Nonprehensile Robotic Manipulation: Controllability and Planning

Kevin M. Lynch
March 1996
CMU-RI-TR-96-05

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

This research was funded in part by NSF under grants IRI-9318496 and IRI-9114208 and NASA Ames under grant NCC 2-713. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NSF, NASA, or the U.S. Government.

Abstract

A good model of the mechanics of a task is a resource for a robot, just as actuators and sensors are resources. The effective use of frictional, gravitational, and dynamic forces can substitute for extra actuators; the expectation derived from a good model can minimize sensing requirements. Despite this, most robot systems attempt to dominate or nullify task mechanics, rather than exploit them. There has been little effort to understand the manipulation capabilities of even the simplest robots under more complete mechanics models.

This thesis addresses that knowledge deficit by studying graspless or *nonprehensile* manipulation. Nonprehensile manipulation exploits task mechanics to achieve a goal state without grasping, allowing simple mechanisms to accomplish complex tasks. With nonprehensile manipulation, a robot can manipulate objects too large or heavy to be grasped and lifted, and a low-degree-of-freedom robot can control more degrees-of-freedom of an object by allowing relative motion between the object and the manipulator.

Two key problems are *determining controllability* of and *motion planning* for nonprehensile manipulation. The first problem is to determine whether the goal state of the object is reachable by nonprehensile manipulation, and the second is to find a manipulator motion to bring the object to the goal state.

Part I studies these problems for quasistatic nonprehensile manipulation by pushing. I elucidate the controllability properties of objects pushed with point and line contact, and I describe a planner that finds stable pushing paths among obstacles. Pushing can also be used to simplify the hardware of a parts feeder; a one-degree-of-freedom robot, positioned over a conveyor, can position and orient any polygonal part on the conveyor by a series of pushes.

Part II of the thesis studies dynamic nonprehensile manipulation. By considering dynamics, I show that even a one-degree-of-freedom robot can take a planar object to a full six-dimensional subset of its state space. Then I describe a planner that finds manipulator trajectories to perform dynamic tasks such as snatching an object from a table, rolling an object on the surface of the arm, and throwing and catching.

To demonstrate the feasibility of nonprehensile manipulation, all planners have been implemented on actual robot systems.

Acknowledgments

The Robotics Institute has been a great place to be a grad student. It's difficult to leave the great facilities and gifted people who have made it such a pleasure to do research here. I'm lucky to have had the world's premier robotics institute right in my hometown, where I could watch my Steelers every winter.

Sports have always been a big part of my life, and some of my most enjoyable times at Carnegie Mellon have come playing for the Rude Dogs in volleyball, the NP-Completions in touch football, and Shakey's Sluggers in softball. We turned the Rude Dogs and the NP-Completions into perennial powerhouses, and as for Shakey's Sluggers, well, there's always next year. I want to especially thank Mark, Bill, Todd, Will, Rich, and David for being great teammates. I'll miss the weight room crowd, and our guru, Rick Statman, whose unique and usually unsolicited opinions created some interesting discussion.

Thanks to the Breaks and Dishes for a memorable coaching opportunity.

Thanks to Jean and Barbara, who have helped me out more times than I can remember. Thanks Tammy, Mark, Greg, and Andrzej for being great officemates and for putting up with my various stupid questions. Thanks to the dinner coop for forcing me to eat right at least once a week: Greg, Jack, Scott, LeAnn, Mei, Susan, Alan, Dave, Mark, Rich, and Tammy. Also thanks to Gary, Jay, Harry, Yoichi, and our Secret Robotics Valentine (you know who you are!)

It's hard to imagine what grad school would have been like without Mark, my housemate, officemate, and teammate. Thanks for everything. And remember, a football isn't round.

Thanks to Matt Mason and Mike Erdmann for making the Manipulation Lab a fun place to work. It's been exciting to participate in the mlab renaissance, along with my labmates Nina, Garth, Yan-bin, Wes, Srinivas, and Tammy. Thanks to former lab members Yu, Ken, Randy, and Alan for showing that it is actually possible to graduate! Thanks to Costa, my right hand man, for his energy in the lab. We were able to do more than I thought possible.

Thanks to my committee for their comments and encouragement: Matt Mason, Mike Erdmann, Ralph Hollis, and Dan Koditschek. For helpful comments on a draft of my thesis, thanks to Garth, Wes, Yan-bin, and Nina. Also thanks to Ben Brown, Tom Mitchell, André Tits, Craig Lawrence, Richard Murray, P. S. Krishnaprasad, Naomi Leonard, Omar Ghattas, and John Maddocks for various contributions to this thesis.

I am very fortunate to have had Matt as my advisor and friend. I have learned much from his good humor, creativity, and integrity. I wonder how much we could have gotten done if we hadn't spent our meetings eating O fries and talking about football? Somehow, though, I think the time was better spent.

Finally, I thank my family for being a constant source of strength, love, and support. It has been a special thrill for me to attend grad school in my hometown, where I could see my family regularly and watch my younger brother grow up.

Most of all I thank my Mom and Dad, who have given me every opportunity in life, and to whom I owe everything.

Contents

Notation

Parts I and II:

$\mathcal{C}$	the configuration manifold $SE(2) = \mathbf{R}^2 \times S^1$
$\mathcal{F}_{\mathcal{W}}$	a coordinate frame fixed in $\mathcal{C}$
$\mathbf{q} = (x_w, y_w, \phi_w)^T$	the configuration of the body in $\mathcal{F}_{\mathcal{W}}$
$\dot{\mathbf{q}}, \ddot{\mathbf{q}}$	the velocity and acceleration of the body in $\mathcal{F}_{\mathcal{W}}$
$T_{\mathbf{q}}\mathcal{C}$	the tangent space at $\mathbf{q}$
$T\mathcal{C}$	the tangent bundle of $\mathcal{C}$
$T_{(\mathbf{q},\dot{\mathbf{q}})}T\mathcal{C}$	the tangent space at $(\mathbf{q}, \dot{\mathbf{q}})$
$TT\mathcal{C}$	the tangent bundle of $T\mathcal{C}$
μ	the friction coefficient between the manipulator and the body
$\mathbf{f} = (f_x, f_y, \tau)^T$	a generalized force (wrench)
$\mathbf{v} = (v_x, v_y, \omega)^T$	a generalized velocity (twist)
$\mathbf{a} = (a_x, a_y, \alpha)^T$	a generalized acceleration
$\mathcal{F}, \mathcal{V}, \mathcal{A}$	sets (or cones) of forces, velocities, and accelerations
$\hat{\mathbf{f}} = (\hat{f}_x, \hat{f}_y, \hat{\tau})^T$	a unit generalized force; similarly for $\hat{\mathbf{v}}$, $\hat{\mathbf{a}}$
$\hat{\mathcal{F}}, \hat{\mathcal{V}}, \hat{\mathcal{A}}$	sets (or cones) of unit forces, velocities, and accelerations
X	a vector field
$\mathcal{X}$	a set of vector fields
$L(\mathcal{X})$	the Lie algebra of a set of vector fields $\mathcal{X}$
$Br(\mathcal{X})$	smallest subset of $L(\mathcal{X})$ containing $\mathcal{X}$ and closed under bracketing
$CH_{S^2}(\cdot)$	the convex hull of a set of unit vectors on S^2
$CH_{\mathbf{R}^3}(\cdot)$	the convex hull of a set of vectors in $\mathbf{R}^3$
,	the piecewise smooth curve describing the perimeter of the body

Part I only:

$\mathcal{P}$	the pusher
$\mathcal{S}$	the slider (pushed object)
$\mathcal{F}_{\mathcal{S}}$	a coordinate frame fixed to the slider
COR($\hat{\mathbf{v}}$)	the center of rotation in $\mathcal{F}_{\mathcal{S}}$ of a unit velocity $\hat{\mathbf{v}}$ in $\mathcal{F}_{\mathcal{S}}$
$s(\mathbf{x})$	the support friction distribution
$\mathcal{C}_{free}$	the open subset of free (noncolliding) configurations in $\mathcal{C}$
$\mathcal{C}_{obs}$	the closed subset complementary to $\mathcal{C}_{free}$

Part II only:

$\mathcal{M}$	the manipulator
Θ	the configuration of the manipulator $\mathcal{M}$
$\dot{\Theta}$	the velocity of the manipulator $\mathcal{M}$
$\mathcal{O}$	the object
$\mathcal{F}_{\mathcal{O}}$	a coordinate frame fixed to the object
ρ	the radius of gyration of the object
$\mathbf{g}$	the gravitational acceleration vector

Chapter 1

Introduction

One of the most basic tasks for a robotic manipulator is to move an object from one place to another. A common solution is to equip the manipulator with a gripper and use pick-and-place. By designing the grasp to resist all forces that could reasonably act on the object during the motion, grasp planning and path planning can usually be decoupled.

If the object is too large to be grasped or too heavy to be carried, however, this approach fails. It underutilizes the resources available to the robot, as it uses only the control forces that can be statically applied at the gripper. In general, the manipulator can apply forces through any surface of any of its links. Other useful sources of control forces include gravity, the frictional kinematic constraints (floor, walls, and obstacles) which make up the robot's environment, and dynamic forces. If the robot can reason about these forces, it can use a richer set of manipulation primitives, including pushing, throwing, and striking. As a result, the set of achievable tasks is increased.

The key idea is this: *a good model of the mechanics of a task is a resource for the robot system, just as actuators, sensors, and computers are resources.* This model can be embedded in the structure of the mechanism, encoded in a feedback scheme, or used explicitly in a manipulation planner. In this thesis, I explore the last approach and study the capabilities of very simple robots endowed with good models of mechanics.

Based on the underlying mechanics model, we can define a taxonomy of manipulation models (Mason and Lynch [143]):

- *Kinematic manipulation.* Kinematics (motion without regard to forces) are considered.
- *Static manipulation.* Kinematics and static forces (such as Coulomb static friction and gravity) are considered.
- *Quasistatic manipulation.* Kinematics, static forces, and quasistatic forces, such as sliding friction, are considered. Inertial forces are negligible.
- *Dynamic manipulation.* Kinematics, static forces, quasistatic forces, and dynamic forces are considered.

(Other manipulation models can be formulated, including aerodynamic, relativistic, and quantum effects. In this thesis, I assume those effects are negligible.)

Typical pick-and-place planners employ kinematic or static models of manipulation. In this thesis I focus on quasistatic and dynamic manipulation.

1.1 Nonprehensile Manipulation

To explore the power of quasistatic and dynamic models, I study graspless or *nonprehensile* manipulation. Nonprehensile manipulation exploits the mechanics of the task to achieve a goal state without grasping, allowing simple mechanisms to accomplish complex tasks. Some common examples of nonprehensile manipulation include pushing a couch, shooting a basketball, rolling dice, carrying a tray of food, and playing ping-pong.

Nonprehensile manipulation offers several potential benefits:

- **New robot primitives.** Nonprehensile manipulation provides an option for a robot when it must manipulate an object too large or heavy to be grasped and lifted.

- **Simpler manipulators.** The structure of the manipulator can be simplified by eliminating the gripper.

- **Flexibility.** Grippers are often designed to manipulate parts of particular shapes and sizes. The geometric simplicity of nonprehensile contact makes it potentially more flexible with respect to the class of manipulable parts. Increased complexity of the robot's motion may compensate for the simplicity of the contact.

- **Increased workspace size.** We usually define the workspace of a robot as the kinematic workspace of its end-effector. In some situations, a more useful concept is the set of reachable configurations for an object manipulated by the robot. If the robot can throw the object to points outside its kinematic workspace, the size of the robot's workspace is effectively increased.

- **Increased workspace dimensionality.** When an object is carried with a grasp, the manipulator must have at least as many degrees of freedom as the number of degrees of freedom of the object we want to control. If we allow the object to move relative to the manipulator, we can control more degrees of freedom than the manipulator itself has. It is not always necessary to attach a motor to every degree of freedom we would like to control.

Nonprehensile manipulation allows us to transfer some of the complexity of a robot system from hardware (actuators and links) to the underlying mechanics model. It is somewhat surprising that so much effort has been spent building highly complex robotic mechanisms, but so little effort has been made to understand the manipulation capabilities of even the simplest robots under more complete mechanics models.

1.1.1 Applications of Nonprehensile Manipulation

This section lists some applications of nonprehensile manipulation, to motivate the problems studied in this thesis.

Parts feeding. Nonprehensile manipulation is particularly suited for parts feeding for automated assembly. The vibratory bowl feeder (Boothroyd *et al.* [35]) and the SONY APOS (Advanced Parts Orienting System) (Hitakawa [92]) are two successful systems that use dynamic nonprehensile manipulation. In both cases, a vibratory motion is combined with specially-shaped fixtures to either bring parts to a desired orientation, or reject them and send them back to be reprocessed.

We have applied results of this thesis to develop a simple planar parts feeding system, dubbed "1JOC" (one-joint-over-conveyor, pronounced "one jock") (Akella *et al.* [6]). Using a single joint manipulator positioned over a conveyor, the 1JOC can bring any polygonal part from a random initial configuration on the conveyor to a desired configuration by a series of pushes. The conveyor is used not only for parts transfer, but also to aid in the robot's manipulation. The 1JOC is described in Chapter 7.

In parts assembly tasks, prehensile manipulation seems to be preferable to nonprehensile manipulation, because small position variations can lead to very different contact forces. With prehensile contact, arbitrary corrective forces can be applied to achieve successful mating, as with the RCC wrist (Whitney [217]). However, APOS provides an example of a simple type of assembly with nonprehensile manipulation—parts are "assembled" in the proper orientation in specially-shaped depressions of a vibrating pallet. A key difference is that the "assembly" is stochastic, and assembly error does not decrease monotonically during the operation. Also, such an approach may not be appropriate when tolerances are tight.

Parts transfer. Heavy crates in warehouses are typically conveyed by means of pallet jacks and forklifts. For these to be effective, however, the crate usually must be mounted on a pallet. Pushing provides a solution for crates and other objects that are not mounted on pallets. In the future, warehouses will be maintained by robots, and pushing will be a useful capability for these robots.

Throwing can be a useful method for transferring parts and tools between robots with disjoint kinematic workspaces. According to Marvin Minsky (Dodd and Rossol [62]):

> It seems to me that assembly lines are silly and when we have good hand-eye robots, they will usually throw the part across the factory to the machine that wants it and that machine will catch it.

Space. Dynamic nonprehensile manipulation can be particularly useful in space, where dynamic effects dominate. For instance, robots may send parts back and forth by throwing.

Air resistance is negligible, and because gravitational effects are very small, throws may be as slow as desired. (In contrast, a gravitational field provides a natural time scale.) It is not necessary to attach motors or jets to the part as it moves from one robot to another; forces are only necessary when a velocity change is needed. Newton's laws are much more reliable than any robot controller.

Two advantages of dynamic nonprehensile manipulation which seem particularly relevant in space are:

1. *Fewer robot failures.* Robots which use motion of the object relative to the manipulator can be made with fewer actuated joints, resulting in fewer actuators to fail. Nonprehensile manipulation also provides an alternative for a fully-actuated robot when actuators fail. This is especially important in space, where repairing the robot may be difficult.

2. *Cost.* Simpler, less massive robots are less expensive to send into space.

Robot programming languages. Someday robot programming languages will include such primitives as "push" and "throw." Box pushing has become such a popular task for mobile robots that at least one mobile robot manufacturer is planning to include pushing simulation software with its simulation environment (Pignon [166]).

1.1.2 An Example

Let's look at one example of nonprehensile manipulation: a nonprehensile "grasp." I will say that an object is "grasped" if it remains fixed to the manipulator as it moves. Using the taxonomy of manipulation models, we can define the following types of grasps (Mason and Lynch [143]):

- *Kinematic grasp.* All relative motion is prevented considering only kinematic constraints. This is a form closure grasp, often used in fixturing.

- *Static grasp.* All relative motion is prevented considering only kinematic constraints and static forces, such as gravity and static friction. A book resting on a table is statically grasped by the table. A force closure grasp (as it is usually defined in the robotics literature) is a static grasp regardless of the external force acting on the object.

- *Quasistatic grasp.* All relative motion is prevented considering only kinematic constraints, static forces, and quasistatic forces, such as sliding and friction. When pushing a block across a table, the block remains fixed to the manipulator in part due to the forces of sliding friction.

- *Dynamic grasp.* All relative motion is prevented considering kinematic constraints, static forces, quasistatic forces, and forces of acceleration. Consider slapping a coin held in one palm on to the back of the other hand. The coin is dynamically grasped by the palm while it is inverted by accelerating the palm downward faster than gravity.

When an object is held with a form or force closure grasp, it can be moved as desired subject only to kinematic and force limitations of the manipulator. In contrast, an object held in a nonprehensile quasistatic or dynamic grasp cannot be moved in an arbitrary manner. *The manipulator must choose motions which use quasistatic or dynamic control forces to stabilize the grasp, so the "grasp" planning and motion planning problems cannot be decoupled.*

This illustrates one of the main difficulties of nonprehensile manipulation. The motion of the object is generally a complex function of the manipulator controls, the states of the manipulator and the object, the contact between the manipulator and the object, and properties of the object itself. The result is that motion planning is much harder than simple kinematic motion planning, and it requires more information about the object. Another issue is the *controllability* of the object: *is it even possible to move the object from the initial state to the goal state using nonprehensile manipulation?*

1.2 Thesis Outline

This thesis studies the problem of transferring a part from one state to another using nonprehensile manipulation. The problems considered are planar. The thesis is divided into two main parts: quasistatic nonprehensile manipulation by pushing, and dynamic nonprehensile manipulation. In each part, I study:

Controllability. Is the desired goal state achievable by nonprehensile manipulation? Motion constraints due to the nonprehensile contact must be considered.

Planning. How do we find a control sequence to achieve the desired state?

A summary of each part is given below.

1.2.1 Part I: Quasistatic Pushing

Controllability

Although there has been a lot of previous work on manipulation by pushing (see Section 1.3.2), there has been little work on the controllability of pushing. Part I of the thesis addresses this problem. The quasistatic motion of a pushed object is a complex function of the friction between the object and the support surface and the friction at the pushing contact. To determine the controllability of a pushed object, I study the set of pushing forces that can be applied to it through its perimeter, and the mapping of these forces through the mechanics of sliding friction to motions of the object. The result of this analysis is a set of feasible motion directions of the object, fixed in the object's frame.

When the object is pushed with point contact, I show

> Any object can be made to follow any path arbitrarily closely by pushing it with point contact, provided the object is not a disk centered at its center of friction with zero friction between the pusher and the object. Thus a two-degree-of-freedom robot (e.g., a point translating in the plane) can maneuver almost any object approximately along any path in the object's three-dimensional configuration space.

The motion of an object pushed at a single point of contact is generally unpredictable, due to indeterminate support friction. With two or more contacts, however, we can achieve a *stable push*, where the object remains fixed to the pusher. For the particular case of line pushing contact, I show

> If the object is a polygon and there is nonzero friction at the pushing contact, then there exists a line contact between the pusher and the object from which the object can be pushed to any configuration in the obstacle-free plane by stable pushes. If the center of friction of the object is known, we can find a set of stable pushes that yields this property.

With two or more line contacts, it may be possible to push the object to follow any trajectory arbitrarily closely.

Planning

Stable pushes are used to synthesize controls for the pushing control system. Chapter 6 describes a planner that finds stable pushing plans to maneuver an object among obstacles. An example plan is shown in Figure 1.1. To demonstrate the feasibility of this approach, the plans have been implemented on an Adept 550 SCARA robot. The plans are executed open-loop, but the mechanics of the stable push make them robust.

Chapter 7 presents an application of the results to a parts feeding problem. The 1JOC positions and orients parts on a conveyor using only one revolute joint. A sequence of pushing motions, punctuated by drift on the conveyor, is sufficient to bring any polygonal part from a three-dimensional set of initial configurations on the conveyor to a single goal configuration. This is a dramatic example of the power of an underactuated manipulator when coupled with a good predictive model.

1.2.2 Part II: Dynamic Nonprehensile Manipulation

Part II studies the *dynamic pick-and-place* problem of designing a manipulator trajectory to take an object from an initial state to a goal state using nonprehensile contact. Dynamic nonprehensile manipulation differs from quasistatic pushing in that the object's state includes velocity. For this reason, I am especially concerned with constraints on the manipulator motion, as the object's state no longer remains fixed as the manipulator moves from one contact to another.

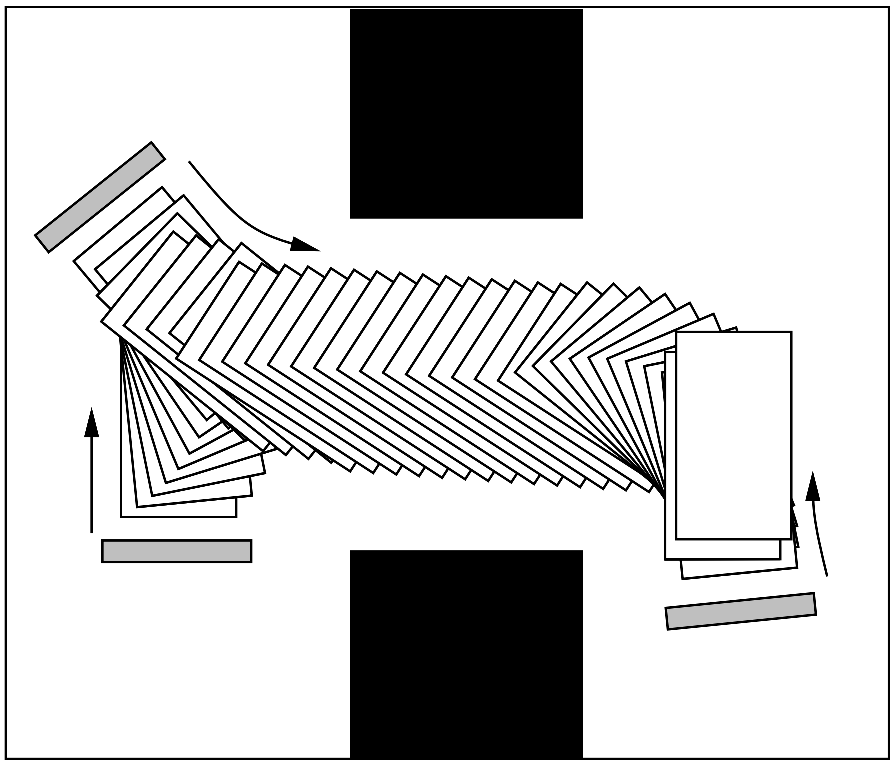


Figure 1.1: Quasistatic nonprehensile manipulation: maneuvering a box through a passageway by stable pushing.

Controllability

Chapter 9 studies reachable sets for an object during dynamic nonprehensile manipulation. I begin by studying the controllability of an object manipulated with point or line contact, assuming no constraints on the motion of the manipulator. The control system then resembles a rigid free-flying object equipped with jets. Because of the problem of changing contacts, however, I then study the accessible sets when the manipulator makes a single sustained contact with the object. In particular, I show that even a one-degree-of-freedom manipulator can take a planar object to a six-dimensional subspace of its six-dimensional state space. In other words, equality constraints on the motion of the manipulator (due to the single joint) do not translate to equality constraints on the configuration or velocity of the object.

Planning

Chapter 10 presents an approach to synthesizing a manipulator trajectory to take an object to a goal state. (The general approach can also be applied to other nonholonomic motion planning problems.) A low-degree-of-freedom manipulator can control an object (with more state variables than the manipulator) by sequencing manipulation phases, including dynamic grasp, roll, and free flight phases. The planning algorithm is a gradient descent optimization which produces manipulator trajectories that increasingly satisfy physical constraints, including friction constraints, while transferring the object to the goal. The objective of the optimization is usually to minimize the required friction coefficient at the contact. An example trajectory generated for a one-degree-of-freedom arm is shown in Figure 1.2.

Chapter 11 describes the implementation of trajectories generated by the dynamic motion planner on a one-degree-of-freedom direct-drive arm. The arm successfully performs dynamic tasks such as snatching an object from a table by accelerating into it, rolling an object on the surface of the arm, and throwing and catching. Despite the simplicity of the one-degree-of-freedom arm, interesting tasks can be performed by varying its (theoretically) infinite-dimensional trajectory.

1.3 Related Research

This section identifies work related to the topic of this thesis. Because the techniques used in this thesis draw on several different fields, including frictional mechanics, motion planning, and nonlinear control, the bibliography of related work is rather extensive. (On the other hand, to keep the bibliography manageable, many works related to the problems of this thesis have been omitted.) It is my hope that this section not only provides citations to other relevant work, but also establishes the perspective with which I view the nonprehensile manipulation problem.

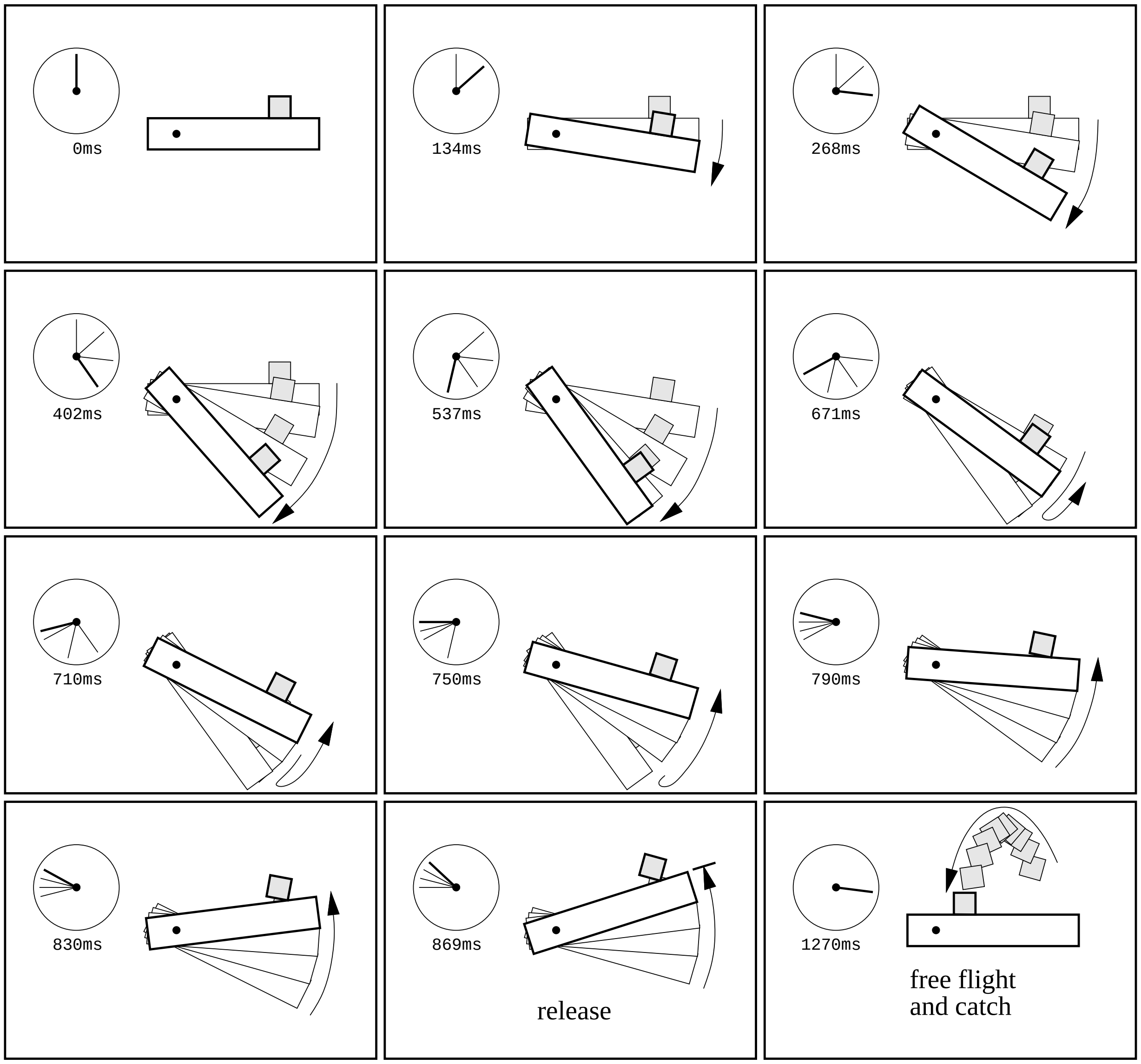


Figure 1.2: Dynamic nonprehensile manipulation: rolling and throwing.

Many of the works cited here can fall under several different categories. I have attempted to place each in the category most relevant to this thesis.

1.3.1 Friction

Friction plays a key role in nonprehensile manipulation. For example, without static friction at an edge contact (planar case) of a quasistatic or dynamic grasp, the set of forces which can be applied to the object has null interior, and therefore the set of manipulator motions which maintains the grasp also has null interior.

In this research, I assume friction conforms to Coulomb's law (Coulomb [57]). The friction cone (Moseley [152]) is a useful geometric interpretation of Coulomb's law. Erdmann [68, 71] generalized the idea of the friction cone by including torques in the representation. Thus a friction cone in the plane is represented as an infinite planar cone emanating from the origin of a three-dimensional force-torque space. For multiple contact points, the cone of possible reaction forces is simply the convex hull of the individual cones. Erdmann showed how to use this generalized friction cone to compute the reaction forces and motions of an object in its configuration space. Extensions to the six-dimensional force-torque space of spatial objects are also discussed.

Graphical methods for representing and combining planar frictional forces are described by Mason [141] and Brost and Mason [44]. Both of these methods ignore force magnitudes to render the force-torque space two-dimensional. The latter approach essentially projects Erdmann's generalized friction cones to the unit sphere in force-torque space. These approaches are suited to problems where force magnitudes are not relevant to determining the direction of motion of an object. Rajan *et al.* [170] described an algebraic approach to determining the motion of a rigid planar object in frictional contact with walls.

Coulomb friction is an approximate model of a complex and poorly understood physical process. It is well known that rigid-body mechanics with Coulomb friction gives rise to inconsistencies and ambiguities (Baraff [19], Dupont [67], Erdmann [68, 69, 70], Lötstedt [126], Lynch and Mason [132], Trinkle *et al.* [211], Wang *et al.* [215]). To overcome this difficulty, compliant contact models have been suggested (Sinha [193]; Wang *et al.* [215]). Alternatively, the friction law could be modified; the field of tribology (Bowden and Tabor [36]) is concerned with finding more accurate descriptions of friction. In this research, however, I assume rigid-body mechanics with Coulomb friction for the following reasons: (1) Coulomb's law is fairly accurate in most situations of interest; (2) the rigid-body state parameters and Coulomb friction coefficient provide an economical description of the task; and (3) the purpose of this thesis is to generate useful robotic manipulation plans, not to examine the micromechanics of contact. Despite the relative simplicity of the contact model, Baraff [19] showed that finding the motion of a rigid object in frictional contact may require time exponential in the number of contacts.

1.3.2 Nonprehensile Manipulation

Many different forms of nonprehensile manipulation have been studied. Of these, pushing has received the most attention.

Pushing

Mason identified pushing as an important manipulation process for manipulating several objects at once, for reducing uncertainty in part orientation, and as a precursor to grasping. Building on early work by Prescott [167] and MacMillan [134] on determining the support friction forces and torques during general sliding motion, Mason [139, 145] derived perhaps the first useful result for planning pushing operations. He discovered a simple rule for determining the rotation sense of a pushed object that depends on the center of friction of the object, not its precise support distribution. This is a key result, as the precise support distribution is usually indeterminate. Peshkin and Sanderson [162] followed this work by finding bounds on the velocity direction of an object pushed at a point contact, and these results were used by Lynch [129] to find stable pushing motions for line contact. When the support distribution of the object is known, the *limit surface* of Goyal *et al.* [84, 85] is a convenient geometric interpretation of the relationship between support friction forces and slider motions. Alexander and Maddocks [9] considered the other extreme, when only the geometric extent of the support area is known, and described techniques to bound the possible motions of a pushed object. A comparison of the quasistatic and dynamic initial motions of a pushed object was given by Pham *et al.* [165].

Brost [41] and Mason and Brost [142] extended Mason's results to identify stable parallel-jaw grasping motions for polygonal objects, and Goldberg [82] developed a planner to find a sequence of parallel-jaw grasps to orient a polygonal part in an unknown initial orientation. Mani and Wilson [136] described a system for orienting a part in an unknown orientation by executing a series of linear pushes with a fence. Peshkin and Sanderson [163], and later Brokowski *et al.* [40], considered a similar problem, where the part is carried on a conveyor belt and reoriented by interactions with fences suspended above the belt. Akella and Mason [7] showed how to use a sequence of linear pushes with a fence to move a polygonal object from any initial configuration to any goal configuration. Brost [43] demonstrated linear pushing motions which result in a goal pusher/object equilibrium configuration by integrating the object's motion on the configuration space obstacle defined by the shapes of the pusher and the object. This is like "catching" the object by pushing it. Balorda [17, 18] has investigated catching by pushing with two points of contact. Mason [140] showed how to synthesize robot pushing motions to slide a block along a wall. Kurisu and Yoshikawa [112] studied the problem of finding open-loop point-contact pushing motions to move an object to a desired configuration.

Open-loop pushing with point contact is inherently unstable, and many researchers have constructed point contact pushing control systems using visual (Inaba and Inoue [97]; Gandolfo *et al.* [80]; Salganicoff *et al.* [180]; Narasimhan [156]), tactile (Okawa and

Yokoyama [158]; Lynch *et al.* [131]), and infrared ranging feedback (unpublished work by Latombe and colleagues [222]). Learning (Mahadevan and Connell [135]; Miura [150]; Salganicoff [181]; Zrimec [227]) and friction parameter estimation (Lynch [130]; Yoshikawa and Kurisu [224]) have also been proposed to improve control of pushing. By careful modeling of frictional and dynamic forces during pushing, Jia and Erdmann [101] demonstrated that it is possible to determine the configuration of a known object from tactile data during a blind, open-loop push. The pushing model provides the information necessary to derive the object's configuration from the tactile data.

Box pushing seems to be a favorite task for mobile robots like Shakey (Fikes and Nilsson [78]) and his successors (Mahadevan and Connell [135]; Okawa and Yokoyama [158]; Latombe and colleagues [222]). Donald *et al.* [65] and Brown and Jennings [45] demonstrated controlled pushing using two cooperating mobile robots which communicate only through the motion of the sliding object. Matarić *et al.* [146] demonstrated box pushing by two walking robots coordinated by a simple set of behavioral rules.

Dynamic Nonprehensile Manipulation

Koditschek [107] provides a thoughtful review of seminal works in the area of dynamic manipulation. Some examples of dynamic nonprehensile manipulation are discussed below.

Parts Reorienting In research closely related to the work described in Part II, Arai and Khatib [11] used dynamic forces to roll a cube sitting on a paddle held by a PUMA robot arm. The paddle has three degrees of motion freedom in the vertical plane. The paddle first imparts an angular velocity to the object, causing it to begin to roll about the desired edge, then accelerates downward to minimize the impact at the end of the roll. The approach assumes no slip at the rolling contact. Part II of this thesis addresses the problem of planning such trajectories automatically, explicitly considering friction constraints and manipulator kinematic and dynamic constraints, especially for low-degree-of-freedom manipulators.

Parts Feeding The SONY APOS parts feeder (Hitakawa [92]) and the bowl feeder (Boothroyd *et al.* [35]) are examples of dynamic nonprehensile manipulation that have been successfully applied in industry. The design of these systems is primarily based on experience and trial-and-error, but recent work on the design of bowl feeders is aimed at partially automating the process (Boothroyd [34]; Caine [48]; Christiansen *et al.* [55]). Singer and Seering [192], Böhringer *et al.* [32], and Swanson *et al.* [207] studied related, simplified problems of using a vibrating planar support surface to bring parts to a desired configuration.

Catching In catching, impact is inevitable. Some recent work on the mechanics of impact includes that of Stronge [200, 201] and Wang and Mason [214] for the planar case and Bhatt and Koechling [25] for the spatial case. Wang [213] considered the problem of arranging a sequence of impacts to catch a disk. Brost [42, 43] ignored the sequence of impacts but was able to design a catcher (for a particular polygonal part dropped in a gravity field) with the following important property: if the part is caught, then it is guaranteed to end up in a known equilibrium configuration. This idea is also used in the APOS parts-orienting system (Hitakawa [92]), which uses a vibrating pallet with specially shaped depressions to capture parts in the desired orientation. The vibration is designed to hold a part in the correct orientation, essentially a dynamic grasp, but to eject a part in the wrong orientation. The result is a tray which is (mostly) full of oriented parts.

The juggling robot of Sakaguchi *et al.* [178, 179] catches a ball with a cup-shaped effector. Velocity-matching and the shape of the cup make the catch robust. Slotine's catching robot uses visual data to estimate the flight of a ball, matches the trajectory of the end-effector to that of the ball, and then grabs it [194].

Burridge *et al.* [47] described three different "mirror laws" for controlled juggling, catching, and palming a ball with a planar paddle. By switching control laws from juggling to catching to palming, the robot transitions from juggling the ball to balancing it on the paddle. Other work on catching focuses on designing the manipulator configuration and compliance to reduce impact forces for tasks such as capturing a satellite (Yoshikawa and Yamada [223]).

Throwing Learning approaches have been applied to improve a robot's ability to throw (Aboaf *et al.* [2]; Schneider and Brown [188]). Throwing a club using a dynamic grasp was previously reported by Mason and Lynch [143, 144].

Batting Batting combines catching and throwing into a single collision. Robot juggling by batting has been demonstrated for one or two pucks with a planar juggler (Bühler and Koditschek [46]), a ball in space (Aboaf *et al.* [3], Rizzi and Koditschek [175]), and two balls in space (Rizzi and Koditschek [176]). The problem of batting a planar polygon to a desired stable orientation was addressed by Zumel and Erdmann [229]. Schaal *et al.* [186, 185] used batting in their "devil sticking" robot. Andersson [10] built a machine to play ping-pong, which is an adversarial form of batting.

Striking an object sitting on a frictional planar surface can be used for micropositioning parts with an accuracy greater than the robot's positioning accuracy (Higuchi [91]; Huang *et al.* [96]).

Running Locomotion can be viewed as "self manipulation." Raibert [168] described a class of running machines that are dynamically stable. These machines can also perform gymnastic feats such as flips (Hodgins and Raibert [93]). McGeer [147] constructed a walking machine that is simplicity itself: there are no motors or sensors. The structure and mass parameters of the walker were chosen so walking down a slight incline is passively stable. A later version of the walker used knees to prevent toe-stubbing [148].

Other Forms of Nonprehensile Manipulation

Tray-tilting is another example of nonprehensile manipulation, where a planar part in a tray is positioned and oriented by a sequence of tilts of the tray. Erdmann and Mason [74] described a planner which considers the task mechanics, and Christiansen [54] constructed an agent to learn to perform the task more reliably. Erdmann *et al.* [75] also considered the case of orienting three-dimensional polyhedra by tilting the perfectly rough support surface.

Sawasaki *et al.* [184] investigated tumbling as a manipulation primitive. During a tumbling operation, two robot fingers move in concert to cause the polyhedral object to rotate about an edge of the object in contact with the floor. Aiyama *et al.* [5] proposed a similar pivoting operation, where the object is rotated about a vertex in contact with the floor. During both of these operations, the floor acts as an extra finger.

Other examples of nonprehensile manipulation include rotating an object in a gravity field using two frictionless point contacts (Abell and Erdmann [1]); cooperative manipulation by two nonprehensile palms (Erdmann [72]; Zumel [228]); and various forms of whole-arm manipulation (Salisbury [182]; Salisbury *et al.* [183]).

1.3.3 Path Planning with Grasp Constraints

To plan pushing paths, we must find (1) a sequence of robot pushing motions and (2) the robot's motion between pushes. When the robot and the object move together, the motion is subject to a set of pushing constraints. A version of this problem was studied by Wilfong [218], who described an algorithm for a convex polygon translating among obstacles and pushing another convex polygon at edge contacts, without considering pushing direction constraints. In the terminology of Alami *et al.* [8], a path between pushing contacts is a *transit path*, and a pushing path is called a *transfer path*. They described an approach to planning both the transit and transfer paths using *manipulation graphs*. In their problem, the robot and the manipulated object translate in a planar workspace with obstacles, and the robot can grasp the object at a discrete set of locations and place the object at a discrete set of configurations. The planner of Dacre-Wright *et al.* [59] addresses the case of a convex planar robot and object where the robot can grasp the object at any contact. Koga and Latombe [110, 109] used manipulation graphs to find plans for a multi-arm robot manipulating objects among obstacles.

The task-level pick-and-place planning system HANDEY is described by Lozano-Pérez *et al.* [127, 128]. Grasps are designed to make carrying the object to the goal configuration feasible in the presence of obstacles. An introduction to motion planning with grasp constraints is given by Latombe [115].

1.3.4 Motion Planning and Control of Underactuated Systems

Results from nonlinear control theory are useful in characterizing the controllability of nonholonomically constrained robot systems. Good introductions to the field of nonlinear geometric control theory are provided by Isidori [98] and Nijmeijer and van der Schaft [157], and the text by Boothby [33] is an excellent introduction to many of the differential geometric concepts used in nonlinear control. The Lie Algebra Rank Condition, an important nonlinear analog to the Kalman controllability rank condition of linear control theory, and its implications are discussed by Brockett [38], Haynes and Hermes [88], Hermann and Krener [89], Jurdjevic [102], Sussmann and Jurdjevic [206], and Sussmann [204, 202]. Although there is no general necessary and sufficient condition for the controllability of nonlinear systems, Sussmann [205] derived a general sufficient condition for small-time local controllability. Even if a nonholonomic system is fully controllable, however, Brockett [39] showed that it may not be stabilizable to a single equilibrium point using smooth state feedback. This suggests the use of algorithmic or nonsmooth control.

An introduction to nonholonomic motion planning and the use of nonlinear control theory in robotics is given in the texts by Latombe [115], Murray *et al.* [154], and Li and Canny [124]. The structure of *mechanical* control systems is used by Bloch *et al.* [28, 29] and Lewis and Murray [122] to derive controllability results. Bloch *et al.* provide a general framework for the control and stabilization of second-order mechanical systems subject to a set of nonholonomic constraints which are linear in the velocity of the system. Lewis and Murray studied "simple" mechanical systems where the Lagrangian is kinetic energy minus potential energy and the directions of applicable control forces are a function of configuration alone. A system is called *equilibrium controllable* if, for any two equilibrium configurations, there exists a control which transfers the system from one to the other. They presented sufficient conditions for this property based on Sussmann's general theorem. Other important work related to the controllability of mechanical systems includes that of San Martin and Crouch [138] and Bloch and Crouch [27].

During nonprehensile manipulation, the object can be considered a link connected to the manipulator by a three (planar) or six (spatial) degree-of-freedom unactuated joint. As shown by Oriolo and Nakamura [159], unactuated joints typically result in nonintegrable second-order constraints. Research on controlling unactuated joints by dynamic coupling includes controlling an inverted pendulum (Higdon and Cannon [90]); controlling the swinging motion of a high-bar robot (Takashima [209]) or the Acrobot (Hauser and Murray [87]; Spong [198]); the control of running and hopping robots (Berkemeier and Fearing [24]); and a series of papers by Arai *et al.* [13, 12, 14] (and related work by Bergerman *et al.* [23] and Yu *et al.* [225]) examining controllability and describing control algorithms for robots with some joints equipped only with brakes. Jain and Rodriguez [100] studied the kinematics and dynamics of underactuated manipulators and described an inverse dynamics algorithm.

Other researchers have studied underactuated manipulation using kinematic coupling instead of dynamic coupling. Examples include pivoting an object within a grasp (Brock [37]; Carlisle *et al.* [52]; Rao *et al.* [171]); rolling an object between two palms (Bicchi [26]); rolling a ball on a plane or another ball (Li and Canny [123]); and controlling an n-link planar manipulator with only two motors using nonholonomic gears (Sørdalen *et al.* [197]. Perhaps the most common example of underactuated manipulation is assembly, where robots possessing just a few degrees-of-freedom are required to assemble a set of parts with a larger number of degrees-of-freedom (Koditschek [105, 106]). Mobile robots are perhaps the most heavily-studied type of kinematically constrained nonholonomic system. Mobile robots are discussed briefly below.

