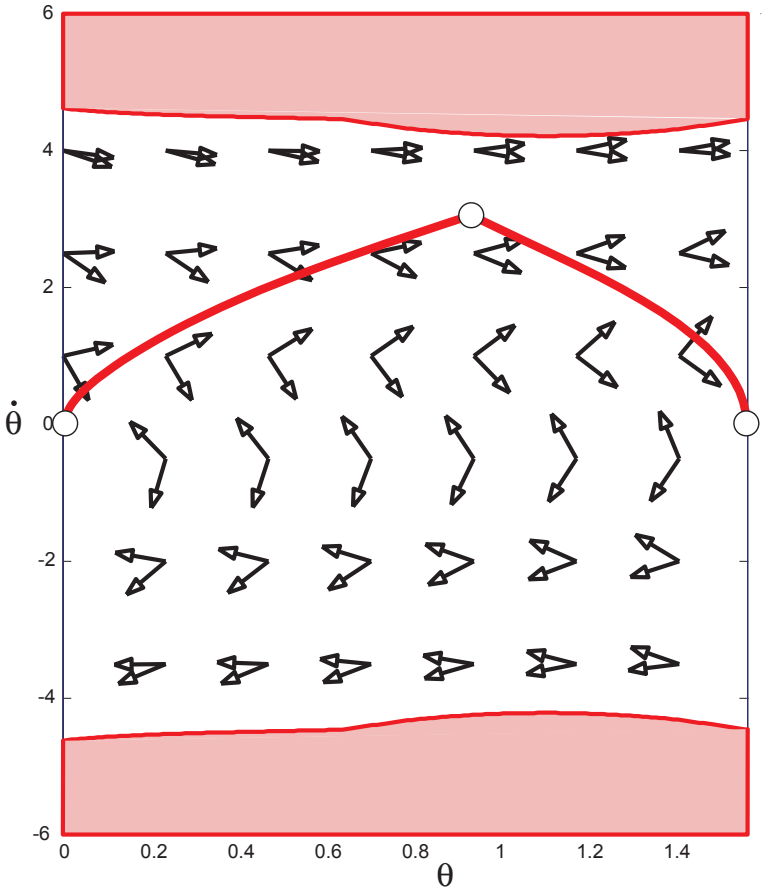


Figure 6.7: A phase plot of the shape freedoms — limits on $\ddot{\theta}$ appear as constraints on the tangent space. A time-optimal feasible trajectory in phase space is also shown. Circles indicate the start location, the switching point and the goal.

We choose a simple technique for finding feasible trajectories. We pick a feasible vector

field and follow it from the start (here, $\theta = 0$). We pick another feasible vector field and back project it from the goal (here, $\theta = \frac{\pi}{2}$). At the point where the two trajectories intersect, we switch vector fields. If the vector fields used are the extremals, then we obtain the time-optimal trajectory. The time-optimal trajectory for the shape freedoms is shown in Fig.6.7. It must be noted that planning, even in 2D, need not always be so easy. For example, the forward and backward trajectories might intersect the limits of the feasible space instead of intersecting each other. In such cases, more complicated planning is required. We refer the reader to [Bobrow et al., 1985, Shin and McKay, 1985] for two solutions to the problem.

One drawback of the above trajectory generation strategy is that there is a discontinuity in the control at the instant the vector fields switch. If the robot has a limit on the rate of change of its acceleration (its *jerk*), then this is not possible. To mitigate this, we truncate the forward and the backward trajectories a user defined time-step before they intersect and merge the two trajectories using a cubic spline. This changes the acceleration profile close to the switch from a step to a ramp, smoothing the trajectory.

The task freedoms

Once a feasible trajectory for the shape freedoms is computed using the technique described in the previous subsection, we can compute the controls $\alpha(t)$ that are required to move the shape freedoms along the desired trajectory using Eqn.6.22. At each point in this trajectory, there is an orthogonal task freedom that can be varied without affecting the trajectory. The control for this freedom lies along the unit vector $\boldsymbol{a}_{\boldsymbol{\theta}}^{\perp}$. Fig.6.8 illustrates the computation of the allowable range for the orthogonal control.

The plot in Fig.6.8 shows the range of the allowable orthogonal control as a function of time. The figure on the top right marked I is a snapshot of the contact acceleration constraint at time $t = 0.5s$.. It shows the polygon of feasible accelerations as well as the $\boldsymbol{a}_{\boldsymbol{\theta}}$ and the $\boldsymbol{a}_{\boldsymbol{\theta}}^{\perp}$ directions. Since we have already computed the shape freedom trajectory, we know the magnitude of the required control along $\boldsymbol{a}_{\boldsymbol{\theta}}$, namely $\alpha(0.5)$. This magnitude is shown as the line segment OA. To find the range of the task freedom control, we erect the perpendicular $\boldsymbol{a}_{\boldsymbol{\theta}}^{\perp}$ at A. The intersection of this perpendicular with the polygon of feasible accelerations, shown as BF, gives us the range of the allowable task freedom control — $\beta(0.5)$ ranges from AB to AF. We repeat this procedure for all t to obtain the $\beta - t$ plot in Fig.6.8.

Notice that there are discontinuities in the slopes of the $\beta(t)$ limits. Two such discontinuities are shown in the figures on the top left marked II and III. Evidently, these discontinuities occur when $\boldsymbol{a}_{\boldsymbol{\theta}}^{\perp}$ passes through a vertex of the feasible polygon, as shown. Another slope discontinuity, shown as IV, occurs due to the patching of the two extremal trajectories using the cubic spline. In this case, the acceleration in-between the dark circles

is approximated to be piecewise linear, due to the spline.

System trajectories

To construct trajectories for the entire system, we need to select $\beta(t)$ from the allowable range of β computed as mentioned in the previous subsection. Any feasible selection of $\beta(t)$, together with the $\alpha(t)$, constitutes a feasible control for the entire system which will, by construction, have the same angular motion of the block as any other feasible selection of $\beta(t)$.

We use a blend of the extremal βs to construct the feasible $\beta(t)$. We define a fraction p such that:

$$\beta_p(t) = (1-p)\beta_{\min} + (p)\beta_{\max} \tag{6.24}$$

A sample of the $\beta_p(t)$ are shown in both the $\beta - t$ plot in Fig.6.8 and the $t = 0.5s$. snapshot.

The system trajectories with the various p are shown in Fig.6.9. For the waiter's problem, choosing $p = 1$ corresponds to choosing the largest feasible acceleration of the palm in the vertical direction for the given $\alpha(t)$. Likewise, choosing $p = 0$ corresponds to choosing the smallest feasible acceleration of the palm in the vertical direction for the given $\alpha(t)$. Hence the two system trajectories for $p = 0$ and $p = 1$ act as bounds for the allowable motion of the palm for the given $\alpha(t)$ — any feasible palm configuration must lie within palm configurations of these two trajectories.

Fig.6.10 is a a finer resolution version of Fig.6.9. Instead of plotting the actual motion of the palm and the block, we have shown the path of the block in configuration space. This allows us to reduce clutter and show the configuration space surface of feasible trajectories for a given $\alpha(t)$.

The decomposition provides us a controllable simulation. For example, in the waiter's problem, if a user were handed two joysticks, one for each degree of freedom of the palm and told to tip the block, he would invariably fail — the block would very easily slide, or lose contact with the palm. Instead, we can give the user control of $\beta(t)$ with the assurance that any value of $\beta(t)$ will satisfy the contact constraints. We have solved the hard problem (tipping the block) and given the control of the freedoms of secondary importance (the motion of the palm) to the user.

6.6 Sliding

Until now we have looked at plans that only involve rolling at the contact between the block and the palm. For the tipping problem, it is relatively easy to incorporate the sliding of the palm as well. This is predominantly because the contact between the block and the palm is of *Type A* (a contact between a vertex of the block and a side of the palm