The *snakeboard* is an interesting system similar to a skateboard, except the directions of the front and rear wheels can be controlled independently. By shifting his or her weight, the rider produces motion of the snakeboard through rolling of the wheels. A simplified model of the snakeboard has been studied by Ostrowski *et al.* [161, 160].

A key feature of underactuated systems is the inability to independently drive all state variables to their desired values using smooth state feedback (Brockett [39]). Thus we need some sort of switching feedback law or open-loop control to reach the goal. Strategies for generating open-loop controls for nonholonomically constrained systems have been proposed by Murray and Sastry [155], Lafferiere and Sussmann [113], Sussmann [202], Sontag [196], and Leonard and Krishnaprasad [121]. Divelbiss and Wen [61] and Fernandes *et al.* [77, 76] both proposed an iterative optimization approach akin to Newton's method, where the controls are approximated by finite Fourier series. At each step, the coefficients of the Fourier terms are adjusted to minimize a merit function. In the work of Divelbiss and Wen, the merit function is given by the error at the final state and potential functions representing obstacles. They illustrated the approach by planning the motion of a tractor pulling a trailer. Fernandes *et al.* used a merit function quadratic in the Fourier coefficients and the error at the final state. They presented examples of controlling the attitude of a falling cat and a space platform/manipulator system.

The approach to trajectory planning for dynamic nonprehensile manipulation employed in this thesis is similar to these works. An important characteristic of the trajectory planning problem for dynamic nonprehensile manipulation is the inequality constraints imposed by unilateral frictional contact.

There are many other nonholonomic systems of interest, including underwater vehicles and space robots; see (Bloch *et al.* [29]; Latombe [115]; Murray *et al.* [154]) for references. We note that dynamic nonprehensile manipulation is similar to controlling the attitude of a satellite (Crouch [58]), except the "jets" are unilateral frictional contacts.

Motion Planning for Mobile Robots

Particularly relevant to our work on planning pushing paths is path planning for car-like mobile robots, which move in an $\mathbf{R}^2 \times S^1$ configuration space and share similar motion constraints. For a car-like mobile robot, a nonholonomic equality constraint is given by the rolling constraint at the wheels, and an inequality constraint is given by the minimum turning radius. Controllability results and motion planners for this class of robots, and mobile robots pulling trailers, have been reported by Fortune and Wilfong [79], Barraquand and Latombe [20, 21], Latombe [114, 115], Laumond [116, 117, 118], and Laumond *et al.* [119]. Jacobs and Canny [99] constructed a motion planner for finding paths among obstacles for a mobile robot which does not reverse. This planner uses a set of canonical shortest C^1 curves with a lower bound on the turning radius, due to Dubins [66]. Reeds and Shepp [173] extended Dubins' results by finding the set of canonical shortest paths which include reversals. This result has also been used in planning paths among obstacles (Laumond *et al.* [119]).

The pushing planner presented in Chapter 6 is an adaptation of an algorithm by Barraquand and Latombe [21] for planning the motion of a car-like mobile robot.

1.3.5 Optimal Trajectory Planning

The problem of finding open-loop controls of nonholonomically constrained systems is related to the optimal trajectory planning problem for fully actuated systems. To increase productivity, much work has focused on time-optimal trajectory planning. In early work, Kahn and Roth [104] studied time-optimal trajectory planning for a manipulator using linearized equations of motion. This approximation allowed them to find the switching curves and derive a bang-bang control to move from one state to another. To simplify the problem when full nonlinear dynamics are modeled, later work assumed that the manipulator path is specified, and only the times along the path are left to be determined. Hollerbach [94] outlined an algorithm that finds a uniform time-scaling of a trajectory to make it feasible given actuator torques. He also showed that it may be necessary to speed up a trajectory to make it dynamically feasible. Vukobratović and Kirćanski [212] split the trajectory into short segments and solved for the actuator effort during each segment using dynamic programming. This permits a nonuniform time-scaling of the trajectory, but it is also much more computationally intensive than Hollerbach's approach.

Shin and McKay [191] and Bobrow *et al.* [30] independently derived similar, and much more efficient, algorithms for determining the time-optimal manipulator trajectory along a given path. The algorithms consider full arm dynamics and actuator torque limits. A computational improvement to the algorithm was reported by Slotine and Yang [195]. Tan and Potts [210] described a time-optimal trajectory planner for a robot modeled using discrete dynamics and considering velocity, torque, and jerk constraints. De Luca *et al.* [60] used sequential quadratic programming to find the timing of knot points in a spline representation of the path. (Other objective functions are also possible.)

These algorithms and later refinements have largely solved the problem of time-scaling a given path. For point-to-point control where the path is unimportant, however, the question remains of how to find both the both the optimal path and its time-scaling. Rajan [169] addressed this problem by first using a simple spline path between the start and goal points. The algorithm of Bobrow *et al.* [30] is used to find the time necessary for that path. Then the path representation is refined by adding another knot point, and the knots are locally varied to find the local minimum-time path. This process is repeated until path refinements yield little improvement. However, this algorithm can suffer from local minima depending on the initial path representation. To address this, Sahar and Hollerbach [177] discretized the robot's joint space and considered all possible paths from the start to the goal. The time between points in the discretized space was found using Hollerbach's scaling algorithm. Shiller and Dubowsky [190] described an alternative approach which is similar in spirit. They also considered all paths on a discretized version of the free space, and they used the algorithm of Bobrow *et al.* to minimize the time for each path. The amount of computation is lessened by pruning the possible paths according to upper bounds on the estimated trajectory time.

A very different approach was taken in a series of papers on *kinodynamic* planning (Canny *et al.* [50], Donald and Xavier [63, 64]). Grid-based algorithms are presented for finding provably-nearly-time-optimal trajectories in obstacle fields for robots with coupled and decoupled dynamics. The resulting actuator profiles are bang-bang and the paths avoid obstacles by a speed-dependent safety margin. Canny *et al.* [49] also proposed an exact algorithm for finding the time-optimal trajectory for a point mass with decoupled dynamics in the obstacle-free plane. All of these kinodynamic algorithms suffer from high computational complexity (Xavier [220]).

Optimal control with objective functions other than time has also been considered. Yen and Nagurka [221] represented robot trajectories as the sum of a fifth-order polynomial and a finite Fourier series and used nonlinear programming to solve for minimum-energy trajectories. Chen [53] took a similar approach, representing the robot trajectory by uniform cubic B-splines and using sequential quadratic programming to handle linear and nonlinear equality and inequality constraints. Recently, Žefran and Kumar [226] proposed another method for handling inequality constraints in optimal control problems.

In an effort to automatically produce physically realistic animation, Witkin and Kass [219] model a character's physical structure and find actuator programs that accomplish a goal while optimizing some function. They use the example of an articulated lamp which is required to perform a hop. The position of the lamp is represented at discrete timesteps, and an initial guess is iteratively modified using sequential quadratic programming. The goal is to minimize some function, such as the total energy expended by the lamps's "muscles," while satisfying constraints that the object's motion be consistent with physical laws. The result is a physically feasible motion which locally optimizes the cost function. This work has been continued and extended; see, for instance, (Cohen [56]).

1.3.6 Minimalism

The minimalist approach to robotics is the attempt to find the simplest systems capable of performing a given task or class of tasks. The complexity of a system may be measured in terms of the sensors, effectors, computation, communication, etc., necessary to accomplish the task (Böhringer *et al.* [31]). This thesis explores minimal effector solutions to manipulation tasks.

McGeer's walking machines [147, 148] are dramatic examples of minimalism; a stable walking gait is obtained without sensors, actuators, or computation. Another example is the tray-tilting system built by Erdmann and Mason [74], which can orient a part in a random configuration in the tray without sensing. Donald *et al.* [65] have begun to develop a theory of information invariants to compare the information content embedded in various subsystems, such as sensing, actuation, computation, and communication. Erdmann [73] makes the case that the true test of a sensor is whether it gives information that allows the system to proceed toward the goal. *Progress cones* bound the set of directions that move toward the goal, and these progress cones can be used to design the simplest effective sensors. Canny and Goldberg [51] argue that RISC (reduced intricacy in sensing and control) robotics using simple devices results in cheaper, more flexible systems.

1.3.7 Sufficiency Results in Manipulation

A primary focus of this thesis is to demonstrate the sufficiency of nonprehensile manipulation for parts transfer. Related results in manipulation include bounds on the number of point fingers necessary for grasping (Mishra *et al.* [149]; Markenscoff *et al.* [137]); the demonstration of the controllability of a ball rolling on a plane or another ball (Li and Canny [123]); and the classification of orientable parts by sensorless parallel-jaw grasping sequences (Goldberg [82]). Goldberg [83] provides an interesting discussion on sufficiency and completeness results in robot motion planning.

1.4 Overview

Part I addresses quasistatic nonprehensile manipulation by pushing. Chapters 4 and 5 discuss the mechanics and controllability of pushing, and Chapter 6 describes an algorithm for finding stable pushing paths among obstacles. Chapter 7 describes an application of the results of Part I to a parts feeding problem.

Part II roughly mirrors the development of Part I. Chapter 9 addresses the controllability of objects during dynamic nonprehensile manipulation, and Chapter 10 describes an algorithm to find dynamic nonprehensile manipulation plans. Simulation and experimental results are given in Chapter 11. A discussion of the results is presented in Chapter 12.

1.5 Publication Note

Much of Chapters 3–6 has previously appeared in (Lynch and Mason [132, 133]). Chapter 7 is joint work with Srinivas Akella, Wes Huang, and Matt Mason, and has previously appeared in [6].

Chapter 2

Definitions

The problems dealt with in this thesis are planar. The configuration space of the object is $\mathcal{C} = SE(2) = \mathbf{R}^2 \times S^1$. A frame $\mathcal{F}_{\mathcal{W}}$ is fixed in the world, and the configuration of the object is written $\mathbf{q} = (x_w, y_w, \phi_w)^T$. The phase space of the object is the tangent bundle $T\mathcal{C} = SE(2) \times \mathbf{R}^3$, and the object's phase is given by $(\mathbf{q}, \dot{\mathbf{q}})$. The *state* of the object is simply its configuration $\mathbf{q}$ in the case of quasistatic pushing, while the state is given by $(\mathbf{q}, \dot{\mathbf{q}})$ in the case of dynamic manipulation. The tangent (velocity) space at $\mathbf{q}$ in the quasistatic case is written $T_{\mathbf{q}}\mathcal{C}$, and the tangent space at $(\mathbf{q}, \dot{\mathbf{q}})$ in the dynamic case is written $T_{(\mathbf{q},\dot{\mathbf{q}})}T\mathcal{C}$.

In this thesis, nonprehensile contact is limited to the two basic planar contact types: point contact and line contact. Line contact is defined by any contact where all the points of contact are collinear, with contact normals perpendicular to the line (see Figure 2.1). Assuming that the friction coefficient at all contacts is the same, any line contact is equivalent to two contact points at the ends of the line segment. Point and line contact are the only types of contact which can occur if the manipulator is a straight edge. The set of available contact points on the object is a closed, piecewise smooth curve , , which typically forms the perimeter of the object.

Friction is assumed to conform to Coulomb's law (Coulomb [57]). The force applied at a frictional contact must satisfy the inequality $|f_t| \leq \mu f_n$, where f_t is the tangential frictional component, f_n is the normal component, and μ is the coefficient of friction. During sliding contact, $|f_t| = \mu f_n$, and the frictional force is directed opposite the direction of sliding. (For simplicity, static and kinetic friction coefficients are assumed equal. Most of the results presented here can be easily generalized to the case where kinetic friction is less than static friction.)

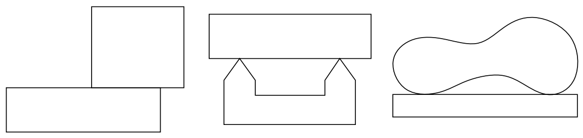


Figure 2.1: Examples of line contact.

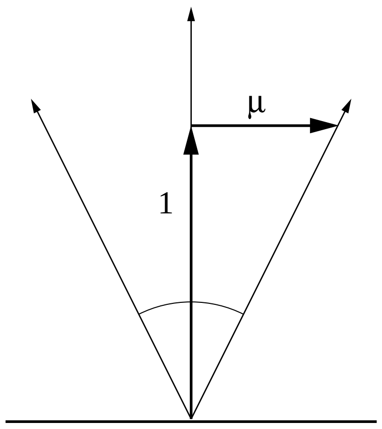


Figure 2.2: A friction cone.

The friction coefficient can be geometrically interpreted as a *friction cone* (generally credited to Moseley [152]). The cone consists of all forces passing through the contact point and making an angle less than or equal to $\tan^{-1}\mu$ with the contact normal (Figure 2.2). By choosing a reference point (usually at the center of mass of the object), the forces and torques implied by the friction cone can be represented as a planar cone in a three-dimensional force-torque space. The set of contact forces arising from multiple friction cones is given by the convex hull of the individual friction cones in this space (Erdmann [68, 71]).

I will use the notation $\mathbf{f} = (f_x, f_y, m)^T$ for generalized forces (wrenches), $\mathbf{v} = (v_x, v_y, \omega)^T$ for generalized velocities (twists), and $\mathbf{a} = (a_x, a_y, \alpha)^T$ for generalized accelerations. These are measured in the body frame, except where noted. A nonzero vector $\mathbf{x}$ is given by the product $x\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the unit vector $\mathbf{x}/|\mathbf{x}|$ and x is the magnitude of $\mathbf{x}$. I will call $\hat{\mathbf{x}}$ the *direction* of $\mathbf{x}$. Thus $\mathbf{f}$ is a force, $\hat{\mathbf{f}}$ is a force direction, and f is a force magnitude. Directions can be represented as points on the unit sphere S^2, giving us the force sphere, the velocity sphere, and the acceleration sphere.

2.1 Controllability Definitions

I will use definitions from nonlinear control theory to discuss the controllability of an object. The system ("the state of the object," or simply "the object") is *controllable from* the state $\mathbf{p}$ if, starting from $\mathbf{p}$, the object can reach every point in the state space. The object is *controllable to* $\mathbf{p}$ if $\mathbf{p}$ is reachable from every point in the state space. The object is *accessible from* $\mathbf{p}$ if the set of states reachable from $\mathbf{p}$ has nonempty interior in the state space.

We can also define local versions of these properties. The object is *small-time accessible from* $\mathbf{p}$ if, for any neighborhood U of $\mathbf{p}$, the set of reachable configurations without leaving U has nonempty interior. The object is *small-time locally controllable from* $\mathbf{p}$ if, for any neighborhood U of $\mathbf{p}$, the set of reachable configurations without leaving U contains a neighborhood of $\mathbf{p}$. Although the phrase "small-time" appears in these terms, time does not appear in their definitions as they are applied in this thesis. For instance, when pushing an object in a tight space, we are not so concerned with the time it takes the manipulator to move between contact configurations. We are more concerned with whether it is possible to maneuver the object in tight spaces.

The phrases "from $\mathbf{p}$" and "to $\mathbf{p}$" can be eliminated in these definitions if they apply to the entire state space. If a system is small-time locally controllable on a subset of its state space, then any path through that subset can be followed arbitrarily closely.

(We note that the terms used for various types of controllability are not completely standard. Other definitions are given by Haynes and Hermes [88], Sussmann and Jurdjevic [206], Hermann and Krener [89], Sussmann [203], and Nijmeijer and van der Schaft [157]. These definitions are not always consistent with each other. Sussmann [204] provides the most complete set of definitions for these nonlinear controllability concepts. For autonomous linear systems, all of these concepts are equivalent.)

2.1.1 Differential Geometry Preliminaries

A nonprehensile system can be viewed as a collection of smooth vector fields on a smooth state manifold, where the vector fields correspond to the possible motions of the object. Determining controllability is then a matter of looking at the state space reachable by following these vector fields. This section reviews some basic facts necessary to apply results from nonlinear control theory, and is intended to aid the reader who is unfamiliar with differential geometry. The presentation is necessarily informal and incomplete. This material can be found in (Boothby [33]; Murray *et al.* [154]; Nijmeijer and van der Schaft [157]; Warner [216]).

The configuration manifold $\mathcal{C}$ is the Lie group $SE(2) = \mathbf{R}^2 \times S^1$, and the tangent bundle $T\mathcal{C}$ is the Lie group $SE(2) \times \mathbf{R}^3$. For the rest of this section, the state manifold of the object is referred to as the m-dimensional manifold M, where it is understood that M corresponds to $\mathcal{C}$ if the system is quasistatic, or $T\mathcal{C}$ if the system is dynamic. The state is denoted $\mathbf{p}$, and the tangent space at $\mathbf{p}$ is $T_{\mathbf{p}}M$.

A *vector field* X on M is a smooth function assigning every point $\mathbf{p}$ of M a tangent vector $X(\mathbf{p})$ in $T_{\mathbf{p}}M$. X is written as a column vector $(x_1, x_2, \ldots, x_m)^T$, where each component x_i is a smooth function of $\mathbf{p}$.

A *Lie algebra* L over $\mathbf{R}$ is a real vector space together with an operator $[\ ,\ ] : L \times L \to L$ (called the *Lie bracket*) with the following properties for all $X, Y, Z \in L$:

1. Bilinearity:
$$[aX + bY, Z] = a[X, Z] + b[Y, Z]$$
$$[X, aY + bZ] = a[X, Y] + b[X, Z]$$

2. Skew commutativity:
$$[X, Y] = -[Y, X]$$

3. The Jacobi identity:

$$[X,[Y,Z]]+[Y,[Z,X]]+[Z,[X,Y]]=0$$

The vector space of all smooth vector fields on M forms an infinite-dimensional Lie algebra, where the Lie bracket $[X,Y]$ of vector fields X and Y is given locally as

$$[X,Y](\mathbf{p})=\frac{\partial Y(\mathbf{p})}{\partial \mathbf{p}}X(\mathbf{p})-\frac{\partial X(\mathbf{p})}{\partial \mathbf{p}}Y(\mathbf{p}).$$

The vector field $[X,Y]$ is essentially the direction of motion obtained by following X for time t, Y for t, $-X$ for t, and $-Y$ for t (for small t). If $[X,Y]=0$, then X and Y *commute*.

The Lie algebra of a finite set of vector fields $\mathcal{X}=\{X_0,X_1,\ldots,X_n\}$ on M is the smallest subalgebra that contains $\mathcal{X}$ and is closed under the Lie bracket.

(The finite-dimensional Lie algebra g of a Lie group G can be identified with the set of tangent vectors to all curves in G at the identity element of G, i.e., the vector space T_eG. The Lie algebra g essentially "generates" the Lie group G. Roughly speaking, the study of controllability asks whether the Lie algebra of vector fields $\mathcal{X}$, corresponding to the control system, can be identified with the tangent space $T_{\mathbf{p}}M$ at points $\mathbf{p}$. If the vector fields are *left invariant* on a Lie group G, it is sufficient to look only at $\mathbf{p}=e$, the identity element of G.)

A *distribution* Δ defined by a set of vector fields $\mathcal{X}$ assigns a linear subspace of the tangent space $T_{\mathbf{p}}M$ at every point $\mathbf{p}$ of M:

$$\Delta_{\mathbf{p}}=\operatorname{span}\{X_0(\mathbf{p}),X_1(\mathbf{p}),\ldots,X_n(\mathbf{p})\}.$$

If the dimension of the subspace $\Delta_{\mathbf{p}}$ does not vary with $\mathbf{p}$, the distribution is *regular*. A distribution Δ is *involutive* if it is closed under the Lie bracket; that is, for all $X,Y\in\Delta$, $[X,Y]\in\Delta$.

A submanifold N of M is an *integral manifold* of a regular distribution Δ if, for all $\mathbf{p}\in N$, $\Delta_{\mathbf{p}}=T_{\mathbf{p}}N$. The relationship between integral manifolds and involutive distributions is given by Frobenius' theorem.

Theorem 2.1 (Frobenius) *A regular distribution is integrable if and only if it is involutive.*

The integral manifolds of a distribution form a *foliation*, and the integral manifold through $\mathbf{p}$ is a *leaf* of the foliation.

2.1.2 Application to Nonlinear Control

Given a finite set of vector fields $\mathcal{X}$ on the manifold M, define $B_0(\mathcal{X})=\mathcal{X}$ and $B_{k+1}(\mathcal{X})=B_k(\mathcal{X})\cup\{[X,Y]\text{ for all }X,Y\in B_k(\mathcal{X})\}$. Then the Lie algebra $L(\mathcal{X})$ (sometimes called the *accessibility algebra*) is spanned by elements of $B_\infty(\mathcal{X})$. We define $Br(\mathcal{X})$ to be the smallest subset of $L(\mathcal{X})$ that contains $\mathcal{X}$ and is closed under bracketing. The elements of $Br(\mathcal{X})$ are not linearly independent; algorithms for constructing a *Philip Hall basis* can be used to identify a subset of $Br(\mathcal{X})$ that forms a basis of $L(\mathcal{X})$ (Serre [189]).

Of all the controllability properties, small-time accessibility can be checked by a direct test of $L(\mathcal{X})$. A control system is small-time accessible from $\mathbf{p}$ if it satisfies the Lie Algebra Rank Condition, which states that $L(\mathcal{X})$, evaluated at $\mathbf{p}$, must span the tangent space $T_{\mathbf{p}}M$. For any neighborhood U of the initial state $\mathbf{p}$, the system can reach a set with nonempty interior in M without leaving U.

If the system satisfies certain special conditions, the Lie Algebra Rank Condition also implies small-time local controllability—the reachable set without leaving U contains a neighborhood of the initial state $\mathbf{p}$. One example is the class of symmetric systems, where all vector fields can be followed forward and backward. Although there are no known necessary and sufficient conditions for small-time local controllability for general nonlinear systems, Sussmann [205] derived a sufficient condition that generalizes many of the sufficient condtions in the literature.

Controllability can be proven by "patching" together open sets of the state space where small-time local controllability holds. In many cases, however, the global controllability of a system cannot be deduced from purely local considerations.

Part I

Quasistatic Pushing

Chapter 3

Quasistatic Pushing

Again if a given force move a given weight a certain distance in a certain time and half the distance in half the time, half the motive power will move half the weight the same distance in the same time.

— Aristotle, *Physica*

In the first part of this thesis I examine quasistatic nonprehensile manipulation by pushing. Pushing provides a simple and practical solution to the problem of positioning and orienting objects in the plane, particularly when the manipulator lacks the size, strength, or dexterity to grasp and lift them. Because the object is not firmly grasped, however, the forces that can be applied are limited, and therefore the possible motions of the object are limited. Unlike pick-and-place, the "grasp" (pushing contact) and manipulator path cannot be decoupled. By modeling the support friction forces, however, we can simultaneously design the pushing contact and manipulator path such that the contact resists all sliding friction forces during the motion.

This problem is made difficult by the indeterminacy of the distribution of support forces of the pushed object. The precise motion of a pushed object is usually unpredictable. If there are two or more pushing points, however, there may exist a space of pushing directions that admit only a single solution to the motion of the object: the motion that causes the object to maintain its configuration relative to the pusher. The object is effectively rigidly attached to the pusher, and the push is called a stable push. A pushing path is formed by stringing together stable pushes, as in Figure 3.1.

The goal is to develop algorithms to automatically find pushing plans to position and orient parts in the plane. Toward this end, I study the following three issues in pushing:

Mechanics. How does an object move when it is pushed? I describe a procedure that identifies a set of stable pushing directions when the pusher makes line contact with the object.

Controllability. The directions an object can move during pushing are limited due to the

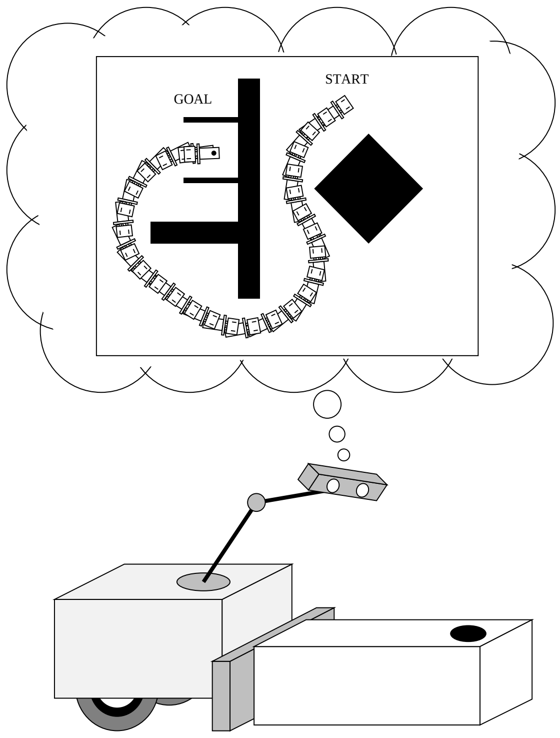


Figure 3.1: A mobile robot pushing a box using stable pushes with line contact.

limited set of forces that can be applied by the pusher. Given these limitations, my study of controllability is motivated by questions of whether or not it is possible to push the object to the goal configuration, with and without obstacles. I examine the local and global controllability of objects pushed with either point contact or stable line contact.

Planning. Pushing paths consist of sequences of stable pushes, and the space of stable pushing directions imposes nonholonomic constraints on the motion of the object. I draw on work on path planning for nonholonomic mobile robots by Barraquand and Latombe [21] to construct a planner to find stable pushing paths among obstacles.

3.1 Assumptions

1. All pushing forces lie in the horizontal support plane, and gravity acts along the vertical.

2. The pusher and slider move in the horizontal plane.

3. Friction properties are uniform over the support plane.

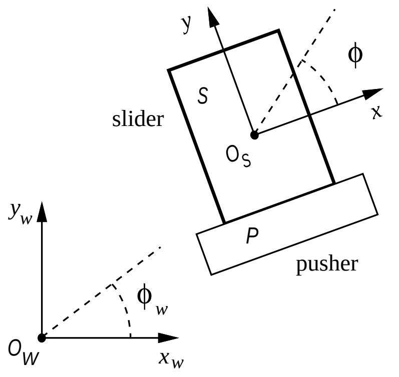


Figure 3.2: The world frame $\mathcal{F}_{\mathcal{W}}$ and the slider frame $\mathcal{F}_{\mathcal{S}}$.

4. Pushing motions are slow enough that inertial forces are negligible. This is the quasistatic assumption. Pushing forces are always balanced by the support frictional forces acting on the object.

3.2 Definitions

The slider $\mathcal{S}$ is a rigid object pushed by a rigid pusher $\mathcal{P}$ at a point or set of points on a closed, piecewise smooth curve , , which typically forms the perimeter of the slider $\mathcal{S}$. A world frame $\mathcal{F}_{\mathcal{W}}$ with origin $O_{\mathcal{W}}$ is fixed in the plane, and a slider frame $\mathcal{F}_{\mathcal{S}}$ with origin $O_{\mathcal{S}}$ is attached to the center of friction of the slider $\mathcal{S}$. (For a uniform coefficient of support friction, the center of friction of the slider is the point in the support plane beneath the center of mass (MacMillan [134]).) The configuration $\mathbf{q} = (x_w, y_w, \phi_w)^T$ describes the position and orientation of the slider frame $\mathcal{F}_{\mathcal{S}}$ relative to the world frame $\mathcal{F}_{\mathcal{W}}$. See Figure 3.2.

Generalized forces $\mathbf{f}$ and velocities $\mathbf{v}$ are always defined with respect to the slider frame $\mathcal{F}_{\mathcal{S}}$. I will sometimes represent a velocity direction by its center of rotation in $\mathcal{F}_{\mathcal{S}}$. The mapping COR$(\cdot)$ maps velocity directions to rotation centers in the slider frame $\mathcal{F}_{\mathcal{S}}$, such that COR$(\hat{\mathbf{v}})$ returns the point about which the velocity direction $\hat{\mathbf{v}}$ is a pure rotation, along with the sense of rotation. The domain of the mapping COR$(\cdot)$ is the velocity sphere, and the range consists of two copies of the plane, one for each rotation sense, and a line at infinity for translations. Figure 3.3 illustrates the mapping from velocity directions to rotation centers.

For the quasistatic pushing problem, we are only concerned with force and velocity directions, not their magnitudes. I assume only that the manipulator is strong enough to move the slider, and that it moves slowly enough to satisfy the quasistatic assumption. A pushing plan generated by the planner in Chapter 6 may be properly thought of as a pushing path, not a trajectory. To generate a manipulator trajectory from this path, times must be assigned to each point along the path such that the quasistatic assumption is satisfied.

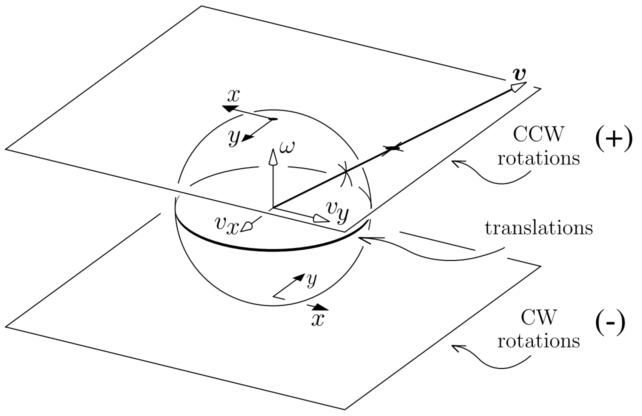


Figure 3.3: The mapping COR($\cdot$) from velocity directions on the unit sphere to rotation centers in the slider frame $\mathcal{F}_S$.

3.3 Overview

In the next chapter I define the pushing control system and some of its basic properties. Using these results, in Chapter 5 I study the mechanics of pushing and the controllability of objects pushed with either point contact or stable line contact. Chapter 6 describes a planning algorithm for repositioning objects among obstacles using stable pushes, and Chapter 7 demonstrates an application of the results of Chapter 5 to a parts feeding problem.

Chapter 4

Controllability with Velocity Constraints

The set of velocity directions that the slider can follow during pushing is limited due to the limited set of force directions that can be applied by the pusher. These limitations constitute a set of nonholonomic constraints: constraints on the velocity of the slider that cannot be integrated to give configuration constraints. For example, a slider that can be pushed in one direction cannot be pulled in the opposite direction by simply reversing the motion of the pusher. Despite these constraints on the motion, we know by experience that it is often possible to move objects to desired configurations by pushing. In this section I formalize these ideas using tools from nonlinear control theory. I defer the problem of determining the motion of a pushed object to Chapter 5.

4.1 The Pushing Control System

The pushing control system can be described abstractly by the autonomous nonlinear control system $\dot{\mathbf{q}} = F(\mathbf{q}, \mathbf{c})$, where $\mathbf{c}$ is the control input describing the pushing contact configuration and the velocity of the pusher in the slider frame $\mathcal{F}_{\mathcal{S}}$. The motion of the slider in the world frame $\mathcal{F}_{\mathcal{W}}$ is a function F of the control input and the configuration of the slider. For the rest of Part I, I will use the following more concrete description of the control system Σ:

$$\Sigma : \quad \dot{\mathbf{q}} = F(\mathbf{q}, \mathbf{c}_u) = X_u(\mathbf{q}),$$

$$\mathbf{q} \in \mathcal{C} = \mathbf{R}^2 \times S^1, \quad u \in \{0, \ldots, n\},$$

$$X_u(\mathbf{q}) = \begin{cases} (0,\ 0,\ 0)^T & \text{if } u = 0 \\ \begin{pmatrix} \cos\phi_w & -\sin\phi_w & 0 \\ \sin\phi_w & \cos\phi_w & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{v}_{ux} \\ \hat{v}_{uy} \\ \hat{\omega}_u \end{pmatrix} & \text{otherwise.} \end{cases}$$

A nonzero u chooses one of n distinct combinations of contact configurations and pushing velocities in the slider frame $\mathcal{F}_{\mathcal{S}}$. Associated with each control $\mathbf{c}_u$ is a vector field X_u describing the motion of the slider in the world frame $\mathcal{F}_{\mathcal{W}}$. The tangent vector $X_u(\mathbf{q})$ is the (unit) velocity of the slider in the world frame $\mathcal{F}_{\mathcal{W}}$, and $\hat{\mathbf{v}}_u$ is the (unit) velocity of the slider in the slider frame $\mathcal{F}_{\mathcal{S}}$. The set of nonzero vector fields X_u is denoted $\mathcal{X}$, and the set of nonzero velocity directions $\hat{\mathbf{v}}_u$ in the slider frame $\mathcal{F}_{\mathcal{S}}$ is $\hat{\mathcal{V}}$. Each of the n nonzero controls results in a distinct velocity direction.

Three aspects of the control system Σ bear mentioning. (1) The absence of a drift vector field (a vector field that is only a function of the state of the slider) implies that the slider will not move when it is not pushed ($u = 0$). This is a consequence of the quasistatic assumption: the slider's kinetic energy is instantly dissipated by support friction, and the slider's state is merely its configuration. (2) For any constant control, the slider's velocity direction is constant in the slider frame $\mathcal{F}_{\mathcal{S}}$. (3) The control system is not necessarily symmetric. In general, it is not possible to follow a vector field X_u backwards. This is due to the unilateral nature of frictional contact: pushing forces must have a nonnegative component in the direction of the contact normal.

4.2 Controllability of the Pushing Control System

For the control system Σ, any controllability property that holds at any $\mathbf{q}$ on an open set of the configuration space $\mathcal{C}$ also holds at any other $\mathbf{q}$ on an open set of $\mathcal{C}$. Similarly, any property that does not hold for some $\mathbf{q} \in \mathcal{C}$ does not hold for any $\mathbf{q} \in \mathcal{C}$. The Lie algebra $L(\mathcal{X})$ is spanned by $B_1(\mathcal{X})$, the vector fields and their Lie brackets.

If $n = 1$ for the control system Σ, then the slider is confined to a one-dimensional integral curve of X_1, and the control system Σ is not accessible. If $n = 2$, the Lie algebra $L(\mathcal{X})$ is spanned by X_1, X_2, and $X_3 = [X_1, X_2]$, where

$$X_1 = (\hat{v}_{1x}\cos\phi_w - \hat{v}_{1y}\sin\phi_w, \hat{v}_{1x}\sin\phi_w + \hat{v}_{1y}\cos\phi_w, \hat{\omega}_1)^T,$$

$$X_2 = (\hat{v}_{2x}\cos\phi_w - \hat{v}_{2y}\sin\phi_w, \hat{v}_{2x}\sin\phi_w + \hat{v}_{2y}\cos\phi_w, \hat{\omega}_2)^T,$$

$$\begin{gathered} X_3 = [X_1, X_2] = \\ (\hat{\omega}_1(-\hat{v}_{2x}\sin\phi_w - \hat{v}_{2y}\cos\phi_w) + \hat{\omega}_2(\hat{v}_{1x}\sin\phi_w + \hat{v}_{1y}\cos\phi_w), \\ \hat{\omega}_1(\hat{v}_{2x}\cos\phi_w - \hat{v}_{2y}\sin\phi_w) + \hat{\omega}_2(-\hat{v}_{1x}\cos\phi_w + \hat{v}_{1y}\sin\phi_w), \\ 0)^T. \end{gathered}$$

The dimension of $L(\mathcal{X})$ is given by $\text{rank}(X_1\ \ X_2\ \ X_3)$. If the determinant of this matrix is nonzero, then its rank is three. A simple calculation yields

$$\det(X_1\ \ X_2\ \ X_3) = (\hat{\omega}_2\hat{v}_{1x} - \hat{\omega}_1\hat{v}_{2x})^2 + (\hat{\omega}_2\hat{v}_{1y} - \hat{\omega}_1\hat{v}_{2y})^2.$$

The determinant is only zero if (1) $\hat{\omega}_1 = \hat{\omega}_2 = 0$ or (2) $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$ are multiples of each other. (Since $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$ are distinct unit vectors, this condition is equivalent to $\hat{\mathbf{v}}_1 = -\hat{\mathbf{v}}_2$.) If condition (1) holds, the slider cannot rotate. If condition (2) holds, the slider is confined to a one-dimensional curve of the configuration space $\mathcal{C}$. If neither of these conditions hold, the Lie bracket operation has essentially created a new linearly independent control vector field and the Lie Algebra Rank Condition is satisfied.

Proposition 4.1 *The control system Σ is small-time accessible if and only if the set of velocity directions $\hat{\mathcal{V}}$ contains two velocity directions, $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$, such that they are not both translations ($\hat{\omega}_1 \neq 0$ or $\hat{\omega}_2 \neq 0$) and $\hat{\mathbf{v}}_1 \neq -\hat{\mathbf{v}}_2$.*

Remark: As noted earlier, small-time accessibility implies accessibility on the configuration space $\mathcal{C}$. Here we note that any control system Σ that is accessible must also be small-time accessible. Thus the conditions of Proposition 4.1 are also necessary and sufficient for accessibility.

Proposition 4.1 is a straightforward generalization of a result due to Barraquand and Latombe [21] that states that the Lie Algebra Rank Condition is satisfied for any car-like mobile robot that can take at least two steering angles. A car-like mobile robot can drive both forward and backward, and this symmetry, coupled with small-time accessibility, implies small-time local controllability. As we have already noted, the pushing control system Σ may not be symmetric, so small-time local controllability does not follow from small-time accessibility.

For some systems, however, accessibility may imply controllability even when the system is not small-time locally controllable (Jurdjevic [102]; Jurdjevic and Sussmann [103]; Brockett [38]; Sussmann [204]; Crouch [58]). Consider the task of setting the minute-hand of a watch if it can only be rotated in a clockwise direction. The configuration of the minute-hand is not small-time locally controllable, but the topology of its configuration manifold S^1 renders the minute-hand's configuration controllable.

For control systems Σ, it is easily shown that accessibility implies controllability on the configuration space $\mathcal{C}$, and the conditions of Proposition 4.1 are also necessary and sufficient for controllability.

Proposition 4.2 *For the control system Σ, accessibility implies controllability. The slider $\mathcal{S}$ may be moved from any configuration to any other configuration in the obstacle-free plane if and only if the set of velocity directions $\hat{\mathcal{V}}$ contains two velocity directions, $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$, such that they are not both translations ($\hat{\omega}_1 \neq 0$ or $\hat{\omega}_2 \neq 0$) and $\hat{\mathbf{v}}_1 \neq -\hat{\mathbf{v}}_2$.*

Proof: Accessibility is a necessary condition for controllability. The following argument shows that it is also sufficient. First consider the case $\hat{\omega}_1 \neq 0$, $\hat{\omega}_2 = 0$. The slider may reach any configuration $\mathbf{q}_{goal}$ from any other configuration $\mathbf{q}_{init}$ by following X_1, then X_2, then X_1. The rotation center COR($\hat{\mathbf{v}}_1$) is located at a point $R_1 = (-\hat{v}_{1y}/\hat{\omega}_1, \hat{v}_{1x}/\hat{\omega}_1)$ in the slider frame $\mathcal{F}_{\mathcal{S}}$. We define a new frame $\mathcal{F}'_{\mathcal{S}}$ with its origin at R_1. The frame $\mathcal{F}'_{\mathcal{S}}$ is aligned with

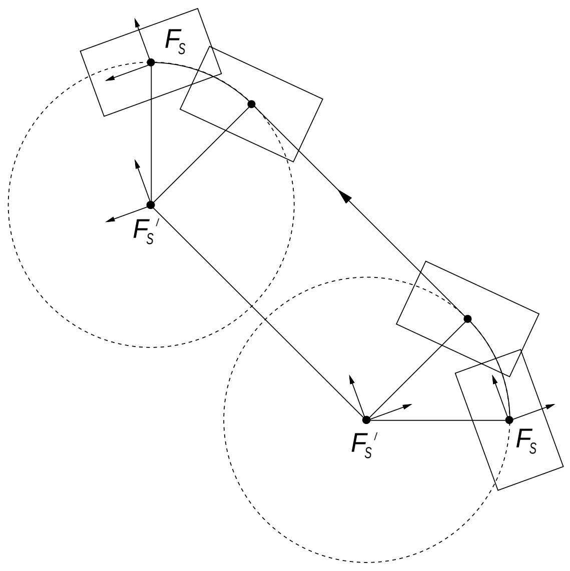


Figure 4.1: An example path using one rotational and one translational velocity direction. The frames $\mathcal{F}_{\mathcal{S}}$ and $\mathcal{F}'_{\mathcal{S}}$ are drawn at the initial and final locations.

and fixed in the slider frame $\mathcal{F}_{\mathcal{S}}$. In the frame $\mathcal{F}'_{\mathcal{S}}$, the two velocity directions are a pure rotation $\hat{\mathbf{v}}'_1 = (0, 0, sgn(\hat{\omega}_1))^T$ and a pure translation $\hat{\mathbf{v}}'_2 = \hat{\mathbf{v}}_2$, and the problem is to transfer the frame $\mathcal{F}'_{\mathcal{S}}$ from $\mathbf{q}'_{init}$ to $\mathbf{q}'_{goal}$. This is achieved by simply rotating the frame $\mathcal{F}'_{\mathcal{S}}$ in place, translating it to the final position, and rotating it to the final orientation. Any of these steps could have zero length. An example is shown in Figure 4.1.

If both $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$ have nonzero angular components, with rotation centers at R_1 and R_2 in the slider frame $\mathcal{F}_{\mathcal{S}}$, respectively, controllability can be demonstrated by showing that the slider can always translate to a configuration from which it can rotate to the goal. A translation is obtained by following X_1 and X_2 such that the total rotation is an integral multiple of 2π. The set of paths that follow X_1 and then X_2 and satisfy this condition defines a circle of final positions of the origin $O_{\mathcal{S}}$ of the slider frame $\mathcal{F}_{\mathcal{S}}$. The radius r of the circle is equal to the distance between R_1 and R_2, and the center of the circle is a distance r from $O_{\mathcal{S}}$ on a ray from $O_{\mathcal{S}}$ parallel to the ray from R_2 through R_1. Translations consisting of paths following X_2 and then X_1 define a similar circle (see Figure 4.2). The locus of points to which the slider frame $\mathcal{F}_{\mathcal{S}}$ can translate with only a single control change is given by these two circles. Each point on the locus is the origin of a similar, translated locus, and the slider frame $\mathcal{F}_{\mathcal{S}}$ can translate to any point in the plane by concatenating translations with one control change. □

Corollary 4.1 follows immediately.

Corollary 4.1 *The number of distinct combinations n of pushing contact configurations and*

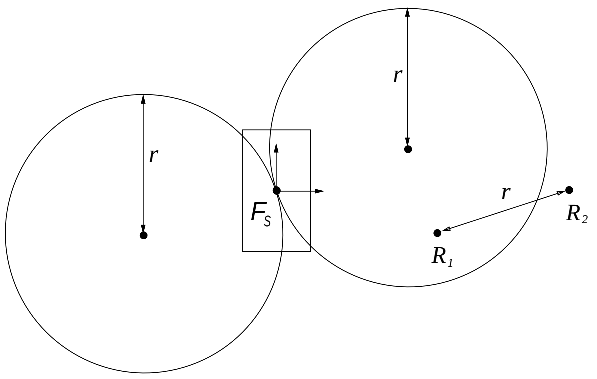


Figure 4.2: The locus of points to which the slider frame $\mathcal{F}_\mathcal{S}$ can translate, using one control change and rotation centers R_1 and R_2 in the slider frame $\mathcal{F}_\mathcal{S}$. The left circle results from rotating about R_1 and then R_2 in the slider frame $\mathcal{F}_\mathcal{S}$, and the right circle is obtained by reversing the order.

pushing directions must be at least two for the configuration of the slider $\mathcal{S}$ to be controllable by pushing. From any set of controls yielding controllability, no more than two are needed.

For a controllable system Σ with two velocity directions ($n = 2$), the slider may have to travel a long distance to reach nearby configurations. Thus the conditions of Proposition 4.2 are not sufficient for small-time local controllability. Before addressing the conditions for small-time local controllability, we establish the following fact.

Proposition 4.3 *Consider a set of velocity directions $\hat{\mathcal{V}}$ and its convex hull $CH_{S^2}(\hat{\mathcal{V}})$ on the velocity sphere. Any path from $\mathbf{q}_1$ to $\mathbf{q}_2$ using velocity directions in $CH_{S^2}(\hat{\mathcal{V}})$ can be followed arbitrarily closely by another path, also from $\mathbf{q}_1$ to $\mathbf{q}_2$, using only velocity directions in $\hat{\mathcal{V}}$.*

Proof: Proposition 5 in Appendix B of (Barraquand and Latombe [21]) proves the case when $\hat{\mathcal{V}}$ consists of two velocity directions. The result for any number of velocity directions follows by induction. □

On any open set of the configuration space $\mathcal{C}$, Proposition 4.3 says that we can consider the available velocity directions to be the convex hull of the velocity direction set $\hat{\mathcal{V}}$. Therefore, if the set of velocity directions $\hat{\mathcal{V}}$ contains four velocity directions that positively span the velocity sphere, the configuration of the slider $\mathcal{S}$ is small-time locally controllable. This also follows from the following theorem:

Theorem 4.1 (Sussmann [203]) *Let $\mathcal{X}$ be a finite set of vector fields on an open set of the state manifold containing $\mathbf{q}$. The set of nonzero tangent vectors at $\mathbf{q}$ is denoted $\mathcal{X}(\mathbf{q})$. Then (1) If $\mathbf{0}$ is in the interior of the convex hull of $\mathcal{X}(\mathbf{q})$, the system is small-time locally*

controllable from $\mathbf{q}$.
(2) If $\mathbf{0}$ *does not belong to the convex hull of* $\mathcal{X}(\mathbf{q})$*, the system is not small-time locally controllable from* $\mathbf{q}$.

To apply this theorem, we define the convex hull in $\mathbf{R}^3$ of a set of unit velocities $\mathcal{X}(\mathbf{q})$ to be $CH_{\mathbf{R}^3}(\mathcal{X}(\mathbf{q}))$ with boundary $\partial CH_{\mathbf{R}^3}(\mathcal{X}(\mathbf{q}))$.

The first half of Theorem 4.1 indicates that if the velocity directions $\hat{\mathcal{V}}$ of the control system Σ positively span the velocity sphere, the control system Σ is small-time locally controllable. The second half of the theorem says that if the velocity directions $\hat{\mathcal{V}}$ are contained in any open hemisphere of the velocity sphere, then Σ is not small-time locally controllable. Theorem 4.1 does not completely answer the question of small-time local controllability at $\mathbf{q}$, however, as it does not address the case when $\mathbf{0}$ lies in $\partial CH_{\mathbf{R}^3}(\mathcal{X}(\mathbf{q}))$. To resolve this case, we must consider the derivatives of $\mathcal{X}(\mathbf{q})$ (Sussmann [203]).

If $\mathbf{0}$ lies in $\partial CH_{\mathbf{R}^3}(\mathcal{X}(\mathbf{q}))$, then there is a unique linear subspace Z of maximum dimension such that $\mathbf{0}$ lies in the interior of $\partial CH_{\mathbf{R}^3}(\mathcal{X}(\mathbf{q}))$ in this space Z. The set $\mathcal{X}_Z$ is the subset of vector fields $X \in \mathcal{X}$ such that the tangent vector $X(\mathbf{q})$ lies in the subspace Z. The dimension of the subspace Z is one if the portion of $\partial CH_{\mathbf{R}^3}(\mathcal{X}(\mathbf{q}))$ through $\mathbf{0}$ is a line segment, and $\mathcal{X}_Z$ consists of two opposite vector fields. The dimension of the subspace Z is two if the portion of $\partial CH_{\mathbf{R}^3}(\mathcal{X}(\mathbf{q}))$ through $\mathbf{0}$ is a planar region, and $\mathcal{X}_Z$ consists of at least three vector fields such that the associated velocity directions span a great circle of the velocity sphere.

The set of all vector fields $[X, Y]$ such that $X, Y \in \mathcal{X}_Z$ is denoted $\mathcal{X}_Z^1$. Applying Sussmann's sufficient condition for small-time local controllability (Sussmann [203]), the control system Σ is small-time locally controllable if the convex hull of $\mathcal{X}(\mathbf{q})$ and $\mathcal{X}_Z^1(\mathbf{q})$ contains $\mathbf{0}$ in its interior. For the control system Σ, Sussmann's sufficient condition is also necessary.

When the dimension of Z is one, the set of vector fields $\mathcal{X}_Z$ consists of two opposite vector fields. The Lie bracket of opposite vector fields is zero, so $\mathcal{X}_Z^1$ consists of the zero vector field. The convex hull of $\mathcal{X}(\mathbf{q})$ and $\mathcal{X}_Z^1(\mathbf{q})$ does not contain $\mathbf{0}$ in its interior, so the control system Σ is not small-time locally controllable.

Now consider the case where the dimension of the subspace Z is two. In this case, the velocity directions corresponding to $\mathcal{X}_Z$ positively span a great circle of the velocity sphere. Provided that this great circle does not lie in the $\omega = 0$ plane, the convex hull of $\mathcal{X}(\mathbf{q})$ and $\mathcal{X}_Z^1(\mathbf{q})$ contains $\mathbf{0}$ in its interior, and the control system Σ is small-time locally controllable. To see this another way, recall that two nonopposite velocity directions that are not both translations are sufficient for small-time accessibility. If both velocity directions can be reversed (a total of four velocity directions), then the system is small-time locally controllable. These velocity directions positively span a great circle of the velocity sphere such that $\hat{\omega}$ is not identically zero. By Proposition 4.3, on any open set of the configuration space $\mathcal{C}$, any set of velocity directions that positively span the same great circle is equivalent.

Proposition 4.4 *The control system* Σ *is small-time locally controllable if and only if the*

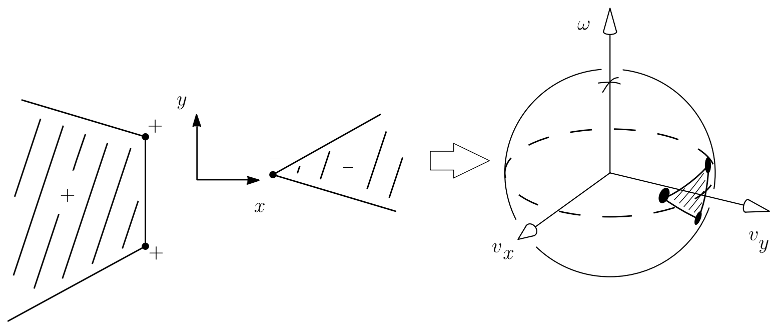


Figure 4.3: The convex hull of three velocity directions, represented in the rotation center space and on the velocity sphere.

set of velocity directions $\hat{\mathcal{V}}$ positively spans a great circle of the velocity sphere that does not lie in the $\omega = 0$ plane.

This result is stated in terms of velocity directions on the velocity sphere, but it can just as easily be stated in terms of rotation centers. The positive span of two rotation centers of the same sense is given by the line segment of rotation centers of the same sense between the points. The positive span of rotation centers of opposite senses is given by all points on the line through the two rotation centers but not between them. The sense of the rotation centers changes at infinity, which corresponds to a translation (see Figure 4.3). A great circle on the velocity sphere is equivalent to a line of rotation centers with both rotation senses. Figure 4.4 gives examples of rotation center sets that yield small-time local controllability.

Corollary 4.2 *The number of distinct combinations n of pushing contact configurations and pushing directions must be at least three for the configuration of the slider $\mathcal{S}$ to be small-time locally controllable by pushing. From any set of controls yielding small-time local controllability, no more than four are needed.*

The second half of Corollary 4.2 follows from Proposition 4.4 by Carathéodory's Theorem (Grünbaum [86]).

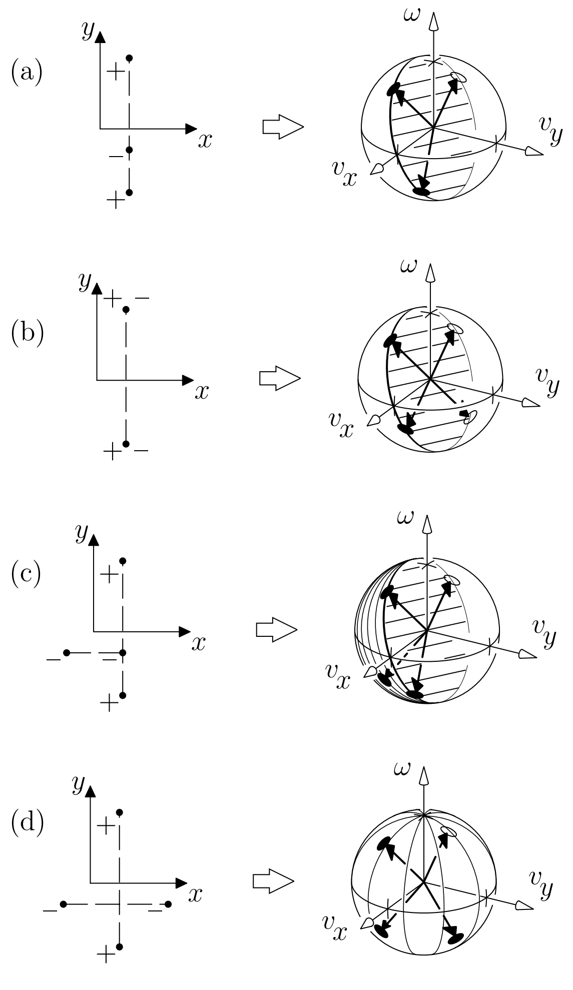


Figure 4.4: Examples of rotation center sets that yield small-time local controllability. (a) These three rotation centers positively span a great circle of the velocity sphere. (b) These four rotation centers positively span a great circle of the velocity sphere. (c) These four rotation centers positively span a closed hemisphere of the velocity sphere. (d) These four rotation centers positively span the velocity sphere.

Chapter 5

Mechanics and Controllability of Pushed Objects

In this chapter I study the mechanics problem of determining the motion of a pushed object. Using these results and those of the previous section, I elucidate the controllability properties of objects pushed with either point contact or stable line contact.

5.1 Mechanics of Pushing

During quasistatic pushing, the force $\mathbf{f}$ applied by the pusher is equal to the frictional force that the slider applies to the support plane. The frictional force $\mathbf{f}$ that the slider applies to the support plane when moving with velocity $\mathbf{v}$ will be expressed in terms of the following:

$\mathbf{x}$	point of contact $(x, y, 0)^T$ between the slider $\mathcal{S}$ and the support surface in the slider reference frame $\mathcal{F}_{\mathcal{S}}$
$d\mathbf{x}$	differential element of support area
$p(\mathbf{x})$	support pressure at $\mathbf{x}$
$\mu_s(\mathbf{x})$	support friction coefficient at $\mathbf{x}$
$s(\mathbf{x})$	the product of $p(\mathbf{x})$ and $\mu_s(\mathbf{x})$, referred to as the support friction distribution
$\mathbf{v}(\mathbf{x})$	the linear velocity of $\mathbf{x}$, given by $(v_x - \omega y, v_y + \omega x, 0)^T$, where the slider velocity $\mathbf{v}$ is $(v_x, v_y, \omega)^T$
$\mathbf{f}_{xy}$	the linear components $(f_x, f_y, 0)^T$ of $\mathbf{f}$

The origin $O_{\mathcal{S}}$ of the slider frame $\mathcal{F}_{\mathcal{S}}$ is located at the center of friction of the slider,

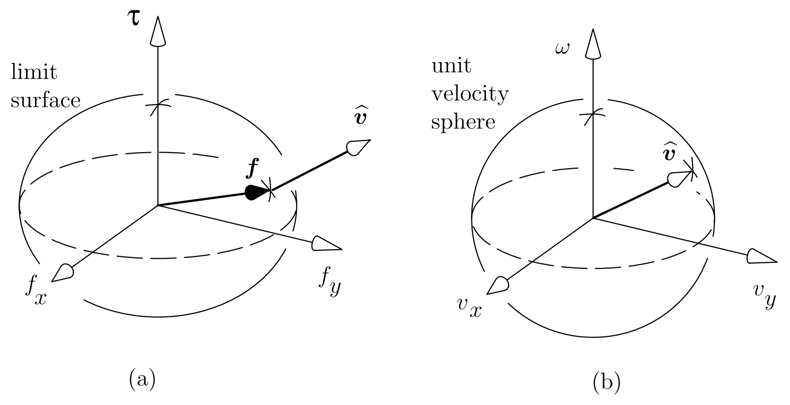


Figure 5.1: Mapping a force $\mathbf{f}$ through a limit surface to a velocity direction $\hat{\mathbf{v}}$.

which is the unique point of the slider such that $\int_{\mathcal{S}} \mathbf{x}s(\mathbf{x})d\mathbf{x} = \mathbf{0}$. When $\mu_s(\mathbf{x})$ is constant, $O_{\mathcal{S}}$ is located directly beneath the center of mass (MacMillan [134]). All forces and velocities are expressed with respect to the slider frame $\mathcal{F}_{\mathcal{S}}$.

The force $\mathbf{f}_{xy}$ and torque τ components of the force $\mathbf{f}$ applied to the support plane by a slider $\mathcal{S}$ are given by Mason [139]:

$$\begin{aligned} \mathbf{f}_{xy} &= \int_{\mathcal{S}} \frac{\mathbf{v}(\mathbf{x})}{|\mathbf{v}(\mathbf{x})|} s(\mathbf{x})d\mathbf{x} \\ \tau\hat{\mathbf{k}} &= \int_{\mathcal{S}} \mathbf{x} \times \frac{\mathbf{v}(\mathbf{x})}{|\mathbf{v}(\mathbf{x})|} s(\mathbf{x})d\mathbf{x}, \end{aligned}$$

where $\hat{\mathbf{k}}$ is the unit vector $(0,0,1)^T$. These expressions simply state that the differential frictional force applied by the slider at each support point $\mathbf{x}$ acts in the direction of the velocity of $\mathbf{x}$ with magnitude $s(\mathbf{x})$.

5.1.1 The Limit Surface

As the slider's velocity direction $\hat{\mathbf{v}}$ moves over the velocity sphere, the force $\mathbf{f}$ moves on a two-dimensional surface in the three-dimensional force space. This closed, convex surface is called the *limit surface* by Goyal *et al.* [84]. The limit surface encloses the set of all forces that can be statically applied to the slider, and during quasistatic motion the applied force lies on the limit surface. The slider's velocity direction vector $\hat{\mathbf{v}}$ is normal to the limit surface at the force $\mathbf{f}$. If the force $\mathbf{f}$ lies on the limit surface with an associated velocity direction $\hat{\mathbf{v}}$, then the force $-\mathbf{f}$ also lies on the limit surface with an associated velocity direction $-\hat{\mathbf{v}}$.

If the support friction distribution $s(\mathbf{x})$ is finite everywhere, the limit surface is smooth and strictly convex, defining a continuous one-to-one mapping from the set of force directions $\hat{\mathbf{f}} \in S^2$ to the set of velocity directions $\hat{\mathbf{v}} \in S^2$. If the applied force has zero torque about the center of friction, the resulting slider velocity is translational and parallel to the applied force.

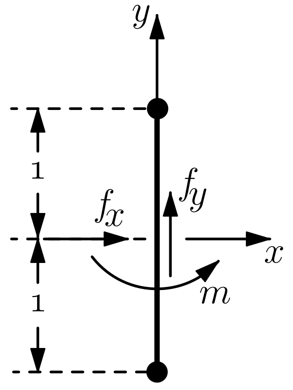


Figure 5.2: A barbell with two support points.

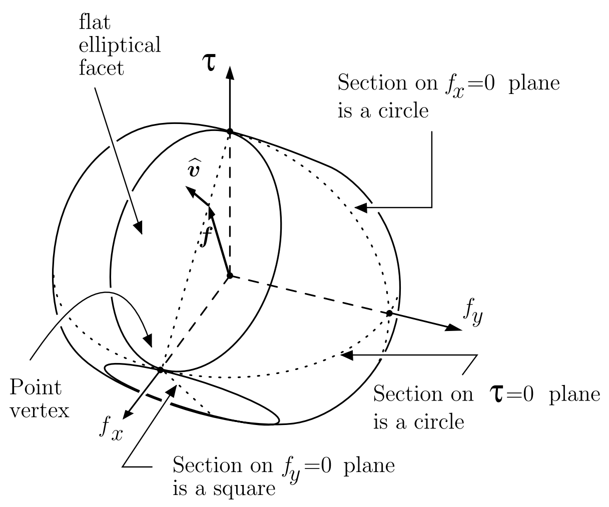
If any single point $\mathbf{x}_0$ supports a finite force with a nonzero coefficient of friction $\mu_s(\mathbf{x}_0)$, however, the mapping is no longer one-to-one. The support pressure $p(\mathbf{x}_0)$ and support friction $s(\mathbf{x}_0)$ is infinite, and the integral of $s(\mathbf{x}_0)$ over the point $\mathbf{x}_0$ is nonzero. The limit surface therefore contains flat facets mapping sets of force directions to the same velocity direction: rotation about $\mathbf{x}_0$.

If the support friction distribution $s(\mathbf{x})$ is infinite only at points on a line, and $s(\mathbf{x})$ integrates to zero over the rest of the support surface, then the limit surface contains vertices. The normals to the limit surface at these vertices are not uniquely defined: the same force maps to a set of possible velocity directions.

A simple limit surface with facets and vertices is that of a barbell: a rod with two points of equal support force at either end (Figures 5.2 and 5.3, taken from Goyal *et al.* [84]). Goyal *et al.* [84, 85] provide more details on the properties of the limit surface.

5.1.2 Solving for the Motion of a Pushed Object

Each contact point between the pusher and the slider may be sticking, breaking free, or sliding to the left or right. The *contact mode* describes the qualitative behavior of each contact point between the pusher and the slider. For each possible contact mode i, there is a space of possible slider velocities $\mathcal{V}_{k,i}$ that are kinematically consistent with that contact mode and the known pusher velocity (Lynch [129]). By Coulomb's law, each contact mode also specifies a polyhedral cone of possible pushing forces in the three-dimensional force space. This cone is the convex hull of the individual friction cones at the sticking contacts and the friction cone edges at the sliding contacts (Erdmann [68, 71]). This composite friction cone is intersected with the limit surface to find a cone of possible velocities $\mathcal{V}_{f,i}$ (Figure 5.4). If $\mathcal{V}_{k,i} \cap \mathcal{V}_{f,i} = \emptyset$, contact mode i cannot occur; otherwise, contact mode i is feasible and any of the velocities in the intersection set is a possible solution to the motion of the slider. There may be more than one solution under Coulomb's law of friction.


flat elliptical facet
τ
Section on f_x=0 plane is a circle
$\widehat{v}$
f
f_y
Point vertex
Section on τ=0 plane is a circle
f_x
Section on f_y=0 plane is a square

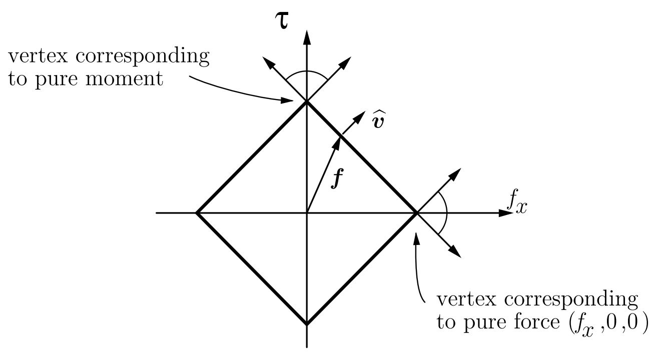

τ
vertex corresponding to pure moment
$\widehat{v}$
f
f_x
vertex corresponding to pure force $(f_x, 0, 0)$

Figure 5.3: The limit surface for the barbell.

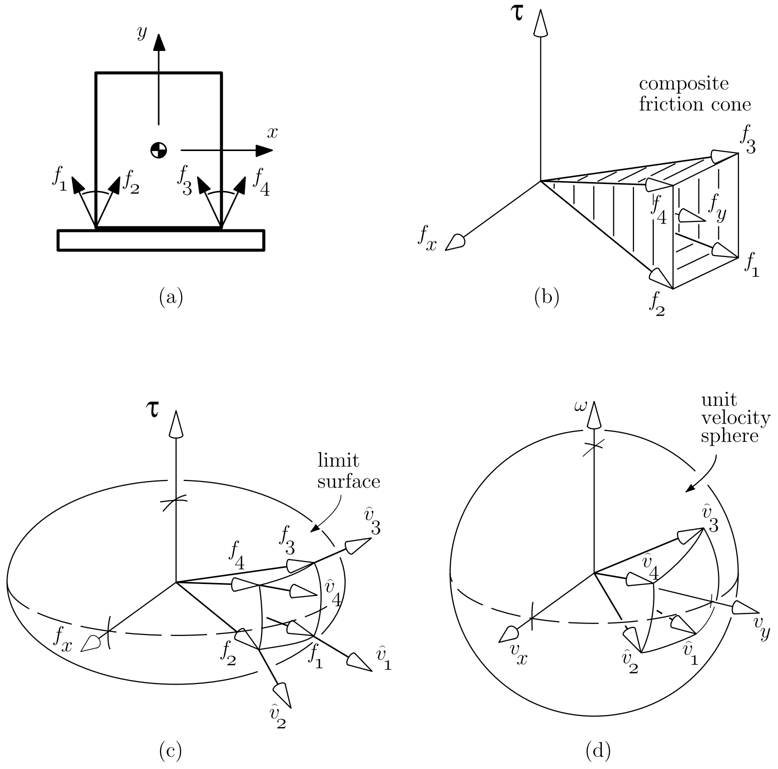


Figure 5.4: (a) The forces that the pusher can apply to the slider during sticking contact are represented by the two friction cones. (b) The convex hull of these friction cones in the three-dimensional force space. The result is a composite friction cone of possible pushing forces. (c) Mapping these forces through the limit surface for the slider. (d) The slider velocity directions corresponding to forces inside the composite friction cone.

5.2 Pushing with Point Contact

5.2.1 Mechanics of Pushing with Point Contact

When the slider is pushed at a single point of contact, there are only three possible contact modes: sticking contact, left sliding, and right sliding. Mason [139] developed search procedures to find the slider motion resulting in force/torque balance (for a known support friction distribution $s(\mathbf{x})$) for sticking and sliding contact. The actual motion is given by the contact mode that is consistent with all kinematic and force constraints. Peshkin and Sanderson [162] and Alexander and Maddocks [9] derived similar search procedures that find the slider motion by minimizing the power loss due to frictional sliding. Peshkin and Sanderson [164] proved that the power minimization approach is equivalent to the force/torque balance approach.

In general, the support friction distribution $s(\mathbf{x})$ is indeterminate. For this reason, several authors have investigated the use of weaker models of the support friction distribution (Mason [139]; Mason and Brost [142]; Peshkin and Sanderson [162]; Alexander and Maddocks [9]). With these weaker models, it is usually impossible to find the exact motion of the slider (see Appendix A). We do not need to be able to determine the motion of a pushed object to prove controllability results, however; we only need general properties of the limit surface and the available pushing contacts.

5.2.2 Controllability by Pushing with Point Contact

Open-loop pushing with point contact is inherently unstable, and many researchers have constructed point contact pushing control systems using visual (Inaba and Inoue [97]; Gandolfo *et al.* [80]; Salganicoff *et al.* [181]; Narasimhan [156]), tactile (Okawa and Yokoyama [158]; Lynch *et al.* [131]), and infrared ranging feedback (unpublished work by Latombe and colleagues [222]). Here I examine the controllability of such a system.

Proposition 5.1 *The configuration of a slider $\mathcal{S}$ with a bounded support friction distribution $s(\mathbf{x})$ is controllable by pushing if and only if the pusher can apply two pushing force directions, $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$, such that they do not both pass through the center of friction ($\hat{\tau}_1 \neq 0$ or $\hat{\tau}_2 \neq 0$) and $\hat{\mathbf{f}}_1 \neq -\hat{\mathbf{f}}_2$.*

Proof: Because the support friction distribution $s(\mathbf{x})$ is bounded, the limit surface has no facets, and therefore the two force directions map through the limit surface to two distinct velocity directions. At least one of the force directions has nonzero torque, so at least one of the velocity directions has a nonzero angular component. Because the two force directions are not opposite, the two velocity directions are also not opposite. Therefore, by Proposition 4.2, the slider is controllable. $\square$

A slider that is uncontrollable by pushing is a disk centered at its center of friction with a pushing friction coefficient of zero. All pushing forces pass through the center of friction, creating zero torque about the center of friction. The slider cannot be rotated (except nondeterministically if its limit surface contains vertices).

Theorem 5.1 is a direct application of Proposition 5.1 to polygonal sliders.

Theorem 5.1 *The configuration of a polygonal slider $\mathcal{S}$ with a bounded support friction distribution $s(\mathbf{x})$ is controllable by pushing with point contact on any edge. It is also controllable from any vertex that has nonzero friction and is not at the center of friction.*

The requirement that $s(\mathbf{x})$ be bounded is necessary; otherwise, we could arrange for all pushing forces through an edge to lie on the same facet of the limit surface.

A slider that is controllable by pushing may have to be pushed a long distance to reach nearby configurations. If the object is small-time locally controllable, however, it can follow any path arbitrarily closely. To find conditions for small-time local controllability of a slider, first recall that the set of available pushing contacts is given by , , a closed, piecewise smooth curve. At each point of , that is not a vertex, the curve , has a unique inwardly-pointing contact normal. At a vertex, we assume that the contact normal can take any direction in the range specified by the contact normals adjacent to the vertex. Each contact point and contact normal specifies a pushing force direction that can be applied to the slider $\mathcal{S}$, and the curve of all such force directions is denoted $\hat{\mathbf{f}}(,)$. Because , is a closed curve, $\hat{\mathbf{f}}(,)$ is a (possibly self-intersecting) closed curve of force directions on the force sphere.

Theorem 5.2 *The configuration of any slider $\mathcal{S}$ with a closed, piecewise smooth curve , of available pushing contact points is small-time locally controllable by pushing with point contact, unless the pushing contact is frictionless and , is a circle centered at the slider's center of friction (a frictionless disk).*

Proof: *Case 1: , not a circle.* Following the argument of Hong *et al.* [95], $\hat{\mathbf{f}}(,)$ must contain at least two pairs of opposite force directions. By the limit surface mapping, these forces yield two pairs of opposite velocity directions that span a great circle of the velocity sphere. By Proposition 4.4, the slider is small-time locally controllable, unless this great circle lies in the $\omega = 0$ plane. In this case we use the result of Mishra, Schwartz, and Sharir [149] which states that $\hat{\mathbf{f}}(,)$ positively spans the force sphere. Therefore $\hat{\mathbf{f}}(,)$ contains forces with positive and negative torque, and the slider can be rotated clockwise or counterclockwise. The $\omega = 0$ great circle and any clockwise and counterclockwise directions positively span the velocity sphere, and the slider is small-time locally controllable.

Case 2: , a circle. Every pair of diametrically opposed points on , gives rise to a pair of opposite velocity directions. If the center of friction is offset from the center of the circle, then only one pair lies in the $\omega = 0$ plane, and therefore any two pairs of opposite velocity directions yield small-time local controllability. If the center of friction is at the center of the circle and there is nonzero friction at the pushing contact, the slider can be translated in any direction and rotated using frictional forces to create torque about the center of friction. The slider is small-time locally controllable. If the contact is frictionless, however, the object cannot be rotated. A frictionless disk centered at its center of friction is the only type of slider that is not small-time locally controllable by point contact pushing. (If the slider's limit surface contains vertices, it may rotate nondeterministically.) □

Theorem 5.2 implies that a two-degree-of-freedom robot (a point moving in the plane) can push any object (other than a frictionless disk) to follow any planar trajectory arbitrarily closely.

5.3 Stable Pushing with Line Contact

5.3.1 Mechanics of Stable Pushing with Line Contact

The previous section examined the controllability of sliders pushed with point contact, but we would also like to synthesize pushing controls. Unfortunately, the motion of a slider pushed with a single point of contact is usually unpredictable, because it depends on the unknown and generally indeterminate support friction distribution $s(\mathbf{x})$. If there are two or more simultaneous pushing contacts, however, there may exist a space of pushing directions that, despite some uncertainty in the support friction distribution $s(\mathbf{x})$, result in a predictable motion of the slider: sticking at all contact points. I call such a push a stable push, and I will use these stable pushes to execute open-loop pushing plans.

In this thesis I focus on stable pushing with line contact. For a given line contact, we can define the following:

$\hat{\mathcal{V}}_{stable}$: The set of pushing directions such that the slider remains fixed to the pusher during the motion.

$\hat{\mathcal{V}}_{\mathcal{F}}$: The set of pushing directions such that one solution to the motion of the slider is to remain fixed to the pusher. This set of velocity directions is found by intersecting the composite friction cone $\mathcal{F}$ from the line contact with the limit surface (Figure 5.4).

If a velocity direction $\hat{\mathbf{v}}$ is in $\hat{\mathcal{V}}_{stable}$, then it is also in $\hat{\mathcal{V}}_{\mathcal{F}}$, but the converse is not necessarily true. Although stable contact is always a possible solution if the pushing direction $\hat{\mathbf{v}}$ is in $\hat{\mathcal{V}}_{\mathcal{F}}$, there may be other solutions. It is necessary to prove all other contact modes inconsistent. In this section, I describe the procedure STABLE for finding a subset of $\hat{\mathcal{V}}_{\mathcal{F}}$ and provide a theorem stating that this subset also belongs to $\hat{\mathcal{V}}_{stable}$.

Usually the support friction distribution $s(\mathbf{x})$ is unknown, and therefore we cannot determine $\hat{\mathcal{V}}_{\mathcal{F}}$ exactly. Instead, I assume that the center of friction and the shape of the slider are known. An algorithm for finding an approximation to $\hat{\mathcal{V}}_{\mathcal{F}}$ for a slider with a known center of friction was presented in (Lynch [129]). This algorithm utilizes results due to Peshkin and Sanderson [162] on the possible support friction distributions of a disk slider.

Here I describe the much simpler procedure STABLE for finding a conservative approximation to $\hat{\mathcal{V}}_{\mathcal{F}}$. I will illustrate the procedure using rotation centers. Without loss of generality, assume that the line contact is horizontal on the page with an upward pointing contact normal, as in Figure 5.5.

<u>Procedure STABLE</u>

(a) The coefficient of friction defines two friction cone edges, at angles $\tan^{-1}\mu$ to the contact normal, where μ is the coefficient of friction. For each edge of the friction cone, draw two lines perpendicular to the friction cone edge such that the entire slider is contained between the two lines. For an applied force at an edge of the friction cone, the resulting rotation center must lie in its respective band (Mason and Brost [142]; Alexander and Maddocks [9]). Counterclockwise (respectively clockwise) rotation centers between the two bands and to the left (resp. right) of the slider correspond to force angles inside the angular limits of the friction cone. See Figure 5.5(a).

(b) For each endpoint of the line contact, draw two lines perpendicular to the line through the center of friction and the endpoint. One of these lines is the perpendicular bisector between the contact point and the center of friction (Mason and Brost [142]). The other is a distance r^2/p from the center of friction and on the opposite side from the endpoint, where p is the distance from the endpoint to the center of friction and r is the distance from the center of friction to the most distant support point of the slider. (This "tip line" should actually be slightly more distant from the center of friction (Peshkin and Sanderson [162]).) The rotation center from a force through this endpoint must lie in the band between these two lines (Mason and Brost [142]). All rotation centers between the two bands correspond to forces that pass between the two endpoints. See Figure 5.5(b).

(c) The intersection of the closed regions found in (a) and (b) yield a set of rotation centers corresponding to forces that are guaranteed to lie on or inside the composite friction cone $\mathcal{F}$ from the line pushing contact. This is a conservative approximation to $\hat{\mathcal{V}}_{\mathcal{F}}$. See Figure 5.5(c).

STABLE misses some tight turning rotation centers but finds all translations belonging to $\hat{\mathcal{V}}_{\mathcal{F}}$. If the composite friction cone $\mathcal{F}$ contains any pure force (zero torque about the center of friction) in its interior, STABLE will find a closed, convex polygon of velocity directions on the velocity sphere with nonempty interior and a range of translation directions. If the composite friction cone $\mathcal{F}$ contains only a single pure force direction, necessarily on the boundary of $\mathcal{F}$, STABLE finds only a single translation in the direction of this pure force. If the composite friction cone $\mathcal{F}$ does not contain a pure force, STABLE finds no velocity directions belonging to $\hat{\mathcal{V}}_{\mathcal{F}}$. In fact, with no information about the support friction distribution $s(\mathbf{x})$ other than the center of friction, no velocity direction is guaranteed to be in $\hat{\mathcal{V}}_{\mathcal{F}}$ if the line contact cannot apply a force through the center of friction.

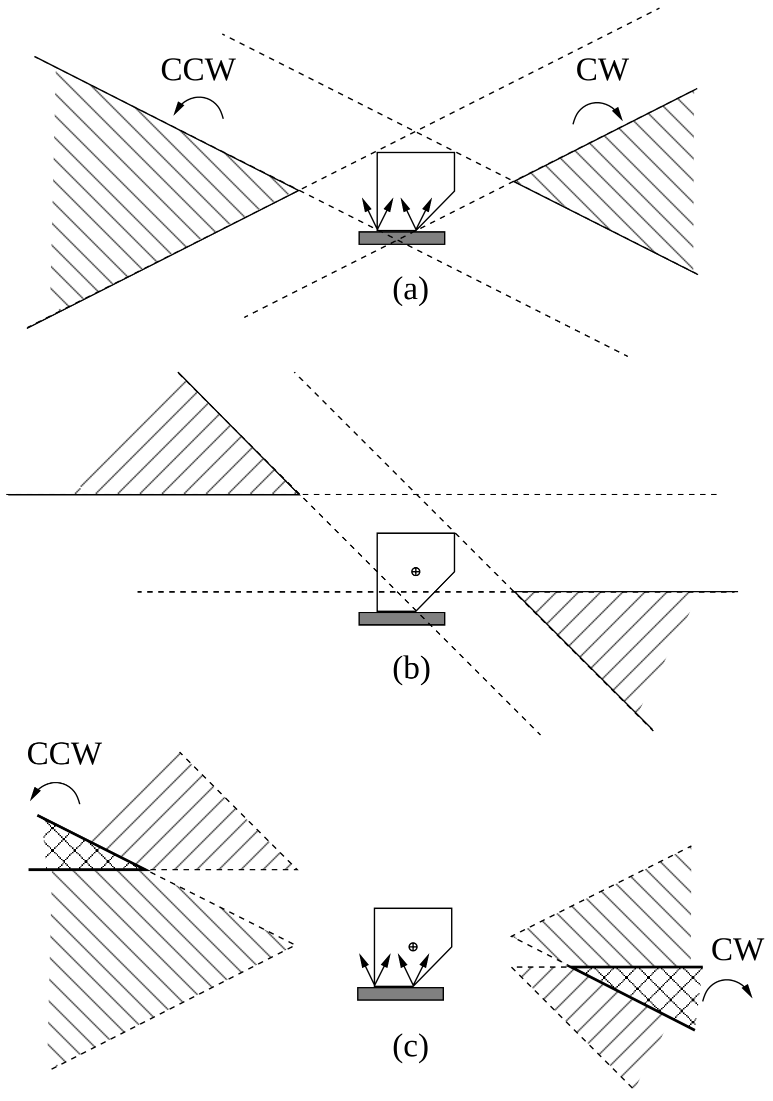


Figure 5.5: Illustration of the procedure STABLE. (a) Rotation centers that can be achieved by forces inside the friction cone angular limits. The friction coefficient is 0.5. (b) Rotation centers that can be achieved by forces passing between the line contact endpoints. (c) The intersection of the closed regions found in (a) and (b) correspond to rotation centers that can be achieved by forces inside the composite friction cone $\mathcal{F}$ defined by the line contact.

Proposition 5.2 *If the pusher $\mathcal{P}$ makes line contact with the slider $\mathcal{S}$ with a composite friction cone $\mathcal{F}$ that does not include a force through the center of friction, then, in the absence of any other information about the support friction distribution $s(\mathbf{x})$, no single velocity direction is guaranteed to be achievable by a force in $\mathcal{F}$.*

Proof: With no information about the support friction distribution $s(\mathbf{x})$, we can always choose the support friction distribution to be concentrated arbitrarily close to the center of friction. In this case, all rotation centers not located at the center of friction correspond to pushing forces passing through or arbitrarily close to the center of friction, which are not included in the composite friction cone $\mathcal{F}$. If the rotation center is located at the center of friction, any support friction distribution $s(\mathbf{x})$ that is symmetric about the center of friction corresponds to a pushing force that is a pure torque. A pure torque cannot be applied by line contact with a finite object. $\square$

Proving that a pushing direction in the set found by STABLE belongs to $\hat{\mathcal{V}}_{stable}$ requires proving all other contact modes inconsistent. Here I state the relevant result, omitting the proof by case analysis.

Theorem 5.3 *Given a pusher in line contact with a slider $\mathcal{S}$ with a bounded support friction distribution $s(\mathbf{x})$, let $\hat{\mathcal{V}}_{\mathcal{F}}$ be the set of rotation centers resulting from forces in the composite friction cone $\mathcal{F}$. Draw two lines perpendicular to the line contact such that the entire slider $\mathcal{S}$ is contained between the two lines. All rotation centers in $\hat{\mathcal{V}}_{\mathcal{F}}$ and outside the two lines are guaranteed to belong to the set of stable pushing directions $\hat{\mathcal{V}}_{stable}$. Therefore, all velocity directions found by* STABLE *belong to $\hat{\mathcal{V}}_{stable}$.*

5.3.2 Controllability by Stable Pushing with Line Contact

If the composite friction cone $\mathcal{F}$ from the line pushing contact contains a pure force in its interior, then STABLE finds a convex set of velocity directions $\hat{\mathcal{V}}_{\mathcal{F}}$ with nonempty interior (including a range of translations). By Proposition 4.2, we get the following.

Theorem 5.4 *If a slider $\mathcal{S}$ is pushed with line contact with a composite friction cone $\mathcal{F}$ such that $\mathcal{F}$ contains a pure force (zero torque about the center of friction of the slider) in its interior, then the configuration of the slider is controllable by the stable pushes found by* STABLE.

If Theorem 5.4 is satisfied, the slider can be pushed to any configuration in the obstacle-free plane using the stable pushes found by STABLE. We can apply Theorem 5.4 to prove the following result regarding polygonal sliders.

Theorem 5.5 *For a polygonal slider $\mathcal{S}$ and a sufficiently long straight-edge pusher $\mathcal{P}$, any edge of the convex hull of $\mathcal{S}$ can be used as the line pushing contact. If the center of friction of $\mathcal{S}$ does not lie on a vertex of the convex hull, and there is nonzero friction at the line* *contacts, then there is at least one line contact from which the slider is controllable by the stable pushes found by* STABLE.

Proof: The center of friction only lies on a vertex of the convex hull if the support friction distribution $s(\mathbf{x})$ integrates to zero everywhere else. Otherwise, there is at least one edge such that the normal of the edge at an interior point of the edge passes through the center of friction. (To find such a point, draw the maximal inscribed circle centered at the center of friction.) A force $\mathbf{f}$ along that normal is in the interior of the composite friction cone for that edge. By Theorem 5.4, the slider is controllable from that edge by the stable pushes of STABLE. □

It is intuitively clear that a slider is not small-time locally controllable by pushing a single edge. It is impossible to "pull" the slider backward, and therefore it cannot follow all paths arbitrarily closely. If we allow the pusher to change contact configurations, however, in some cases the configuration of the slider may be made small-time locally controllable by stable pushes with line contact. The following theorem gives a sufficient condition for a slider to be small-time locally controllable by line contact pushing.

Theorem 5.6 *Given a set of composite friction cones from line contacts between the pusher $\mathcal{P}$ and the slider $\mathcal{S}$. Define $\mathcal{F}_f$ to be the set of pure forces (zero torque) such that each $\mathbf{f} \in \mathcal{F}_f$ is interior to at least one of the composite friction cones. Then if $\mathcal{F}_f$ positively spans the force plane, $\mathcal{S}$ is small-time locally controllable by the stable pushes found by* STABLE. *Further, we can always choose three stable pushing directions such that the slider is small-time locally controllable.*

Proof: If the conditions of Theorem 5.6 are satisfied, then STABLE will find a set of translation directions that positively spans the $\omega = 0$ great circle. Because each of these translation directions has a neighborhood of velocity directions also in the set found by STABLE, the velocity directions found by STABLE positively span the velocity sphere. □

The slider of Figure 5.5 is small-time locally controllable by stable pushing at the two line contacts shown in Figure 5.6. Another example is given in Figure 5.7. In addition, any rectangular object or regular $2k$-gon is small-time locally controllable by the stable pushes found by STABLE, provided the center of friction of the object is in the interior of the slider and there is nonzero friction at the edge contacts. It suffices to show there are two opposing edges with opposing pure forces in the interior of their respective composite friction cones.

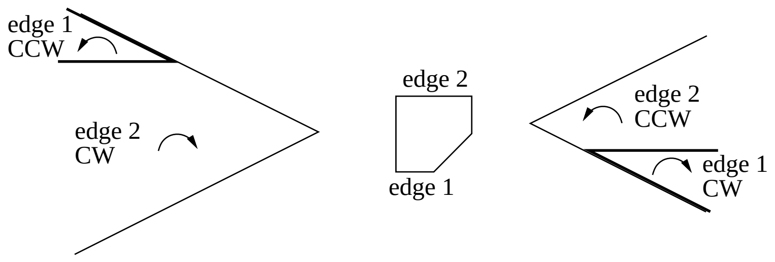


Figure 5.6: The slider of Figure 5.5 is small-time locally controllable by stable pushing at two edges using pushes found by Stable. The friction coefficient is 0.5.

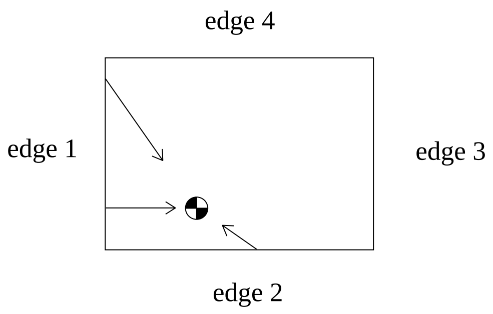


Figure 5.7: The slider is small-time locally controllable by pushing edges 1 and 3 or edges 2 and 4, provided friction is nonzero. Interestingly, the slider is also small-time locally controllable by pushing edges 1 and 2 if the friction coefficient is greater than 1.0. For example, the three pure forces shown positively span the force plane. If these forces are interior to their respective composite friction cones, the object is small-time locally controllable by Theorem 5.6.

Chapter 6

Planning Stable Pushing Paths among Obstacles

The stable pushing directions impose nonholonomic constraints on the motion of the pusher, and the problem is to plan free pushing paths among obstacles subject to these nonholonomic constraints. This is similar to the problem of planning paths for car-like mobile robots, the subject of much recent robotics research. The configuration space of a car-like mobile robot is also $\mathbf{R}^2 \times S^1$, but its feasible velocity direction set is only one-dimensional, corresponding to the angle of the steering wheel. Despite this, Laumond [116] showed that a car-like robot that can reverse can reach any configuration in any open connected subset of its free configuration space. Barraquand and Latombe [21] showed that only two steering directions are required.