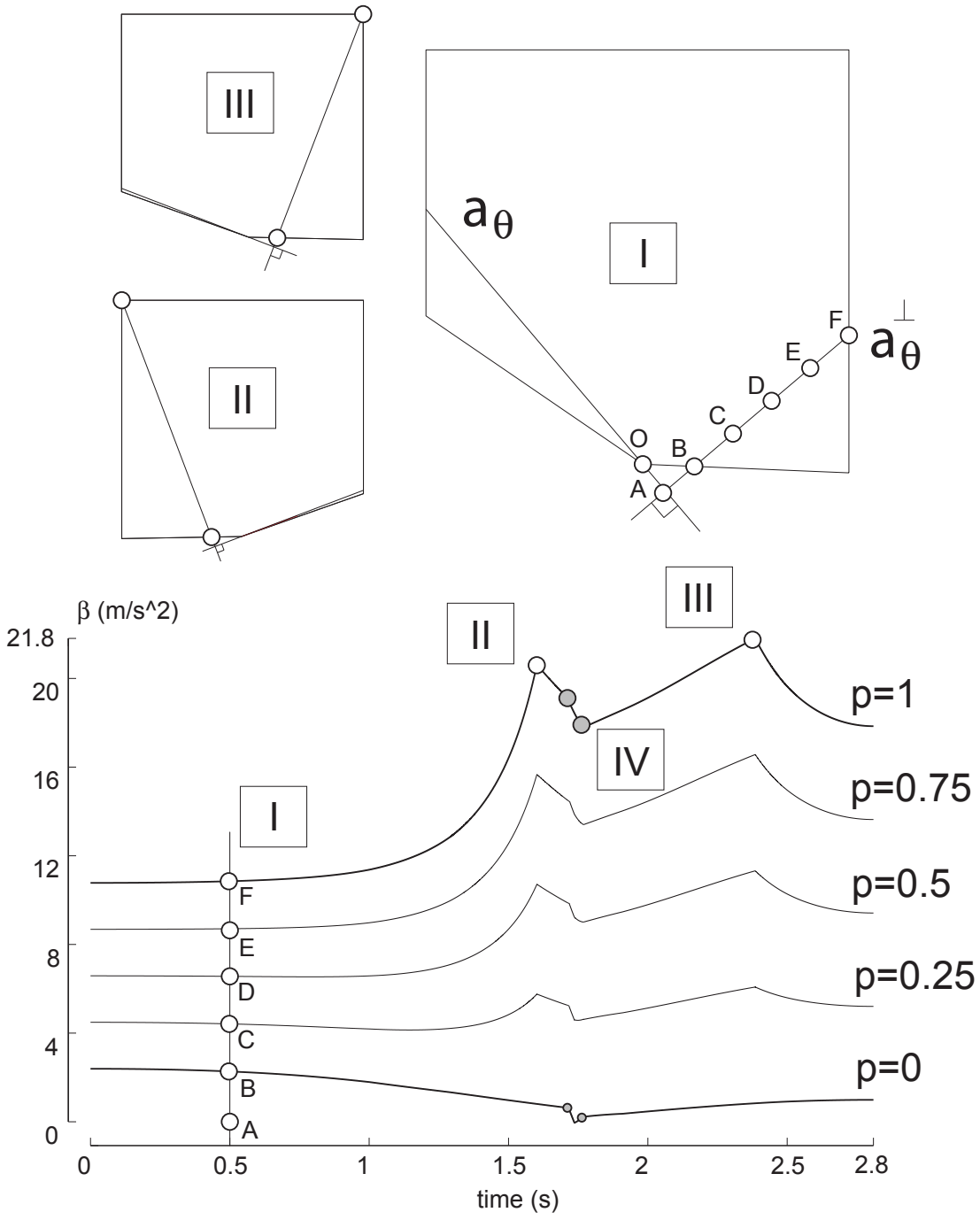


Figure 6.8: Limits on the acceleration of the task freedoms

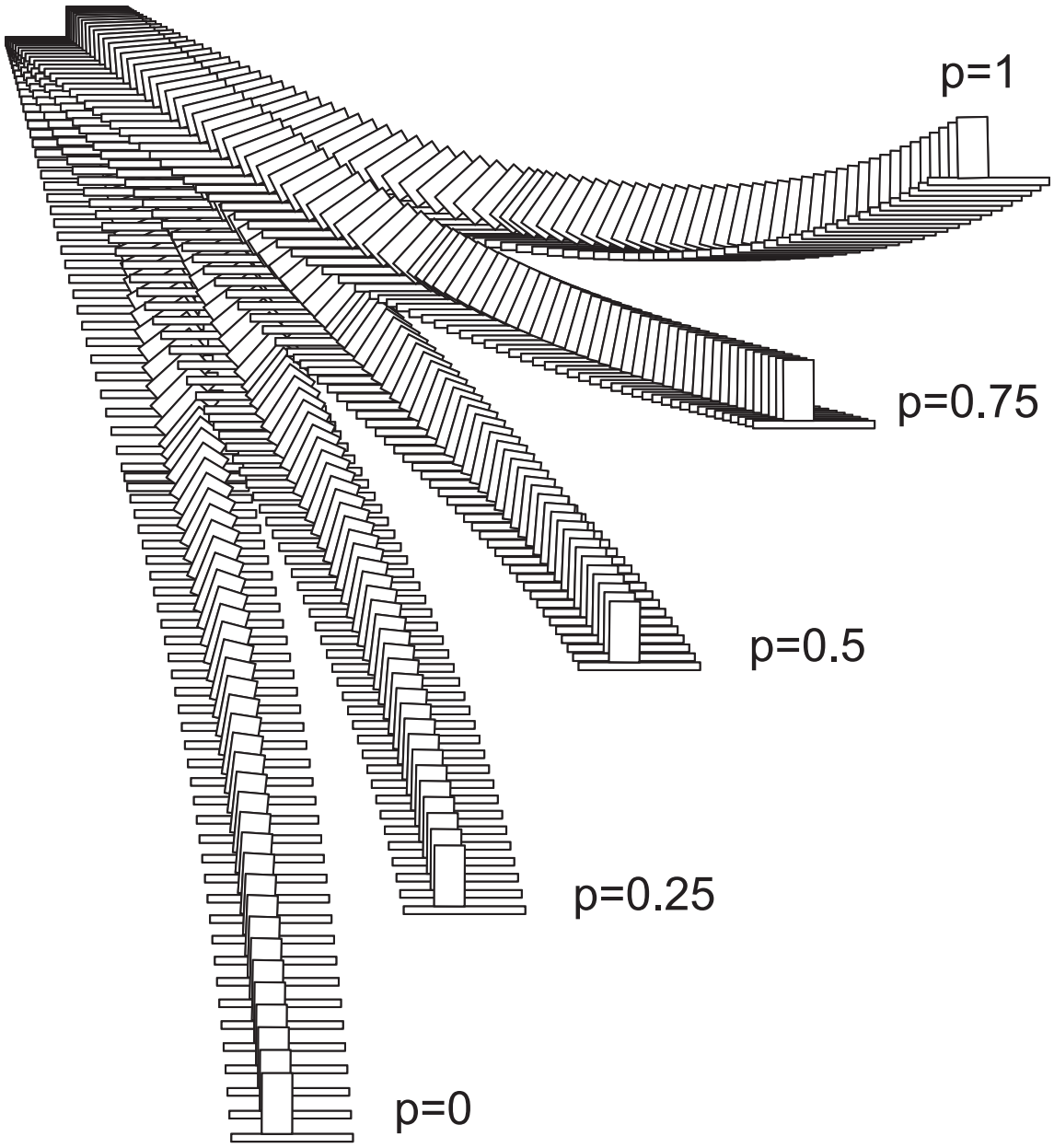


Figure 6.9: A sampling of the set of all feasible palm motions that move the block from $\theta = 0$ to $\theta = \frac{\pi}{2}$ in 2.5 seconds and have the exact same angular motion $\theta(t)$ of the block

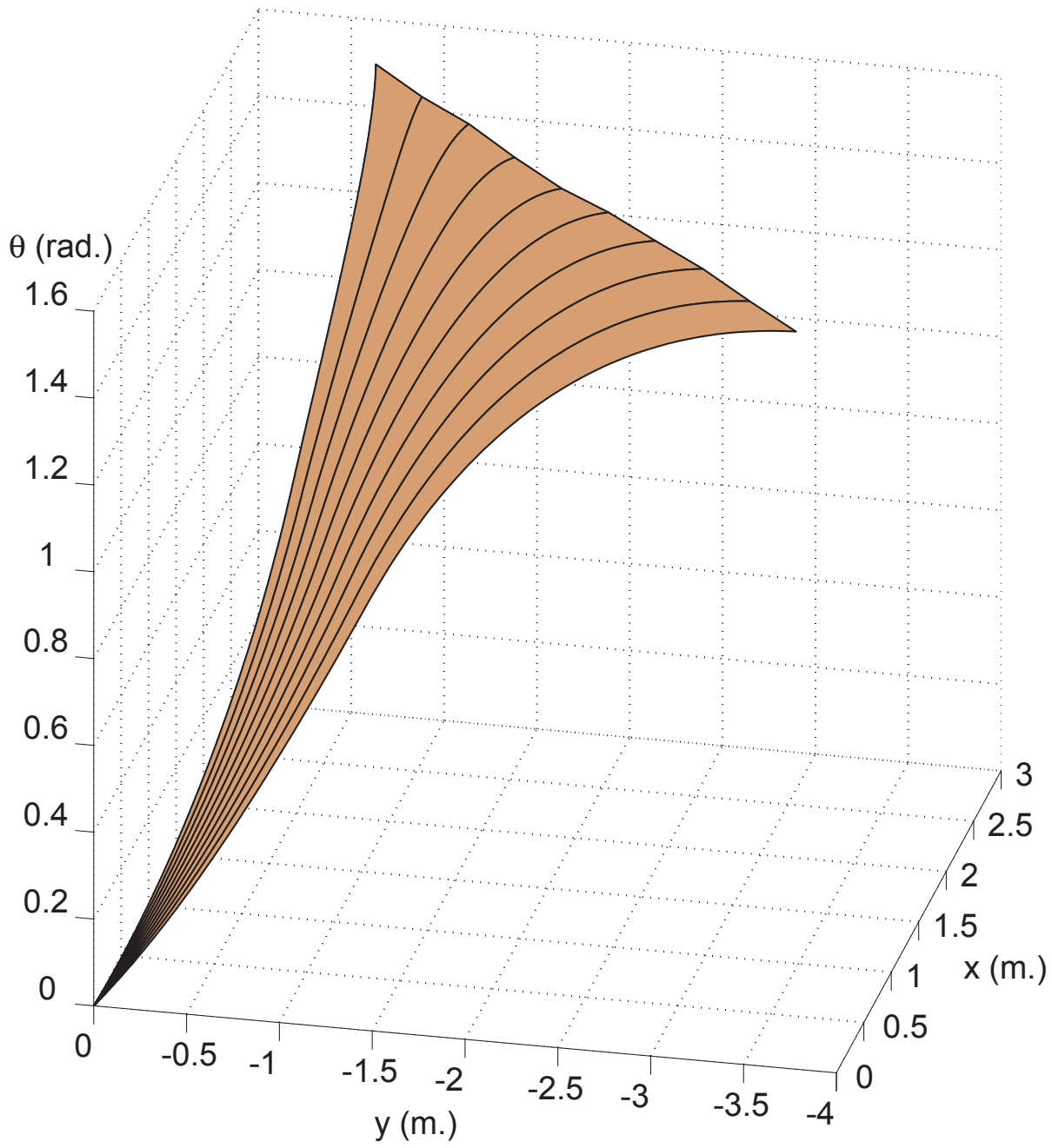


Figure 6.10: The surface of feasible system trajectories, for a given $\alpha(t)$, parametrised by $p \in [0, 1]$ projected onto the system configuration space

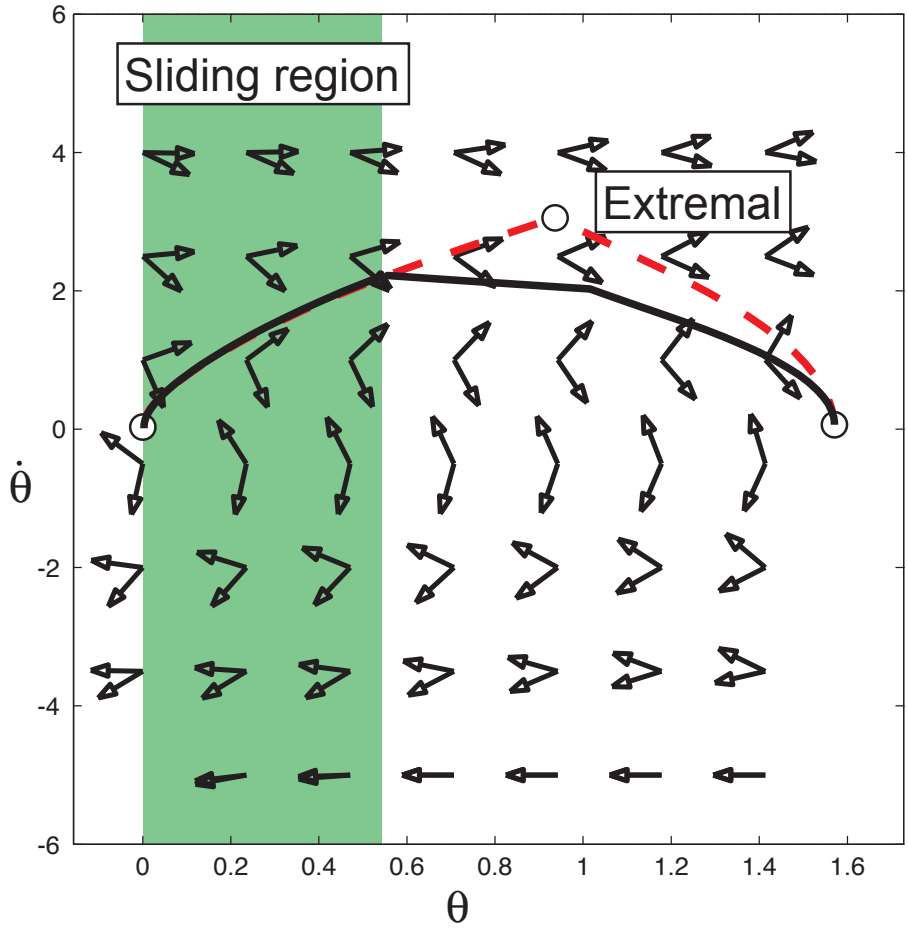


Figure 6.11: A trajectory that involves both sliding and rolling at the contact. The block slides in the sliding region and reverts back to rolling out of the region.

[Lozano-Pérez, 1983]). As a result, the moment applied by any contact force on the block by the palm remains the same regardless of the position of the block relative to the palm. This is illustrated in Fig.6.12. Hence sliding the palm relative to the block does not affect the motion of the task freedoms.

An instance of a plan involving sliding and rolling is shown in Fig.6.11. The dotted line denotes the extremal trajectory — one where sliding of the palm must occur. The extremal is computed by following the limits of the allowable tangents at each point of of the state space. The solid line denotes a trajectory where both rolling and sliding occur. When the solid line coincides with the extremal, we are allowed to slide the palm relative to the block (the allowable region is colored in the plot). However we need to ensure that by the time the trajectory reaches the end of the allowable region, the sliding must cease — the relative velocity of the palm with respect to the block must be brought to zero. At the end of the

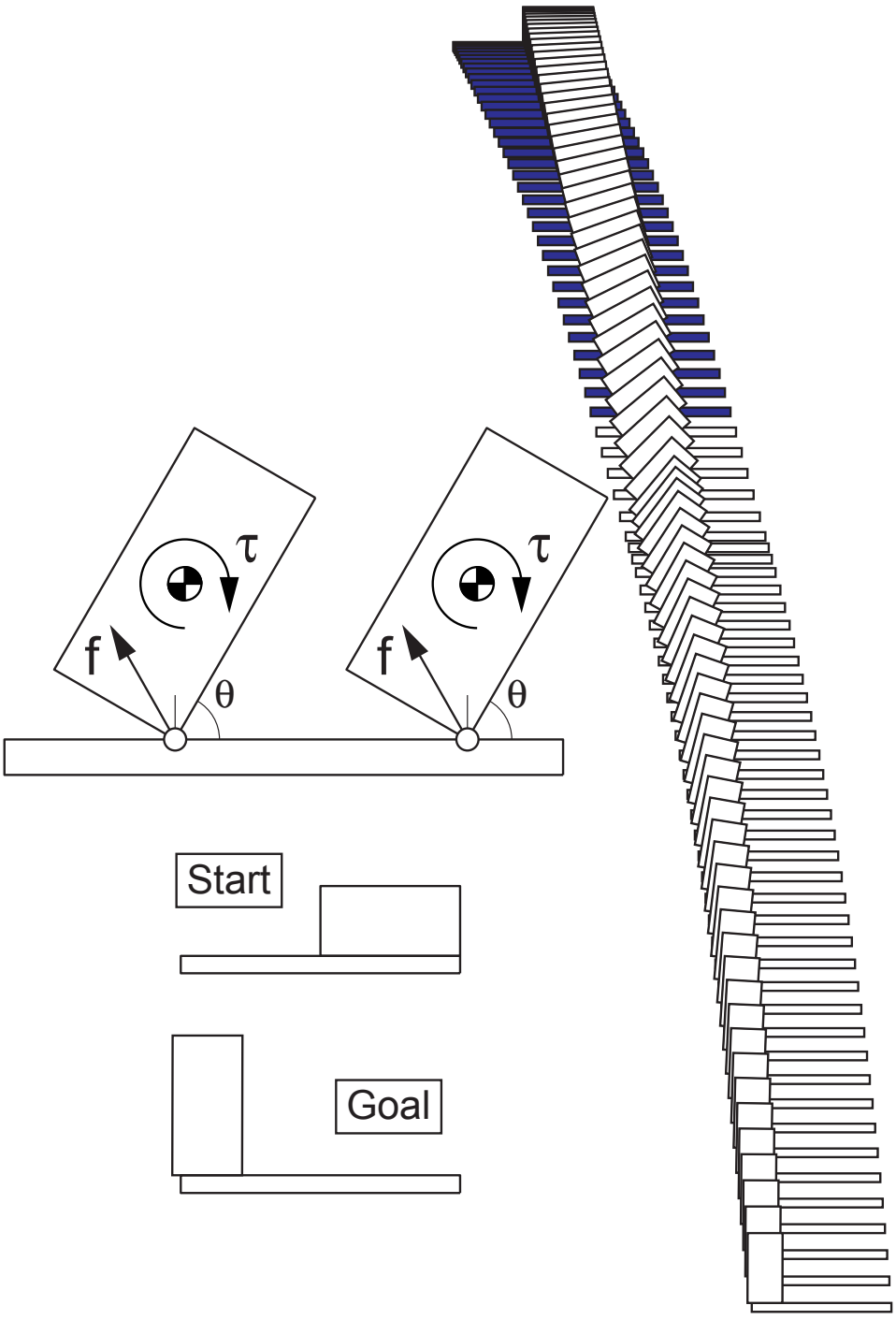


Figure 6.12: An illustration of a Type A contact and a palm motion that results in both sliding (shown by the dark palms) and rolling at the contact

allowable region, the solid line leaves the extremal and rolling contact is initiated. The block then maintains rolling contact until it reaches the goal.

The motion of the palm for the trajectory given in Fig.6.11 is shown in Fig.6.12. The times where the palm is dark denote sliding. For this problem, we used a simple constant acceleration followed by a constant deceleration for the motion of the palm relative to the block during the sliding phase.

6.7 Feedback control for shape freedoms

The main theme of the task and shape decomposition paradigm is the importance of constraint satisfaction. The basic idea is that, given a choice, it is less undesirable for the system to have an error in the task freedoms rather than have it fail completely due to an error in the shape freedoms. Carrying over the idea to the waiter's problem, we will be more tolerant to positioning error for the palm than to error that causes the block to slip on the palm or lose contact. This is because once the constraint is broken, the system transitions to a completely new dynamic regime (sliding instead of rolling for the block) where accomplishing the task might be impossible.

A second advantage of writing a controller for just the shape freedoms is that the task freedoms are still unconstrained and can be controlled at will without affecting the controller, due to the orthogonal projection.

A third advantage is the reduction of the dimensionality of the space in which the controller resides. For the waiter's problem, we will use the numerical technique of backwards dynamic programming to construct a feedback controller. Albeit a powerful technique, one drawback of dynamic programming is that the running time grows exponentially with the number of dimensions. By choosing the low dimensional shape space to write our controller in, we can obtain greater speed and higher resolution for dynamic programming.

We will follow this theme and write a feedback controller for the shape freedoms of the waiter's problem. The controller will ensure that the contact acceleration constraints are not broken in case there is control noise.