Many path planners for car-like mobile robots have been proposed; see Chapter 9 of (Latombe [115]) for a survey. For planning pushing paths, I chose to adapt an algorithm by Barraquand and Latombe [21] due to its simplicity and its adaptability to two-dimensional velocity direction sets. The resulting algorithm closely parallels that described by Barraquand and Latombe. More details on the algorithm can be found in (Barraquand and Latombe [21]).

6.1 Algorithm

To plan stable pushing paths, we first choose a discrete set of m line pushing contacts. For each pushing contact $i = 1, \ldots, m$, we calculate the set of stable pushing directions using the procedure Stable. In light of Proposition 4.3, the planner uses discrete sets of velocity directions $\hat{\mathcal{V}}^i_{extremal}$, where $\hat{\mathcal{V}}^i_{extremal}$ is the set of vertices of the convex polygon of velocity directions found by Stable for pushing contact i. The union of $\hat{\mathcal{V}}^i_{extremal}$ for all i is denoted $\hat{\mathcal{V}}_{extremal}$.

The push planner, shown below in pseudo-code, is a simple best-first search using a variation of Dijkstra's algorithm (Aho, Hopcroft, and Ullman [4]). The planner constructs a tree T of configurations reached in the search and a list *OPEN* of configurations in T whose successors have not yet been generated. Configurations in *OPEN* are sorted by the costs of their paths. The planner either returns failure or a path to a user-specified goal neighborhood. Note that the planner is not exact, as it only finds a path to a goal neighborhood.

```
program push_planner
    initialize $T$, OPEN with start configuration $q_{init}$
    while OPEN not empty
        $q \leftarrow$ first in OPEN, remove from OPEN
        if $q$ is in the goal region
            report success
        if $q$ is not near previously occupied configuration
            mark $q$ occupied
            for each pushing contact $i = 1, \ldots, m$
                for each pushing direction in $\hat{\mathcal{V}}^i_{extremal}$
                    integrate forward from $q$ a small distance
                    compute cost of path to the new configuration $q_{new}$
                    if cost <= MAXCOST and path is collision-free
                        make $q_{new}$ a successor to $q$ in $T$
                        place $q_{new}$ in OPEN, sorted by cost
    report failure
end
```

6.2 Details of the Implementation

6.2.1 Collision Detection

The workspace is an open rectangular subset of $\mathbf{R}^2$, and obstacles are represented as closed polygons. For a particular pushing contact configuration, $\mathcal{PS}$ is the closed region of the workspace occupied by the pusher $\mathcal{P}$ and the slider $\mathcal{S}$. $\mathcal{PS}(\mathbf{q})$ is the closed region of the workspace occupied by the pusher and the slider at the configuration $\mathbf{q}$. The obstacle space $\mathcal{C}_{obs}$ is the closed set of configurations $\mathbf{q}$ such that $\mathcal{PS}(\mathbf{q})$ intersects an obstacle or the walls bounding the workspace. The free space $\mathcal{C}_{free}$ is the open set of the configuration space $\mathcal{C}$ complementary to $\mathcal{C}_{obs}$.

The free space $\mathcal{C}_{free}$ changes with the pushing contact, but for simplicity the planner uses a single representation of free space for all pushing contacts. This simplification is appropriate when the pusher $\mathcal{P}$ is much smaller or much larger than the slider $\mathcal{S}$. The region occupied by the pusher and the slider is treated as the smallest disk that encloses them for all pushing contact configurations. A collision occurs when this disk intersects the polygonal obstacles. This representation of the free space $\mathcal{C}_{free}$ is not exact, but the resulting code is simple and the collision-detection routine is fast.

The planner checks for collisions at each new configuration in the search, not along the path. For this reason, the disk approximation to $\mathcal{PS}$ is grown by the maximum distance any point in $\mathcal{PS}$ can move in a single step. This ensures that if a new configuration is free, then the path that transferred it there is also free. This approach is simple and fast, but somewhat conservative: the planner may not find paths through tight spots where paths exist.

6.2.2 Cost Function

In the current implementation, the cost of a path is the sum of an integer a times the number of pushing steps, an integer b times the number of control changes, and an integer c times the number of changes of the pushing contact. These values must all be nonnegative. The user can control the maximum permissible cost *MAXCOST* for a path.

Because path costs are always integral, the sorted list *OPEN* is represented by a 1-d array of linked lists, where each array index represents the cost of the paths to the configurations in its linked list. Inserting a new configuration into *OPEN* consists of simply appending it to the end of the appropriate linked list, and therefore takes constant time.

6.2.3 Pruning

The planner prunes configurations that are sufficiently close to configurations that have been reached with the same or lower cost and the same pushing contact. Two configurations are considered sufficiently close if they occupy the same cell of a predefined grid on the configuration space.

6.2.4 Parameters

The user must specify the parameters defining the size of the goal neighborhood $\mathcal{G}(\mathbf{q}_{goal})$, the length of the integration step, and the resolution of the configuration space grid used to check for prior occupancy. These parameters are interdependent. The resolution of the grid should be sufficiently fine that the application of any control moves the configuration to a new grid cell, and the goal neighborhood should be large enough that $\mathcal{PS}$ does not easily jump over it. The user must also specify the maximum number of configurations that the planner will explore before returning failure.

6.2.5 Changing the Pushing Contact

We assume that the pusher $\mathcal{P}$ can change pushing contacts at any time. If the slider is pushed by a manipulator capable of moving above the obstacles, such as a robot arm pushing objects on a table, this is a reasonable assumption. If the pusher is constrained to move in the plane, such as a mobile robot, however, this planner does not address the problem of finding paths between pushing contacts.

6.2.6 Complexity

The size of the configuration space grid is mc^3, where m is the number of pushing contacts and c is the number of discretization intervals along the x_w, y_w, and ϕ_w axes. Assuming we do not otherwise limit the size of the search, the space complexity of the planner is $O(mc^3)$. Each configuration must undergo a collision-check. In our implementation, this takes $O(n)$ time, where n is the number of vertices in the environment. Inserting a new configuration into the sorted list *OPEN* takes constant time. The time complexity is therefore $O(mnc^3)$.

In practice, planning is faster in cluttered spaces, where obstacles remove large portions of the configuration space. Also, the user-specified maximum number of search nodes can be reached before a solution is found or *OPEN* becomes empty. This is particularly an issue when there are many pushing edges and extremal controls.

6.3 Properties of the Planner

The push planner inherits properties of the planner described by Barraquand and Latombe [21]. Changing contact configurations in the pushing case is essentially equivalent to shifting between forward and reverse in the car-like robot case. (Note that we assume there is always a path for the pusher from one feasible pushing contact to another.) With an exact collision-detection routine, it is possible to choose search parameters such that the following properties hold:

- Completeness. *If there is a feasible pushing path from the initial configuration $\mathbf{q}_{init}$ to the goal neighborhood $\mathcal{G}(\mathbf{q}_{goal})$ using the stable pushing directions found by* Stable, *then the planner will find a pushing path.*

- Optimality. *If the cost of the pushing path is given by the number of changes of the pushing contact, then the planner will find a pushing path with λ changes of the pushing contact, where λ is the minimum number of contact changes for any feasible pushing path connecting the initial configuration $\mathbf{q}_{init}$ to the goal neighborhood $\mathcal{G}(\mathbf{q}_{goal})$ using velocity directions found by* Stable.

Now assume that the extent of the pusher $\mathcal{P}$ is negligible so that $\mathcal{PS}$ is equivalent to the slider $\mathcal{S}$. If $\hat{\mathcal{V}}_{extremal}$ satisfies the conditions for small-time local controllability, then, with an exact collision-detection routine, it is possible to choose search parameters such that the following property holds:

- Existence of a solution. *If the initial configuration* $\mathbf{q}_{init}$ *and the goal configuration* $\mathbf{q}_{goal}$ *lie in the same connected component of the slider's free configuration space* $\mathcal{C}_{free}$*, then the planner will find a pushing path connecting* $\mathbf{q}_{init}$ *to the goal neighborhood* $\mathcal{G}(\mathbf{q}_{goal})$.

The proofs of these properties do not suggest how small to choose the integration step or how to choose the other search parameters.

6.4 Planner Results

The planner is implemented in C on a Sun SPARC 20. This section presents some pushing paths generated for the slider of Figure 5.6.

In the first two examples, a mobile robot pushes the slider from edge 1 in Figure 5.6. Because the mobile robot pushes from only a single edge, the slider is not small-time locally controllable. The extremal velocity directions $\hat{\mathcal{V}}_{extremal}$ found by STABLE consist of two rotations and two translations. The goal region is about $\pm 25\%$ the length of the slider and ± 2 degrees.

In Figure 6.1, the slider is pushed to the goal by an omnidirectional mobile robot. This path took about seven seconds to generate. (In all of the examples, approximately 30,000 search nodes are created every second.) The same problem is presented to a car-like mobile robot, which imposes additional constraints on the possible pushing directions. Figure 6.2 shows the intersection of the stable pushing directions with the set of possible robot velocity directions, limited by the rolling constraint of parallel wheels and a minimum turning radius. Combining the constraints, we see that the robot cannot execute a left-turn stable push: the extremal velocity directions consist of a straight-ahead motion and a right turn. Figure 6.3 shows the path generated for the car-like mobile robot. The robot is forced to take a longer path due to its kinematic constraints. Planning time was one second. In both examples, the cost function is $a = 1$, $b = 5$, and $c = \infty$ (edge changes are disallowed). A nonzero value of b tends to smooth the paths by eliminating excessive control changes.

In the next two examples, a robot arm pushes the slider on a table, and the arm is capable of lifting up and changing the pushing contact. The pusher can contact the slider at the two opposite edges shown in Figure 5.6. The resulting sets of stable pushing directions yield small-time local controllability. For each edge, the planner uses the four extremal velocity directions found by STABLE. The area of the pusher $\mathcal{P}$ in the plane is assumed to be negligible.

Figures 6.4 and 6.5 show pushing paths that solve the same problem using different cost functions. In these examples, the goal region is about $\pm 10\%$ the length of the slider and ± 2 degrees. In Figure 6.4, the cost function is $a = 1$, $b = 5$, and $c = 0$—a short path with few control changes is preferred. The resulting path took 35 seconds to find. In Figure 6.5,

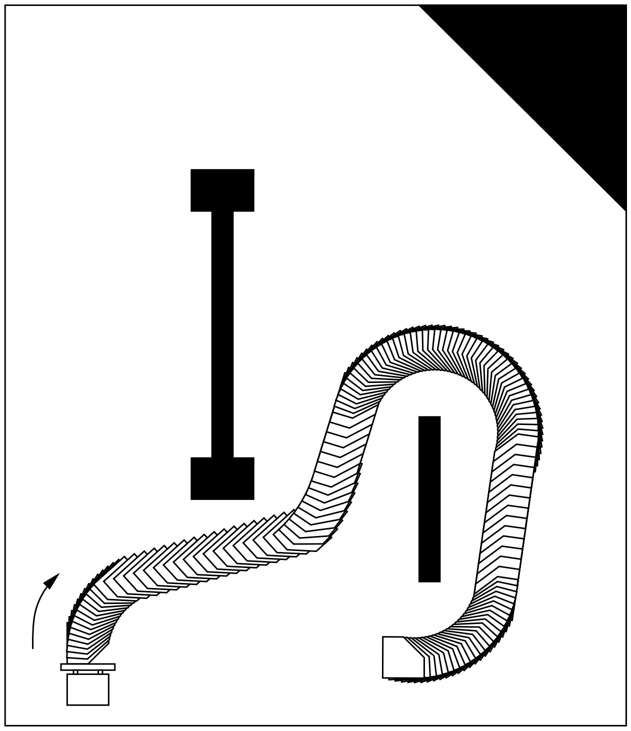


Figure 6.1: The slider being pushed to the goal by an omnidirectional mobile robot. For clarity, the mobile robot is only drawn at the beginning of the path.

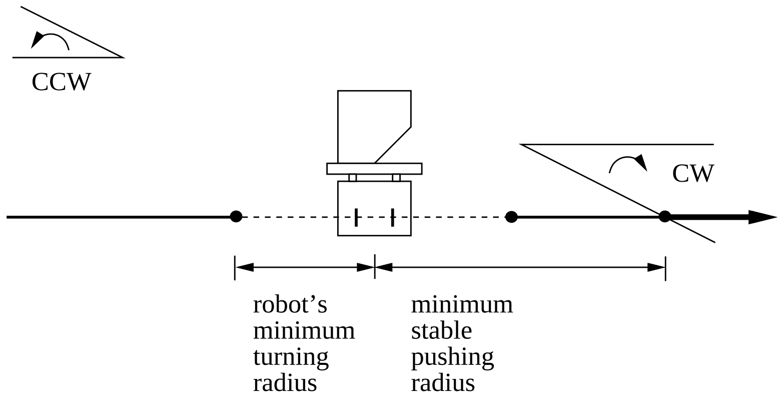


Figure 6.2: Intersecting the feasible velocity directions of the car-like mobile robot pusher (represented as rotation centers) with the stable pushing directions for the slider. The intersection prohibits left turns.

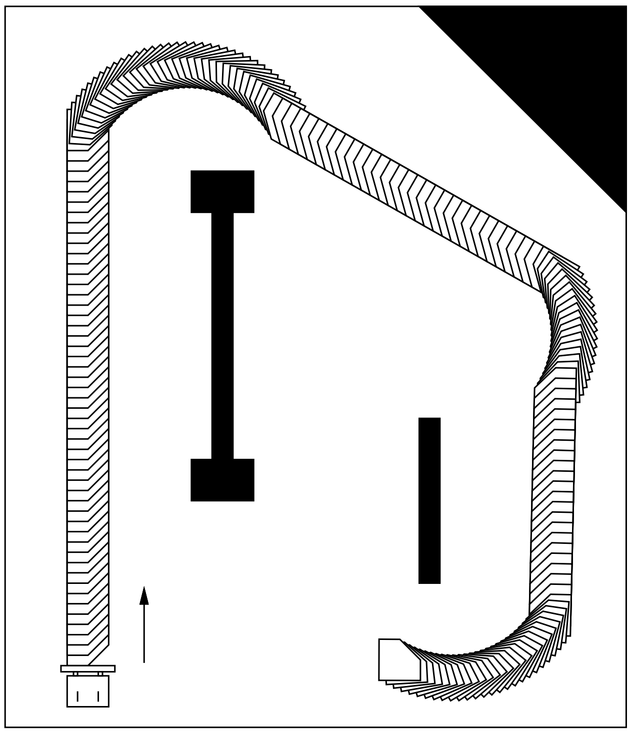


Figure 6.3: The slider being pushed to the goal by a car-like mobile robot.

the cost function is $a = 0$, $b = 1$, and $c = 10$—a path that minimizes contact changes is preferred. The planner found this path in two seconds.

Figure 6.6 presents another example of maneuvering the slider in a cluttered workspace using two pushing edges. The size of the goal region is the same as the previous two examples, and the cost function is $a = 0$, $b = 1$, and $c = 10$. Planning time was 43 seconds.

Our experience shows that this simple push planner can quickly and reliably solve complex manipulation problems. The user must choose the integration step size with care, however. If the step is too small, the search will reach its memory capacity before it reaches the goal region; if the step is too large, tight paths will be missed and the goal size may have to be increased.

6.5 Implementation

Costa Nikou and I have developed a graphical user interface (GUI), written in Tcl/Tk, for the specification of pushing problems. The user can create a polygonal object using the mouse and specify the edges to be pushed, along with the coefficient of friction. The GUI then calculates the set of extremal stable velocity directions for each pushing edge using the procedure STABLE. The user is notified of the result, and whether or not the object is small-time locally controllable by pushing these edges.

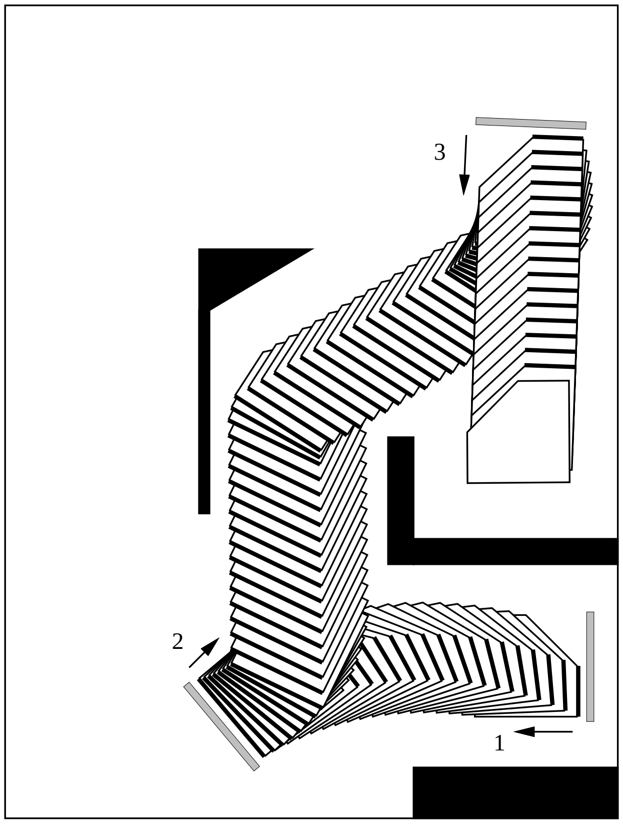


Figure 6.4: A pushing plan found with cost function $a = 1$, $b = 5$, and $c = 0$.

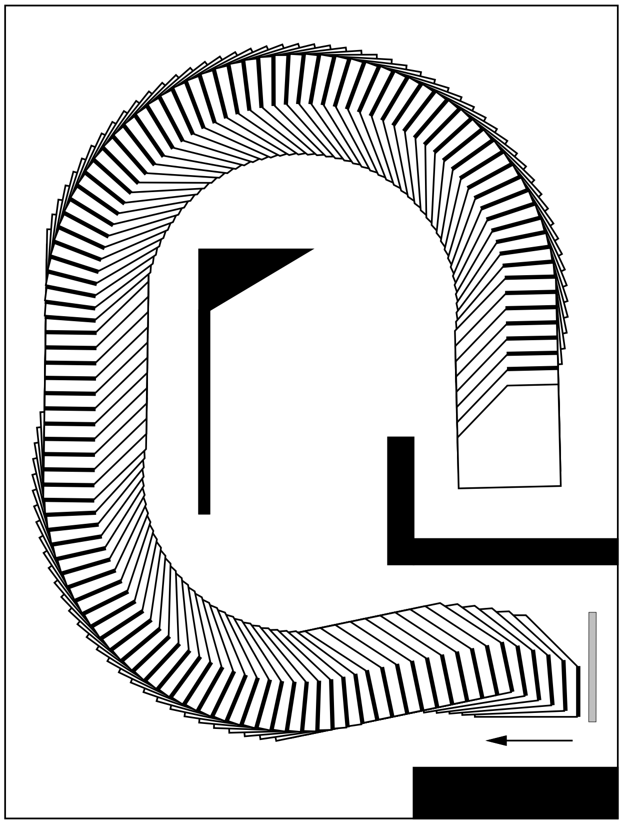


Figure 6.5: A pushing plan found with cost function $a = 0$, $b = 1$, and $c = 10$.

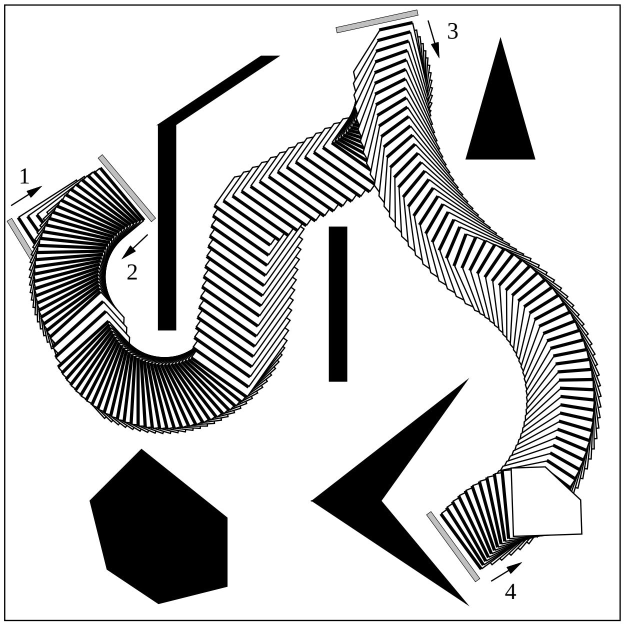
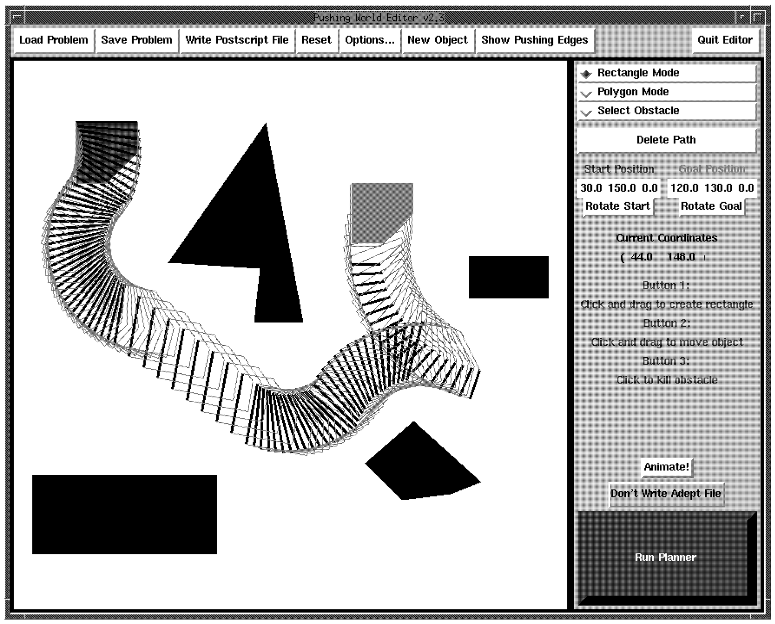


Figure 6.6: A pushing plan found with cost function $a = 0$, $b = 1$, and $c = 10$.



Start Position 30.0 150.0 0.0

Goal Position 120.0 130.0 0.0

Current Coordinates (44.0 148.0)

Button 1:
Click and drag to create rectangle
Button 2:
Click and drag to move object
Button 3:
Click to kill obstacle

Figure 6.7: The graphical user interface.

The user specifies a pushing problem by drawing obstacles in the space and moving a part to its initial and goal configurations. The user can also alter the default cost function and other planner parameters. The GUI invokes the planner, and if it finds a path it is displayed on the screen (Figure 6.7). An option to animate the pushing path is also introduced.

Finally, we implement the resulting plan on an Adept 550 tabletop robot, which holds a small flat pusher in its gripper (Figure 6.8). The planner writes a V+ program which is sent to the Adept's control computer. In our experiments, we print out a copy of the "pushing world," including the obstacles and the start and goal configurations, and we place the paper at a designated spot in the robot's workspace. Then the robot pushes the object sitting on the paper "world." By simply printing the paper world, it is easy to create new obstacle fields and guarantee that the computer's model corresponds exactly with the scene.

We tested the implementation with extruded laminar parts with planar dimensions on the order of 30 mm. To increase pushing friction, for tighter maneuvers, we sometimes covered the perimeters of the parts with sandpaper. The pushing plans are very robust; error in the final configuration of the object is essentially just the robot's positioning error. (Note that the final configuration can be anywhere in the goal region specified to the planner.) Parts other than extruded lamina sometimes experienced out-of-plane rolling during pushing, indicating that the assumption of planar forces is violated.

Interestingly, the gripper that came with the Adept 550 was designed to grasp very small objects, and it is too small to grasp many of the test objects we used in our experiments. Pushing provides a practical way for this particular robot to manipulate parts in the plane without changing the hardware.

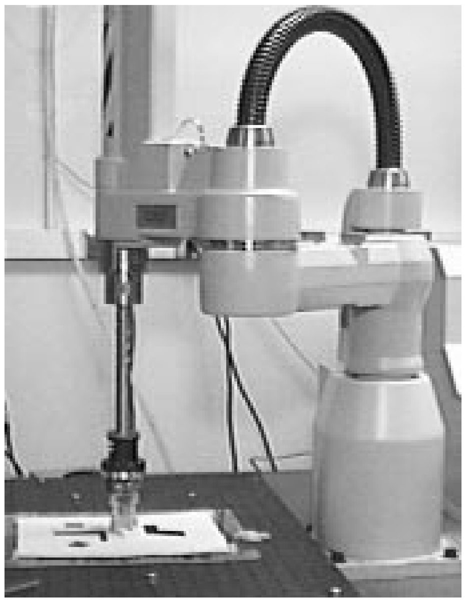


Figure 6.8: The Adept robot pushing an object.

6.6 Variations on the Planner

To shrink the size of the goal neighborhood without sacrificing much speed in planning, the integration step and grid cell sizes could be decreased in the neighborhood of the goal. This would allow fine positioning near the goal.

The planner could be modified to find a path to an exact goal configuration instead of a goal neighborhood. A very simple way to implement this is to plan a path to a goal neighborhood as before, then back up along the path until a configuration is found from which $\mathcal{PS}$ can move exactly to the goal configuration using a single pushing direction in $\hat{\mathcal{V}}_{stable}$. This is possible due to the fact that the space of stable pushing directions has nonempty interior on the velocity sphere. (This is not true for the case of a car-like mobile robot pusher.) The preimage of the goal configuration under all constant stable pushing directions therefore encloses a volume of the configuration space $\mathcal{C}$ with nonempty interior. An even better solution is to define this preimage as the goal region. Another approach is to find an exact holonomic path and transform it into one that obeys the nonholonomic constraints (Laumond *et al.* [119]).

The examples presented in Section 6.4 use stable translational velocity directions that border on being unstable. These pushing paths may be made robust to bounded uncertainty in the pushing friction coefficient and the location of the center of friction by propagating this uncertainty through the procedure STABLE. The result is to shrink the set returned by STABLE. Brost [41, 43] has applied this idea to several manipulation tasks.

6.7 Pushing Multiple Objects

Imagine a mobile robot with a "plow" pushing multiple sliding objects, each in line contact with the plow. The set of pushing directions that move the system as a single rigid body is simply the intersection of the stable pushing directions for each object. The planner can then be applied to the entire system.

A more interesting example is pushing a chain of objects, where the manipulator pushes one slider which pushes another slider. Consider two sliders, $\mathcal{S}_1$ and $\mathcal{S}_2$, initially in contact with each other, and pushing $\mathcal{S}_1$ so that the pusher, $\mathcal{S}_1$, and $\mathcal{S}_2$ move together as a single rigid body. Once we find the pushing direction constraints, the push planner can be used directly.

To find the stable pushing direction constraints, we can do the following. Define $\mathcal{F}_1$ to be the cone of forces that can be applied to $\mathcal{S}_1$ through its contact with the pusher, $\mathcal{F}_2$ to be the cone of forces that $\mathcal{S}_1$ can apply to $\mathcal{S}_2$, and $\mathcal{F}_\cap = \mathcal{F}_1 \cap \mathcal{F}_2$. (Assume a common reference frame, not necessarily attached to the center of friction of either slider.) Ignoring $\mathcal{S}_2$, $\hat{\mathcal{V}}_1$ is the set of stable pushing directions of $\mathcal{S}_1$ from forces in $\mathcal{F}_1$. $\hat{\mathcal{V}}_2$ is the set of stable pushing directions of $\mathcal{S}_2$ from forces in $\mathcal{F}_2$. $\hat{\mathcal{V}}_\cap$ is the set of stable pushing directions of $\mathcal{S}_2$ from forces in $\mathcal{F}_\cap$. Then $\hat{\mathcal{V}}_{stable} = \hat{\mathcal{V}}_1 \cap \hat{\mathcal{V}}_\cap$ is a set of stable pushing directions for the entire system.[1]

To see this, consider that the force necessary for the motion of $\mathcal{S}_2$ is within the cone $\mathcal{F}_2$, and this force (which is also in $\mathcal{F}_1$), summed with the force to move $\mathcal{S}_1$, must lie within the cone $\mathcal{F}_1$. This reasoning can be extended to chains of more than two sliders. To actually find the velocity direction sets $\hat{\mathcal{V}}_1$ and $\hat{\mathcal{V}}_\cap$, we may be able to apply the procedure STABLE. In general, however, the force cone $\mathcal{F}_\cap$ does not fit the special structure assumed by STABLE, so a more general algorithm, such as that described in (Lynch [129]), must be used.

In principle we can do better if we know the relative magnitude of the support friction of $\mathcal{S}_1$ and $\mathcal{S}_2$. For example, if the weight of $\mathcal{S}_1$ is much greater than that of $\mathcal{S}_2$, then the force felt by the pusher is not affected by the motion of $\mathcal{S}_2$, and a set of stable pushing directions is $\hat{\mathcal{V}}_{stable} = \hat{\mathcal{V}}_1 \cap \hat{\mathcal{V}}_2$.

6.8 Open Problems

This planner ignores the issue of finding the path for the pusher from one feasible pushing contact to another, but assumes that a free path always exists. If the pusher is constrained to move in the plane of the obstacles, this problem must be addressed.

The push planner is asymptotically optimal in the sense that if a feasible pushing path exists, then with an appropriate choice of search parameters, it can find a path that minimizes the number of changes in the pushing contact. I can make no claim, however, that the

[1]Note that pushing directions in $\hat{\mathcal{V}}_{stable}$ may fail to be stable if there is an ambiguity due to Coulomb friction. Unlike the case of pushing a single object with line contact (Theorem 5.3), I have not proven that all other contact modes are impossible.

resulting path will be "short." An open problem is how to find shortest paths (by some metric) for a polygonal set of feasible velocity directions. Dubins [66] found a set of canonical paths in the obstacle-free plane that is guaranteed to include the minimum arclength path for a car that can only go forward. This result has been utilized by Jacobs and Canny [99] in planning paths among obstacles. Reeds and Shepp [173] enumerated a set of canonical paths, guaranteed to include the shortest path in the obstacle-free plane, for a car-like mobile robot that can reverse. This result has also been used in planning paths among obstacles (Laumond *et al.* [119]). As far as I am aware, there have been no similar results for more general sets of velocity directions.

I have assumed that the set of possible pushing contacts is specified by the user. An interesting problem is how to choose the set of pushing contacts based on knowledge of the pusher, the slider, the environment, and the initial and goal configurations. I have focused on line contact pushing, but a similar approach can be used with other types of pushing contacts, particularly using the algorithm of (Lynch [129]).

To manipulate an unknown object, a robot could estimate the friction parameters of the object (Yoshikawa and Kurisu [224]; Lynch [130]), perform the mechanics analysis, and use the results in the planner described here. Alternatively, the robot could empirically determine a set of stable pushing directions and plug these directly into the planner.

The pushing paths shown here are stabilized by using more than one contact point between the pusher and the slider. No sensing is required—the inherent mechanics of the task essentially close a tight feedback loop. Nevertheless, unmodeled effects, such as variations in the support plane, could cause the pusher to lose control of the motion of the object. The planner described here could be considered the feedforward component of a pushing control system.

Chapter 7

Application to Parts Feeding

Controllability and small-time local controllability are desirable properties for a manipulation system, but they are not really necessary for parts feeding, where the goal is to move an object to a particular configuration for later processing. We require only that our parts feeding system possess the *feeding property*:

> A system has the *feeding property* over a set of parts $\mathcal{P}$ and set of initial configurations $\mathcal{I}$ if, given any part in $\mathcal{P}$, there is some output configuration $\mathbf{q}$ such that the system can move the part to $\mathbf{q}$ from any location in $\mathcal{I}$.

In other words, the part should be "controllable to" some configuration. This chapter demonstrates that a one joint robot can possess this property over a useful set of initial configurations and a broad class of planar parts.

The key is to use a single revolute joint to push the parts around on a constant speed conveyor belt. This approach, referred to as "1JOC" (one-joint-over-conveyor, pronounced "one jock"), was initially conceived as a variation on the Adept Flex Feeder. The Flex Feeder uses a system of conveyors to circulate parts, presenting them with random orientation to a camera and robotic manipulator. Those parts that are in a graspable configuration may then be picked up by the robot and assembled into a product, placed in a pallet, or otherwise processed.

The question addressed in this chapter is whether, at least in parts feeding applications, we could replace the SCARA robot with a simpler and more flexible robot, and also replace the Flex Feeder's servoed programmable conveyor with a fixed speed conveyor. Figure 7.1 shows a possible variation on the Flex Feeder, where the SCARA robot has been replaced by a fence with a single revolute joint. By a sequence of pushing operations, punctuated by drift along the conveyor, the fence positions and orients a part and directs it into the entry point of a feeder track which carries the part to the next station.

[1]This chapter is joint work with Srinivas Akella, Wes Huang, and Matt Mason, and previously appeared in (Akella *et al.* [6]).

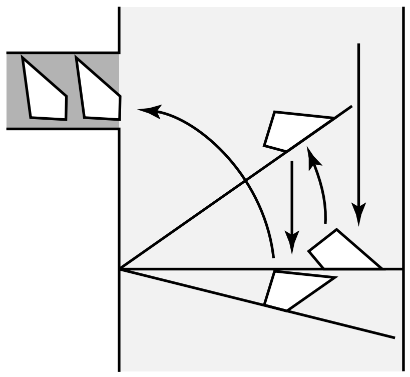


Figure 7.1: We can feed a part by alternately pushing it with the fence and letting it drift along the conveyor.

The main result of this chapter is:

> It is possible to move an arbitrary polygon from a broad range of initial configurations to a specific goal; and that goal can be chosen from a broad range of possible goals. (This is a generalization of the feeding property.)

We begin by developing a progression of models leading to the 1JOC. We prove the feeding property for the 1JOC model and describe the implementation of a planner as well as experimental results.

7.1 A Progression of Models

The problem of pushing a part across a moving conveyor is complicated by the mechanics of pushing. Some insights can be gained by first considering an idealized model: the part matches the fence's motion whenever desired, and matches the conveyor's motion otherwise. In this section we examine a progression of models leading to the 1JOC system.

One Rotation Center First consider an idealization of a rotating fence—an infinite turntable to which the part can be affixed. This turntable can give the part an arbitrary angular velocity about its pivot.

From any initial point in the part configuration space, the reachable set is one dimensional. If we consider the full pushing mechanics for a rotating fence, the reachable set is also just one dimensional. If the fence can swing all the way around and push on the other side, however, the part's configuration may be accessible. But we will not consider this case further.

Two Rotation Centers Now consider a pair of overlapping (yet independent) ideal turntables. At any given instant, the part is affixed to one of the two turntables and rotates about the center of that turntable. We assume that we can instantaneously switch the part from one turntable to the other.

There are two cases to consider:

- Both turntables are bidirectional. If each turntable can be driven in either forward or reverse, the part's configuration is small-time locally controllable.

- One or both turntables are unidirectional. The part's configuration is controllable but not small-time locally controllable.

We can model the conveyor as the limiting case of a turntable whose pivot approaches infinity. The vector fields $X_1 = (0, 1, 0)^T$ and $X_2 = (-y, x, 1)^T$ correspond to the motion of the conveyor and the turntable, respectively, where the origin is at the center of the turntable. When the part tracks the conveyor, the part motion is given by $\dot{\mathbf{q}} = vX_1$, where v is the velocity of the conveyor in the y direction. When the part tracks the turntable, the part motion is given by $\dot{\mathbf{q}} = \omega X_2$, where ω is the angular velocity of the turntable. The Lie bracket $[X_1, X_2]$ of these two vector fields is $(-1, 0, 0)^T$. The bracket is linearly independent of X_1 and X_2, yielding a third controlled freedom (provided both v and ω can be nonzero). The system is small-time locally controllable if both v and ω can be positive or negative, and simply controllable if the sign of either is fixed.

The 1JOC system consisting of a conveyor and a rotating fence resembles this system with a fixed v. One difference is that rotating the part about the pivot is only stable on a subset of the configuration space, specifically when an edge of the part is aligned with the fence.

Detailed Pushing Model We now consider pushing mechanics in the model with a conveyor and a rotating fence. We assume that the friction coefficient at the pushing contact and the distribution of support forces are known, and we place no restriction on the fence motions. Although this model makes unrealistic assumptions about the available information, any negative results will also apply to less detailed models.

Figure 7.2 shows that the system is not small-time locally controllable for the case of a thin rod. If the fence pushes from below, the rod's angular velocity is positive. If the fence swings around and pushes the part from above, the angular velocity is still positive. There is no maneuver combining these actions and conveyor drift that can locally achieve negative rotations.

Practical Pushing Models The detailed pushing model is difficult to use because it requires full knowledge of the support friction distribution $s(\mathbf{x})$. We can define more abstract models by restricting the allowable actions. In particular, we would like to use actions with predictable outcomes, such as stable pushes, that do not impose unrealistic demands for information.

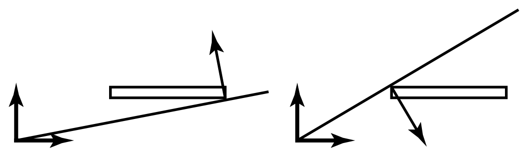


Figure 7.2: The thin rod is not small-time locally controllable from this configuration—it can only be rotated with a positive angular velocity, regardless of whether the fence pushes from above or below.

Any abstraction of the detailed pushing model will inherit its limitations, thus it is immediately clear that small-time local controllability is impossible.

7.2 The Feeding Property

In this section we focus on a particular model for the 1JOC and prove that it can transfer any polygonal part from any valid input configuration to a single goal configuration. To prove this property, we utilize a subset of the actions available to the 1JOC and show that they are sufficient for the feeding property.

7.2.1 The 1JOC Model

The fence is a line that pivots about a fixed point on the line. The origin of a fixed world frame $\mathcal{F}_{\mathcal{W}}$ coincides with the pivot point. The conveyor is the half-plane $x > 0$, with a constant drift velocity v in the $-y$ direction. The fence angle θ is measured with respect to the x axis of $\mathcal{F}_{\mathcal{W}}$, and its angular velocity is given by ω. The fence angular velocity ω is our single control input.

The part can be any polygon, but because the manipulator is a line, the part can always be treated as its convex hull. The center of mass of the part lies in the interior of the convex hull and the coefficient of friction between the fence and the part is nonzero. When we refer to a "part," we assume these properties.

It is convenient to represent the linear and angular velocity of the part relative to the conveyor in a frame $\mathcal{F}_{\mathcal{S}}$ attached to the center of mass of the part. Velocity directions are represented as rotation centers in the frame $\mathcal{F}_{\mathcal{S}}$. Note that the support friction acting on the part during pushing is determined by the motion of the part relative to the conveyor, not the world frame $\mathcal{F}_{\mathcal{W}}$.

The configuration of the part frame $\mathcal{F}_{\mathcal{S}}$ in the world frame $\mathcal{F}_{\mathcal{W}}$ is given by $(x, y, \phi)^T$. (For notational simplicity in this chapter, the subscript is dropped from the usual world coordinates, $(x_w, y_w, \phi_w)^T$.) When an edge of the part is aligned with the fence, the contact radius r defines the distance from the fence's pivot point to the closest point on the edge (the *contact vertex*), and the configuration of the part can be given in the polar coordinates (r, θ). The location of the point at r on the fence is (r_x, r_y) in the world frame. See Figure 7.3.

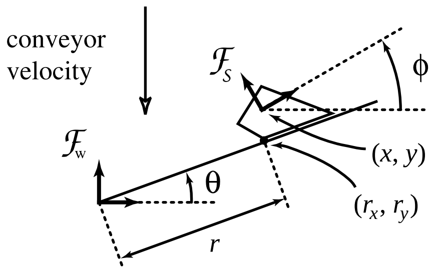


Figure 7.3: 1JOC notation.

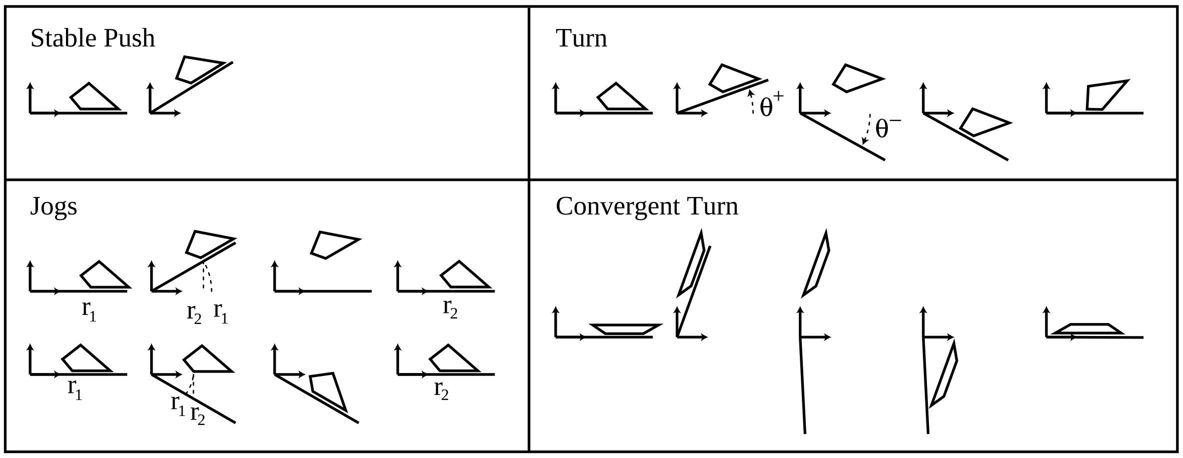


Figure 7.4: Step by step illustrations of the 1JOC primitives.

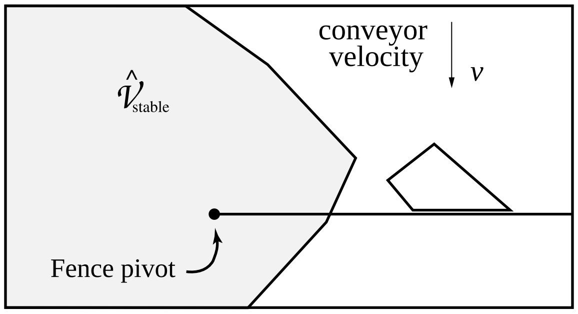
7.2.2 1JOC Primitives

We now describe the basic 1JOC primitives which yield the feeding property. We assume the fence is initially held at 0 degrees, perpendicular to the conveyor velocity, until the part contacts and comes to rest against the fence. See Figure 7.4 for an illustration of the primitives.

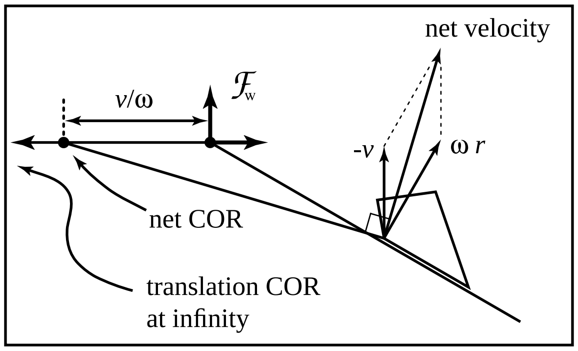
Stable Pushes

The simplest example of a stable push occurs when the fence is held at 0 degrees, and a part on the conveyor drifts into contact and comes to rest on the fence. When the part is at rest, the fence is executing a stable push. The pushing direction, as seen by the part, is opposite the conveyor's velocity direction.

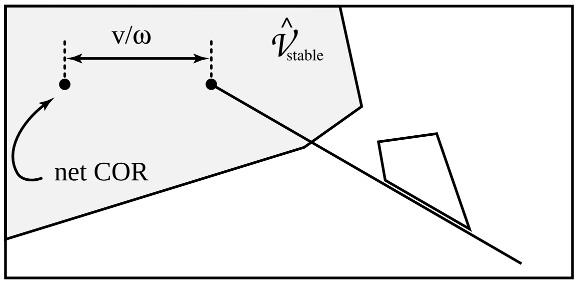
Of course, a stable push may also occur while the fence is rotating. The procedure STABLE is used to find a set of stable pushing directions $\hat{\mathcal{V}}_{stable}$, fixed in the part frame $\mathcal{F}_{\mathcal{S}}$. The fence is guaranteed to execute a stable push if the edge is aligned with the fence and the motion of the fence and the conveyor combine to yield a pushing direction in $\hat{\mathcal{V}}_{stable}$. See Figure 7.5 for details.