Background

The optimal control problem deals with the generation of trajectories and controls for a system that minimize a given cost function. The work of Bellman [Bellman, 1957] pioneered the use of dynamic programming to solve optimal control problems. Bellman proposed a numerical technique for finding the optimal cost from any point in a state space to a goal. The gradient of the optimal cost function can be used to generate the corresponding optimal feedback controller.

A related area of research, kinodynamic planning, focusses on the motion planning problem of a robot subject to both kinematic and dynamic constraints. Most of the early research on kinodynamic planning focussed on planning a time-optimal motion for a point mass in a 2D or 3D obstacle field undergoing Newtonian dynamics subject to velocity and acceleration bounds.

Donald *et al.*[Donald et al., 1993] provided an approximation algorithm that ran in polynomial time. The fundamental idea of their approach was to convert the continuous trajectory generation problem into a discrete graph search. They achieved this by constructing a uniform grid in state space with the spacing equal to applying a constant acceleration for a given time interval. The acceleration and the time interval were chosen to touch, but not break, the velocity and acceleration bounds. They then posed the trajectory generation problem as a graph search in the discrete state space from the grid point nearest to the start to the grid point nearest to the goal.

While the philosophy of the approach is identical to dynamic programming, the key contribution of Donald *et al.*was their insight into the selection of the discretization parameters. For the point mass problem, they were able to relate the granularity of the state space discretization and the choice of the time interval to a speed-dependent obstacle avoidance margin. However, it is unclear how this relation can be extended to problems with more complicated dynamics.

Most of the problems for which dynamic programming has been used to obtain numerical solutions can be formulated as deterministic discrete-time variational control problems. Continuous-time variational control problems can be treated by assuming that the control is piecewise constant and making appropriate transformations to the discrete-time case.

In the subsequent paragraphs, we provide a brief description of the dynamic programming technique. We refer the reader to Bellman's book [Bellman, 1957] for a more thorough treatment.

The general case of the discrete-time problem is formulated as follows.

Given:

1. A system described by a nonlinear difference equation

$$\boldsymbol{x}(k+1) = \boldsymbol{\Phi}[\boldsymbol{x}(k), \boldsymbol{u}(k), k] \tag{6.25}$$

where $\boldsymbol{x}$ is an n-dimensional state vector, $\boldsymbol{u}$ is an m-dimensional control, k is an index for the stage variable, and $\boldsymbol{\Phi}$ is an n-dimensional vector functional.

2. A variational performance criterion

$$J = \sum_{k=0}^{K} L[\boldsymbol{x}(k), \boldsymbol{u}(k), k] \tag{6.26}$$

where J is the total cost and L is the cost for a single stage.

3. Constraints

$$\boldsymbol{x} \in \boldsymbol{X}(k) \tag{6.27}$$

$$\boldsymbol{u} \in \boldsymbol{U}(\boldsymbol{x}, k) \tag{6.28}$$

where $\boldsymbol{X}(k)$ is a set of admissible states at stage k, and $\boldsymbol{U}(\boldsymbol{x}, k)$ is a set of admissible controls at state $\boldsymbol{x}$, stage k.

4. An initial state

$$\boldsymbol{x}(0) = \boldsymbol{s} \tag{6.29}$$

Find:

The control sequence $\boldsymbol{u}(0), \cdots, \boldsymbol{u}(k)$ such that J is minimized subject to the system equation, the constraints and the initial condition.

If we denote by $I(\boldsymbol{x}, k)$ the minimum cost for state $\boldsymbol{x}$ at stage k, Bellman's principle of optimality can be written as:

$$I(\boldsymbol{x}, k) = \min_{\boldsymbol{u}} \{L(\boldsymbol{x}, \boldsymbol{u}, k) + I[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u}, k), k+1]\} \tag{6.30}$$

It states that the minimum cost for state $\boldsymbol{x}$ at stage k is found by choosing the control that minimizes the sum of the cost to be paid at the present stage and the minimum cost in going to the end from the stage $K+1$ which results from applying this control. The optimal control at state $\boldsymbol{x}$ and stage k, denoted by $\hat{\boldsymbol{u}}(\boldsymbol{x}, k)$ is directly obtained as the value of $\boldsymbol{u}$ for which the minimum in Eqn.6.30 is attained.

Since Eqn.6.30 determines $I(\boldsymbol{x}, k)$ and $\hat{\boldsymbol{u}}(\boldsymbol{x}, k)$ in terms of $I(\boldsymbol{x}, k+1)$, it must be solved backwards in k. The optimization over a sequence of controls is thus reduced to a sequence of optimizations over a single control. Note that the minimum cost at the next state, $I[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{u}, k-1), k]$ is found by linearly interpolating the values of $I(\boldsymbol{x}, k)$ at the quantized states.

The same procedure can be used to compute the optimal control sequence for *any* initial state starting at *any* stage. If the initial state is not a quantized state, an additional interpolation might be necessary. Thus, the solution to many optimization problems is

obtained in the same calculation as for the original problem. Bellman called this *invariant embedding*.

Feedback controller

A feedback controller provides the optimal control to be applied at any state of the system that will result in a trajectory that minimizes a given cost function. We used backward dynamic programming to construct a feedback controller for the waiter's problem. The cost function we optimized was time-optimality, *i.e.*the feedback controller gave us paths that took the least amount of time to get to the goal.

We discretize the $(\theta, \dot{\theta})$ space with a uniform grid. A sampling of the grid is shown in Fig.6.13. The grid points are much closer in reality (there are 99 grid points in between consecutive horizontal grid points and 49 grid points in between consecutive vertical grid points) but we do not show all of them, to reduce clutter. At each grid point, we compute the contact acceleration constraint and compute the allowable α. We divide the range of allowable α into 10 equally spaced controls and apply these constant controls for a time-step of $1ms$. The orbits of the shape freedoms on the application of the controls is shown in Fig.6.13. The goal region is shown as the dark rectangle.

The output of the dynamic program is an optimal cost function that gives the optimal cost (here, minimum time) to get from any point in the state space to the goal. The optimal control is one that moves the trajectory in a direction that is *closest* to the gradient of the optimal cost function. The level sets of the optimal cost function and a time-optimal trajectory from $\theta = 0$ to $\theta = \frac{\pi}{2}$ are shown in Fig.6.14.

6.8 Summary

In this chapter, we have demonstrated a new technique for solving the trajectory generation problem for dynamic contact manipulation. We have shown how the projected dynamics can be used to generate analytical solutions and controllable simulations — where the user can control a set of the freedoms of the system without violating the dynamic constraints. We have also presented the construction of feedback controllers for the shape freedoms using the numerical technique of dynamic programming.

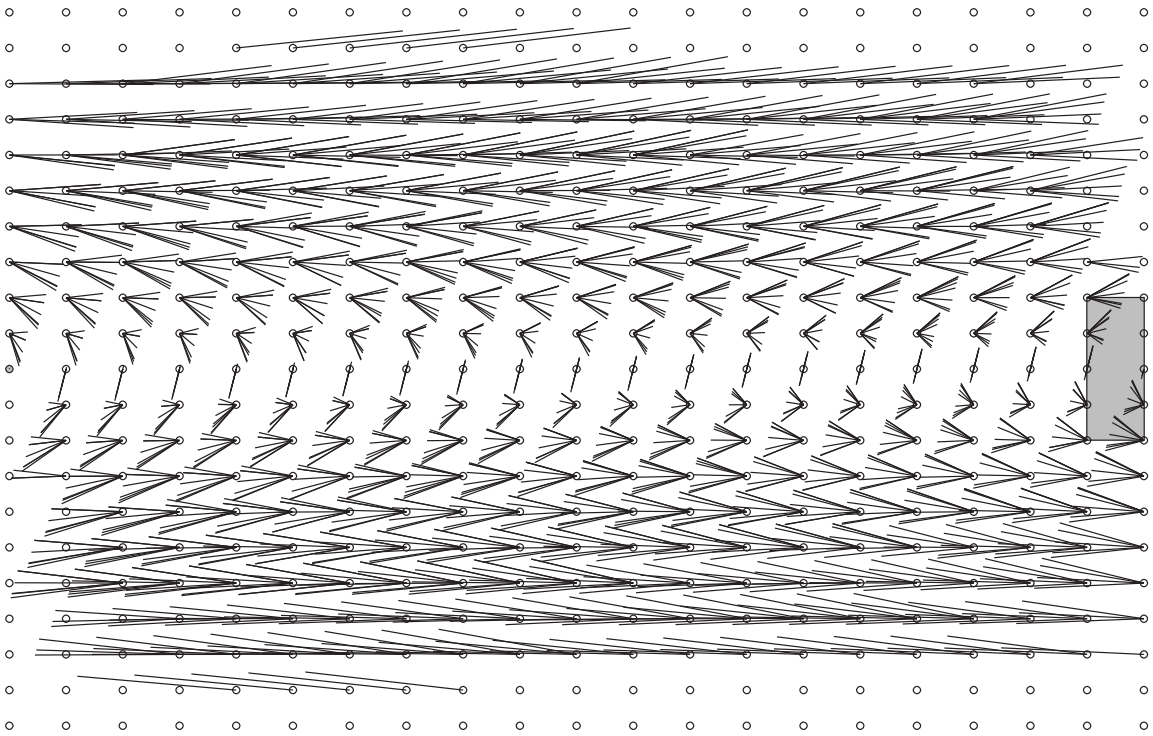


Figure 6.13: Discretization for dynamic programming: the circles denote discrete states. At each discrete state we choose 10 feasible controls. Each curve at a state is an orbit of the system over one time-step with a piecewise constant discrete control. There are 10 curves at each discrete state. The goal region is denoted by the dark rectangle.