For a given edge contact, we can determine $\hat{\mathcal{V}}_{stable}$, the set of pushing rotation centers that keep the object fixed to the pusher. Note that $\hat{\mathcal{V}}_{stable}$ is fixed in the part frame, not the world frame.



The conveyor velocity v and the fence angular velocity ω combine to form a net velocity at each point. This can be expressed as a net COR, which lies on the line through the fence pivot and perpendicular to the conveyor velocity.



Provided the net COR is contained in $\hat{\mathcal{V}}_{stable}$, the part will remain stationary with respect to the fence for that combination of conveyor and fence velocities. Note that as the fence rotates, $\hat{\mathcal{V}}_{stable}$ rotates with the part.

Figure 7.5: Testing the stability of a push on a moving conveyor.

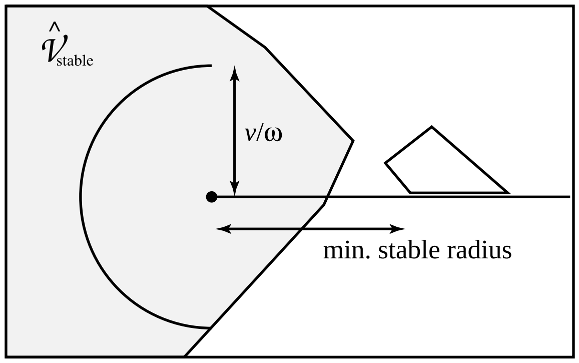


Figure 7.6: In the part frame $\mathcal{F}_{\mathcal{P}}$, the locus of net COR form a semicircle of radius ω/v about the fence pivot as the fence swings from -90 to 90 degrees. Given a fence angular velocity ω and a conveyor velocity v, we pick a minimum stable radius so that this semicircle lies completely within $\hat{\mathcal{V}}_{stable}$. Recall that $\hat{\mathcal{V}}_{stable}$ is fixed with respect to the part, so increasing the contact radius moves the semicircle deeper into $\hat{\mathcal{V}}_{stable}$. This conservative estimate of the minimum stable radius guarantees stable pushing in the counterclockwise direction at any fence angle in $[-90°, 90°]$.

For simplicity, we will only consider the edges of the part that yield a stable push while the fence is held at 0 degrees (the pushing direction is a translation normal to the edge). The existence of these stable edges is indicated by Theorem 5.5, restated here for convenience.

Lemma 1 *All polygonal parts have at least one edge such that the normal translational pushing direction, along with a neighborhood of this pushing direction, belongs to $\hat{\mathcal{V}}_{stable}$. Such edges are called* stable edges.

Remark: The object may also have *metastable edges*—edges such that the center of mass lies directly above an edge endpoint when the fence is held at 0 degrees. If the fence rotates any nonzero amount in the "wrong" direction, the edge will become unstable. We will avoid these edges. If the part initially comes to rest on the fence on one of these edges, we can perturb the fence slightly to bring the part to a stable edge.

Lemma 1 says that a stable edge is also stable for a neighborhood of fixed fence angles around 0 degrees. In addition, if the contact radius r is sufficiently large, the fence pivot will be inside the set of stable rotation centers $\hat{\mathcal{V}}_{stable}$ fixed in the part frame $\mathcal{F}_{\mathcal{S}}$. If the fence angular velocity ω is large enough relative to the conveyor velocity v, the combined pushing motion is nearly a pure rotation about the pivot, and the pushing motion is stable. Therefore, it is possible to stably push the part from any θ_0 to any θ_1 in a counterclockwise (CCW) direction, where $90° > \theta_1 > \theta_0 > -90°$. Figure 7.6 illustrates a method for finding the *minimum stable radius*: the minimum contact radius r such that the push is stable for all fence angles in the range $[-90, 90]$ for a given fence angular velocity ω and conveyor velocity v.

For a push to be stable when the fence is rotating CCW, the contact radius r must be sufficiently large. We can change r by performing jogs, described below.

Jogs

Jogs are simple maneuvers that allow us to change the contact radius r from the fence pivot to the part without changing the stable resting edge.

Lemma 2 *A part resting on a stable edge can be moved from any initial contact radius r on the fence to any desired r in the range $(0, \infty)$ by a series of jogs.*

Proof: We use the fact that a neighborhood of pushing directions about the normal translation is stable for a stable edge. In fact, there is a range $[\theta^{min}, \theta^{max}]$ of fixed fence angles about 0 degrees that are also stable. The fence does not have to be motionless to execute a stable push, however; if the fence angular velocity ω is small enough with respect to the conveyor velocity v, then the fence can execute a stable push while moving in the angle range $[\theta^{min}, \theta^{max}]$, regardless of the contact radius r.

To decrease the contact radius r, the fence is raised to an angle θ^+ ($\theta^{max} \geq \theta^+ > 0$), keeping the part fixed to the fence by a stable push. This changes the value of r_x to $r_x \cos\theta^+$. The fence then drops to 0 degrees, immediately releasing the part. The part drifts back into contact with the fence and settles on the same stable edge. We assume there is no slip between the part and the fence. (The no-slip assumption is important for open-loop feeding plans.) The result of the jog is to decrease the contact radius from r to $r\cos\theta^+$.

To increase the contact radius r, the fence is quickly lowered to an angle θ^- ($\theta^{min} \leq \theta^- < 0$), releasing the part. The part drifts back into contact with the fence and settles on the same stable edge. The fence is then raised to 0 degrees again, pushing the part with a stable push. The contact radius of the part has changed to $r/\cos\theta^-$.

A series of jogs, either increasing or decreasing r, can bring the part to any r in the range $(0, \infty)$. $\square$

A jog is directly analogous to the Lie bracket motion for the ideal turntable plus conveyor system of Section 7.1. The difference from the ideal system is that the part only tracks the fence when it is in stable contact. Nonetheless, we can now show that the configuration of the part is accessible with just stable pushes, jogs, and the conveyor drift.

Theorem 7.1 *The configuration of a polygonal part in stable edge contact with a fence held at 0 degrees is small-time accessible using stable pushes, jogs, and conveyor drift.*

Proof: Given any neighborhood of the part configuration while it is in stable edge contact with the fence at 0 degrees, it is possible to jog the part a nonzero distance in both directions without leaving this neighborhood. The effect of the jog is to change the x coordinate of the part configuration, exactly like the Lie bracket motion of Section 7.1. The other two vector fields of Section 7.1 can be obtained directly by conveyor drift and stable pushes. □

We call $\mathcal{I}$ the set of initial part configurations that drift to stable edge contact with the fence at 0 degrees. From a stable edge contact, Theorem 7.1 indicates that the configuration of the part is small-time accessible. Therefore, the configuration of the part is accessible from the set $\mathcal{I}$.

We still have not proven the feeding property. To get this property, we need the ability to turn a part to a new stable edge.

Turns

Turns allow us to change the edge of the part aligned with the fence, subject to the constraint that the initial and final edges both be stable edges. A turn consists of raising the fence to an angle $\theta^+ \geq 0$ by executing a stable push; dropping the fence to an angle $\theta^- \leq 0$, immediately releasing the part; and reacquiring the part on the new edge with a stable push, raising the fence back to 0 degrees. The final push commences at the moment the new edge contacts the fence.

Lemma 3 *Any polygonal part has at most one stable edge from which it is impossible to turn the part to the next CW stable edge.* Exception: *a part has two such stable edges if they are the only stable edges and they are parallel to each other.*

Proof: The fence can execute a stable push to any angle less than 90 degrees, and it can reacquire the part with a stable push at any angle greater than −90 degrees. This indicates that it is possible to reach any final stable edge less than 180 degrees CW of the initial edge. If the part has two or more stable edges, there can only be one stable edge from which the next CW stable edge is greater than 180 degrees removed. For "exception" parts, there are two stable edges which are exactly 180 degrees removed from each other. □

There are a few important points to make about turns:

- The contact radius r, for both the initial and final edges, must be large enough that the pushing directions always remain in $\hat{\mathcal{V}}_{stable}$.

- The quantity $\theta^+ - \theta^-$ is determined by the part geometry, but the actual values of θ^+ and θ^- are not. To some extent, the contact radius r after the turn can be controlled by the choice of these values. If the angle difference between the initial and final edge is 90 degrees or more, r can be changed to any value in the range $(0, \infty)$ during the turn.

- This primitive requires precise knowledge of the conveyor's motion—the final stable push must begin precisely when the new edge contacts the fence.

Consider the case where turning a part to a new stable edge requires raising the fence to $90 - \delta$ and catching the part at $-90 + \epsilon$, where δ and ϵ are small positive values. It may be necessary to move the part to a large contact radius r before executing the turn to ensure that the part remains completely on the conveyor ($x > 0$) during the initial push.

Convergent turns

To perform a turn between two stable edges which are parallel, the fence must, in principle, be raised to 90 degrees, pushing the part off the conveyor. To turn such parts, we can introduce a convergent turn: raise the fence to $90 - \delta$, allow the part to drift down, and begin pushing it again at $-90 + \epsilon$. Assuming no slip at the pushing contact, the part will converge to the new stable edge as the fence is raised to 0 degrees. Convergent turns allow us to strengthen Lemma 3 to apply to all polygonal parts, with no exceptions. Here, the sole function of convergent turns is to handle parts with only two stable edges, which are parallel.

7.2.3 The Feeding Property

Using the lemmas proven above, we can now demonstrate the feeding property for the 1JOC.

Theorem 7.2 *The 1JOC possesses the feeding property:*

- *for all polygonal parts*
- *for the three-dimensional space $\mathcal{I}$ of initial configurations such that the part initially comes to rest on a stable edge, completely on the conveyor, against the fence fixed at 0 degrees, and*
- *using only stable pushes, jogs, turns, convergent turns, and conveyor drift.*

Furthermore, the goal configuration can be chosen from a three-dimensional subset of the configuration space.

Proof: If the part has a stable edge such that it cannot be turned to a new stable edge, then this edge must be chosen as the goal edge. Otherwise, any stable edge can be chosen as the goal edge. By Lemma 3 (strengthened by the convergent turn of Section 7.2.2), this goal edge can be reached using turns. Once at the goal edge, the part can be jogged to any contact radius r in the range $(0, \infty)$ (Lemma 2). By Lemma 1, the part is stable against the fence for a range of fence angles θ. The goal edge therefore allows a two-dimensional set of final stable configurations of the part in the (r, θ) space. If the fence is quickly lowered to 90 degrees, releasing the part, this set drifts to a set of reachable configurations with nonempty interior in the world configuration space. Any configuration in this set can be chosen as the goal. □

If the application of the 1JOC is to stuff parts into a feeder track, then the fence can be rotated 90 degrees, pushing the part off the conveyor into the feeder track, instead of releasing it to continue on the conveyor.

7.3 A Feeding Planner

The feeding property means that there exists a sequence of jogs and turns to feed any polygonal part, assuming infinite fence length and an infinite conveyor half-plane. A 1JOC with finite fence and conveyor dimensions may not be able to feed some parts. For a given part however, we can always find a fence and conveyor with finite dimensions to feed it. We describe how to find a sequence of jogs and turns to feed a given part with a fence and conveyor of known dimensions in the minimum amount of time.

We assume that we know the shape and center of mass of the part, the coefficient of friction between the fence and part, the conveyor velocity, and the fence and conveyor dimensions. The coefficient of friction between the part and conveyor is assumed constant, and we pick the fence angular velocity to be one of three discrete values: ω_{stable} (for stable pushes), ω_{drop} (when lowering the fence), and zero. We assume the fence receives parts that are singulated.

A part drifts down the conveyor until it comes to rest against the fence. The feeding problem consists of getting the part from this start configuration to the goal configuration (see Figure 7.7), where the configurations are specified by the edge aligned with the fence, and the contact radius (distance from pivot to the closest vertex of the resting edge). We assume that the initial fence orientation is zero, and, for convenience, that the final fence orientation is zero as well.

Turns change the part orientation; they rotate the part CCW so the new edge aligned with the fence is CW to the initial edge. Turns thus enable transitions to stable edges in the CW direction. We consider all CW sequences of edges beginning at the start edge e_s and ending at the goal edge e_g which do not require rotating the part more than 360 degrees. Take an example sequence $S = \{e_s, e_j, e_k, e_g\}$. If we can find a valid sequence of jogs and turns to move the part in sequence through the edges in S and change the contact radius by the required amount, we have a feasible plan. Of all feasible plans, we take the one which requires minimum time.

7.3.1 A Simple Example

Consider a part resting on edge e_j (orientation ϕ_j) at radius r_j which is to be moved to the neighboring CW stable edge e_k (orientation ϕ_k) at radius r_k (see Figure 7.7). Here the sequence of edges is $S = \{e_j, e_k\}$. We look for a feasible solution (and ignore minimizing the

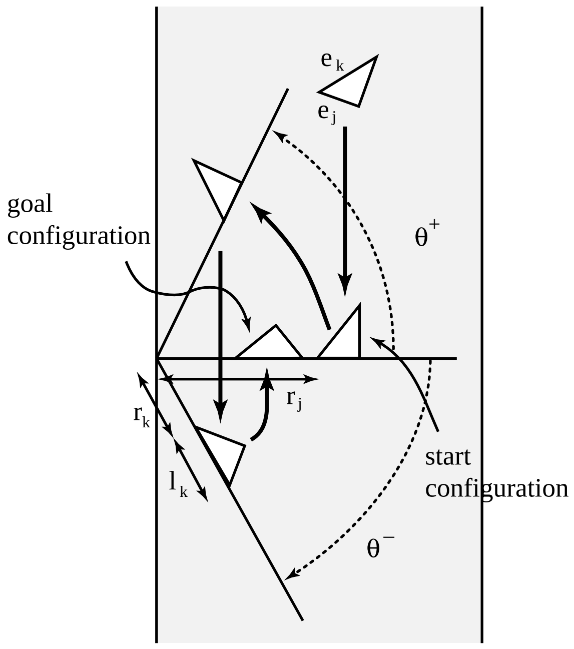


Figure 7.7: A feeding plan to move a triangle from start configuration (e_j, r_j) to goal configuration (e_k, r_k) by a turn with raise and drop angles of θ^+ and θ^-. Length of edge e_k is l_k.

time). Assume r_j and r_k are greater than the minimum stable radii for the corresponding edges, and are close enough in magnitude that no jog is required. A plan consists of determining the fence angles θ^+ and θ^- for the turn. From the geometry, we have:

$$r_k = r_j \frac{\cos\theta^+}{\cos\theta^-} - l_k$$

$$\theta^+ - \theta^- = \phi_k - \phi_j$$

Solving these equations, we find:

$$\theta^- = \tan^{-1}\left(\cot(\phi_k - \phi_j) - \frac{r_k + l_k}{r_j \sin(\phi_k - \phi_j)}\right)$$
$$\theta^+ = \phi_k - \phi_j + \theta^-$$

This simple example has a closed form solution that provides us with a plan. (Note that when $(\phi_k - \phi_j) \geq 90°$, the radius change can be arbitrarily large or small.)

7.3.2 Nonlinear Programming Approach

For the general case, we must find a sequence of jogs and turns that effects the desired configuration change and minimizes time. For a given sequence of edges, we must determine

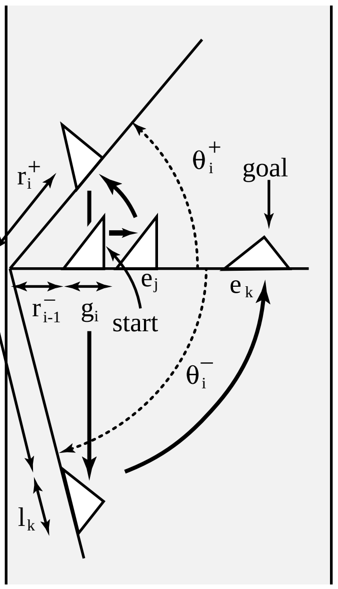


Figure 7.8: The ith transition from edge e_j to edge e_k with a translation of g_i, start and end turn radii r_i^+ and r_i^-, and raise and drop angles θ_i^+ and θ_i^-.

the parameters for each turn while ensuring that each contact edge in a turn has a contact radius that is no less than its minimum stable radius. If the contact radius is less than the minimum radius, we must translate the part prior to executing the turn. Such a *translation* may consist of a jog or a series of jogs.

A feeding plan to transfer a part from its start configuration (e_s, r_s) to its goal configuration (e_g, r_g) consists of several stages, each of which accomplishes an edge transition. Each stage consists of a translation followed by a turn; the translation for a given stage may be zero. A plan with the edge transition sequence S will take $n-1$ stages, where $n = |S|$. A final translation may also be required after the goal edge has been reached.

Consider the ith stage in a plan (see Figure 7.8) to accomplish the transition from edge e_j to edge e_k. First, we do a translation of g_i, which moves the part to a contact radius of r_i^+. Since we are about to do a turn, this radius must be greater than the minimum stable radius for edge e_j, r_j^{stable}. We then perform a turn with a raise angle of θ_i^+ and a drop angle of θ_i^-. The stable push at the end of the turn brings the fence to $\theta = 0$ with the new contact edge e_k. For this stable push on edge e_k, the contact radius r_i^- must be greater than the minimum stable radius for e_k, r_k^{stable}. We also have to ensure that the fence raise and drop angles are in valid ranges, that the part is within fence and conveyor bounds, and that the fence contacts the part only at the end of the drift phase.

Assume the fence pivot is on the conveyor left edge, and that the conveyor half-length exceeds its width. The resulting constraints are (for $i = 1, \ldots, n-1$):

$$r_i^+ = r_{i-1}^- + g_i \tag{7.1}$$

$$r_i^- = \frac{(r_i^+ - d_j)\cos\theta_i^+}{\cos\theta_i^-} - (l_k + d_k) \tag{7.2}$$

$$\theta_i^+ - \theta_i^- = \phi_k - \phi_j \tag{7.3}$$

$$r_j^{max} \geq r_i^+ \geq r_j^{stable} \tag{7.4}$$

$$r_k^{max} \geq r_i^- \geq r_k^{stable} \tag{7.5}$$

$$90 > \theta_i^+ \geq 0 \tag{7.6}$$

$$0 \geq \theta_i^- > -90 \tag{7.7}$$

$$(r_i^+ + a_j^v)\cos\theta_i^+ - p_j^v \sin\theta_i^+ > 0, \forall v \in V \tag{7.8}$$

$$\omega_{drop} r_i^+ \cos\theta_i^+ > v_c \tag{7.9}$$

$$\frac{(r_i^+ - d_j)\sin\theta_i^+ - (r_i^- + l_k + d_k)\sin\theta_i^-}{v_c} > \frac{\theta_i^+ - \theta_i^-}{\omega_{drop}} \tag{7.10}$$

where l_k is the length of edge k, ϕ_j and ϕ_k are the orientations of edges e_j and e_k, v_c is the conveyor velocity, r_j^{max} and r_k^{max} are maximum valid contact radii for edges e_j and e_k, V is the set of part vertices, and a_j^v and p_j^v are the components along and perpendicular to edge e_j of the vector from the contact vertex to a part vertex v. If e_j and e_k are neighboring edges, $d_j = d_k = 0$. Else, d_j and d_k are the distances from the (virtual) intersection vertex of (extended) edges e_j and e_k to the contact vertices of these edges.

The start and goal constraints are:

$$r_0^- = r_s \tag{7.11}$$

$$r_g = r_{n-1}^- + g_n \tag{7.12}$$

These define a set of nonlinear constraints over the variables g_i, θ_i^+, θ_i^-, r_i^+, and r_i^-. (We could also eliminate the equality constraints and solve for fewer variables.) Since the radius at the start of a turn depends on the radius at the end of the previous turn, an optimal solution must consider all variables simultaneously.

We want a solution which minimizes the total time to feed the part. The time taken for each stage of the plan is the sum of the jog time, fence raise time, drift time on the conveyor, and fence recovery time. We approximate the total time by the objective function we minimize:

$$t_n g_n^2 + \sum_{i=1}^{n-1}(t_i g_i^2 + \frac{\theta_i^+}{\omega_{stable}} - \frac{\theta_i^-}{\omega_{stable}} + \frac{(r_i^+ - d_j)\sin\theta_i^+ - (r_i^- + l_k + d_k)\sin\theta_i^-}{v_c})$$

The approximate time cost coefficient of the ith jog series is $t_i = (k\cos\alpha_i)/(v_c(r_i^+ + r_{i-1}^-))$ where k is a constant chosen to be 1000, and $\alpha_i = \min(|\theta_i^{min}|, |\theta_i^{max}|)$ is a safe jog angle for the ith jog series.

Our problem as stated above is a nonlinear programming problem whose feasible solutions yield valid feeding plans. Once we have a solution for the g_i, θ_i^-, θ_i^+, r_i^+, and r_i^- values which

(a) (b) (c)

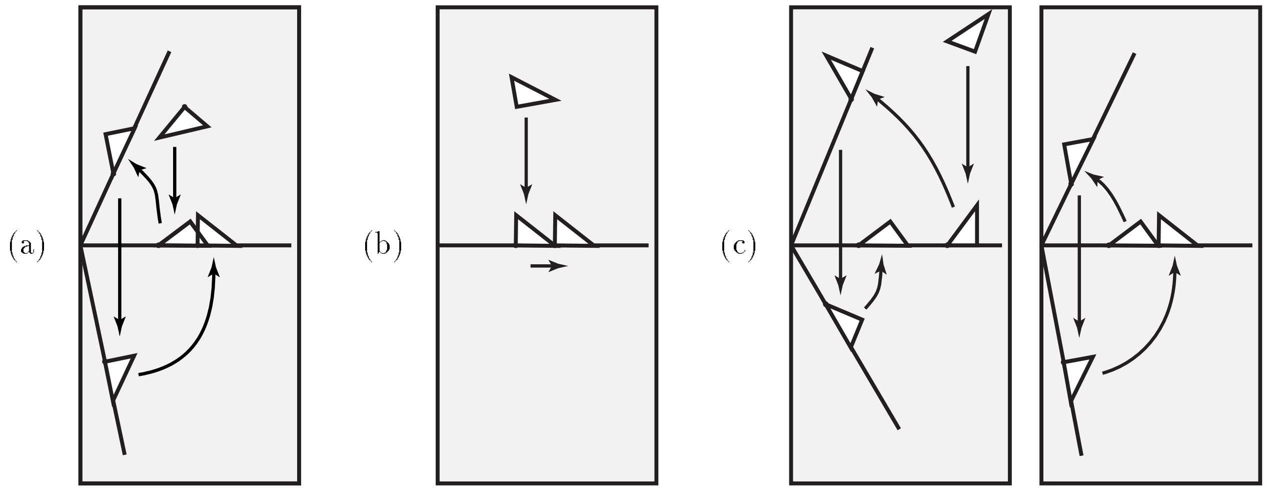


Figure 7.9: Example feeding plans for a triangle. The goal configuration ($\phi_g = 269.34, r_g = 150$) is the same, while the start configurations are different: (a) ($\phi_s = 126.49, r_s = 100$) (b) ($\phi_s = 269.34, r_s = 100$) (c) ($\phi_s = 0, r_s = 200$).

satisfies the constraints (7.1)–(7.12) and minimizes our objective function, we must still determine a series of jogs to execute the translations g_i.

An inward translation distance $g_i < 0$ requires a series of decreasing jogs, such that $r_i^+ = r_{i-1}^-(\cos\gamma_i)^m \cos\rho_i$ where m is a nonnegative integer, and $\theta_i^{max} \geq \gamma_i \geq \rho_i \geq 0$. So this radius decreasing translation is accomplished by a series of m jogs of angle γ_i followed by a jog of ρ_i. Similarly, an outward translation distance $g_i > 0$ requires a series of increasing jogs, such that $r_{i-1}^- = r_i^+(\cos\gamma_i)^m \cos\rho_i$ where m is a nonnegative integer, and $0 \geq \rho_i \geq \gamma_i \geq \theta_i^{min}$.

A feasible solution to the above problem always exists if the fence and conveyor are large enough that each stable edge has a valid contact radius greater than its minimum stable radius. When this condition is satisfied, a plan with a maximum of three stages to get the part from the start to the goal configuration exists, although it may not be optimal.

7.3.3 Generating Feeding Plans

From the part description and coefficient of friction, we compute the minimum stable radii for the stable edges and the transition angles between these edges. For the start and goal configurations, we generate candidate edge transition sequences. For each such sequence, the corresponding objective function and constraints are input to the commercial nonlinear programming package GINO [125] to solve the problem. A feasible solution to the nonlinear programming formulation provides us with a valid feeding plan. See the example plans in Figure 7.9.

Currently we use minimum stable radius values empirically determined to be robust. A conservative estimate of the minimum stable radius can be determined as illustrated in Figure 7.6. Also, the planning procedure can be extended to use convergent turns to handle parts with only two stable edges which are parallel.

7.4 Implementation

We have implemented several feeding plans on a conveyor with the fence being (somewhat ironically) actuated by one joint of an Adept 550 SCARA robot. The conveyor speed and fence rotational speed have been selected to be 20 mm/s and 30 degrees/s respectively. The fence is covered with a foam material to avoid slip between the part and the fence.

Currently we place the part at the start configuration and observe the position of the part after plan execution. In a fully implemented setup, singulated parts would come down the conveyor in random configurations. The initial part orientation and contact radius would be determined by an overhead camera, and a feeding plan to get the part to the goal configuration without any further sensing would be generated.

Figure 7.9 illustrates three plans we ran on our setup which achieve the same goal configuration from different starting configurations. Plan (a) requires a single turn to reach the goal, plan (b) requires just a series of jogs, and plan (c) requires two turns. These plans had position errors at the goal configuration ranging from 0 mm to 3 mm. The errors are sensitive to part shape measurements, fence turn angles, and timing in the turns. Occasional slip also causes some error.

7.5 Variations on a Theme

The 1JOC approach uses a fixed velocity conveyor in combination with a single servoed joint to obtain the diversity of motions required for planar parts feeding. We have shown by proof and demonstration that the 1JOC is capable of useful planar manipulation: any polygon is controllable from a broad range of initial configurations to any goal chosen from a broad range of goal configurations.

We designed the system and the set of actions to simplify analysis and planning. There are many variations on the approach which may be more suitable in different contexts. Some variations on the system configuration are: a curved fence; a prismatic fence in place of the revolute fence; a rotary table in place of the conveyor; or use of gravity rather than a conveyor. Some variations on the actions are: optimization of fence rotation rates instead of using fixed values; allowing objects to slip along the fence; complete revolutions of the object to reduce jogs; and speeds high enough to require dynamic analysis instead of quasistatic. To handle three-dimensional objects, we could use techniques similar to Bicchi and Sorrentino's to roll objects on the belt [26].

When addressing potential industrial applications, it is important to consider the whole system, including interactions with surrounding equipment. Several different scenarios have occurred to us: a 1JOC to pose objects followed by a simple pick-and-place device; a 1JOC to singulate parts for a SCARA; or two or three 1JOCs pipelined to singulate and feed parts.

Part II

Dynamic Manipulation

Chapter 8

Dynamic Manipulation

The change of motion is proportional to the motive force impressed; and is made in the direction of the right line in which that force is impressed.
— Sir Isaac Newton, *Principia*

Imagine now that we turn off the friction between the object and the surface it moves on, so the object no longer comes to rest after we stop pushing it. We have moved from the Aristotelian world of quasistatic manipulation to the Newtonian world of dynamic manipulation.

In the second half of this thesis I focus on the *dynamic pick-and-place* problem of finding a manipulator motion to move an object from one state to another. This problem differs from traditional pick-and-place in that the state of the object includes its velocity. The contact is also nonprehensile, which places constraints on the motion of the object.

The key to dynamic nonprehensile manipulation is to arrange the manipulator motion such that dynamic coupling forces through the frictional nonprehensile contact, along with gravity, integrate over time to cause the desired motion of the object. By sequencing control of the object's state variables during dynamic grasping, sliding, rolling, and free-flight phases, it is often possible to control more degrees-of-freedom of the object than the manipulator itself has.

Much of the discussion will focus on the simplest possible manipulator: a single revolute joint. This focus stems from two important considerations. (1) Despite its simplicity, a one-degree-of-freedom arm is capable of interesting manipulation in the plane. In general, it can cause a planar object to reach a six-dimensional subset of its six-dimensional state space. (2) We have a one-degree-of-freedom direct-drive arm available in our lab to test our nonprehensile dynamic strategies. This arm moves in the vertical plane.

Part II is organized as follows. Chapter 9 discusses the controllability of an object by dynamic nonprehensile manipulation. Because of the difficulty of releasing and recontacting a moving object, particular attention is paid to the case where the manipulator makes a single sustained contact. Chapter 10 outlines an optimization procedure for trajectory planning for dynamic manipulation. Chapter 11 describes results of the planner which have been implemented on the one-degree-of-freedom arm. The robot successfully executes dynamic tasks such as snatching an object from a table, rolling an object on its surface, and throwing and catching. Finally I conclude with a discussion of the results in Chapter 12.

8.1 Definitions

A world frame $\mathcal{F}_\mathcal{W}$ is fixed in the world, and an object frame $\mathcal{F}_\mathcal{O}$ is attached to the center of mass of the object. The configuration of the object frame $\mathcal{F}_\mathcal{O}$ in the world frame $\mathcal{F}_\mathcal{W}$ is denoted $\mathbf{q} = (x_w, y_w, \phi_w)^T$, and the velocity of the object in the world frame is denoted $\dot{\mathbf{q}} = (\dot{x}_w, \dot{y}_w, \dot{\phi}_w)^T$. The state of the object is $(\mathbf{q}, \dot{\mathbf{q}})$. The mass of the object is m and its radius of gyration of the object is ρ, making its inertia $m\rho^2$. (In what follows, we are not concerned with the mass m, so only the radius of gyration ρ is of interest.)

The state of the manipulator $\mathcal{M}$ is written $(\Theta, \dot{\Theta})$, where Θ is the joint configuration vector. Despite the notation, each joint can be either prismatic or revolute.

Chapter 9

Controllability of Dynamic Manipulation

This chapter examines the controllability of dynamic nonprehensile manipulation. I begin by developing a geometric interpretation of the mapping from applied forces to accelerations of the object, taken from (Lynch and Mason [132]). This geometric interpretation is simply to aid the intuition, and is analogous to the limit surface of Part I. A key difference is that the limit surface has no closed-form representation, and there is no closed-form solution to the motion of a pushed object.

I then study the controllability of an object manipulated with point or line contact, assuming no constraints on the motion of the manipulator. Because of the drift field, however, manipulator constraints play a more prominent role than in quasistatic pushing—the object does not sit still as the contact is changed. I therefore study the accessibility of an object during a single sustained contact. Examples show that even a single-degree-of-freedom manipulator can take an object to an open subset of its state space.

To develop some intuition as to how to control an object's state, the chapter concludes with a discussion of algorithmic control, the basis of the dynamic manipulation planner of Chapter 10.

9.1 The Dynamic Analog to the Limit Surface

The acceleration of the object $\mathbf{a} = (a_x, a_y, \alpha)^T$ is related to the applied force $\mathbf{f} = (f_x, f_y, \tau)^T$ by $\mathbf{f} = \mathbf{M}\mathbf{a}$, where

$$\mathbf{M} = \begin{pmatrix} M_{xx} & M_{xy} & M_{x\phi} \\ M_{yx} & M_{yy} & M_{y\phi} \\ M_{\phi x} & M_{\phi y} & M_{\phi\phi} \end{pmatrix}.$$

When torques are measured about the center of mass of the object, $\mathbf{M}$ is simply the diagonal matrix

$$\mathbf{M} = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m\rho^2 \end{pmatrix}.$$

The equations of motion are easily stated algebraically, but to aid the intuition, and to continue the geometric development of Part I, note that the mapping between the applied force and the resultant acceleration is governed by an ellipsoidal surface analogous to the limit surface of quasistatic pushing. An applied force $(f_x, f_y, \tau)^T$ maps to an acceleration $(a_x, a_y, \alpha)^T$ normal to this ellipsoid at $(f_x, f_y, \tau)^T$.

To see this, consider the ellipsoid $F(\mathbf{f}) = \frac{1}{2}\mathbf{f}^T\mathbf{A}\mathbf{f} = c^2$, where c is a scaling factor depending on the force magnitude f, and $\mathbf{A} = \mathbf{M}^{-1}$:

$$\mathbf{A} = \frac{1}{m\rho^2}\begin{pmatrix} \rho^2 & 0 & 0 \\ 0 & \rho^2 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Taking the gradient of this ellipsoid, we find:

$$\nabla F = (\frac{\partial F}{\partial f_x}, \frac{\partial F}{\partial f_y}, \frac{\partial F}{\partial \tau})^T = (\frac{f_x}{m}, \frac{f_y}{m}, \frac{\tau}{m\rho^2})^T = (a_x, a_y, \alpha)^T.$$

The gradient of the ellipsoid at the applied force is equal to the resultant acceleration.

If torques are measured about a reference point other than the object's center of mass, the force space undergoes a shear. For example, assume forces and torques are measured in a frame $\mathcal{F}'_{\mathcal{O}}$, which is aligned with $\mathcal{F}_{\mathcal{O}}$ with an origin at (x, y) in $\mathcal{F}_{\mathcal{O}}$. A force $\mathbf{f}$ in $\mathcal{F}_{\mathcal{O}}$ is related to the force $\mathbf{f}'$ in $\mathcal{F}'_{\mathcal{O}}$ by $\mathbf{f} = \mathbf{S}\mathbf{f}'$, where the shear matrix $\mathbf{S}$ is given by

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -y & x & 1 \end{pmatrix}.$$

Substituting for $\mathbf{f}$, the ellipsoid is given by $\frac{1}{2}\mathbf{f}'^T\mathbf{S}^T\mathbf{A}\mathbf{S}\mathbf{f}' = c^2$. The matrix $\mathbf{A}$ is positive definite, and by the law of inertia (Strang [199]), the matrix $\mathbf{A}' = \mathbf{S}^T\mathbf{A}\mathbf{S}$ is also positive definite. Thus we have a sheared ellipsoid in the new force space given by $F'(\mathbf{f}') = \frac{1}{2}\mathbf{f}'^T\mathbf{A}'\mathbf{f}' = c^2$, where

$$\mathbf{A}' = \frac{1}{m\rho^2}\begin{pmatrix} \rho^2 + y^2 & -xy & -y \\ -xy & \rho^2 + x^2 & x \\ -y & x & 1 \end{pmatrix}.$$

The gradient $\nabla F'$ with respect to $\mathbf{f}'$ gives the acceleration of the slider in the reference frame $\mathcal{F}'_{\mathcal{O}}$.

To determine the set of possible acceleration directions for a point or line contact, we intersect the composite friction cone $\mathcal{F}$ with the ellipsoid to find an acceleration cone $\mathcal{A}$. (The ellipsoid can be normalized to $c = 1$.) This is directly analogous to intersecting a composite friction cone with a limit surface (Figure 5.4), except the exact acceleration cone can be found in closed form. It is simply the convex hull of accelerations arising from extremal forces in the composite friction cone $\mathcal{F}$.

For simplicity, we choose the unit of distance to be the object's radius of gyration ρ and the unit of mass to be the object's mass m. The ellipsoid then becomes a unit sphere, and the object's acceleration is equal to the force in the frame $\mathcal{F}_{\mathcal{O}}$ ($\mathbf{f} = \mathbf{a}$). Therefore, if the unit force cone $\hat{\mathcal{F}}$ is contained in a hemisphere of the force sphere, the unit acceleration cone $\hat{\mathcal{A}}$ is contained in a hemisphere of the acceleration sphere; if $\hat{\mathcal{F}}$ positively spans a great circle, $\hat{\mathcal{A}}$ positively spans a great circle; and if $\hat{\mathcal{F}}$ positively spans the sphere, then $\hat{\mathcal{A}}$ also positively spans the sphere. These properties are preserved for reference frames $\mathcal{F}'_{\mathcal{O}}$ and other choices of the unit of distance.

Although this thesis deals with planar problems, the ellipsoid is readily generalized to a six-dimensional ellipsoid for objects in space. (A simple distance reparameterization will not generally turn this ellipsoid into a sphere, but the spanning properties hold.)

In the rest of this chapter, we will use nondimensional units obtained by multiplying or dividing, as necessary, by a reference mass (the mass m of the object), a reference length (the radius of gyration ρ), and a reference time, chosen so gravity is unit. In the absence of gravity, the reference time is arbitrary.

9.2 No Manipulator Constraints

We begin by examining the case of no constraints on the motion of the manipulator $\mathcal{M}$, which makes point or line contact with the object $\mathcal{O}$. The results of this section are analogous to those of Chapter 5 for a pushed object, except the support friction forces are now assumed to be negligible. As a result, the state of the object $\mathcal{O}$ includes velocities and the system contains a drift term (the object continues to move when the manipulator is not in contact).

To visualize the control system, imagine an object floating on an air table which may be tilted to yield a gravitational acceleration in the plane of the table. The object is pushed or batted by a manipulator. Alternatively, the object can be considered to be a planar free-flying rigid body with gas jets attached to its perimeter. The angle the gas jets can take with respect to the normal of the perimeter is determined by the friction coefficient μ.

The control system is written

$$(\dot{\mathbf{q}}, \ddot{\mathbf{q}}) = X_0(\mathbf{q}, \dot{\mathbf{q}}) + uX_i(\mathbf{q}, \dot{\mathbf{q}}), \quad i \in \{1, \ldots, n\},$$

where X_0 is a drift vector field, $u \in [0, \infty)$ is a scalar control, and X_i is the control vector field, where i chooses which control vector field is used. (Note that only one control force is applied at a time.) $\mathcal{X}$ is the set of vector fields $X_j, j = 0, \ldots, n$. The vector field X_i, $i \in 1, \ldots, n$, corresponds to a force direction $\hat{\mathbf{f}}_i$ (and resulting acceleration direction $\hat{\mathbf{a}}_i$) fixed in the object frame $\mathcal{F}_{\mathcal{O}}$. The set of force directions $\cup_i \hat{\mathbf{f}}_i$ is denoted $\hat{\mathcal{F}}$, and the set of acceleration directions $\cup_i \hat{\mathbf{a}}_i$ is written $\hat{\mathcal{A}}$. Forces arise from point or line contact on the perimeter Γ of the object.

The state of $\mathcal{O}$ is written $(\mathbf{q}, \dot{\mathbf{q}}) = (x_w, y_w, \phi_w, \dot{x}_w, \dot{y}_w, \dot{\phi}_w)^T$, and the tangent vector is $(\dot{\mathbf{q}}, \ddot{\mathbf{q}}) = (\dot{x}_w, \dot{y}_w, \dot{\phi}_w, \ddot{x}_w, \ddot{y}_w, \ddot{\phi}_w)^T$. The drift field X_0 is written $(\dot{x}_w, \dot{y}_w, \dot{\phi}_w, 0, 0, 0)^T + \mathbf{g}$, where $\mathbf{g}$ is the gravitational acceleration vector $(0,0,0,0,g,0)^T$. g may be either 1 or 0.

9.2.1 Point Contact

We first study the case of a single control, $n = 1$. To determine accessibility, we can examine the Lie algebra generated by the vector fields X_0 and X_1. For simplicity, assume the control force applied to the object is parallel to the y-axis of the object, so the control vector field X_1 is written $(0, 0, 0, -\sin\phi_w, \cos\phi_w, \alpha)^T$. (Note that a pure angular acceleration is not possible by a force through the perimeter of the object.) Define the following vector fields in $Br(\mathcal{X})$:

$$\begin{aligned}
X_2 &= [X_0, X_1] = (\sin\phi_w, -\cos\phi_w, -\alpha, -\dot{\phi}_w\cos\phi_w, -\dot{\phi}_w\sin\phi_w, 0)^T \\
X_3 &= [X_1, [X_0, X_1]] = (0, 0, 0, -2\alpha\cos\phi_w, -2\alpha\sin\phi_w, 0)^T \\
X_4 &= [X_1, [X_0, [X_0, X_1]]] = (2\alpha\cos\phi_w, 2\alpha\sin\phi_w, 0, 2\alpha\dot{\phi}_w\sin\phi_w, -2\alpha\dot{\phi}_w\cos\phi_w, 0)^T \\
X_5 &= [X_1, [X_1, [X_0, [X_0, X_1]]]] = (0, 0, 0, 2\alpha^2\sin\phi_w, -2\alpha^2\cos\phi_w, 0)^T \\
X_6 &= [X_0, [X_1, [X_1, [X_0, [X_0, X_1]]]]] \\
&= (-2\alpha^2\sin\phi_w, 2\alpha^2\cos\phi_w, 0, 2\alpha^2\dot{\phi}_w\cos\phi_w, 2\alpha^2\dot{\phi}_w\sin\phi_w, 0)^T.
\end{aligned}$$

We find that

$$\det(X_1\ X_2\ X_3\ X_4\ X_5\ X_6) = -16\alpha^8,$$

indicating that these six vector fields span the tangent space $T_{(\mathbf{q},\dot{\mathbf{q}})}T\mathcal{C}$ at any state $(\mathbf{q}, \dot{\mathbf{q}})$, provided $\alpha \neq 0$—the control force must not pass through the center of mass. Unlike the quasistatic case, accessibility is obtained with a single control due to the drift field X_0.

The tangent vectors $X_1(\mathbf{q}, \dot{\mathbf{q}})$, $X_3(\mathbf{q}, \dot{\mathbf{q}})$, and $X_5(\mathbf{q}, \dot{\mathbf{q}})$ span the acceleration space at $(\mathbf{q}, \dot{\mathbf{q}})$, and $X_2(\mathbf{q}, \dot{\mathbf{q}})$, $X_4(\mathbf{q}, \dot{\mathbf{q}})$, and $X_6(\mathbf{q}, \dot{\mathbf{q}})$ span the velocity space.

Proposition 9.1 *The state of the planar object $\mathcal{O}$ is small-time accessible if and only if $\hat{\mathcal{F}}$ contains a force direction which does not pass through the center of mass of $\mathcal{O}$.*

Clearly $n = 1$ is never sufficient for controllability; the angular velocity of the object can only change in one direction. It can be shown that $n = 2$ is sufficient for controllability, provided the signs of $\hat{\alpha}_1$ and $\hat{\alpha}_2$ are opposite.

Proposition 9.2 *The state of the planar object $\mathcal{O}$ is controllable if and only if $\hat{\mathcal{F}}$ contains force directions $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$ such that $\hat{\tau}_1 > 0$ and $\hat{\tau}_2 < 0$.*

Sketch of Proof: It is sufficient to show that the object $\mathcal{O}$ is controllable to the zero state with zero gravity.

If the object initially has nonzero velocity, it is clear that its velocity can be brought to zero, because $\hat{\tau}_1$ and $\hat{\tau}_2$ have opposite sign. After the initial velocity is canceled, the object's state is $(\mathbf{q}_1, \mathbf{0})$. By sequencing the force directions $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$, $\mathcal{O}$ can be brought to a new resting state $(\mathbf{q}_2, \mathbf{0})$, where $\mathcal{O}$ has rotated an angle π. (The new state $(\mathbf{q}_2, \mathbf{0})$ should be in the interior of the reachable set from $(\mathbf{q}_1, \mathbf{0})$, which is possible because the system is accessible by Proposition 9.1.) Now clearly $(\mathbf{q}_1, \mathbf{0})$ is reachable from $(\mathbf{q}_2, \mathbf{0})$ by the same control sequence that takes $(\mathbf{q}_1, \mathbf{0})$ to $(\mathbf{q}_2, \mathbf{0})$. Furthermore, $(\mathbf{q}_1, \mathbf{0})$ is in the interior of the set reachable from $(\mathbf{q}_2, \mathbf{0})$. Thus $\mathcal{O}$ can reach a neighborhood of resting configurations of $(\mathbf{q}_1, \mathbf{0})$ by first traveling to $(\mathbf{q}_2, \mathbf{0})$. The object is controllable to the zero state by patching together these neighborhoods.

Krishnaprasad [111] proved a similar result for the case $\hat{\mathbf{f}}_1 = -\hat{\mathbf{f}}_2$. □

The force directions $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$ may arise from two different point contacts, or from a single point contact (not at the center of mass) with nonzero friction.

We now consider conditions for small-time local controllability about a state $(\mathbf{q}, \mathbf{0})$. The object $\mathcal{O}$ is small-time locally controllable about $(\mathbf{q}, \mathbf{0})$ if, given any neighborhood U of $(\mathbf{q}, \mathbf{0})$, the set of trajectories remaining in U contains $(\mathbf{q}, \mathbf{0})$ in its interior.

Proposition 9.3 *In the presence of gravity, the planar object $\mathcal{O}$ is small-time locally controllable at all states $(\mathbf{q}, \mathbf{0})$ if and only if the set of force directions $\hat{\mathcal{F}}$ positively spans the force sphere. A minimum of four controls ($n \geq 4$) is necessary.*

Proof: If $\hat{\mathcal{F}}$ positively spans the force sphere, $\hat{\mathcal{A}}$ positively spans the acceleration sphere. In the presence of gravity, the set of accelerations $\tilde{\mathcal{A}}(u) = \cup_{i=1,\ldots,n}\ \tilde{\mathbf{a}}_i(u)/|\tilde{\mathbf{a}}_i(u)|$, where $\tilde{\mathbf{a}}_i(u) = u\hat{\mathbf{a}}_i + \mathbf{g}$ for u sufficiently large, also positively span the acceleration sphere. Because $\mathbf{0}$ lies in the interior of $CH_{\mathbf{R}^3}(\hat{\mathcal{A}})$ and $CH_{\mathbf{R}^3}(\tilde{\mathcal{A}}(u))$, the convex hull in $\mathbf{R}^3$, the object $\mathcal{O}$ is small-time locally controllable by Theorem 4.1.

To see that the condition is necessary, we assume that the accelerations $\hat{\mathcal{A}}$ positively span a closed hemisphere of the acceleration sphere, and we show that there are configurations from which the object is not small-time locally controllable. The gravity vector $\mathbf{g}$, expressed in the object's frame $\mathcal{F}_{\mathcal{O}}$, traces out a circle on the $(\ddot{x}, \ddot{y})$-plane as the object's angle ϕ_w varies from 0 to 2π. This gravity vector "pulls" the accelerations in $\hat{\mathcal{A}}$ toward it. If the gravity vector lies in the interior of the closed hemisphere, $CH_{\mathbf{R}^3}(\tilde{\mathcal{A}}(u))$, for any u, does not include $\mathbf{0}$. By Theorem 4.1, the object is not small-time locally controllable. For any closed hemisphere of the acceleration sphere, there is a range of object angles ϕ_w of measure π such that the gravity vector is in the interior of the closed hemisphere. The exceptions are the closed hemispheres $\ddot{\phi} \geq 0$ and $\ddot{\phi} \leq 0$. In these cases, angular acceleration in only one direction is possible, and the object is not small-time locally controllable. □

The condition of Proposition 9.3 is the familiar condition for a "force-closure" grasp of a planar object; the difference is that for a grasp, all forces are simultaneously active. We can directly apply various theorems regarding the existence of positive grasps.

Theorem 9.1 *Any planar object $\mathcal{O}$ with a closed, piecewise smooth curve , of available contact points is small-time locally controllable at all states $(\mathbf{q}, \mathbf{0})$, with or without gravity, unless the contact is frictionless and , is a circle.*

Proof: See, e.g., (Mishra *et al.* [149]; Markenscoff *et al.* [137]). □

We can also apply results concerning the number of fingers (respectively controls) sufficient for force closure (respectively small-time local controllability), with and without friction at the contacts, but they are not listed here because their application is direct. It should be noted, however, that sufficient conditions for a force-closure grasp of a spatial object are also sufficient conditions for small-time local controllability of the object.

Weaker sufficient conditions on the set of control forces $\hat{\mathcal{F}}$ can be found for small-time local controllability for zero gravity and for subsets of the space $(\mathbf{q}, \mathbf{0})$ in the presence of gravity.

Proposition 9.4 *The planar object $\mathcal{O}$ is small-time locally controllable about a state $(\mathbf{q}, \mathbf{0})$ if the negated gravitational force direction, expressed in the object's frame $\mathcal{F}_{\mathcal{O}}$, is in the interior of $CH_{S^2}(\hat{\mathcal{F}})$, the convex hull of the force directions $\hat{\mathcal{F}}$ on the force sphere.*

Proof: Under the conditions of the proposition, the gravitational acceleration and the accelerations $\hat{\mathcal{A}}$ positively span the acceleration sphere. □

If the condition of Proposition 9.4 is satisfied, $n \geq 3$ and the object is small-time locally controllable on the simply connected three-dimensional subset of its configuration space $\{\mathbf{q} \in \mathcal{C} \mid \phi_{min} < \phi_w < \phi_{max}\}$, for a suitably defined world frame $\mathcal{F}_{\mathcal{W}}$.

Proposition 9.5 addresses small-time local controllability in zero gravity, and is relevant to controlling the position and attitude of a free-flying planar robot with gas jets.

Proposition 9.5 *In the absence of gravity, the object $\mathcal{O}$ is small-time locally controllable about any state $(\mathbf{q}, \mathbf{0})$ if the set of forces $\hat{\mathcal{F}}$ positively span a great circle of the force sphere that (1) does not lie in the $\tau = 0$ plane, and (2) does not lie in a plane orthogonal to the $\tau = 0$ plane.*

Remark: The condition of Proposition 9.5 is satisfied by any set of force lines that intersect at a single point, provided the force lines positively span the plane and the intersection point is not at the object's center of mass (see Figure 9.1).

Proof: Following Lewis and Murray [122], we consider the system

$$(\mathbf{q}, \dot{\mathbf{q}}) = X_0(\mathbf{q}, \dot{\mathbf{q}}) + u_1 X_1(\mathbf{q}, \dot{\mathbf{q}}) + u_2 X_2(\mathbf{q}, \dot{\mathbf{q}}),$$

where u_1 and u_2 are unbounded scalar controls, $X_1(\mathbf{q}, \dot{\mathbf{q}}) = (0, 0, 0, \cos\phi_w, \sin\phi_w, 0)^T$ and $X_2(\mathbf{q}, \dot{\mathbf{q}}) = (0, 0, 0, -\sin\phi_w, \cos\phi_w, \alpha)^T$. In other words, the force $\mathbf{f}_1$ acts through the center

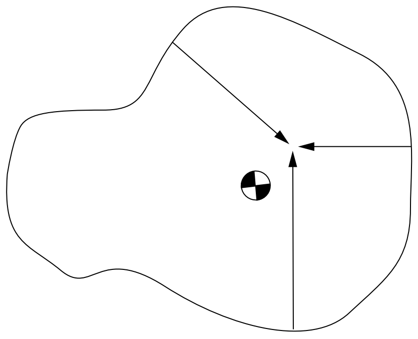


Figure 9.1: Without gravity, the object is small-time locally controllable at any state $(\mathbf{q}, \mathbf{0})$ by the three forces shown.

of mass in the x direction of the frame $\mathcal{F}_{\mathcal{O}}$, and the force $\mathbf{f}_2$ acts in the y direction with some torque about the center of mass. We construct the brackets

$$\begin{aligned} X_3 &= [X_0, X_1] = (-\cos\phi_w, -\sin\phi_w, 0, -\dot{\phi}_w \sin\phi_w, \dot{\phi}_w \cos\phi_w, 0)^T \\ X_4 &= [X_0, X_2] = (\sin\phi_w, -\cos\phi_w, -\alpha, -\dot{\phi}_w \cos\phi_w, -\dot{\phi}_w \sin\phi_w, 0)^T \\ X_5 &= [X_1, [X_0, X_2]] = (0, 0, 0, -\alpha\sin\phi_w, \alpha\cos\phi_w, 0)^T \\ X_6 &= [X_0, [X_1, [X_0, X_2]]] = (\alpha\sin\phi_w, -\alpha\cos\phi_w, 0, -\alpha\dot{\phi}_w\cos\phi_w, -\alpha\dot{\phi}_w\sin\phi_w, 0)^T. \end{aligned}$$

At any state $(\mathbf{q}, \mathbf{0})$, the tangent vectors of X_3, X_4, and X_6 span the velocity space, and the tangent vectors of X_1, X_2, and X_5 span the acceleration space, provided $\alpha \neq 0$. The system satisfies the Lie Algebra Rank Condition.

Now we give some definitions necessary to apply Sussmann's sufficient condition for small-time local controllability [205]. For an element $B \in Br(\mathcal{X})$, we define $\delta_i(B)$ as the number of times X_i appears in B, and the degree of B is $\sum_{i=0}^{n} \delta_i(B)$. B is called a "bad" bracket if $\delta_0(B)$ is odd and $\delta_i(B)$ is even for all $i \in \{1, \ldots, n\}$, and B is a "good" bracket otherwise. A "bad" bracket B is "neutralized" at a state $\mathbf{p}$ if B, evaluated at $\mathbf{p}$, is the linear combination of "good" brackets of lower degree evaluated at $\mathbf{p}$. Sussmann [205] proved that if a system satisfies the Lie Algebra Rank Condition at $\mathbf{p}$ and all "bad" brackets evaluated at $\mathbf{p}$ are neutralized, then the system is small-time locally controllable at $\mathbf{p}$.

For our case, the Lie Algebra Rank Condition is satisfied by brackets up to degree four, so the only "bad" brackets to be neutralized are the drift field (which vanishes at $\dot{\mathbf{q}} = \mathbf{0}$) and the "bad" brackets of degree three,

$$[X_1, [X_0, X_1]] = (0, 0, 0, 0, 0, 0)^T$$

$$[X_2, [X_0, X_2]] = (0, 0, 0, -2\alpha\cos\phi_w, -2\alpha\sin\phi_w, 0)^T,$$

which are clearly neutralized. Therefore, the system is small-time locally controllable.

Considering X_1, $-X_1$, X_2, and $-X_2$ as four separate control vector fields with nonnegative controls, the corresponding force directions $\hat{\mathbf{f}}_1$, $-\hat{\mathbf{f}}_1$, $\hat{\mathbf{f}}_2$, and $-\hat{\mathbf{f}}_2$ positively span a great circle of the force sphere that does not lie in the $\tau = 0$ plane or a plane orthogonal to the $\tau = 0$ plane. Applying Sussmann's general theorem, we see that any set of forces which positively spans the same great circle is sufficient for small-time local controllability. $\square$

Proposition 9.5 allows us to strengthen Theorem 9.1 for the case of zero gravity.

Theorem 9.2 *In the absence of gravity, any object $\mathcal{O}$ with a closed, piecewise smooth curve , of available contact points is small-time locally controllable at all states* $(\mathbf{q}, \mathbf{0})$*, unless the contact is frictionless and , is a circle centered at the object's center of mass.*

9.2.2 Line Contact

When the manipulator and object are in line contact, the convex cone of contact force directions $\hat{\mathcal{F}}$ maps to a convex cone of acceleration directions $\hat{\mathcal{A}}$. The object is in a dynamic grasp (remains fixed to the manipulator) if the manipulator acceleration, summed with the negated gravitational acceleration, yields an acceleration direction in $\hat{\mathcal{A}}$.[1]

This section briefly lists some simple results on the controllability of an object by dynamic grasps with line contact. When friction is zero, $\hat{\mathcal{F}}$ is the convex combination of two force directions, and when friction is nonzero, it is the convex combination of four force directions. We can consider the extremal force directions in $\hat{\mathcal{F}}$ to be the controls of the system.

Theorem 9.3 *For a polygonal object $\mathcal{O}$ and a sufficiently long straight-edge manipulator $\mathcal{M}$, any edge of the convex hull of $\mathcal{O}$ can be used as the line contact. If the center of mass of $\mathcal{O}$ does not lie on a vertex of the convex hull:*

- *There is at least one line contact from which $\mathcal{O}$ is controllable.*

If we further assume nonzero friction and nonzero gravity:

- *There is at least one line contact from which $\mathcal{O}$ is small-time locally controllable on a simply connected three-dimensional subset of the configuration space $\mathcal{C}$.*

Proof: There is at least one edge such that the normal of the edge at an interior point of the edge passes through the center of mass of $\mathcal{O}$. Therefore, $\hat{\mathcal{F}}$ contains force directions with torques of opposite signs and the object $\mathcal{O}$ is controllable by Proposition 9.2. If there is friction at this edge contact, a pure force is interior to $\hat{\mathcal{F}}$, and there is a simply connected three-dimensional subset of $\mathcal{C}$, $\{\mathbf{q} \in \mathcal{C} \mid \phi_{min} < \phi_w < \phi_{max}\}$, for a suitably defined world frame $\mathcal{F}_{\mathcal{W}}$, that satisfies the condition of Proposition 9.4 for small-time local controllability in the presence of gravity. $\square$

If the manipulator can switch between dynamic grasps on the object, we have the following.

[1]The object may move relative to the manipulator if there is an ambiguity due to Coulomb friction. I ignore this possibility.

Proposition 9.6 *Given a set of composite friction cones from line contacts between the manipulator $\mathcal{M}$ and the object $\mathcal{O}$. Define $\hat{\mathcal{F}}_f$ to be the set of pure force directions (zero torque) such that each $\hat{\mathbf{f}} \in \hat{\mathcal{F}}_f$ is interior to at least one of the composite friction cones. Then if $\hat{\mathcal{F}}_f$ positively spans the force plane, $\mathcal{O}$ is small-time locally controllable at all states $(\mathbf{q}, \mathbf{0})$ by dynamic grasps at these line contacts, with or without gravity.*

Proof: Because each of the spanning pure forces has a neighborhood in one of the force cones, the set of force directions positively spans the force sphere. $\square$

9.3 With Manipulator Constraints

The results of the previous section address the theoretical capabilities of dynamic nonprehensile manipulation from the point of view of the object alone. Just as important, however, are properties of the manipulator which is controlling the object. While an object may be controllable by point contact, the manipulator may not be able to achieve the contacts and motions necessary to bring the object to the desired state. Also, unlike the case of quasistatic pushing, the object continues to move as the manipulator changes contacts.

Ideally, we would like to understand the global reachability properties of a manipulator/object system—given the kinematic and dynamic specifications of a manipulator $\mathcal{M}$, the mass, radius of gyration, and shape of an object $\mathcal{O}$, the friction between them, and their initial state, where can the manipulator take the object? Unfortunately such a question appears to be very difficult to answer, so we are forced to look locally. Because the contact is nonprehensile, in any neighborhood of a manipulator/object state, the best we can hope for is small-time accessibility in the absence of gravity (with gravity, we may have small-time local controllability at some configurations of the system).

Proposition 9.1 says that a single control force, fixed in the object frame, is sufficient for small-time accessibility. This indicates that a one-degree-of-freedom manipulator may be sufficient for small-time accessibility. This would be an interesting result, as it implies that even the simplest of robots is capable of performing interesting dynamic manipulation in the plane—the object's reachable state space is a full-dimensional subset of its six-dimensional state space.

We examine this possibility for a single prismatic joint and a single revolute joint.

9.3.1 Example 1: Single Prismatic Joint

Consider the system of Figure 9.2. The manipulator $\mathcal{M}$ is a single prismatic joint, and the object $\mathcal{O}$ is a thin rod in point contact with $\mathcal{M}$ at an angle $\phi_w \in (0, \pi)$. The mass of the rod is concentrated at the endpoints, making the distance from the rod's center of mass to the contact equal to the unit radius of gyration.

The configuration of the system is $\mathbf{q}' = (\mathbf{q}, y_m) \in \mathcal{C}'$, where y_m is the position of $\mathcal{M}$ in the world frame $\mathcal{F}_{\mathcal{W}}$. The state space of the system is the eight-dimensional manifold

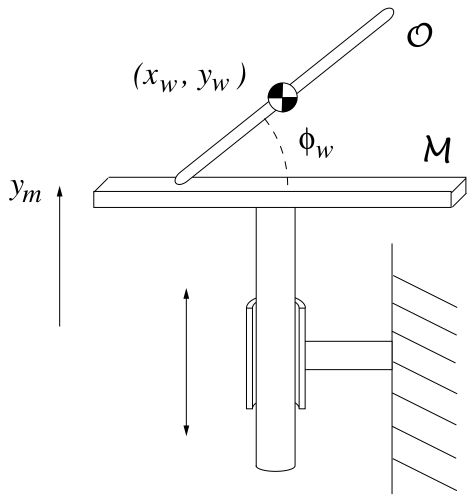


Figure 9.2: A one-degree-of-freedom prismatic robot manipulating a rod.

$TC' = \mathbf{R}^3 \times S^1 \times \mathbf{R}^4$, and the set of contact states of the system is the seven-dimensional submanifold $\{(\mathbf{q}', \dot{\mathbf{q}}') \in TC' \mid F(\mathbf{q}') = y_w - y_m - \sin\phi_w = 0\}$. We assume that the manipulator stays in contact with the rod endpoint at all times, and it may apply zero force (simply "following" the rod) or apply a nonzero force.

The conditions that $\mathcal{O}$ remain in contact with $\mathcal{M}$ are given by

$$\begin{aligned} \frac{dF(\mathbf{q}'(t))}{dt} &= 0 \\ \frac{d^2F(\mathbf{q}'(t))}{dt^2} &= 0. \end{aligned}$$

Erdmann [68, 71] refers to these constraints as the *First and Second Variation Constraints*, respectively. These constraints state that the velocity and acceleration of the system normal to the constraint surface must be zero. For the system of Figure 9.2, these constraints are written

$$\begin{aligned} \frac{dF(\mathbf{q}'(t))}{dt} &= \dot{y}_w - \dot{y}_m - \dot{\phi}_w \cos\phi_w = 0 \\ \frac{d^2F(\mathbf{q}'(t))}{dt^2} &= \ddot{y}_w - \ddot{y}_m + \dot{\phi}_w^2 \sin\phi_w - \ddot{\phi}_w \cos\phi_w = 0. \end{aligned} \quad (9.1)$$

(The state variables' dependence on time is omitted for clarity.) If these equations are satisfied at the initial state for some manipulator control $\ddot{y}_m$, they remain satisfied for any $\ddot{y}_m$ that does not "pull away" from the rod.

Frictionless or Slipping Contact. First consider the case of slipping contact. That is, in a neighborhood of the initial system state, only slipping contact is possible for any control $\ddot{y}_m$. For simplicity, assume the contact is frictionless. Then the acceleration (force) of $\mathcal{O}$ must satisfy the constraints

$$\begin{aligned} \ddot{x}_w &= 0 && (9.2) \\ \ddot{y}_w \cos\phi_w - \ddot{\phi}_w &= 0. && (9.3) \end{aligned}$$

Equation (9.2) constrains the direction of the force, and Equation (9.3) constrains the force to pass through the contact point. Using Equations (9.1)–(9.3), we solve for the acceleration (force) of $\mathcal{O}$ as a function of the manipulator control $\ddot{y}_m$ and the system state $(\mathbf{q}', \dot{\mathbf{q}}')$:

$$\begin{aligned} \ddot{x}_w &= 0 \\ \ddot{y}_w &= K \\ \ddot{\phi}_w &= -K\cos\phi_w, \end{aligned}$$

where

$$K = \frac{\ddot{y}_m - \dot{\phi}_w^2 \sin\phi_w}{1 + \cos^2\phi_w}.$$

Now we can write the drift vector field $X_0(\mathbf{q}, \dot{\mathbf{q}})$ and the control vector field $X_1(\ddot{y}_m, \mathbf{q}, \dot{\mathbf{q}})$ (which are the projections of the vector fields on the system state space $T\mathcal{C}'$ to vector fields on the object's state space $T\mathcal{C}$) as follows:

$$\begin{aligned} X_0(\mathbf{q}, \dot{\mathbf{q}}) &= (\dot{x}_w, \dot{y}_w, \dot{\phi}_w, 0, 0, 0)^T \\ X_1(\ddot{y}_m, \mathbf{q}, \dot{\mathbf{q}}) &= (0, 0, 0, \ddot{x}_w, \ddot{y}_w, \ddot{\phi}_w)^T, \end{aligned}$$

where $\ddot{x}_w$, $\ddot{y}_w$, and $\ddot{\phi}_w$ are solved for above. Note that the control vector field is not linear in the control $\ddot{y}_m$. The drift field is written without a gravity term, but one can be included in any direction without changing the results here.

It is clear that $\mathcal{O}$ is not small-time accessible, nor even accessible, as Equation (9.2) integrates to yield the velocity constraint $\dot{x}_w = c_1$ and the position constraint $x_w = c_1 t + c_2$. The situation is identical with slipping contact and nonzero friction. In that case, the force is constrained to act along a friction cone edge instead of the contact normal.

Although $\mathcal{O}$ is not accessible on its full state space, it is small-time accessible on its reduced state space $(y_w, \phi_w, \dot{y}_w, \dot{\phi}_w)$.

Sticking Contact. With sticking contact, the force direction constraint (Equation (9.2)) is replaced by a kinematic constraint that the contact point does not change when the manipulator applies a force. This is expressed by another second variation constraint:

$$\begin{aligned} G(\mathbf{q}'(t)) &= x_w - \cos\phi_w = 0 \\ \frac{dG(\mathbf{q}'(t))}{dt} &= \dot{x}_w + \dot{\phi}_w \sin\phi_w = 0 \\ \frac{d^2G(\mathbf{q}'(t))}{dt^2} &= \ddot{x}_w + \dot{\phi}_w^2 \cos\phi_w + \ddot{\phi}_w \sin\phi_w = 0. && (9.4) \end{aligned}$$

The constraint that the force passes through the contact point is

$$\ddot{x}_w \sin\phi_w + \ddot{y}_w \cos\phi_w + \ddot{\phi}_w = 0. \tag{9.5}$$

Solving Equations (9.1), (9.4), and (9.5) for the acceleration of $\mathcal{O}$, we get

$$\begin{aligned}
\ddot{x}_w &= -\dot{\phi}_w^2 \sec\phi_w + (\ddot{y}_m \tan\phi_w)/2 \\
\ddot{y}_w &= \ddot{y}_m/2 \\
\ddot{\phi}_w &= \dot{\phi}_w^2 \tan\phi_w - (\ddot{y}_m \sec\phi_w)/2
\end{aligned}$$

Again we can write the drift and control vector fields:

$$\begin{aligned}
X_0(\mathbf{q},\dot{\mathbf{q}}) &= (\dot{x}_w, \dot{y}_w, \dot{\phi}_w, 0, 0, 0)^T \\
X_1(\ddot{y}_m, \mathbf{q}, \dot{\mathbf{q}}) &= (0, 0, 0, \ddot{x}_w, \ddot{y}_w, \ddot{\phi}_w)^T.
\end{aligned}$$

As in the proof of Proposition 9.1, we construct the vector fields

$$\begin{aligned}
X_2 &= [X_0, X_1] \\
X_3 &= [X_1, [X_0, X_1]] \\
X_4 &= [X_1, [X_0, [X_0, X_1]]] \\
X_5 &= [X_1, [X_1, [X_0, [X_0, X_1]]]] \\
X_6 &= [X_0, [X_1, [X_1, [X_0, [X_0, X_1]]]]].
\end{aligned}$$

(Due to their complexity, the bracket terms are not written out.) For the particular case of $\dot{\phi}_w = 0$ (initial angular velocity of the rod is zero), we get

$$\det(X_1\ X_2\ X_3\ X_4\ X_5\ X_6) = \frac{-\ddot{y}_m^{12} \sec^{12}\phi_w (\tan\phi_w - \sec\phi_w)^2 (\sec\phi_w + \tan\phi_w)^2}{256}.$$

This determinant is undefined at $\phi_w = \pi/2$ and $\phi_w = 3\pi/2$, but it is clear that $\mathcal{O}$ is not accessible at these configurations. All forces pass through the center of mass, so the rod cannot be rotated. At all other configurations, provided $\ddot{y}_m$ can be nonzero, these vector fields span, and $\mathcal{O}$ is small-time accessible. X_1, X_3, and X_5 span the acceleration space at $(\mathbf{q}, \dot{\mathbf{q}})$, and X_2, X_4, and X_6 span the velocity space.

It is the ability to apply forces in different directions at a sticking contact (depending on the object state) that makes the difference in accessibility between slipping and sticking contact.

Figure 9.3 gives two examples of prismatic manipulator/object systems, one of which is small-time accessible and one of which is not.

(Note that if the contact point between $\mathcal{O}$ and $\mathcal{M}$ remains sticking during the entire motion, the state of $\mathcal{O}$ cannot be accessible. The contact point slides when no force is applied to the object. Therefore there may be some initial slip when the manipulator again begins to apply a force, or an impact to instantly switch from sliding to sticking contact.)

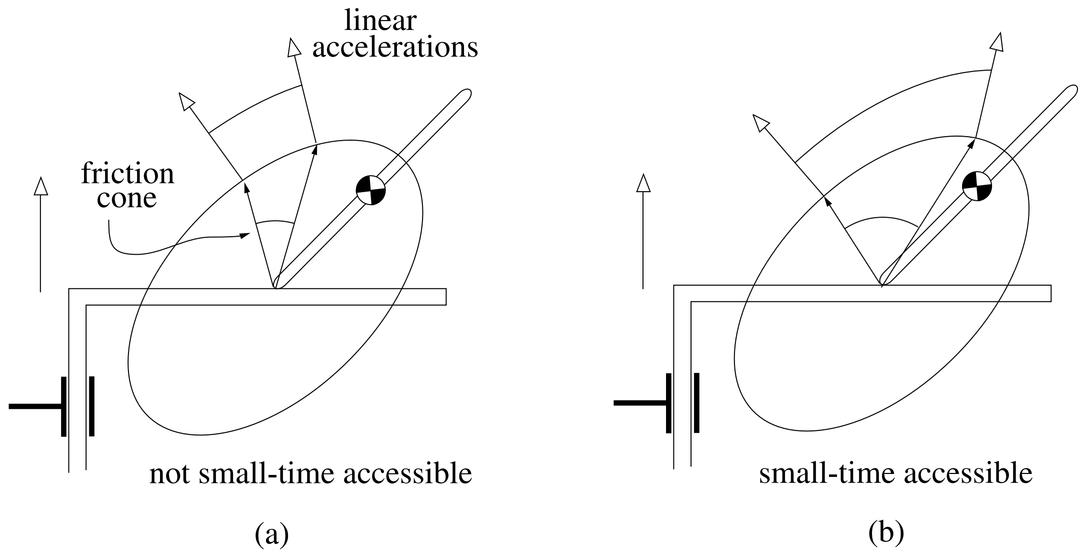


Figure 9.3: The systems (a) and (b) are identical, except for the friction coefficient at the contact. The rod is at an angle $\pi/4$ to the prismatic manipulator $\mathcal{M}$, gravity is zero, and the system is initially at rest. In each figure is drawn the ellipse mapping the friction cone at the contact point to an acceleration cone of acceleration directions of the contact point on the rod. (a) The acceleration cone does not include the direction of the manipulator acceleration. The contact must slip to the left when the manipulator accelerates into the object, resulting in a force along the right edge of the friction cone. The rod's state is not small-time accessible. (b) For this larger friction cone, the acceleration cone includes the direction of the manipulator acceleration. The contact is sticking when the manipulator accelerates into the rod. The rod's state is small-time accessible.

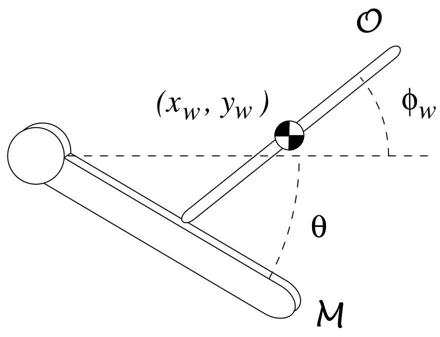


Figure 9.4: A one-degree-of-freedom revolute robot manipulating a rod.

9.3.2 Example 2: Single Revolute Joint

Now consider the system of Figure 9.4. The manipulator $\mathcal{M}$ is a single revolute joint, and the object $\mathcal{O}$ is a rod as before. A world frame $\mathcal{F}_\mathcal{W}$ is fixed to the axis of the revolute joint. The configuration of the system is $\mathbf{q}' = (\mathbf{q}, \theta) \in \mathcal{C}'$, where θ is the angle of the revolute joint. Assuming the single link is thin, the seven-dimensional contact submanifold of the eight-dimensional state space is given by $\{(\mathbf{q}, \dot{\mathbf{q}}) \in T\mathcal{C}' \mid F(\mathbf{q}') = \cos\theta(y_w - \sin\phi_w) + \sin\theta(\cos\phi_w + x_w) = 0\}$. After collecting acceleration, centrifugal, and Coriolis terms, the second variation constraint is written

$$\begin{aligned}\frac{d^2F(\mathbf{q}'(t))}{dt^2} = \; & \ddot{x}_w(-\sin\theta) + \\ & \ddot{y}_w(\cos\theta) + \\ & \ddot{\phi}_w(-\cos(\phi_w - \theta)) + \\ & \ddot{\theta}(\cos(\phi_w - \theta) - x_w\cos\theta - y_w\sin\theta) + \\ & \dot{\phi}_w^2(\sin(\phi_w - \theta)) + \\ & \dot{\theta}^2(\sin(\phi_w - \theta) + x_w\sin\theta - y_w\cos\theta) + \\ & \dot{\theta}\dot{x}_w(-2\cos\theta) + \\ & \dot{\theta}\dot{y}_w(-2\sin\theta). \end{aligned} \tag{9.6}$$

Assuming the contact is frictionless, we have two constraints on the line of force:

$$\ddot{x}_w\cos\theta + \ddot{y}_w\sin\theta = 0 \tag{9.7}$$
$$\ddot{y}_w\cos(\phi_w - \theta) + \ddot{\phi}_w\cos\theta = 0. \tag{9.8}$$

Solving Equations (9.6)–(9.8) for the acceleration of the object in terms of the control $\ddot{\theta}$ and the system state $(\mathbf{q}', \dot{\mathbf{q}}')$, we get

$$\ddot{x}_w = -K\sin\theta \tag{9.9}$$
$$\ddot{y}_w = K\cos\theta \tag{9.10}$$
$$\ddot{\phi}_w = -K\cos(\phi_w - \theta), \tag{9.11}$$

where

$$\begin{aligned} K \ = 2 \ \ & (\cos(\phi_w - \theta)(-\ddot{\theta}) + \\ & \sin(\phi_w - \theta)(-\dot{\phi}_w^2 + 2\dot{\phi}_w\dot{\theta} - \dot{\theta}^2) + \\ & \cos\theta(\ddot{\theta}x_w + 2\dot{\theta}\dot{x}_w + \dot{\theta}^2 y_w) + \\ & \sin\theta(\ddot{\theta}y_w + 2\dot{\theta}\dot{y}_w - \dot{\theta}^2 x_w)) \\ & /(3 + \cos(2(\phi_w - \theta))). \end{aligned}$$

Equations (9.9)–(9.11) have the structure we expect—the force is normal to the manipulator link and the torque about the center of mass of $\mathcal{O}$ depends on the relative angle between the object and the manipulator. The equations also show that even for this simplest of systems, the force is a complex function of the state and control.

Using the object acceleration calculated above, we can again write the drift and control vector fields:

$$\begin{aligned} X_0(\mathbf{q}', \dot{\mathbf{q}}') &= (\dot{x}_w, \dot{y}_w, \dot{\phi}_w, \dot{\theta}, 0, 0, 0, 0)^T \\ X_1(\ddot{\theta}, \mathbf{q}', \dot{\mathbf{q}}') &= (0, 0, 0, 0, \ddot{x}_w, \ddot{y}_w, \ddot{\phi}_w, \ddot{\theta})^T. \end{aligned}$$

Because the acceleration of the object is a function of the manipulator state $(\theta, \dot{\theta})$, not just the control $\ddot{\theta}$, we cannot immediately project the vector fields to vector fields on the object's state space $T\mathcal{C}$ as we did with the prismatic joint. We must look at the vector fields X_0 and X_1, and their Lie brackets, on the full eight-dimensional manifold $T\mathcal{C}'$. After we have constructed the Lie brackets, we can look at their projection to $T\mathcal{C}$ to determine the accessibility of $\mathcal{O}$.

As before, we construct the vector fields X_2, X_3, X_4, X_5, and X_6. These vector fields are complex trigonometric functions; in fact, X_6 consists of thousands of terms and took several minutes to compute using Mathematica running on a Sun SPARC 20. Instead of attempting to determine conditions under which these vector fields satisfy the Lie Algebra Rank Condition, I chose a representative state of the system and numerically evaluated the tangent vectors there.

Consider the system state $(\mathbf{q}', \dot{\mathbf{q}}') = ((1+\sqrt{2})/\sqrt{2}, 1/\sqrt{2}, \pi/4, 0, 0, 0, 0, 0)^T$—the system is at rest with a manipulator joint angle $\theta = 0$, and the rod is at a 45 degree angle and contacting the manipulator at the point $(1, 0)$. Choosing a control $\ddot{\theta} = 1$, the tangent vectors at $(\mathbf{q}', \dot{\mathbf{q}}')$, projected to the object's tangent space $T_{(\mathbf{q},\dot{\mathbf{q}})}T\mathcal{C}$, are

$$\begin{array}{rclcccccccl} X_1(\ddot{\theta}, \mathbf{q}, \dot{\mathbf{q}}) & = & (& 0, & 0, & 0, & 0, & 0.667, & -0.471 &)^T \\ X_2(\ddot{\theta}, \mathbf{q}, \dot{\mathbf{q}}) & = & (& 0, & -0.667, & 0.471, & 0, & 0, & 0 &)^T \\ X_3(\ddot{\theta}, \mathbf{q}, \dot{\mathbf{q}}) & = & (& 0, & 0, & 0, & -1.333, & -1.752, & -0.148 &)^T \\ X_4(\ddot{\theta}, \mathbf{q}, \dot{\mathbf{q}}) & = & (& 1.333, & 1.752, & 0.148, & 0, & 0, & 0 &)^T \\ X_5(\ddot{\theta}, \mathbf{q}, \dot{\mathbf{q}}) & = & (& 0, & 0, & 0, & 3.505, & 1.598, & 3.475 &)^T \\ X_6(\ddot{\theta}, \mathbf{q}, \dot{\mathbf{q}}) & = & (& -3.505, & -1.598, & -3.475, & 0, & 0, & 0 &)^T. \end{array}$$

These tangent vectors span $T_{(\mathbf{q},\dot{\mathbf{q}})}T\mathcal{C}$, so the object's state is small-time accessible. As before, note that X_1, X_3, and X_5 span the acceleration space at $(\mathbf{q}, \dot{\mathbf{q}})$, and X_2, X_4, and X_6 span the velocity space.

Unlike the case of a frictionless prismatic joint, the object's state space can be accessible with a frictionless revolute joint because the force angle varies with the joint angle θ.

9.3.3 Discussion

The examples above show that even a one-degree-of-freedom robot can take an object to a six-dimensional subset of its state space. This is similar to the 1JOC system—in both cases, accessibility is obtained by bracketing a single control with a drift field. We should therefore be able to do interesting planar dynamic manipulation even with the simplest of robots. Further, it appears that a revolute joint is preferable to a prismatic joint, because small-time accessibility holds even with slipping contact. Also, a prismatic robot initially in line contact with an object may be unable to rotate the object to any other type of contact configuration. On the other hand, a revolute joint can instantly bring the object to point contact by rotating the object.

The examples above consider edge-vertex contacts only, but curved objects or manipulators can easily be handled in the contact constraint (and therefore the second variation constraint).

The system is *dynamically singular* at a system state $(\mathbf{q}', \dot{\mathbf{q}}')$ if the set of motion directions of $\mathcal{O}$ loses rank on $T_{(\mathbf{q},\dot{\mathbf{q}})}T\mathcal{C}$. (The object may still be accessible if the system can break the dynamic singularity at some time T.) Figure 9.5 gives examples of dynamically singular systems.

In short, the system is not small-time accessible if the applied force direction (1) is fixed in the world frame (as with a frictionless prismatic joint) or (2) always passes through the object's center of mass (as with a frictionless disk, or if the center of mass of the object is coincident with the contact point). If neither of these conditions hold, I conjecture that the object is small-time accessible by a single-revolute-joint robot.

9.4 Algorithmic Control

The examples of accessibility above say nothing about the shape of the object's accessible state space. Calculating the shape of the accessible state space appears to be very hard in general, depending on factors such as the shape and mass properties of the object, the kinematic structure of the robot, control limits, gravity, and friction. The proofs also provide little insight into how controls can be generated to reach a goal state. This section describes an intuitive approach to controlling the degrees-of-freedom of the object using *algorithmic* control, which is the basis of the control algorithm described in Chapter 10.

The idea behind algorithmic control is to only control a subset of the state variables

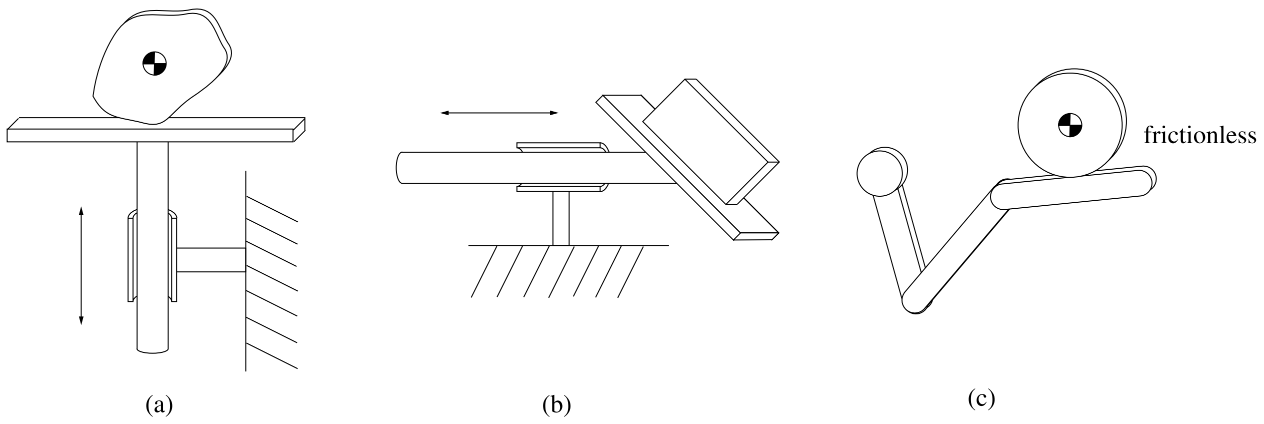


Figure 9.5: Dynamically singular nonprehensile systems. (a) The system is initially at rest. At this configuration, any acceleration of the prismatic joint either results in zero force applied to the object, or a force through the center of mass. This dynamic singularity is unstable. (b) If the object is initially at rest in line contact with the manipulator, no acceleration of the manipulator can cause the object to rotate. This singularity persists over time. Even if the robot has two or more prismatic joints, the system may be singular with line contact. (c) If the disk is frictionless and centered at its center of mass, every state of the system is dynamically singular. The initial angular velocity of the object cannot be changed.

at any given time, but to switch between the controlled subset such that the goal state is reached. For example, a unicycle can be driven to the zero state by first reorienting it so that it points toward the origin, then driving it to the origin, and finally reorienting it to the goal angle. In general, we must account for the possibility of drift in the variables that are not directly controlled.

In the context of dynamic nonprehensile manipulation, we can define the following control phases.

1. *Dynamic grasp.* Because the object remains fixed to the manipulator, up to $\min(2n, 6)$ of the object's state variables can be directly controlled by an n-joint manipulator.

2. *Slip.* Controlled slip provides control of two state variables, the slipping distance and the slipping velocity.

3. *Roll.* Rolling provides control of two state variables, the rolling angle and angular velocity.

4. *Slip and roll.* Slip and roll occur simultaneously, giving control of up to four state variables.

5. *Free flight.* After the object is released, it follows a one-dimensional path through its state space, parameterized by its time of flight t.

A control algorithm can sequence these phases. The dimension of the accessible state space is the sum of the independent freedoms of each phase, up to a maximum of six.

Example. An n-degree-of-freedom manipulator carries an object with a dynamic grasp, then releases it. The dimension of the accessible state space of the object is upper-bounded by $\min(2n+1, 6)$. The "controls" are the release state of the object, plus the time of flight (assuming the arrival time at the goal state is a "don't care").

If $n = 1$ and the joint is revolute, the dimension of the object's accessible state space is three using this control strategy. While the dimension of the accessible state space is independent of gravity, the shape of the accessible state space is not. For example, with a gravity field, the three goal configuration variables $\mathbf{q}_g$ of the goal state $(\mathbf{q}_g, \dot{\mathbf{q}}_g)$ can be chosen independently. If there is no gravity, they cannot. This is because gravity provides a coupling between the time of flight of the object and its velocity in the direction of the gravity field.

Example. A one-degree-of-freedom revolute manipulator carries the object by a dynamic grasp to a state at which the object begins to slip. The object stops slipping after a distance, and begins to roll about a vertex. The manipulator then releases the object to free flight. The dimension of the accessible state space of the object is $2+1+2+1=6$. Notice that, in this case, slip contributed only one dimension to the object's accessible state space, because the slip velocity is required to come to zero before the roll begins.

The dynamic grasp, slip and roll, and free-flight phases are described briefly below.

Dynamic Grasp A dynamic grasp allows the object to move with the manipulator, much as an object in a prehensile grasp would. Static grasps are an important subclass of dynamic grasps, defined as the subset of dynamic grasps where $\dot{\mathbf{q}} = \ddot{\mathbf{q}} = \mathbf{0}$. We can further define a *robust* static grasp to be a static grasp that is robust to infinitesimal disturbance forces. For example, a box resting on a horizontal frictionless surface is in a static grasp, but it is not robust. A robust static grasp with line contact is only possible with gravity and nonzero friction, and the negated gravitational force must be in the interior of the composite friction cone $\mathcal{F}$.

For a given line contact between the object and manipulator, if there is a manipulator configuration Θ that results in a robust static grasp, then Θ is an element of a connected open subset of robust static grasp configurations. Because the object's state is small-time locally controllable (Theorem 9.3) at such configurations, it is possible to move from any robust static grasp to any other robust static grasp (with the same line contact) by "slow" motion. On the other hand, the ability to move from $(\mathbf{q}_1, \dot{\mathbf{q}}_1)$ to $(\mathbf{q}_2, \dot{\mathbf{q}}_2)$ with a dynamic grasp does not generally guarantee the ability to move from $(\mathbf{q}_2, \dot{\mathbf{q}}_2)$ to $(\mathbf{q}_1, \dot{\mathbf{q}}_1)$.

Slip and Roll We simply mention that it is often possible to control two state variables during slipping and rolling contact. For example, the slip distance and velocity of a block

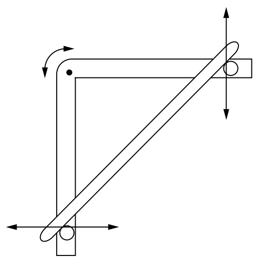


Figure 9.6: Regardless of the manipulator's acceleration, it cannot instantaneously break contact with the rod at both points.

resting on a table can be independently controlled by the tilt angle of the table and the time of the tilt. Oriolo and Nakamura [159] showed that the angle and angular velocity of an unactuated revolute joint can be controlled by dynamic coupling through the joint (except for very special cases, such as when the center of mass of the link is at the joint). The same reasoning can be directly applied to rolling contact.