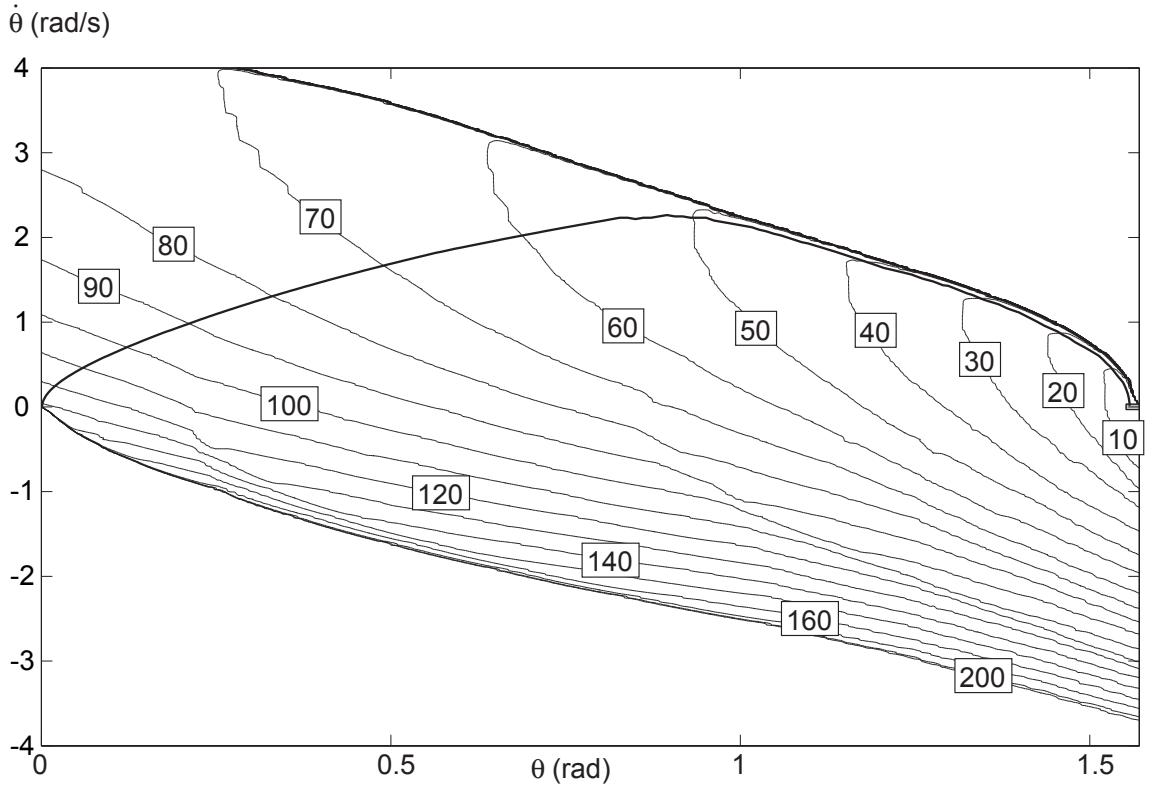


Figure 6.14: Feedback control: isocontours of the time-optimal cost-to-go function and their values in milliseconds are shown. A time-optimal trajectory from $\theta = 0$ is shown as a thick curve. The trajectory takes $139.21ms.$ to reach the goal.

Chapter 7

Conclusion

7.1 Contributions

This thesis explores the planning and control of dynamic contact manipulation. Our main contributions are:

- *The contact acceleration constraint*: We have shown that both actuator limits and Coulomb friction constraints are important for dynamic manipulation. In §2, we have derived the contact acceleration constraint, a mapping of both sets of constraints into a unified space and have shown, for the first time, how both these constraints interact to decide feasible system trajectories.

- *The task and shape decomposition paradigm*: While the contact acceleration constraint provides an instantaneous solution to the set of feasible controls, computing feasible trajectories using this information is still a hard problem. To solve this problem, we have introduced the task and shape decomposition paradigm in §4. While the contact acceleration constraint is a specific contribution to dynamic manipulation, the task and shape decomposition paradigm is a broader contribution to constrained dynamical systems. We use this paradigm to derived *analytical* solutions for a constrained nonholonomic control problem (§5) and a dynamic manipulation problem (§6).

7.2 Future directions

In this section, we explore how our techniques can be used in areas like humanoid robots and computer graphics to generate robust dynamic trajectories. Since this is future work, we shall provide the germ of the ideas and some preliminary results.

Walking

Walking is dual to manipulation. Just like a hand manipulates an object to move it, a walker manipulates the ground to move itself. Human beings are extremely versatile walkers — we can walk on rough terrain, slippery terrain and even unknown terrain (when we are walking in the dark). An important requirement of walking is the maintenance of a fixed contact between the foot and the ground. Hence, an important question to answer is:

> What are the controls that can be applied to a walking robot that generate a stable gait while maintaining a fixed contact mode between the walker and the ground?

To illustrate the use of our techniques, we will derive the contact acceleration constraint for a simple walking robot, in fact, the *Simplest Walking Model* proposed by Garcia *et al.*[Garcia et al., 1998]. The model, shown in Fig.7.1, comprises of two planar *legs* joined at the *hip* by a revolute joint. There are point masses at the tips of the legs and at the hip. The leg that is in contact with the ground is called the *stance leg* while the leg that is distal from the ground is called the *swing leg*. During a walk cycle, the swing leg swings from behind the stance leg to in front of it and impacts the ground with an inelastic collision. Instantaneous with the impact, the stance leg lifts off the ground and the two legs switch roles.

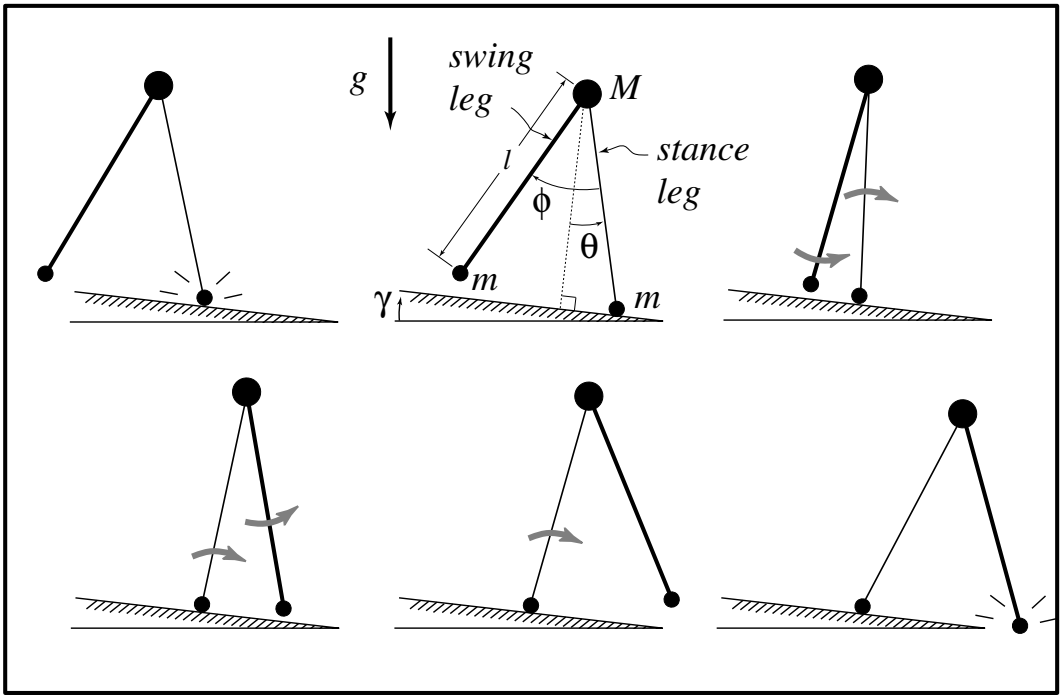


Figure 7.1: The Simplest Walking Model, reproduced from [Garcia et al., 1998]

Garcia *et al.*analyzed this model as a passive dynamic walker, one with no actuation or control. Research on passive dynamic walking was pioneered by McGeer [McGeer, 1990] who built numerous unactuated machines that, when set into motion on top of a gentle slope, would walk down the slope using just their natural dynamics. Any kinetic energy lost during the inelastic impact with the ground is recovered from gravity.

To draw a parallel with manipulation, we place this model in a slightly different setting. The walker is placed on an articulated palm. The palm is allowed to move horizontally and vertically, but is not allowed to rotate. The goal is to compute palm motions that make the walker walk on the palm.

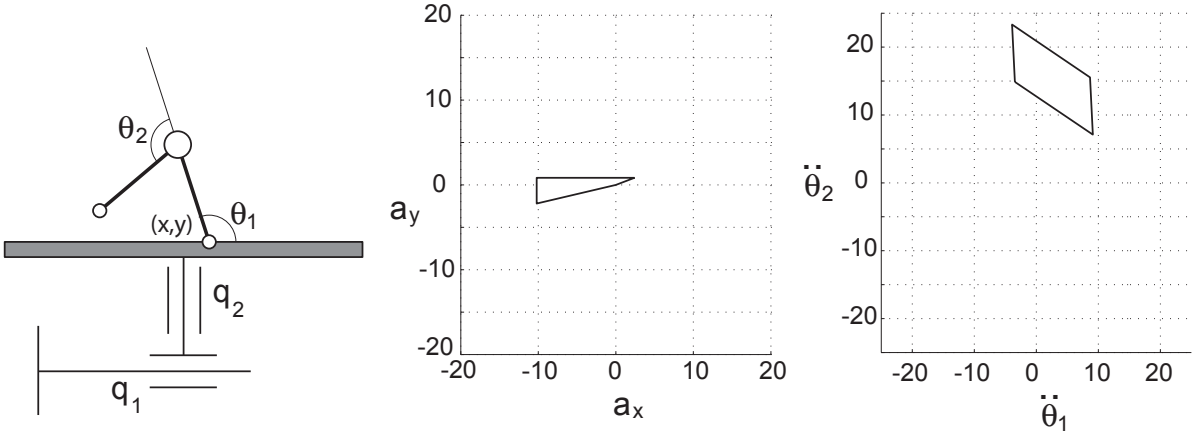


Figure 7.2: Walking by manipulation — the simplest walking model rests on an articulated palm. The plots show the constraint in contact acceleration space and the allowable angular accelerations of the joints that maintain a fixed contact between the palm and the walker.

We define the configuration of the walker $\boldsymbol{q_o}$ by $(x, y, \theta_1, \theta_2)$, where (x, y) are the coordinates of its contact with the palm and (θ_1, θ_2) are the joint angles of the links, as shown in Fig.7.2. We define the configuration of the palm $\boldsymbol{q_r}$ by (q_1, q_2). We control the motion of the palm $(\ddot{q}_1, \ddot{q}_2)$.

A rolling contact between the walker and the palm imposes the constraint:

$$\mathsf{J}\dot{\boldsymbol{q}}_{\boldsymbol{r}} = G^T \dot{\boldsymbol{q}}_{\boldsymbol{o}} \tag{7.1}$$

where

$$\mathsf{J} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathsf{G} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Coulomb friction at the contact imposes the contact force constraint:

$$\boldsymbol{f} \in \mathcal{F} \tag{7.2}$$

We can write the dynamics of the walker as:

$$\mathsf{M}\ddot{\boldsymbol{q}}_{\boldsymbol{o}} + \boldsymbol{B}(\boldsymbol{q}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{o}}) + \boldsymbol{n}_{\boldsymbol{o}} = \mathsf{G}\boldsymbol{f} \tag{7.3}$$

For a given configuration $\boldsymbol{q} = (\boldsymbol{q}_{\boldsymbol{o}}, \boldsymbol{q}_{\boldsymbol{r}})$ and velocity $\dot{\boldsymbol{q}} = (\dot{\boldsymbol{q}}_{\boldsymbol{o}}, \dot{\boldsymbol{q}}_{\boldsymbol{r}})$, the allowable joint acceleration of the robot $\ddot{\boldsymbol{q}}_{\boldsymbol{r}}$ that satisfies both the velocity and the force constraints are given by the contact acceleration constraint.

Applying the contact acceleration constraint, we get:

$$\begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} + \boldsymbol{V}(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) \in \mathsf{G}^{\mathsf{T}}\mathsf{M}^{-1}\mathsf{G}(\theta_1, \theta_2)(\mathcal{F}) \tag{7.4}$$

What is important here is that the constraints depend only on the angles (θ_1, θ_2) and their rates $(\dot{\theta}_1, \dot{\theta}_2)$ and not on the spatial location of the palm (x, y) or the velocity of the palm $(\dot{x}, \dot{y})$. The constraint in contact acceleration space, and the corresponding constraints on the angular rotations of the walker, for a given state, are shown in Fig.7.2.

Computing an *optimal gait* using this information is still an open problem. There is no clear notion of optimality for walking robots, even for the simplest walking model. A minimum energy gait results in the walker tip-toeing, taking tiny steps, while a gait that maximizes the stride length is not stable to control error and gets dangerously close to breaking the friction constraints. Of course, one could fix a trajectory for the walker and write a controller that servos the system to that trajectory [Morimoto et al., 2003]. However, by doing so, we are forcing the walker to follow a trajectory that might not be natural to it. We refer the reader to Wisse's thesis [Wisse, 2004] for a detailed discussion on optimality criteria for walking robots.

Motion retargeting

The term motion retargeting refers to the technique of editing an existing motion sequence to achieve a desired effect. The motion sequence is usually obtained from human motion capture data. The desired effect is usually to use this data to animate creatures that are kinematically dissimilar to humans, like ogres, Hobbits and green swamp creatures.

For the case of animating manipulation tasks, it may not be very difficult to come up with a reasonable guess for a path for the character and the object, for example by extracting just a few keyframes from motion capture data and editing them to fit a new situation or by having an animator set up some key poses for a totally new scenario.

However, getting the timing right for dynamic manipulation can be much more difficult for an animator to do than coming up with the key poses. Human beings are extremely good at observing manipulation tasks performed by others and estimating the inertial and frictional parameters of objects. Likewise, we are also very good at spotting errors in such animations. Motion retargeting of manipulation thus requires an accurate and dynamically feasible selection of the time-scaling of the animator's path. Hence, an important question is:

> Given an animation path, either from motion capture data or from interpolating an artist's keyframes, what is the space of dynamically feasible time-scalings of this path for a new set of inertial and frictional parameters?

Figure 7.3: Motion capture of a manipulation task, reproduced from [Pollard and Hodgins, 2002]

We used the motion capture data from [Pollard and Hodgins, 2002] to extract a feasible path. As shown in Fig.7.3, the character tumbles a box while maintaining a fixed contact between the box and the ground. During the course of manipulation, the character demonstrates various contact modes — her right and left hands slide and remain fixed on the box. We can parametrize the pose of the box and the motion of the character's hands by the angle θ that the box makes with the horizontal. A trace of the entire motion from $\theta = 0$ at the top left to $\theta = \frac{\pi}{2}$ is shown in Fig.7.4. The path is divided into stages (I-VI) depending

on the contact mode of the system. For example, in stage I, both hands maintain a fixed contact with the box.

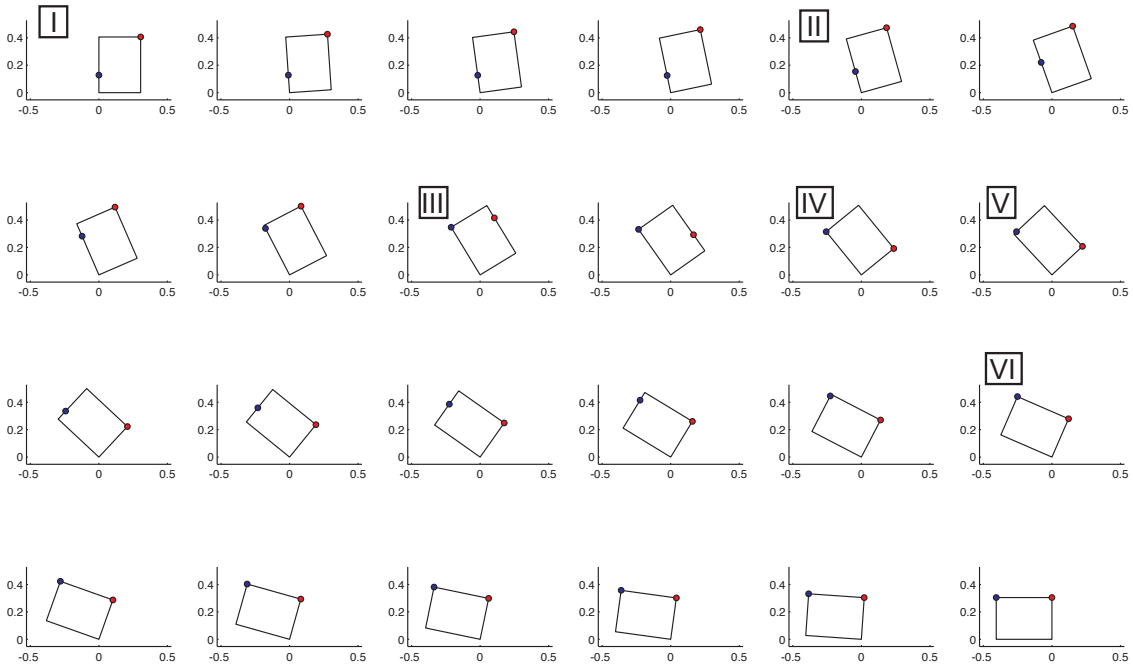


Figure 7.4: Motion of the block and the fingers. During all the sliding stages, motion of the finger along the side of the box is chosen to be linear in θ. Motion is divided into six stages (I-VI) based on the contact mode.

We took the parametrized path and applied the contact acceleration constraint to it. The procedure is identical to the one used to compute the feasible accelerations for the block standing problem in §3.

Fig.7.5 shows the phase plot for stage I. A limit of $10m/s^2$ is set on the acceleration $\ddot{\theta}$. The inertial properties and the coefficients of friction are chosen to be different from the properties measured during motion capture. A feasible retargeting time-scaling is a trajectory in this phase space that satisfies the tangency constraints at each point.

An interesting area of future work is the selection of an optimal time-scaling — one that is not only feasible but also physically realistic. The contact acceleration constraint provides a principled technique for computing feasible time-scalings of animations. One can imagine parametrizing this space of feasible trajectories and applying them to find the one that is most visually appealing.