Release and Free Flight To release a contact, the system must satisfy an inequality constraint of the form $(\mathbf{a}_o - \mathbf{a}_m)^T\hat{\mathbf{n}} > 0$, where $\hat{\mathbf{n}}$ is the contact normal pointing into the object; $\mathbf{a}_m$ is the acceleration of the contact point on the manipulator, a function of the state and acceleration of the manipulator; and $\mathbf{a}_o$ is the acceleration of the contact point on the object assuming free flight, a function of the state of the object. Releasing a line contact is equivalent to releasing the contact points at each end of the line segment, and therefore places two inequality constraints on the manipulator acceleration. Considering the three-dimensional acceleration space of the last link of the manipulator, the boundaries of the half-spaces defined by the constraints are two-dimensional planes, and their intersection is a line. This line moves continuously as a function of the state of the manipulator and the object.

If we assume the manipulator jerk is finite, then the acceleration is continuous. To release a line contact, the continuous manipulator acceleration must pass through the line formed by the intersection of the boundaries of the two half-space constraints. Generically, the manipulator requires two degrees-of-freedom (the set of attainable manipulator accelerations forms a plane in the three-dimensional acceleration space, which generically will intersect a line). If jerk is not required to be finite, even a one-degree-of-freedom manipulator can release a line contact.

Even with no constraints on manipulator jerk or acceleration, however, it may be impossible to release a point or line contact if the set of attainable manipulator accelerations lies outside the open half-spaces defined by the constraints (see Figure 9.6).

Chapter 10

Planning for Dynamic Manipulation

In the previous chapter we developed an understanding of the theoretical capabilities of dynamic nonprehensile manipulation. This chapter addresses the planning problem for dynamic nonprehensile manipulation, which can be stated as follows:

> Given an object, a manipulator, and their initial states, design the manipulator trajectory to bring the object to the goal state.

This statement of the problem permits repeated contacts, which are beyond the scope of this thesis. Instead I focus on the restricted problem:

> *Given:*
>
> 1. *A polygonal object $\mathcal{O}$ and its center of mass and radius of gyration ρ,*
> 2. *a manipulator $\mathcal{M}$ (including kinematic and dynamic specifications), and*
> 3. *$\mathcal{O}$ and $\mathcal{M}$ initially in point or line frictional contact,*
>
> *find a manipulator trajectory that will transfer the object to the goal state using a single sustained contact.*

We also need to choose between manipulator trajectories that accomplish the goal. Because nonprehensile manipulation relies so critically on friction at the contact, we prefer the trajectory that minimizes the required friction at the contact. This trajectory is maximally robust to variations in friction.

10.1 A Simple Example

Let's begin our investigation with a simple problem: designing the trajectory of a one-degree-of-freedom arm to carry an object from one state to another with a dynamic grasp. Because the object remains fixed to the manipulator, the manipulator/object state space is two-dimensional and may be represented by $(\theta, \dot{\theta}) \in S^1 \times \mathbf{R}^1$, where θ is the angle of the

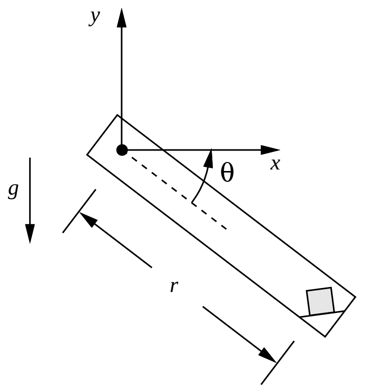


Figure 10.1: A simple one-degree-of-freedom manipulator contacting a square. The contact edge is at a 45 degree angle to the length of the manipulator.

manipulator mass	3.0 kg
manipulator inertia (about motor axis)	1.6 kg m^2
maximum motor torque	40 Nm
minimum motor torque	-40 Nm
square dimensions	10 cm x 10 cm
square mass	0.1 kg
square inertia (about its center of mass)	1.68 x 10^{-4} kg m^2
distance r from axis to square CM	1.0 m

Table 10.1: Parameters relevant to the manipulator/square system.

manipulator. A picture of the system is given in Figure 10.1, and parameters relevant to the system are listed in Table 10.1.

The frictional contact between the manipulator and object defines a three-dimensional convex force cone $\mathcal{F}$, represented in the object's frame. This force cone maps to a three-dimensional acceleration cone $\mathcal{A}$ of possible accelerations of the object. To maintain the dynamic grasp, the manipulator must choose an angular acceleration $\ddot{\theta}$, as a function of the current state $(\theta, \dot{\theta})$, such that the total acceleration of the manipulator in the object frame (including negated gravity) is contained in $\mathcal{A}$. This dynamic grasp constraint can be represented in the system's state space as a set of vector field cones: from any point in the state space, the vector field cone describes the motions of the system that maintain a dynamic grasp. Any valid dynamic grasp trajectory through the state space will be an integral curve of tangent vectors inside the vector field cones.

Figure 10.2 shows the dynamic grasp vector field cones for the manipulator/object system of Figure 10.1 with a friction coefficient $\mu = 0.175$. There is an "island" (Shin and McKay [191]) in the center of the figure; no feasible manipulator acceleration will keep

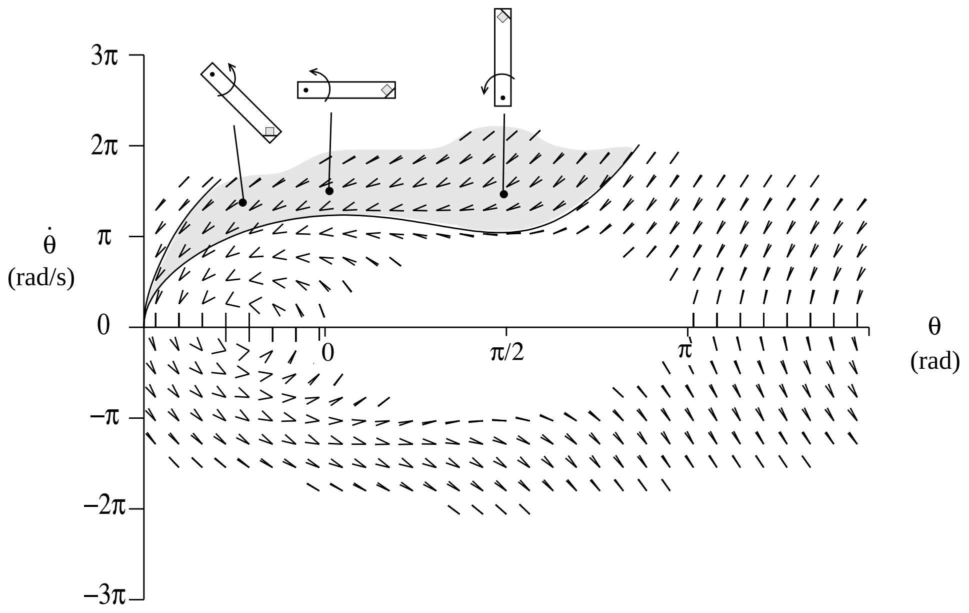


Figure 10.2: Tangent vector cones that maintain the dynamic grasp for $\mu = 0.175$ and specified motor torque limits.

the dynamic grasp at these states. This is a state space obstacle in dynamic grasp trajectory planning. Also shown in the figure is the state space reachable from $(-\pi/2, 0)$, found by integrating the bounding tangent vectors of the vector field cones.

Notice that no equilibrium point is reachable from $(-\pi/2, 0)$ for the friction coefficient $\mu = 0.175$. If we increase μ, however, the vector field cones get larger, until eventually a stable equilibrium point can be reached. At such points, the object is in a static grasp. For this system, a stable equilibrium point can be reached if $\mu \geq 0.265$. Interestingly, the robot must reverse its path ($\dot{\theta}$ becomes negative) to reach a stable equilibrium for $0.265 \leq \mu < 0.35$. (This is a fundamental property of dynamic nonprehensile manipulation: an optimal trajectory may have path reversals.) For $\mu \geq 0.35$, a stable equilibrium can be reached without path reversal (θ increases monotonically).

To plan a dynamic grasp carry to a goal state (a static grasp or a release point for a throw), we must find a curve connecting the two states subject to the dynamic grasp constraints. If we disallow path reversals, we can employ the ACOT algorithm of Shin and McKay [191] (or a similar approach) to find a trajectory between the states. (In their time-optimal path following problem, "islands" can arise only if there is friction at the joints.) The ACOT algorithm finds the fastest path through the vector field cones or returns failure if none exists. We are more concerned with minimizing the friction coefficient, so we can increase μ from zero until ACOT can find a solution. Binary search on μ could speed the computation. Note that when $\mu = 0$, each cone collapses to a single tangent vector, and the reachable state space from any state is at most a one-dimensional curve.

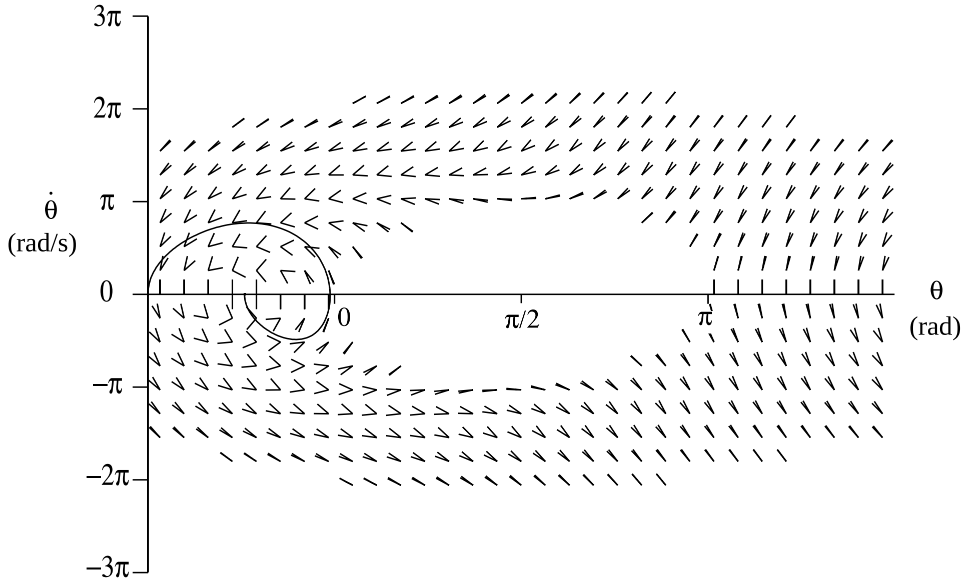


Figure 10.3: The tangent vector cone field for $\mu = 0.265$ and a manipulator motion that brings the square to a static grasp. This is the manipulator motion that requires the minimum value of friction to bring the object to a static grasp. Notice that there is a path reversal.

This phase plane idea can be extended to apply to manipulators with any number of degrees of freedom, provided the path of the object (or equivalently, the path of the manipulator) is given in advance. The path is parameterized by s, and the phase plane is parameterized by $(s, \dot{s})$. As before, the dynamic grasp constraints define a tangent vector cone at each state, and a path-following trajectory moves monotonically in s. Again we can search on μ until the ACOT algorithm finds the minimum friction solution.

Unfortunately, it is easy to choose a path between two configurations such that no time-scaling will produce a valid dynamic grasp trajectory, even when one exists. To avoid this problem, we would like to be able to find the path and trajectory simultaneously. In addition, our dynamic manipulation planner should allow relative motion between the object and the manipulator. Clearly we need a more general approach.

10.2 The General Case

This section describes a general approach to dynamic nonprehensile manipulation planning. We are especially interested in the following dynamic tasks:

1. *Snatch:* transfer an object initially at rest on a table to a statically stable grasp on the manipulator. The manipulator accelerates into the object, transferring control of the object from the table to the manipulator.

2. *Roll:* roll a polygonal object sitting on the manipulator from one statically stable edge to another statically stable edge.

3. *Throw:* throw the object to a desired goal state. The object is carried with a dynamically stable grasp and released instantaneously (no slipping or rolling) at a point where the free-flight dynamics will take the object to the goal state. This goal state may be a new position for the object on the arm.

4. *Rolling throw:* allow the object to begin rolling before throwing it. By controlling the roll angle and velocity before the release, the dimension of the object's accessible state space is increased by two.

The two types of contact which occur in these tasks are sticking at all contacts (a dynamic grasp) and rolling. This thesis does not consider slipping contacts.

To solve these problems I cast trajectory planning as a constrained nonlinear optimization problem, where the system's initial state and goal state (or state manifold) are specified as constraints to the optimization. The trajectory is also subject to a set of linear and nonlinear equality and inequality constraints arising from constraints on the manipulator motion and the dynamics governing the object's motion relative to the manipulator. Because dynamic nonprehensile manipulation relies on friction between the object and the manipulator, and friction coefficients are often uncertain and varying, the optimization is usually asked to minimize the required friction coefficient for successful manipulation. Unlike other work on optimizing the time or energy of a robot's motion, we are more concerned with making the manipulation maximally robust to variations in the friction coefficient.

In formulating the nonlinear program, we have several choices concerning the design variables and constraints. Three possibilities are:

1. *Design the motion of the object.* Here the design variables specify the object's motion, subject to the constraints that the manipulator can follow the object and apply forces as necessary during the object's motion.

2. *Design the motion of the manipulator.* The design variables specify the motion of the manipulator, subject to the constraint that the object reaches the desired goal state. The object's motion relative to the manipulator (if any) is simulated. The equivalent of this approach was used by Fernandes *et al.* [76, 77].

3. *Simultaneously design the motions of the object and the manipulator.* The design variables specify the motion of both the object and the manipulator, subject to the constraints that dynamic equations of motion are satisfied. Witkin and Kass [219] used a similar approach by simultaneously solving for the positions of and forces acting on an articulated body.

The space of all trajectories of the manipulator and object is, in theory, infinite-dimensional. In practice, we can only solve for finite-dimensional approximations to the optimal trajectories. An important issue is how to choose the finite parameterizations used for the trajectories of the manipulator and the object. In particular, they must have "enough" degrees-of-freedom to permit a solution to the problem when one exists. At the very least, we should have as many degrees-of-freedom in the trajectories as there are independent equality constraints in the problem formulation. In the case of choice (3) above, we must take care to choose trajectory types for the object and manipulator so that they can be dynamically consistent with each other.

The problem formulations in this thesis adhere to choice (2) above. For low-degree-of-freedom manipulators, this choice generally yields the most compact problem formulation in terms of the number of design variables and the number of constraints. This choice requires the motion of the object to be simulated. It is well known that there can be multiple solutions to the motion of rigid bodies in frictional contact. To handle this problem, we can avoid manipulator motions that result in ambiguities of the object's motion, or adopt friction models with compliance. Here we will simply assume that if the "expected" contact mode is dynamically and kinematically consistent, then it is the actual contact mode.

The manipulator is treated as an acceleration source, so the trajectory found by the nonlinear program is specified as an acceleration profile, not a joint torque history. Of course these two representations are related through the dynamics of the manipulator/object system, but it is more natural to abstract away the dynamic properties of the manipulator and treat it as an accelerating kinematic constraint. To execute an acceleration profile on a real robot, it must be translated through the system dynamics to a torque profile. We will briefly discuss this problem in Chapter 11.

10.3 Problem Specification

To see how a dynamic nonprehensile manipulation planning problem is formulated as a nonlinear program, we first need some notation. Every task is assumed to consist of a sequence of manipulation phases made up of one or more of the following: a (dynamic) grasp phase g, a roll phase r, and a flight phase f. The nonlinear program is written to handle any sequence of manipulation phases S from the simple finite state machine (Rayward-Smith [172]):

$$\begin{aligned} S &\rightarrow ABC \\ A &\rightarrow g|\varepsilon \\ B &\rightarrow r|rg|\varepsilon \\ C &\rightarrow f|\varepsilon \end{aligned}$$

where ε is the empty string. With this notation, a throw (as defined above) is denoted gf, a roll is denoted grg, and a rolling throw is denoted grf. A snatch can be either g or rg. We assume that there is no rebound from the impact at the end of a roll (the transition from r

to g). The instant the new edge contacts, if the dynamic grasp conditions are met, then the object is assumed to be in a dynamic grasp.

The times of the manipulation phases are t_{g1} for the first dynamic grasp phase g, t_{roll} for the rolling phase r, t_{g2} for the second g phase, and t_{flight} for the flight phase f. If a phase is omitted, its corresponding time duration is zero. We also define the "running" times:

$$\begin{aligned} T_{g1} &= t_{g1} \\ T_{roll} &= T_{g1} + t_{roll} \\ T_{g2} &= T_{roll} + t_{g2} \\ T_{flight} &= T_{g2} + t_{flight} \end{aligned}$$

Finally, define $T = T_{g2}$, where T is the total time the manipulator is in contact with the object during the manipulation.

These primitive manipulations—the snatch, the roll, the throw, and the rolling throw—can be composed or "glued together" to form more complex manipulations. The glue between phases is the static grasp. A primitive that ends with a static grasp can be connected to another primitive that begins with a static grasp by moving the manipulator slowly to the start position for the next manipulation. If there are no joint limits or obstacles in the space, there is a trivial dynamic grasp trajectory between the two static grasps.

In the next three subsections we describe the three fundamental elements of the nonlinear program: the design variables, the constraints, and the objective function.

10.3.1 Design Variables

The design variables consist of the variables $\mathbf{x}$ specifying the trajectory of the manipulator over the interval $[0, T]$; the times of each phase of the manipulation, t_{g1}, t_{roll}, t_{g2}, and t_{flight} (some subset of these will be applicable based on the problem specification); and the friction coefficient μ between the manipulator and the object. These variables are not independent; the trajectory of the manipulator implicitly defines the time of each phase. However, the problem formulation is much simpler if we make each of these variables explicit and constrain them to be dynamically consistent. Also, we usually cannot control the friction coefficient μ; it is determined by the materials comprising the object and the manipulator. However, in most of the optimizations the objective is to minimize the required friction coefficient to make the plan optimally robust to friction variations. It is convenient to represent μ as a design variable and explicitly enforce the resulting friction constraints.

Many different finite parameterizations of manipulator trajectories have been explored, including polynomials, Fourier bases, summed Fourier and polynomial functions, splines, and piecewise constant acceleration segments. In this thesis, trajectories are represented as uniform cubic B-splines (Bartels [22]).

For an n-joint robot, $\mathbf{x} = (\mathbf{x}^1\ \mathbf{x}^2\ \ldots\ \mathbf{x}^n)$, where $\mathbf{x}^i$ is the vector of knot points for the cubic B-spline position history of joint i. The time of each knot point x^i_j is given by t_j,

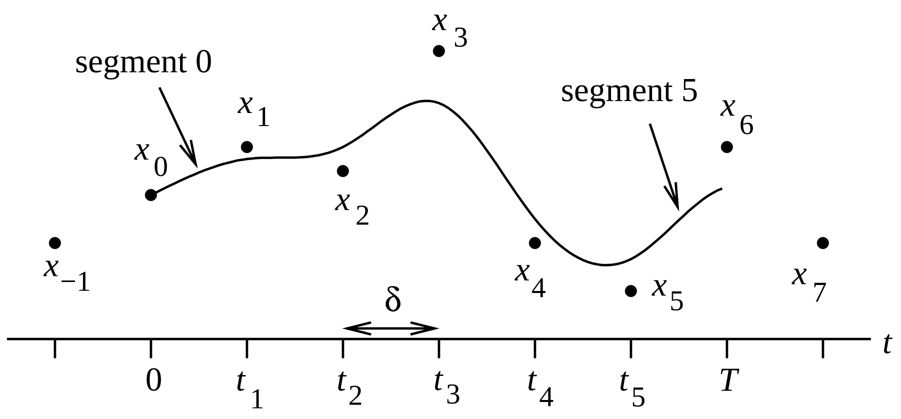


Figure 10.4: An example cubic B-spline with nine knot points and six segments.

and the knot points are evenly spaced in time. The position of the joint passes "near" the knot points; the actual position at each time is obtained by taking a weighted sum of the four knot points which are closest in time (two previous in time and two later in time). The weighting basis functions are cubic polynomials of time. Therefore, the position is C^2 and piecewise cubic, the velocity is C^1 and piecewise quadratic, and the acceleration is C^0 and piecewise linear (constant jerk segments).

More formally, consider a trajectory of time T consisting of m segments of time $\delta = T/m$. (This is the knot point spacing in time.) The m segments are numbered $j = 0, \ldots, m-1$, and the four closest knot points at time t during segment $j = \lfloor t/\delta \rfloor$ are denoted x^i_{j-1}, x^i_j, x^i_{j+1}, and x^i_{j+2}. We define the function $\lambda_j(t)$, which evaluates to 0 when the time t is at the beginning of segment j and 1 when t is at the end of segment j:

$$\begin{aligned} \lambda_j(t) &= (t - t_j)/\delta \\ t_j &= \delta j. \end{aligned}$$

Now for each time segment $j = 0, \ldots, m-1$, the position is given by

$$\theta^i_j(t) = x_{j-1}\, b_{-1}(\lambda_j(t)) + x_j\, b_0(\lambda_j(t)) + x_{j+1}\, b_1(\lambda_j(t)) + x_{j+2}\, b_2(\lambda_j(t)), \tag{10.1}$$

where

$$\begin{aligned} b_{-1}(\gamma) &= (1 - 3\gamma + 3\gamma^2 - \gamma^3)/6 && (10.2)\\ b_0(\gamma) &= (4 - 6\gamma^2 + 3\gamma^3)/6 && (10.3)\\ b_1(\gamma) &= (1 + 3\gamma + 3\gamma^2 - 3\gamma^3)/6 && (10.4)\\ b_2(\gamma) &= \gamma^3/6. && (10.5) \end{aligned}$$

See Figure 10.4.

The velocity and acceleration of the joint are obtained by simply differentiating Equation 10.1:

$$\frac{d^k\theta^i_j(t)}{dt^k} = x_{j-1}\, b^{(k)}_{-1}(\lambda_j(t)) + x_j\, b^{(k)}_0(\lambda_j(t)) + x_{j+1}\, b^{(k)}_1(\lambda_j(t)) + x_{j+2}\, b^{(k)}_2(\lambda_j(t)).$$

The functions $b^{(k)}_h(\gamma)$, $h = -1, \ldots, 2$, are obtained by differentiating Equations 10.2–10.5 k times.

The vector $\mathbf{x}^i$ representing the trajectory of joint i is an $(m+3)$-vector $(x^i_{-1}\ x^i_0\ \ldots\ x^i_{m+1})$. By choosing m sufficiently high, cubic B-splines can approximate any trajectory arbitrarily closely.

10.3.2 Constraints

Constraints in the optimization arise from limitations on the motion of the manipulator and constraints on the motion of the object. These latter are determined by the inequality constraints of Coulomb friction and the equality constraints of Newton's laws. To simplify the notation, the dependencies of each of these constraints on the design variables is omitted.

Manipulator constraints

1. **Position constraints**

$$\mathbf{p}(\Theta(t), \mathbf{q}) \leq \mathbf{0},\ t \in [0, T],$$

where $\mathbf{p}$ is a vector-valued function representing joint limits of the manipulator and obstacles in the world.

2. **Joint velocity constraints**

$$\dot{\Theta}_{min} \leq \dot{\Theta}(t) \leq \dot{\Theta}_{max},\ t \in [0, T]$$

These constraints encode motor velocity limits.

3. **Joint torque constraints**

$$\boldsymbol{\tau}_{min} \leq \boldsymbol{\tau}(t) = \mathbf{M}(\Theta(t))\,\ddot{\Theta}(t) + \mathbf{C}(\Theta(t), \dot{\Theta}(t)) + \mathbf{G}(\Theta(t)) + \boldsymbol{\tau}_{obj}(t) \leq \boldsymbol{\tau}_{max},\ t \in [0, T]$$

where $\mathbf{M}(\Theta)$ is the $n \times n$ mass matrix, $\mathbf{C}(\Theta, \dot{\Theta})$ is an n-vector representing the centrifugal and Coriolis effects, $\mathbf{G}(\Theta)$ is the n-vector of gravitational torques, and $\boldsymbol{\tau}_{obj}$ is the n-vector of joint torques required to accelerate the object along its trajectory. In general, the minimum and maximum torques are also a function of the joint velocities, and there is a friction torque at each joint.

4. **Torque derivative constraints**

$$\dot{\boldsymbol{\tau}}_{min} \leq \frac{d\boldsymbol{\tau}(t)}{dt} \leq \dot{\boldsymbol{\tau}}_{max},\ t \in [0, T]$$

5. **Initial state constraints**

$$\begin{aligned}\Theta(0) &= \Theta_0\\ \dot{\Theta}(0) &= \dot{\Theta}_0\end{aligned}$$

Object constraints During a dynamic grasp or rolling phase, contact friction constraints must be enforced to prevent slipping or breaking contact. These force constraints encode both the unilateral nature of contact (forces can only be applied into the object) and the finite friction coefficient μ.

To maintain a dynamic grasp, the sum of the negated gravitational vector $-\mathbf{g}$ and the manipulator's acceleration, measured in the object frame, must fall inside the cone $\mathcal{A}$ of accelerations due to forces at the contact. This cone is found by mapping the composite friction cone through the mass parameters of the object, as described earlier. When the contact is a line contact, the cone is bounded by four edges, $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4$, numbered so that the interior of the cone lies to the left as we move from $\mathbf{a}_1$ to $\mathbf{a}_2$, etc. The magnitude of these vectors is not important, but for simplicity assume they are unit vectors. We now form the 3×4 matrix

$$\mathbf{A}_g = (\mathbf{a}_2 \times \mathbf{a}_1 \mid \mathbf{a}_3 \times \mathbf{a}_2 \mid \mathbf{a}_4 \times \mathbf{a}_3 \mid \mathbf{a}_1 \times \mathbf{a}_4),$$

where each column of $\mathbf{A}_g$ is an outward-pointing normal to a face of the acceleration cone $\mathcal{A}$. The acceleration of the manipulator must have a nonpositive dot product with each column vector of $\mathbf{A}_g$.

1. **Dynamic grasp constraints** (dynamic grasp phase only)

$$\mathbf{A}_{gi}^T\,(\dot{\mathbf{J}}_{gi}(\Theta(t))\,\dot{\Theta}(t) + \mathbf{J}_{gi}(\Theta(t))\,\ddot{\Theta}(t) - \mathbf{g}) \leq \mathbf{0},\ t \in [0, T_{g1}] \vee t \in [T_{roll}, T_{g2}],$$

where i is 1 or 2, depending on the current dynamic grasp phase. During a dynamic grasp phase, $\mathbf{J}_{gi}$ and $\dot{\mathbf{J}}_{gi}$, the manipulator Jacobian and its time derivative, are measured at the center of mass of the object, so that all accelerations can be represented in the object frame.

During rolling contact, a single point of the object is in contact with the manipulator, so the cone $\mathcal{A}$ of feasible object accelerations is a planar cone in the acceleration space, bounded by the two edges $\mathbf{a}_l$ and $\mathbf{a}_r$ (the "left" and "right" cone edges, respectively). Define the vector $\mathbf{a}_\perp$ normal to the plane of this cone: $\mathbf{a}_\perp = \mathbf{a}_r \times \mathbf{a}_l$. We form the matrix $\mathbf{A}_r = (\mathbf{a}_\perp \times \mathbf{a}_l \mid \mathbf{a}_r \times \mathbf{a}_\perp)$, where each column of $\mathbf{A}_r$ lies in the $(\mathbf{a}_l, \mathbf{a}_r)$ plane and is an outward-pointing normal to an edge of the acceleration cone. The acceleration of the object must have a nonpositive dot product with each column vector of $\mathbf{A}_r$.

The object acceleration $\mathbf{a}_{roll} = (a_{roll,x}, a_{roll,y}, \alpha_{roll})^T$ required to maintain the rolling contact is determined by assuming a pin joint at the contact and finding the object acceleration consistent with the motion of the manipulator. The acceleration $\mathbf{a} = (a_x, a_y, \alpha)^T$ of the contact point on the robot (including negated gravity) is given by

$$(a_x, a_y, \alpha)^T = \dot{\mathbf{J}}_r(\Theta(t))\,\dot{\Theta}(t) + \mathbf{J}_r(\Theta(t))\,\ddot{\Theta}(t) - \mathbf{g},$$

where $\mathbf{J}_r$ and $\dot{\mathbf{J}}_r$ are measured at the contact point on the manipulator. The constraint that the linear acceleration (a_x, a_y) matches the acceleration of the contact point on the object is expressed

$$(a_x, a_y) = -\omega^2\mathbf{r} + (-r_y\alpha, r_x\alpha) + (a_{roll,x}, a_{roll,y}),$$

where ω is the angular velocity of the object, $\mathbf{r} = (r_x, r_y)$ is the vector from the CM of the object to the contact point in the world frame, and α is the angular acceleration of the object. This gives us two constraints on $\mathbf{a}_{roll}$; the third is given by

$$\alpha_{roll} = \frac{1}{\rho^2}(r_x a_{roll,y} - r_y a_{roll,x}).$$

After a little manipulation, we get

$$\begin{aligned}
a_{roll,x} &= \frac{a_x(\rho^2 + r_x^2) + a_y r_x r_y + r_x\omega^2(\rho^2 + r_x^2 + r_y^2)}{\rho^2 + r_x^2 + r_y^2} \\
a_{roll,y} &= \frac{a_y(\rho^2 + r_y^2) + a_x r_x r_y + r_y\omega^2(\rho^2 + r_x^2 + r_y^2)}{\rho^2 + r_x^2 + r_y^2} \\
\alpha_{roll} &= \frac{r_x a_{roll,y} - r_y a_{roll,x}}{\rho^2}.
\end{aligned}$$

Now we can write the rolling friction constraints.

2. **Rolling friction constraints** (rolling phase only)

$$\mathbf{A}_r^T\,\mathbf{a}_{roll} \leq \mathbf{0},\; t \in [T_g, T_{roll}]$$

Remark: If both the manipulator trajectory and the rolling trajectory are represented by the design variables, as in problem formulation (3), then $\mathbf{a}_{roll}$ is directly specified by the current iterate. In this case we have the extra rolling friction equality constraint $\mathbf{a}_\perp^T\mathbf{a}_{roll} = 0$, which ensures that the force applied to the object passes through the contact point (that is, in the $(\mathbf{a}_l, \mathbf{a}_r)$ plane).

3. **Roll angle constraints** (rolling phase only)

$$\psi_{min} \leq \psi(t) \leq \psi_{max},\; t \in [T_g, T_{roll}]$$

where ψ is the angle of the object relative to the manipulator. This constraint prevents the object from penetrating the manipulator during the rolling phase. The relative angle ψ at the beginning of the roll is chosen to be zero by convention.

4. **Roll completed constraint** (for rolls only)

$$\psi(T_{roll}) = \psi_{goal}$$

ψ_{goal} corresponds to a new edge in contact with the manipulator, at ψ_{min} or ψ_{max}.

5. **Release state constraints** (for throws only)

$$\mathbf{r}(\mathbf{q}, \dot{\mathbf{q}}, t_{flight}) = \mathbf{0}$$

These constraints specify that the object reaches the goal submanifold by free flight, where $(\mathbf{q}, \dot{\mathbf{q}})$ is the release state and t_{flight} is the time of flight. The goal submanifold is usually specified by goal values of some subset of the state variables. The number of independent constraints that can be placed on the goal state manifold is, in general, upper-bounded by $2n+1$ in the case of a throw, and $2n+3$ in the case of a rolling throw, where n is the number of joints of the manipulator.

6. **Clean release constraints**. If a throw is specified, these constraints should ensure that the object is released instantaneously and cleanly (no slip) once the object has reached its goal free-flight trajectory. As we have already seen, however, such a release may be impossible with a continuous acceleration profile. For this reason, these constraints are not included in the optimization. In the execution, a release is accomplished by hitting the arm with a "bang" (a large torque), releasing the object nearly instantaneously. (As we saw in Chapter 9, however, significant slipping may be unavoidable even with an infinitely large bang. This thesis does not consider such problems.)

In principle, the manipulator constraints (1)–(4) and object constraints (1)–(3) should be satisfied at all times during their domain of applicability. In practice, the optimization can only handle a finite number of constraints. For this reason, the constraints are only enforced at p uniformly-sampled points during each manipulation phase. During a rolling phase of time t_{roll}, for instance, there is time t_{roll}/p between constraint checks. p should be chosen sufficiently high to ensure that the trajectory approximately satisfies the constraints at all times. In the examples shown, p is chosen between 20 and 50.

10.3.3 Objective Function

Unless otherwise stated, the objective is to minimize the required friction coefficient μ between the object and manipulator.

10.4 Sequential Quadratic Programming

Sequential quadratic programming (SQP) is used to solve the nonlinear program. SQP is a generalization of Newton's method for unconstrained optimization in that it finds a step away from the current iterate by minimizing a quadratic model of the problem. At each iteration, SQP determines the direction to step by solving a quadratic subprogram, where the objective function is a quadratic approximation at the current point and nonlinear constraints are linearized.

The constrained nonlinear program can be written (Moré and Wright [151])

$$\min f(x)$$

$$\begin{aligned} \text{subject to} \quad c_i(x) &\le 0, \quad i \in \mathcal{I} \\ c_i(x) &= 0, \quad i \in \mathcal{E} \end{aligned}$$

where $x \in \mathbf{R}^n$ is the iterate, f is the objective function, each c_i is a constraint mapping $\mathbf{R}^n$ to $\mathbf{R}$, and $\mathcal{I}$ and $\mathcal{E}$ are index sets for inequality and equality constraints, respectively. The Lagrangian function is defined

$$L(x,\lambda) = f(x) + \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i c_i(x)$$

where the λ_i are the Lagrange multipliers. At each iterate, the direction of the step is computed by a quadratic programming subproblem of the form

$$\min \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d$$

$$\begin{aligned} \text{subject to} \quad c_i(x_k) + \nabla c_i(x_k)^T d &\le 0, \quad i \in \mathcal{I} \\ c_i(x_k) + \nabla c_i(x_k)^T d &= 0, \quad i \in \mathcal{E} \end{aligned}$$

where x_k is the current iterate, the constraints c_i are local linear approximations, and H_k is a positive definite estimate of the Hessian of the Lagrangian ($\nabla^2_{xx} L(x_k, \lambda_k)$). (The set $\mathcal{I}$ may be an "active" set of inequality constraints rather than the full set.) The solution d_k to this subproblem defines the direction of descent. The distance moved in the step direction is determined by a line search to minimize a merit function consisting of the objective function and a penalty function based on the violation of the constraints.

At points near the solution, SQP converges at a second-order rate if exact Hessian information is used. In other words, if the optimal solution is (x^*, λ^*) and $||x_k - x^*||$ and $||\lambda_k - \lambda^*||$ are of order ϵ for sufficiently small ϵ, the errors in x_{k+1} and λ_{k+1} are of order ϵ^2 (Gill *et al.* [81]). Most algorithms, including the one used in this work, use the BFGS update rule (or similar quasi-Newton update rule) to generate a positive definite approximation to the Hessian, which results in superlinear convergence (Moré and Wright [151]).

A number of packages are available to solve SQP problems, including implementations in the NAG and IMSL libraries. The results reported in this thesis were obtained using CFSQP (C code for Feasible Sequential Quadratic Programming, Lawrence *et al.* [120]) using QLD (Schittkowski [187]) to solve each quadratic programming subproblem. CFSQP is a variant of the general approach described above, and it is unique in that it maintains "feasible" iterates during the optimization—once an iterate is found that satisfies all linear constraints and nonlinear inequality constraints, all subsequent iterates will also satisfy these constraints. (For problems without an rg subsequence, this property assures that each iterate corresponds to a physically valid motion, though the goal state may not be achieved.) The merit function is the sum of the objective function and a weighted sum of the violations of the nonlinear equality constraints.

As with all iterative gradient-based optimization routines, SQP finds a local optimum which is not necessarily the global optimum. In addition, the finite-dimensional parameterization of the manipulator trajectory artificially limits the space of possible trajectories. The particular local optimum achieved by SQP depends on the shape of the feasible space and the initial guess.

10.5 Putting It Together

A dynamic task is specified by a *problem specification* file. This file encodes the dynamic task planning problem the user would like to solve. Each problem specification file consists of:

1. The name of an *object specification* file. This file contains a geometric description of the polygon, along with the center of mass of the object and its radius of gyration ρ.

2. Descriptions of the initial state and the desired goal state; the sequence of manipulation phases to use (e.g., g, rg, gf, grg, or grf); the number of constraint checks during each phase; an initial guess at the solution; and parameters related to the optimization routine (such as the convergence criteria and maximum number of iterations).

The problem specification file is passed as an argument to the optimization routine. The optimization routine calls the CFSQP code and passes it constraint and objective values when requested. The optimization also uses the following helper functions:

1. A function that returns joint positions, velocities, and accelerations for the specified time and current iterate. The examples in this thesis use cubic B-splines, but the code is written modularly to permit plugging-in different trajectory code. I have also experimented with polynomial trajectories, but the optimization appeared to converge more easily with cubic B-splines, perhaps because of the local effect of each knot point.

2. Functions which calculate the Jacobians $\mathbf{J}_{g1}, \mathbf{J}_r, \mathbf{J}_{g2}$ and their time derivatives $\dot{\mathbf{J}}_{g1}, \dot{\mathbf{J}}_r, \dot{\mathbf{J}}_{g2}$.

3. A function which simulates the rolling motion of the object using fourth-order Runge-Kutta simulation. This simulation should be as efficient as possible, to speed up the optimization. There is a tradeoff between the computational complexity (the size of the

time step) and the accuracy of the simulation. The results reported in this thesis use simulation time steps of about 10 ms. To justify this choice, several "typical" rolling motions (e.g., a pendulum falling in a gravity field) were tested with time steps as small as 0.5 ms. With fourth-order Runge-Kutta, 10 ms time steps returned angular positions and velocities typically less than 0.1% different than those obtained using 0.5 ms time steps. (In contrast, simple Euler integration required a much smaller time step to converge to the same accuracy.) This error was deemed insignificant relative to other errors that might occur in the physical implementation of a plan.

(For simplicity, the number of simulation steps during a roll is a fixed integral multiple of the number of constraint checks p during the roll. Thus the simulation step time varies with t_{roll}, but 10 ms is typical.)

In addition to functions that return the objective and constraint values for the current iterate, SQP also requires the gradients of the objective and constraints with respect to the design variables. These can be obtained symbolically by using the chain rule to take the partial derivatives of the objective and constraints with respect to the design variables. Instead of calculating symbolic derivatives of the fourth-order Runge-Kutta simulation, finite difference gradients are used. Simulation results are cached so they do not have to be re-evaluated for each finite difference gradient request.

All functions are implemented in C.

Chapter 11

Dynamic Manipulation Experiments

This chapter describes the implementation of some dynamic manipulation trajectories found by the optimization.

11.1 Experimental Setup

11.1.1 The NSK Arm

Our testbed for dynamic nonprehensile manipulation is a one-degree-of-freedom arm driven by an NSK Megatorque direct-drive motor with a maximum torque of 40 Nm. The arm is a centrally-mounted, 122 cm-long section of 4 in x 4 in (10.16 cm x 10.16 cm) extruded aluminum square tubing with a weight of about 3.6 kg. The motor is mounted on a column approximately 122 cm above the floor, and the arm swings in a vertical plane. The arm is also equipped with a 17.7 cm x 10.0 cm "palm," which is a piece of angle bracket mounted at a 45 degree angle at one end of the arm. The manipulation surfaces used by the arm are the palm and the top 122 cm x 10.16 cm surface of the arm ("whole arm manipulation"). These surfaces are covered by a soft foam, about 5 mm in thickness, which gives high friction with the test objects along with low restitution upon impact. A picture of the arm is shown in Figure 11.1, and the world coordinate frame attached to the base of the arm is shown in Figure 11.2.

The arm's control computer is a 386 PC. A serial line to the RS-232C port of the motor driver chooses the operating mode for the motor. The motor's commanded torque is specified by analog voltage commands in the $\pm$ 10 V range, sent from a D/A card in the PC to the motor driver. Quadrature resolver feedback of 153,600 counts/rev is fed into a decoder board on the PC, and velocity and acceleration feedback is obtained by differencing and double differencing, respectively. The servo rate is 1 kHz.

Even for such a simple manipulator, the demands of dynamic manipulation present unusual difficulties in control. Ideally, the manipulator should appear to the object as an acceleration source. To achieve this, I developed a model of the torque available from the

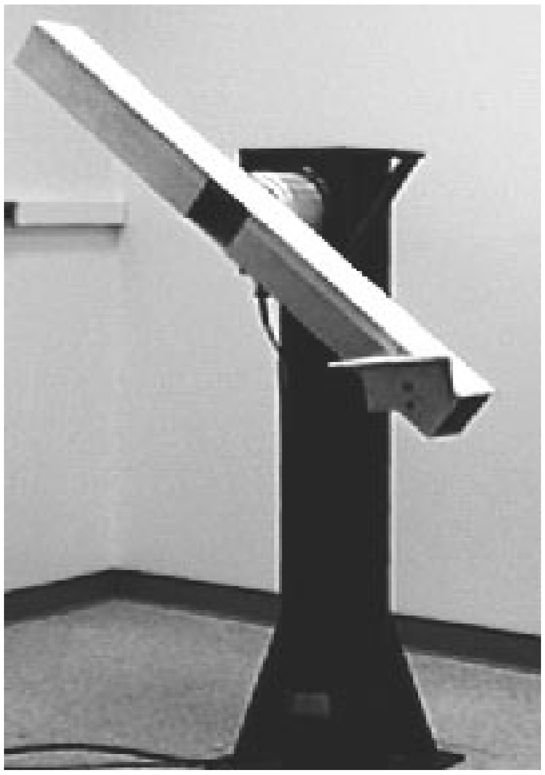


Figure 11.1: The one-degree-of-freedom NSK arm.

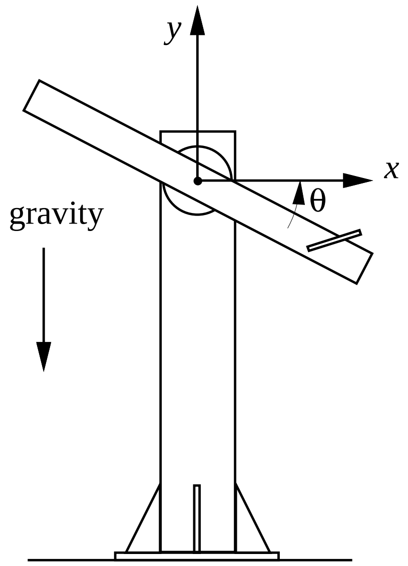


Figure 11.2: The world coordinate frame.

motor at different speeds. I discovered that friction in the motor is significant (up to 5% of the total torque available) and that the speed-torque characteristics are highly nonlinear. The details of the motor modeling are given in Appendix B.

Execution of all experiments proceeds as follows:

1. Using the dynamic model of the arm, a feedforward torque profile is generated to follow the trajectory specified by the planner.

2. The trajectory is executed once with PD feedback. (The gains of the PD feedback are intentionally chosen to be small.) The results of the run are stored, including errors due to unmodeled effects (including the mass of the object, which, in our experiments, is small with respect to the manipulator's mass).

3. One iteration of "learning control" is applied to the results of the previous run to generate a new feedforward control that compensates for unmodeled effects. We use an algorithm due to Atkeson [16] (see Appendix B).

4. The feedforward control is now fixed for all time, and future runs use this feedforward control along with PD feedback.

This execution procedure is somewhat arbitrary; other learning control algorithms could be applied, and they could be applied multiple times, so the manipulator continues to improve its trajectory following. For reliability and data acquisition, however, it was undesirable to have the manipulator trajectory changing (however slightly) between runs.

A single iteration of Atkeson's learning algorithm generally improved trajectory following (see Appendix B for examples). After multiple runs, it could begin to overcompensate and become unstable. In practice, the precise motor and arm dynamic models, the single learning iteration, and small PD error corrections resulted in very faithful trajectory following. Because initial trajectory following is improved with PD feedback, the improvement due to the learning iteration is less dramatic (and sometimes negligible) compared to the case of no PD feedback.