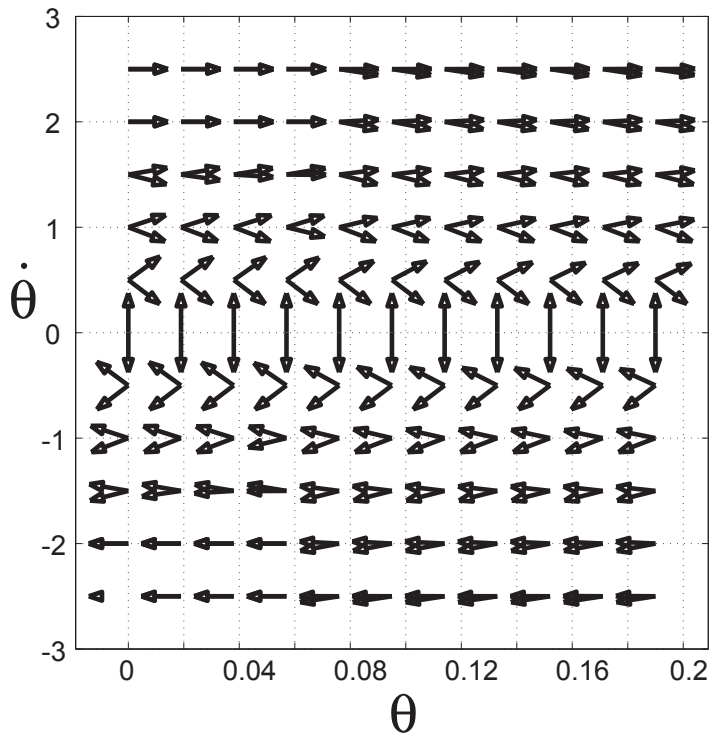


Figure 7.5: Phase plot for Stage I

7.3 Closing thoughts

The problem of generating robust dynamic trajectories for manipulation is still open. This is rather surprising, since even a child can perform complex manipulation tasks involving 3D contacts and controlled contact mode changes. Clearly, the child is not constructing friction cones in its head and computing their intersections, but posses some *instinct*, some innate ability to simplify the problem and solve the task. We hope we have convinced the reader that all the theorems and equations we have introduced in this thesis have helped us simplify the problem, and not obfuscate it. Manipulation is the *art* of moving things, and like any art form, it must be simple and elegant.

References

[Anitescu and Potra, 1997] Anitescu, M. and Potra, F. (1997). Formulating multi-rigid-body contact problems with friction as solvable linear complimentarity problems. In *ASME Journal of nonlinear dynamics*, volume 14, pages 231–247.

[Baillieul, 1985] Baillieul, J. (1985). Kinematic programming alternatives for redundant manipulators. In *IEEE Int. Conf. on Robotics and Automation*, pages 722–728.

[Balkcom and Trinkle, 2002] Balkcom, D. and Trinkle, J. (2002). Computing wrench cones for planar rigid body contact tasks. In *International Journal of Robotics Research*, volume 21, pages 1053–1066.

[Barraquand and Latombe, 1993] Barraquand, J. and Latombe, J.-C. (1993). Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155.

[Bellman, 1957] Bellman, R. (1957). Dynamic programming. Princeton University Press.

[Bicchi, 1994] Bicchi, A. (1994). On the problem of decomposing grasp and manipulation forces in multiple whole-limb manipulation. *International Journal of Robotics and Autonomous Systems*, 13:127–147.

[Bicchi and Marigo, 2002] Bicchi, A. and Marigo, A. (2002). Dexterous grippers: Putting nonholonomy to work for fine manipulation. In *International Journal of Robotics Research*, volume 21, pages 427–442.

[Blind et al., 2001] Blind, S., McCullough, C., Akella, S., and Ponce, J. (2001). Manipulating parts with an array of pins : A method and a machine. In *Int. Journal of Robotics Research*, volume 20, pages 808–810.

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

[Brock, 1988] Brock, D. (1988). Enhancing the dexterity of a robot hand using controlled slip. In *IEEE International Conference on Robotics and Automation*.

[Brost and Mason, 1989] Brost, R. and Mason, M. (1989). Graphical analysis of planar rigid-body dynamics with multiple frictional contacts. In *5th International Symposium on Robotics Research*, pages 293–300.

[Brost, 1991] Brost, R. C. (1991). *Analysis and Planning of Planar Manipulation Tasks*. PhD thesis, The Robotics Institute, Carnegie Mellon University. CMU-CS-TR-91-149.

[Butler and Tomizuka, 1992] Butler, J. and Tomizuka, M. (1992). A sub-optimal reference generation technique for robotic manipulators following specified paths. In *ASME Journal of dynamic systems, measurement, and control*, volume 114, pages 524–527.

[Cherif and Gupta, 1999] Cherif, M. and Gupta, K. (1999). Planning quasi-static fingertip manipulations for reconfiguring objects. In *IEEE Transactions on robotics and automation*, volume 15, pages 837–848.

[Chow, 1939] Chow, W. (1939). Uber systeme von linearen partiallen differentialgleichungen erster ordnung. *Math Ann.*, 117:98–105.

[Cole et al., 1989] Cole, A., Hauser, J., and Sastry, S. (1989). Kinematics and control of multifingered hands with rolling contact. In *IEEE Transactions on robotics and automation*, volume 34.

[Cole et al., 1992] Cole, A., Hsu, P., and Sastry, S. (1992). Dynamic control of sliding by robot hands for regrasping. In *IEEE Transactions on robotics and automation*, volume 8.

[Collins, 1975] Collins, G. (1975). Quantifier elemination for real closed fields by cylindrical algebraic decomposition. In *2nd GI conference on automata theory and formal languages*, volume 22, pages 134–183.

[Coulomb, 1781] Coulomb, C. (1781). *Théorie des Machines simples, en ayant égard au frottement de leurs parties et a la roideur des Corages*. Académie des Sciences.

[Craig, 1989] Craig, J. (1989). *Introduction to Robotics*. Addison Wesley.

[Donald et al., 1994] Donald, B., Jennings, J., and Rus, D. (1994). Analyzing teams of cooperating mobile robots. In *IEEE Int. Conf. on Robotics and Automation*.

[Donald et al., 1993] Donald, B., Xavier, P., canny, J., and Reif, J. (1993). Kinodynamic motion planning. In *journal for the association of computer machinery*, volume 40, pages 378–400.

[Erdmann, 1994] Erdmann, M. (1994). On a representation of friction in configuration space. In *International Journal of Robotics Research*, volume 13, pages 240–271.

[Erdmann, 1998] Erdmann, M. (1998). An exploration of nonprehensile two-palm manipulation. In *Int. Journal of Robotics Research*, volume 17.

[Erdmann and Mason, 1988] Erdmann, M. and Mason, M. (1988). An exploration of sensorless manipulation. In *Int. Journal of Rob. Res.*, volume 4, pages 369–378.

[Fearing, 1986] Fearing, R. (1986). Implementing a force strategy for object re-orientation. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 96–102.

[Garcia et al., 1998] Garcia, M., Ruina, A., Chatterjee, A., and Coleman, M. (1998). The simplest walking model: Stability, complexity, and scaling. In *ASME Journal of Biomechanical Engineering*, volume 120, pages 281–288.

[Goodwine and Burdick, 1996] Goodwine, B. and Burdick, J. (1996). Controllability with unilateral control inputs. In *IEEE Conference on Decision and Control*.

[Gupta, 1995] Gupta, K. (1995). Motion planning for re-orientation using finger tracking: landmarks in so(3)$\times\omega$. In *IEEE International Conference on Robotics and Automation*, pages 446–451.

[Gurvits and Li, 1993] Gurvits, L. and Li, Z. (1993). Smooth time-periodic feedback solutions for nonholonomic motion planning. In *Nonholonomic Motion Planning, Kluwer*, pages 53–108.

[Hanafusa et al., 1981] Hanafusa, H., Yoshikawa, T., and Nakamura, Y. (1981). Analysis and control of articulated robot with redundancy. In *IFAC 8th Triennial World Congress*, volume 4, pages 1927–1932.

[Harada and Kaneko, 2002] Harada, K. and Kaneko, M. (2002). A sufficient condition for manipulation of envelope family. In *IEEE Transactions on robotics and automation*, volume 18, pages 597–607.

[Harada et al., 2000] Harada, K., Kaneko, M., and Tsuji, T. (2000). Rolling-based manipulation for multiple objects. In *IEEE Transactions on robotics and automation*, volume 16, pages 457–468.

[Hollerbach, 1984] Hollerbach, J. (1984). Dynamic scaling of manipulator trajectories. In *Journal of Dynamic systems, Measurement and Control*, volume 106, pages 102–106.

[Jacobs et al., 1991] Jacobs, P., Laumond, J.-P., , and Taix, M. (1991). Efficient motion planners for nonholonomic mobile robots. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*.

[Kahn, 1969] Kahn, M. (1969). The near-minimum-time control of open-loop articulated kinematic chains. In *Stanford Artificial Intelligence Laboratory*, volume 106.

[Kahn and Roth, 1971] Kahn, M. and Roth, B. (1971). The near-minimum-time control of open-loop articulated kinematic chains. In *Journal of Dynamic systems, Measurement and Control*, volume 93, pages 164–172.

[Kerr and Roth, 1986] Kerr, J. and Roth, B. (1986). Analysis of multifingered hands. In *International Journal of Robotics Research*, volume 4, pages 3–17.

[Khatib et al., 1996] Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., Casal, A., and Baader, A. (1996). Force strategies for cooperative tasks in multiple mobile manipulation systems. In *Robotics Research : The Fifth Int. Symposium*.

[Kolmanovsky and McClamroch, 1995] Kolmanovsky, I. and McClamroch, N. (1995). Developments in nonholonomic control problems. In *IEEE Control systems magazine*, pages 20–36.

[Lafferriere and Sussmann, 1993] Lafferriere, G. and Sussmann, H. (1993). A differential geometric approach to motion planning. In *Nonholonomic Motion Planning*, pages 235–270.

[Laumond et al., 1994] Laumond, J.-P., Sekhavat, S., and Vaisset, M. (1994). Collision-free motion planning for a nonholonomic mobile robot with trailers. In *Proceedings of the IFAC symposium on robot control*.

[LaValle and Kuffner, 2001] LaValle, S. and Kuffner, J. (2001). Randomized kinodynamic planning. In *International Journal of Robotics Research*, volume 20, pages 378–400.

[Li et al., 1989] Li, Z., Canny, J., and Sastry, S. (1989). On motion planning for dexterous manipulation. i. the problem formulation. In *IEEE International Conference on Robotics and Automation*, pages 775–780.

[Li and Sastry, 1988] Li, Z. and Sastry, S. (1988). Task-oriented optimal grasping by multifingered robot hands. In *IEEE Journal of Robotics and Automation*, volume 4.

[Liu and Li, 2001] Liu, G. and Li, Z. (2001). A unified geometric approach to modeling and control of constrained mechanical systems. In *IEEE Transactions on robotics and automation*, volume XX.

[Lotstedt, 1981] Lotstedt, P. (1981). Coulomb friction in two-dimensional rigid body problems. In *Zeitschrift fur Angewandte Mathematik und Mechanik*, volume 61, pages 605–615.

[Lotstedt, 1982] Lotstedt, P. (1982). Mechanical systems of rigid bodies subject to unilateral constraints. In *Society for Industrial and Applied Mathematics*, volume 42, pages 281–296.

[Lotstedt, 1984] Lotstedt, P. (1984). Numerical simulation of time-dependent contacts and friction problems in rigid body mechanics. In *Society for Industrial and Applied Mathematics*, volume 5, pages 370–393.

[Lozano-Pérez, 1983] Lozano-Pérez, T. (1983). Spatial planning: a configuration space approach. In *IEEE Transactions on Computers*, volume 32, pages 108–120.

[Lynch, 1999] Lynch, K. (1999). Controllability of a planar body with unilateral thrusters. In *Automatic Control, IEEE Transactions on*, volume 44, pages 1206–1211.

[Lynch and Mason, 1999] Lynch, K. and Mason, M. (1999). Dynamic nonprehensile manipulation: Controllability, planning, and experiments. In *International Journal of Robotics Research*, volume 18.

[Lynch, 1996] Lynch, K. M. (1996). *Nonprehensile robotic manipulation : controllability and planning*. PhD thesis, The Robotics Institute, Carnegie Mellon University. CMU-RI-TR-96-05.

[Maeda et al., 2001] Maeda, Y., Kijimoto, H., Aiyama, Y., and Arai, T. (2001). Planning of graspless manipulation by multiple robot fingers. In *IEEE International Conference on Robotics and Automation*, pages 2474–2479.

[Mason, 1991] Mason, M. (1991). Two graphical methods for planar contact problems. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 443–448.

[Mason et al., 1999] Mason, M., Pai, D., Rus, D., Howell, J., Taylor, L., and Erdmann, M. (1999). Experiments with desktop mobile manipulators. In *Int. Symp. on Experimental Robotics*.

[McGeer, 1990] McGeer, T. (1990). Passive dynamic walking. In *International Journal of Robotics Research*, volume 9, pages 62–82.

[Mirtich and Canny, 1992] Mirtich, B. and Canny, J. (1992). Using skeletons for nonholonomic path planning among obstacles. In *IEEE Int. conf. on robotics and automation*, pages 2533–2540.

[Morimoto et al., 2003] Morimoto, J., Zeglin, G., and Atkeson, C. (2003). Minmax differential dynamic programming: Application to a biped walking robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Nakamura, 1991] Nakamura, Y. (1991). Advanced robotics-redundancy and optimization. In *Addison-Wesley*.

[Nguyen, 1988] Nguyen, V. (1988). Constructing force-closure grasps. In *International Journal of Robotics Research*, volume 7(3), pages 3–16.

[Nilsson, 1984] Nilsson, N. (1984). Shakey the robot. Technical Report 223, SRI International.

[Okamura et al., 2002] Okamura, A. M., Smaby, N., and Cutkosky, M. R. (2002). An overview of dexterous manipulation. In *IEEE International Conference on Robotics and Automation*, pages 255–262.

[Pai et al., 1994] Pai, D., Barman, R., and Ralph, S. (1994). Platonic beasts : a new family of multilimbed robots. In *IEEE Int. Conf. on Robotics and Automation*.

[Painlevé, 1895] Painlevé, P. (1895). *Sur les lois du frottement de glissement*. Académie des Sciences.

[Paljug et al., 1994] Paljug, E., Yun, X., and Kumar, V. (1994). Control of rolling contacts in multi-arm manipulation. In *IEEE Transactions on robotics and automation*, volume 10, pages 441–452.

[Pollard and Hodgins, 2002] Pollard, N. S. and Hodgins, J. K. (2002). Generalizing demonstrated manipulation tasks. In *Workshop on the Algorithmic Foundations of Robotics*.

[Rus, 1997] Rus, D. (1997). In-hand manipulation of piecewise-smooth 3d objects. In *International Journal of Robotics Research*, volume 18, pages 355–381.

[Sacks, 1998] Sacks, E. (1998). Practical sliced configuration spaces for curved planar pairs. *Int. Journal of Robotics Research*, 18:59–63.

[Sacks, 2001] Sacks, E. (2001). Configuration space path planning for planar mechanical systems. In *IEEE Int. Conf. on Robotics and Automation*.

[Sahar and Hollerbach, 1985] Sahar, G. and Hollerbach, J. (1985). Planning a minimum-time trajectories for robot arms. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 751 – 758.

[Sarkar et al., 1997a] Sarkar, N., Yun, X., and Kumar, V. (1997a). Control of contact interactions with acatastatic nonholonomic constraints. In *International Journal of Robotics Research*, volume 16.

[Sarkar et al., 1997b] Sarkar, N., Yun, X., and Kumar, V. (1997b). Dynamic control of 3-d rolling contacts in two-arm manipulation. In *IEEE Transactions on robotics and automation*, volume 13, pages 357–374.

[Schwartz and Sharir, 1983a] Schwartz, J. and Sharir, M. (1983a). On the piano movers' problem i: the case of a rigid polygonal body moving amidst polygonal barriers. In *Comm. Pure and Applied Mathematics*, volume 36, pages 345–398.

[Schwartz and Sharir, 1983b] Schwartz, J. and Sharir, M. (1983b). On the piano movers' problem ii: general techniques for computing topological properties of real algebraic manifolds. In *Advances in Applied Mathematics*, volume 4, pages 298–351.

[Schwartz and Sharir, 1983c] Schwartz, J. and Sharir, M. (1983c). On the piano movers' problem iii: coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. In *International Journal of Robotics Research*, volume 2, pages 46–75.

[Schwartz and Sharir, 1984] Schwartz, J. and Sharir, M. (1984). On the piano movers' problem v: the case of a rod moving in three dimensional space amidst polyhedral obstacles. In *Comm. Pure and Applied Mathematics*, volume 37, pages 815–845.

[Shin and McKay, 1985] Shin, K. and McKay, N. (1985). Minimum-time control of robotic manipulators with geometric path constraints. In *IEEE Transactions on Automatic Control*, volume 30, pages 531–541.

[Simunovic, 1975] Simunovic, S. (1975). Force information in assembly processes. In *5th International Symposium on Industrial Robots*.

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

[Srinivasa et al., 2002] Srinivasa, S., Baker, C., Sacks, E., Reshko, G., Erdmann, M., and Mason, M. (2002). Experiments with nonholonomic manipulation. In *IEEE International Conference on Robotics and Automation*.

[Srinivasa et al., 2003] Srinivasa, S., Erdmann, M., and Mason, M. (2003). Bilateral time-scaling for the control of task freedoms of a constrained nonholonomic system. In *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan.

[Srinivasa et al., 2005a] Srinivasa, S., Erdmann, M., and Mason, M. (2005a). Control synthesis for dynamic contact manipulation. In *IEEE International Conference on Robotics and Automation*.

[Srinivasa et al., 2005b] Srinivasa, S., Erdmann, M., and Mason, M. (2005b). Using projected dynamics to plan dynamic contact manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Sussman, 1978] Sussman, H. (1978). Lie brackets and local controllability : A sufficient condition for local scalar-input systems. *SIAM J. on control and opt.*, 21(5).

[Sussmann and W., 1991] Sussmann, H. and W., L. (1991). Limits of highly oscillatory controls and approximation of general paths by admissible trajectories. In *IEEE International conference on decision and control*.

[Thompson, 1977] Thompson, A. (1977). The navigation system of the jpl robot. In *Proc. 5th Int. Joint Conf. on Artificial Intelligence*.

[Trinkle and Hunter, 1991] Trinkle, J. and Hunter, J. J. (1991). A framework for planning dexterous manipulation. In *IEEE International Conference on Robotics and Automation*, pages 1245–1251.

[Trinkle et al., 1997] Trinkle, J., Pang, J., Sudarsky, S., and Lo, G. (1997). On dynamic multi-rigid-body contact problems with coulomb friction. In *Zeitschrift fur Angewandte Mathematik und Mechanik*, volume 77, pages 267–279.

[Trinkle and Paul, 1990] Trinkle, J. and Paul, R. (1990). Planning for dexterous manipulation with sliding contacts. In *International Journal of Robotics Research*, volume 9, pages 24–48.

[Trinkle and Zeng, 1995] Trinkle, J. and Zeng, D. (1995). Prediction of the quasistatic planar motion of a contacted rigid body. In *IEEE Transactions on robotics and automation*, volume 11, pages 229–246.

[Wen and Desrochers, 1986] Wen, J. and Desrochers, A. (1986). Sub-time-optimal control strategies for robotic manipulators. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 118–124.

[Whitney, 1982] Whitney, D. (1982). Quasi-static assembly of compliantly supported rigid parts. In *ASME Journal of dynamic systems, measurement, and control*, volume 104, pages 65–77.

[Wisse, 2004] Wisse, M. (2004). Essentials of dynamic walking: analysis and design of two-legged robots. In *PhD Thesis, TU Delft*.

[Zheng et al., 2000] Zheng, X.-Z., Nakshima, R., and Yoshikawa, T. (2000). On dynamic control of object motion and finger sliding in manipulation with multifingered hands. In *IEEE Transactions on robotics and automation*, volume 16, pages 469–481.