The robot is typically programmed by dumping a cubic B-spline trajectory generated by the optimization directly to the robot's controller. However, I have also created a graphical user interface, written in Tcl/Tk, that allows the user to manually create a spline trajectory by dragging knot points (Figure 11.3). This interface can be used to generate initial guesses for the optimization procedure, or to manually modify a solution found by the optimization.

For throwing problems, the controller appends a release sequence to the end of the feedforward trajectory. The release is effected by maximally accelerating the arm away from the object for 100 ms. After that, the arm follows a bang-bang trajectory to reach its final configuration in minimum time. Since the arm sometimes acts as the catcher for the thrown object, it is important that it reaches its final configuration as quickly as possible.

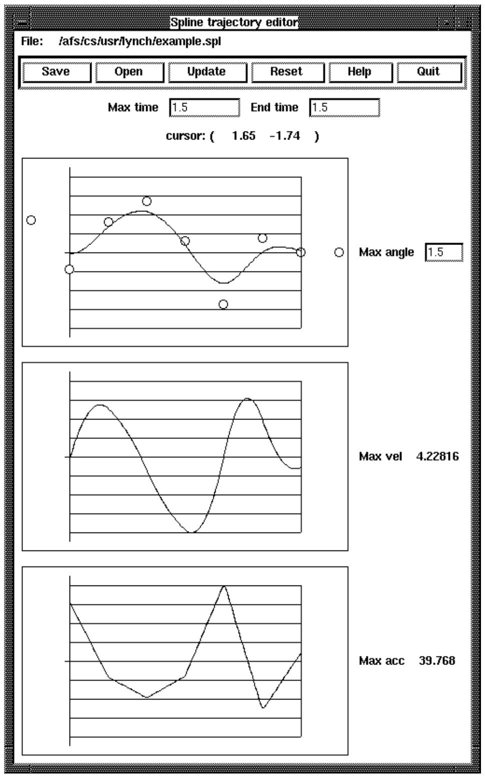


Figure 11.3: The spline trajectory editor.

11.1.2 The Test Objects

The objects used in the experiments were five lightweight, wooden, rectilinear objects. The planar profile was square for three of the objects and rectangular for two. In all cases, the center of mass was at the center of the object. Table 11.1 gives more information on each.

The objects are lightweight for safety reasons, and to minimize the disturbance to the robot's trajectory due to the mass of the object. Object C, the largest object, is actually composed of four small wooden blocks at each corner of the square, joined by wooden dowels.

11.1.3 Departures from the Ideal Model

For simplicity in planning, I have assumed rigid-body mechanics with Coulomb friction. The actual system deviates somewhat from the ideal.

1. The foam on the arm's surface compresses slightly under the weight of the objects. This can be especially important during rolling, as it effectively rounds the rolling corner.

2. Extra "friction" arises from the compression of the foam. Thus heavier objects appear to have higher friction than lighter objects. This violates the Coulomb assumption

object	material	planar dimensions (cm)	out of plane dimension (cm)	ρ^2 (cm^2)
A	basswood	8.8 x 8.8	8.8	12.9
B	basswood	7.6 x 7.6	7.6	9.6
C	basswood	27.0 x 27.0	8.9	225.0
D	balsa	12.6 x 3.7	7.6	14.4
E	balsa	12.6 x 7.6	4.1	18.0

Table 11.1: Objects used in the dynamic manipulation experiments.

that friction only depends on the materials in contact.

3. The foam may act as a low-pass filter, smoothing the time sequence of forces experienced by the object and limiting the dynamic response.

Despite these violations, rigid-body mechanics with Coulomb friction is a useful approximation for planning.

11.2 Experiments

This section presents a variety of dynamic manipulation tasks, all of which have been successfully implemented on the NSK arm. Some of the problem specifications required a bit of empirical tuning, primarily to deal with unmodeled impact and the "rounding" effect of rolling on foam, but in all cases the trajectory implemented on the arm is the unmodified output of the optimization.

The manipulator constraints used in each problem are acceleration constraints (± 60 rad/s^2) (torque constraints for the approximately symmetric arm take the simple form of acceleration constraints) and initial and final state constraints. Our robot has no joint limits, eliminating position constraints, and velocity constraints were not an issue. Object constraints included dynamic grasp constraints, rolling constraints for problems with rolls, and goal state constraints.

For each type of experiment (snatch, roll, etc.) the text describing all experiments is given first, and then the figures. The run times reported are for the optimization running on a Sun SPARC 20.

11.2.1 Snatching

Both of the snatching examples use the phase sequence rg and roll the object onto the palm. The end angle of the arm is unconstrained, provided the final configuration is a static grasp.

A pure g snatch on a vertical face of an object is avoided because it requires a very large friction coefficient.

Snatch Example 1 The arm begins at $\theta = -90°$ and snatches object D from a supporting table into the palm. The palm and the object are initially in contact. Because of the geometry of the contact, as the arm begins to move and the object begins to roll toward the palm, the object breaks contact with the table immediately. We can therefore ignore contact forces between the object and the table.

The spline trajectory is represented by seven knot points. After 89 iterations and 24 seconds, the planner finds the trajectory shown in Figures 11.4 and 11.5, which requires a friction coefficient of 0.451. Notice that the arm reverses its path, much like the dynamic carry of Figure 10.3. The implementation of this task on the robot is robust; it works every time. I also tried the same trajectory at 150% speed and 67% speed. In the first case, the arm snatches the block, but then throws it when it reverses direction, and never attains a static grasp. In the second case, the arm merely pushes the block off the table. As expected, the speed of execution matters.

Snatch Example 2 In this example, object E is snatched from the table. The trajectory is represented by nine knot points, and after 200 iterations and 120 seconds, the optimization returns the trajectory shown in Figures 11.6 and 11.7. The required friction coefficient is 0.609. When I implemented this trajectory on the arm, the initial compression of the foam at the contact was insufficient to realize this friction coefficient. After covering the contact face of the object with sandpaper, the snatch was consistently successful (Figure 11.8). If the initial contact is slightly too high on the object, however, it never rotates fully onto the palm, and it eventually falls off. Notice in Figure 11.6 that the simulated rolling speed drops to zero at one point during the roll.

The primary reason the friction coefficient is higher than the previous example is that the line from the pivot to the center of mass of the object is closer to horizontal, making object E harder to snatch than object D. If the line from the pivot to the center of mass drops to horizontal or below, the object is impossible to snatch regardless of the friction coefficient.

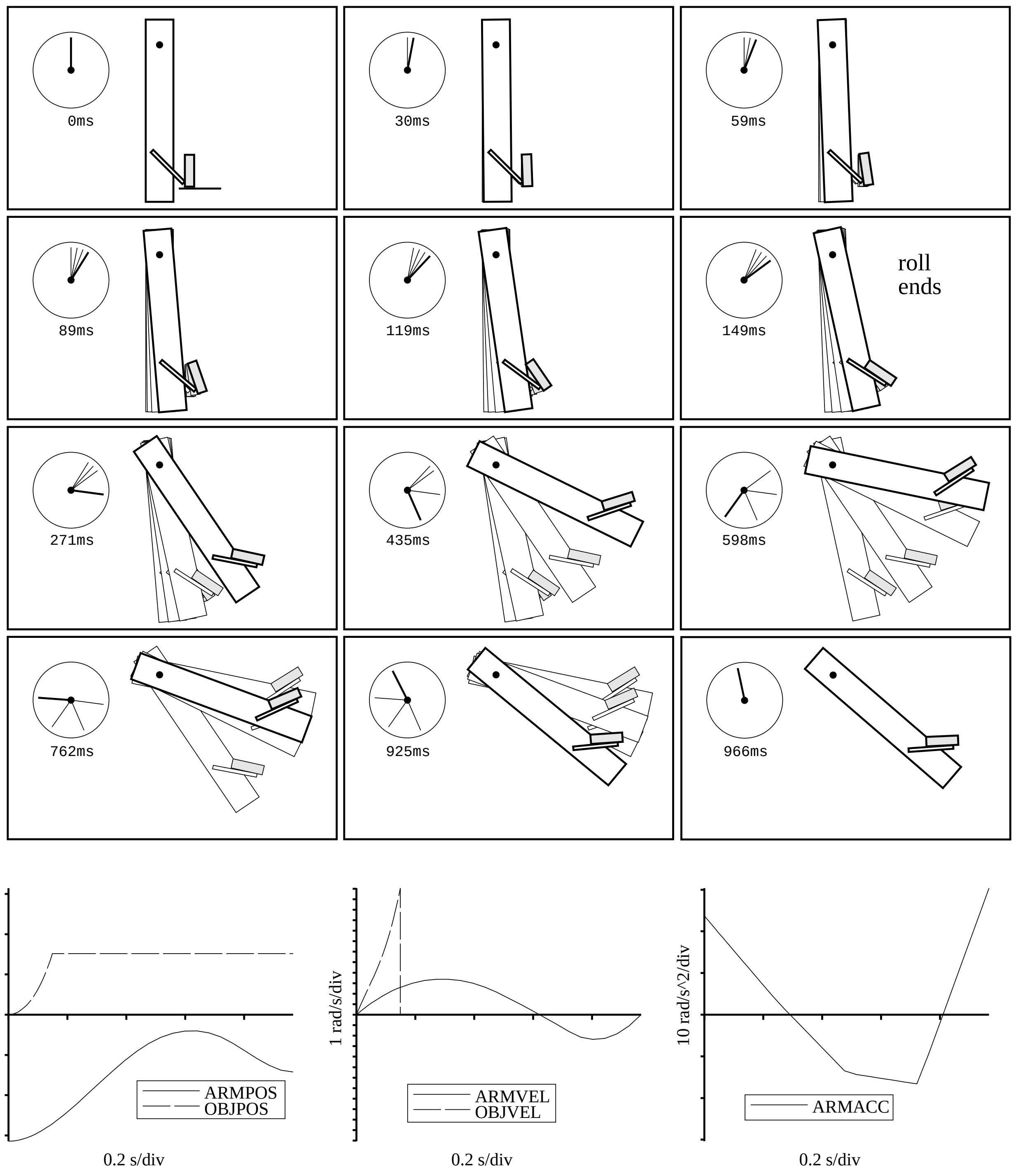


Figure 11.4: Snatch example 1: the trajectory found by the optimization. Friction is 0.451. The plots show the angle of the arm (ARMPOS) and the angle of the object (OBJPOS) relative to it. Note that the time between frames is not constant, so an equal number of frames of both phases can be seen. The clock indicates the time of each frame, and the previous three frames create a "motion blur."

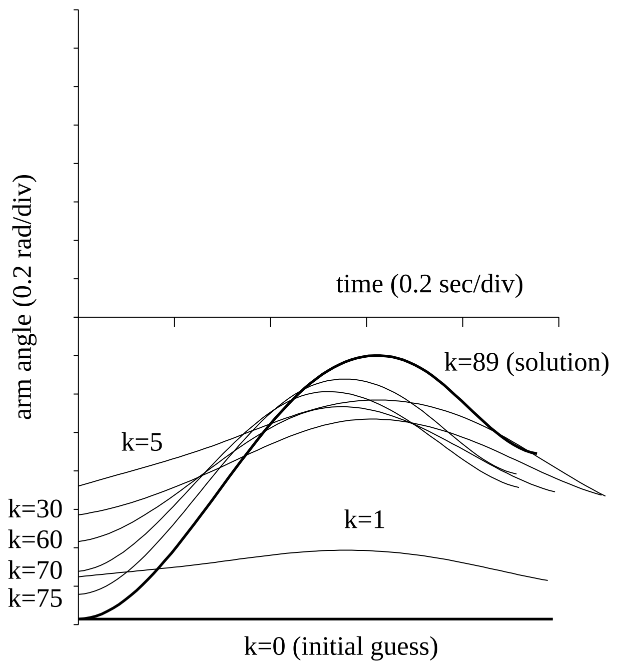
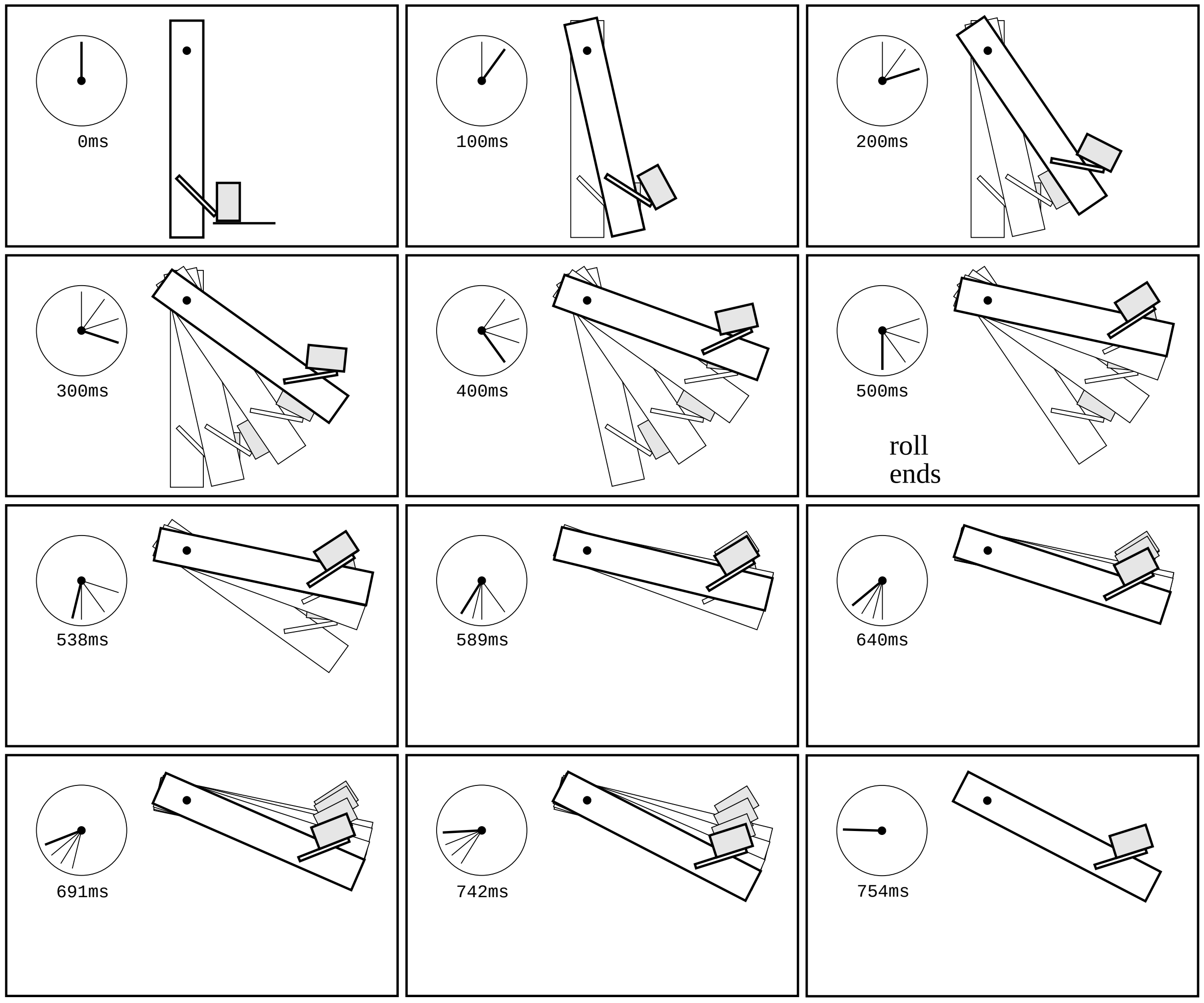


Figure 11.5: Snatch example 1: the initial trajectory guess, the solution, and some intermediate iterates.



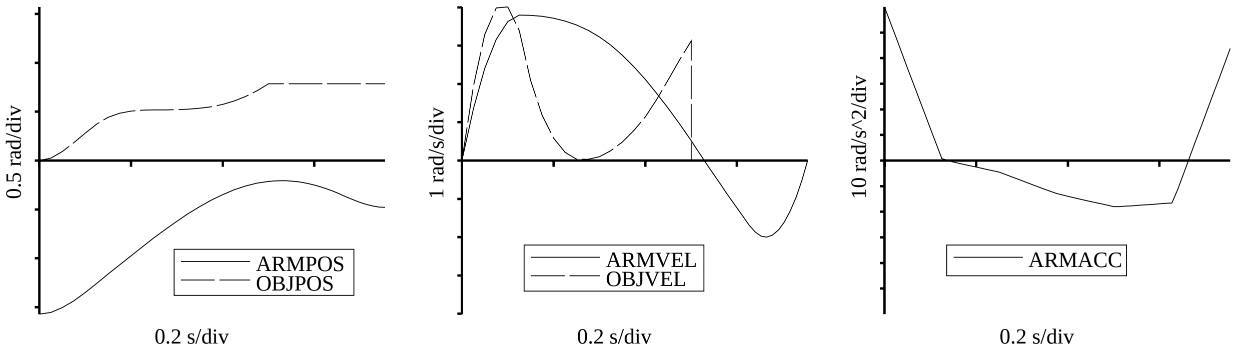


Figure 11.6: Snatch example 2: the trajectory found by the optimization. Friction is 0.609.

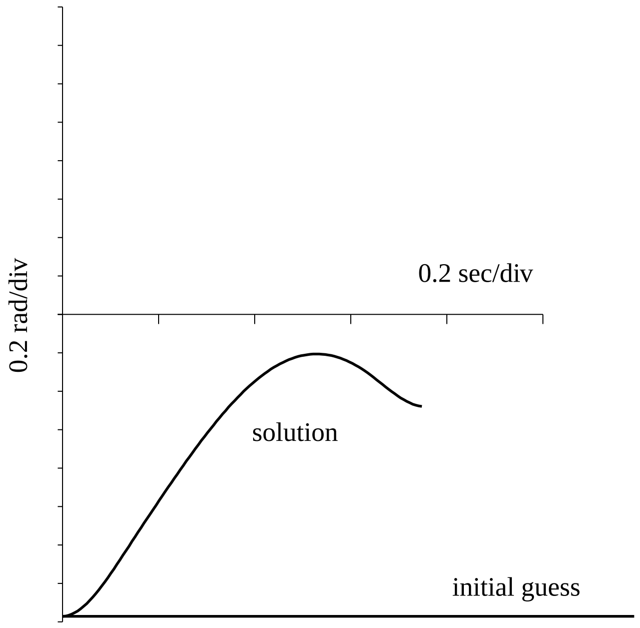


Figure 11.7: Snatch example 2: the initial trajectory guess and the solution.

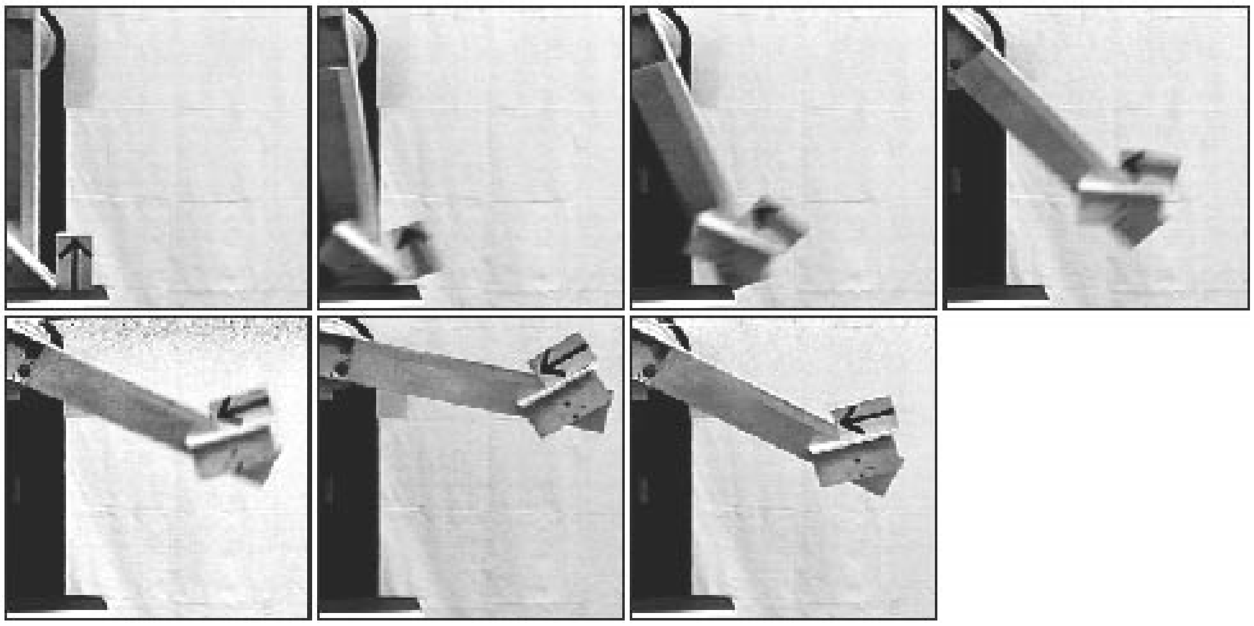


Figure 11.8: Snatch example 2: a video record of the implementation on the robot.

11.2.2 Rolling

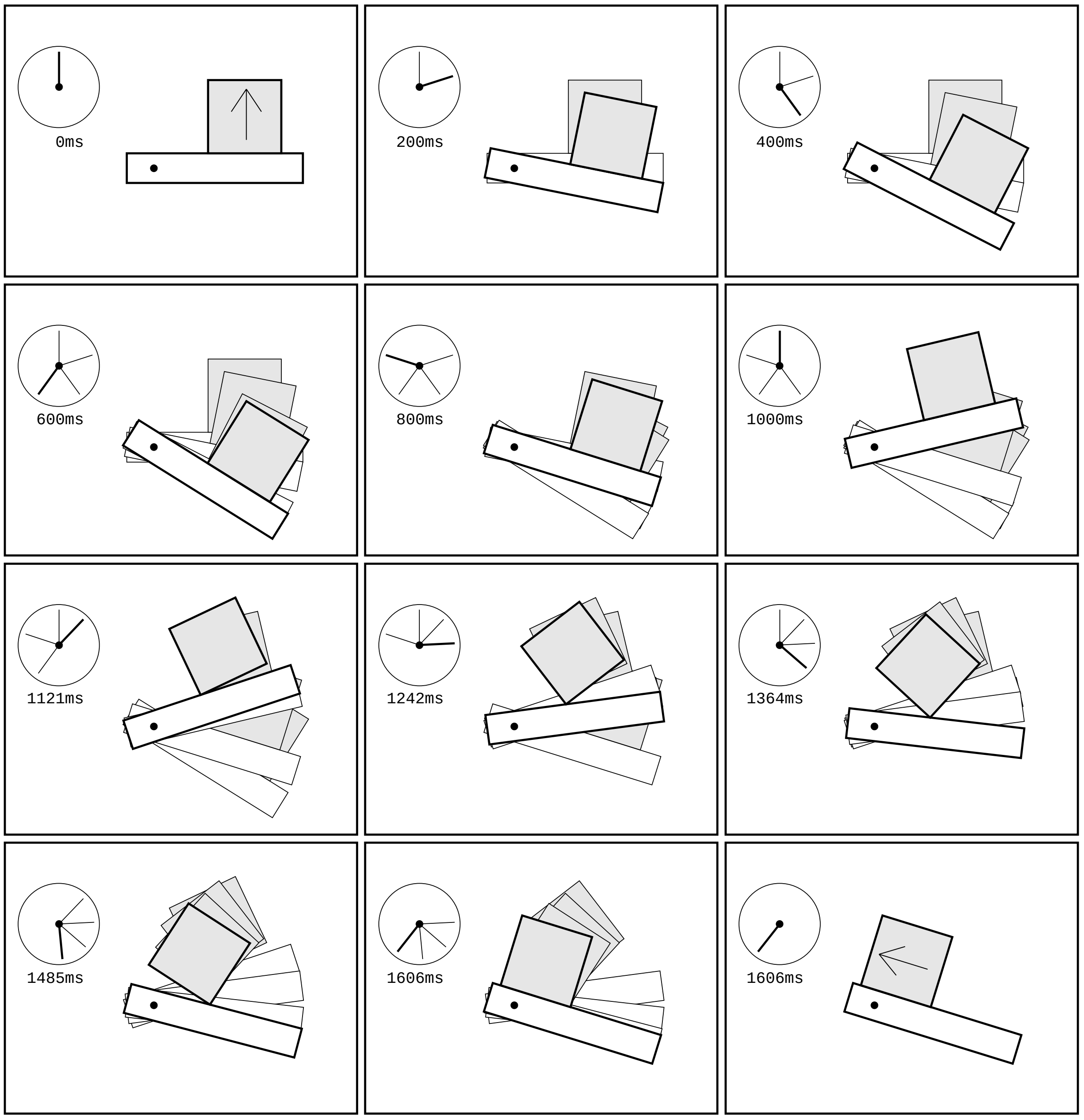
The manipulation phase sequence is grg, where the duration of the final g phase is zero. In other words, the arm stops moving as soon as the block has finished its roll, and this final configuration is a statically stable grasp.

Roll Example Unlike all of the other examples, I chose to minimize the squared angular velocity of the object relative to the manipulator at the end of the roll, to reduce impact. Friction was fixed and set high ($\mu = 1.5$).

The initial state of the system is the arm and object at rest, with the arm at $0°$ and the near (pivot) vertex of object C at $x = 20$ cm. The goal state is a $\pi/2$ rotation of the square relative to the arm, ending in a static grasp. (The arm angle is unconstrained.) The trajectory of the arm is represented by nine knot points. After 32 seconds and 55 iterations, the optimization found a trajectory that yields an impact velocity of approximately 3.1 rad/s. The designed trajectory and its implementation on the robot are shown in Figures 11.9, 11.10, and 11.11.

Notice that the solution found is to "throw" the object such that its angular velocity at the beginning of the roll is sufficient to to take the center of mass of the object over the vertex. The constraint $\theta(T) \geq -0.3$ rad was added to the optimization, without which the arm would tend to stop at $-\pi/4$ rad, when the center of mass is directly over the vertex. While this theoretically minimizes the impact velocity, it is not robust, and even the slightest impact sends the object rolling back in the other direction.

The implementation of this dynamic task is very robust, and the rebound on impact is small.



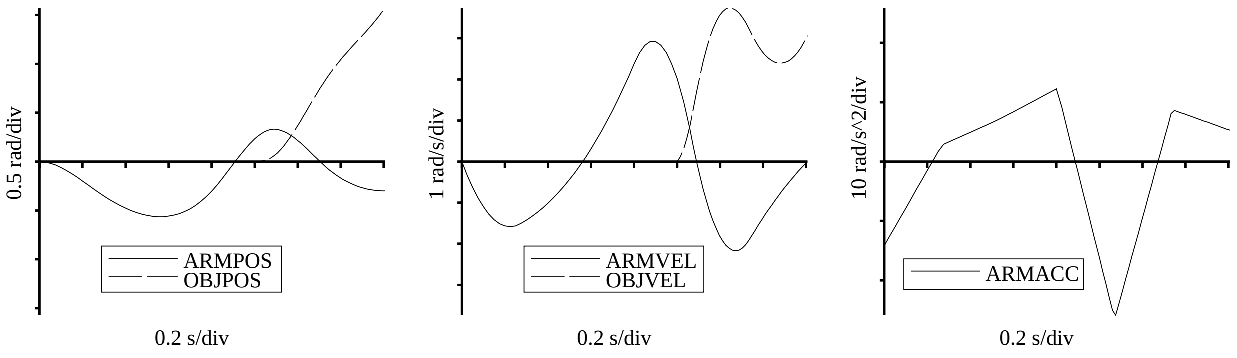


Figure 11.9: Roll example: the trajectory found by the optimization.

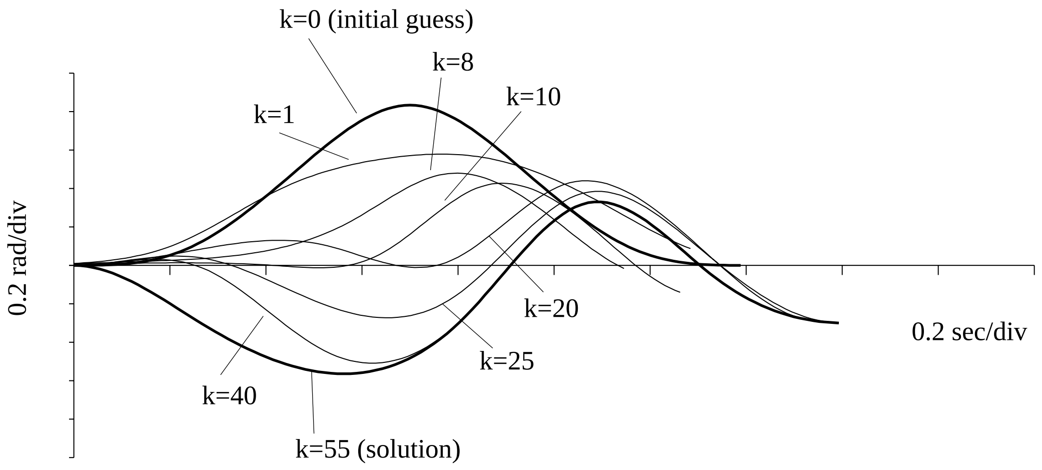


Figure 11.10: Roll example: the initial trajectory guess, the solution, and some intermediate iterates.

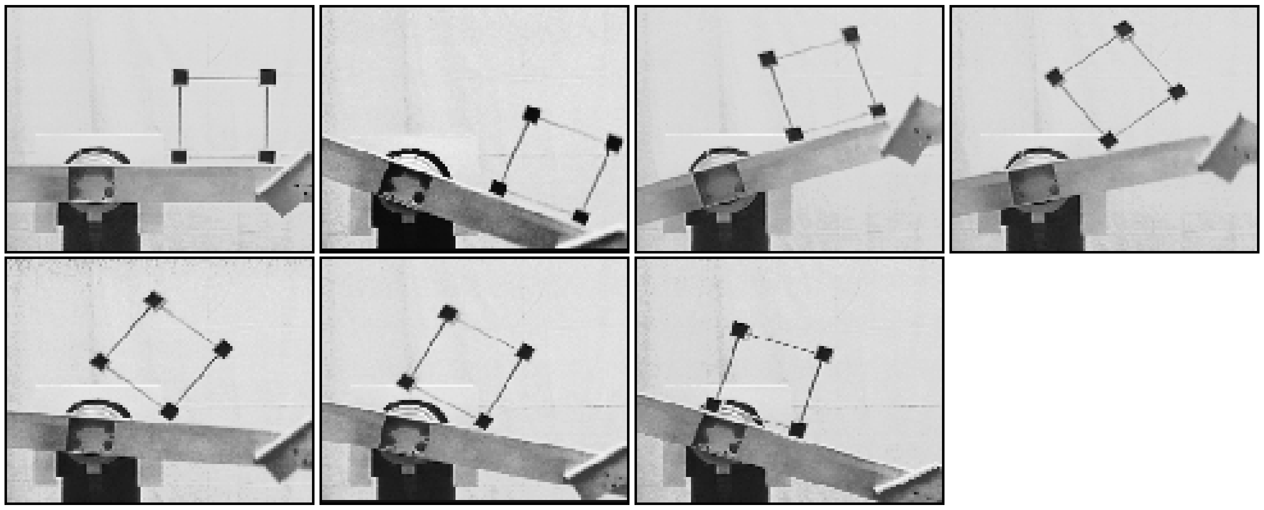


Figure 11.11: Roll example: a video record of the implementation on the robot.

11.2.3 Throwing (and Sometimes Catching)

This section illustrates throwing examples with the phase sequence gf. With a throw, we can specify as many as $2n+1$ independent constraints on the goal state, where n is the number of joints of the manipulator. With the one-degree-of-freedom arm, I usually choose the goal configuration and leave the goal velocity free.

Although throwing to a specified configuration is the goal of the optimization, not throwing to a catch, for most of the throwing examples I have chosen the goal configuration to correspond to a catch configuration. This allows for interesting manipulation on the arm, such as flipping. I empirically determined goal states that are robust for catching. In particular, the angle ϕ is important at impact; in general, the initial impact occurs at a vertex, and the vector from this vertex to the center of mass determines whether the impact increases or decreases each component of the object's velocity.

Section 11.2.4 gives an example of explicitly designing a throw to result in a catch.

Throw Example 1 The first example is to simply flip the cube B 1/4 revolution CCW in place. The block is initially sitting on the arm at 0° with its far vertex at $x = 35$ cm. To make the catch robust, I specified that the block should slightly overrotate, to 1.8 radians instead of 1.57 radians. The trajectory is represented by nine knot points. After 300 iterations and 59 seconds, the optimization found the trajectory shown in Figures 11.12 and 11.13, which requires a friction coefficient of 0.710. The implementation on the arm, shown in Figure 11.14, is robust.

Throw Example 2 In this example, the goal is to stand up object E. The object is initially lying on its long edge, with its far vertex at $x = 40$ cm, and the goal configuration is a 1/4 revolution CCW to straddling the $x = 20$ cm mark. The nine-knot trajectory shown in Figures 11.15 and 11.16 took 62 iterations and 23 seconds to find. The implementation is shown in Figure 11.17.

Over ten consecutive throws, the average final position was $x = 18.9$ cm, an error of 1.1 cm, and all final positions were within 2 mm of this average.

Throw Example 3 This example is similar to the previous example, except the goal is to stand the block up in place, so the far vertex at the catch is at $x = 40$ cm. To make the catch robust, the goal configuration was slightly overrotated (1.7 radians).

This throw requires a significantly higher friction coefficient. The nine-knot trajectory shown in Figures 11.18 and 11.19 was found after 49 iterations and 14 seconds. The implementation is shown in Figure 11.20.

Throw Example 4 This throw is "glued" to the end of snatch example 2. After snatching object E, the arm throws it to standing up on the palm, a 1/4 revolution CCW. The catching configuration is chosen to be standing on the palm when the arm is at $-45°$ (i.e., the palm is horizontal). To make the catch robust, the object is overrotated to 1.8 radians.

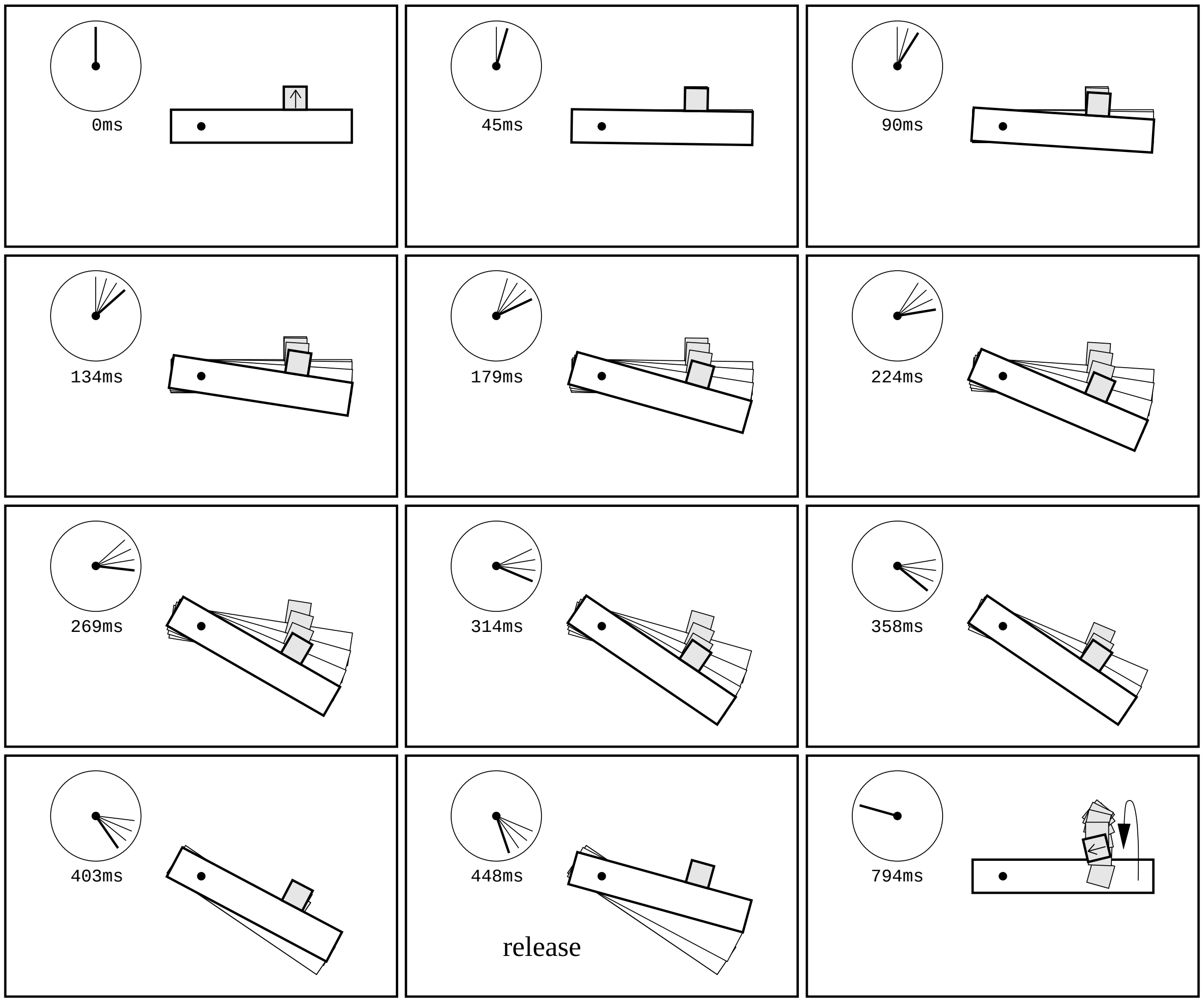
The solution is more of a drop than a throw. The optimization takes 98 iterations and 13 seconds to find the seven-knot trajectory shown in Figures 11.21 and 11.22. The throw is experimentally robust (Figure 11.23), although occasionally the catch fails when glued to the snatch if the object tilts slightly out of the plane during the snatch.

Throw Example 5 To check the accuracy of a throw, I generated a trajectory to throw object E from the palm to a point in free space. This throw was executed 20 times, and the flight of the object was recorded by a 60 Hz vision system. The goal configuration is $x = 65.0$ cm, $y = 30.0$ cm, and $\phi = 3.14$ rad from the object's start angle. The seven-knot trajectory in Figures 11.24 and 11.25 was obtained after 50 iterations and 7 seconds.

The results of the experiment are shown in Figure 11.26. I fit a parabola to the motion of the center of mass of the object in each throw. Also shown are the object configurations along each trajectory that are closest in (x, y) to reaching the goal configuration. The average "closest" configuration for the throws was $(63.78, 29.73, 3.31)$, for an error of $(1.22, 0.27, 0.17)$ with standard deviations of $(0.145, 0.032, 0.023)$.

This experiment shows that the throws are very repeatable, which is encouraging, but there is some systematic error. The data are consistent with the hypothesis that the object is being released slightly later in the trajectory: the object is thrown somewhat higher, with less x velocity than the planned trajectory. The most likely reason for the difference is the assumption of an instantaneous release. In reality, there is a slight delay between the torque "bang" command and the response of the arm. This problem could be somewhat exacerbated by the soft foam.

To achieve greater accuracy in the throw, the motor frequency response and the foam's response could be modeled and compensated for.



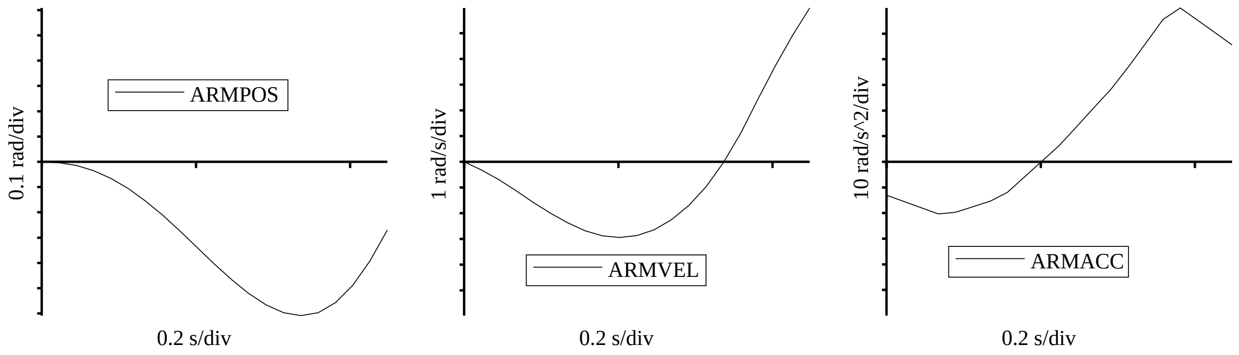


Figure 11.12: Throw example 1: the trajectory found by the optimization. Friction is 0.710.

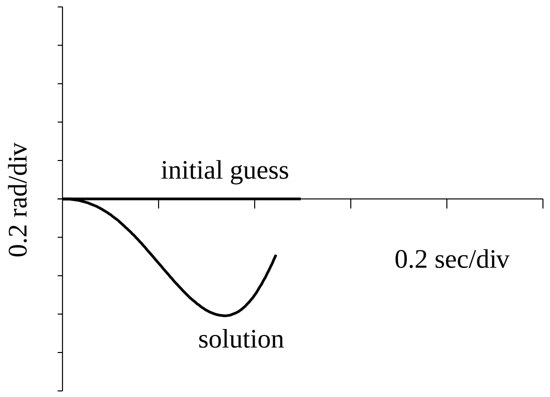


Figure 11.13: Throw example 1: the initial trajectory guess and the solution.

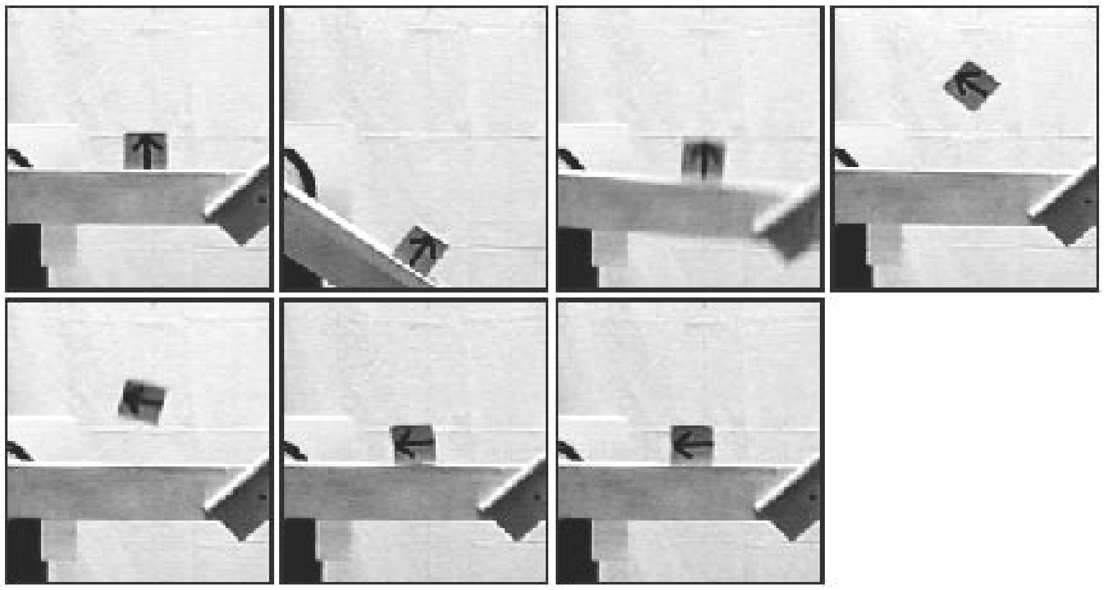


Figure 11.14: Throw example 1: a video record of the implementation on the robot.

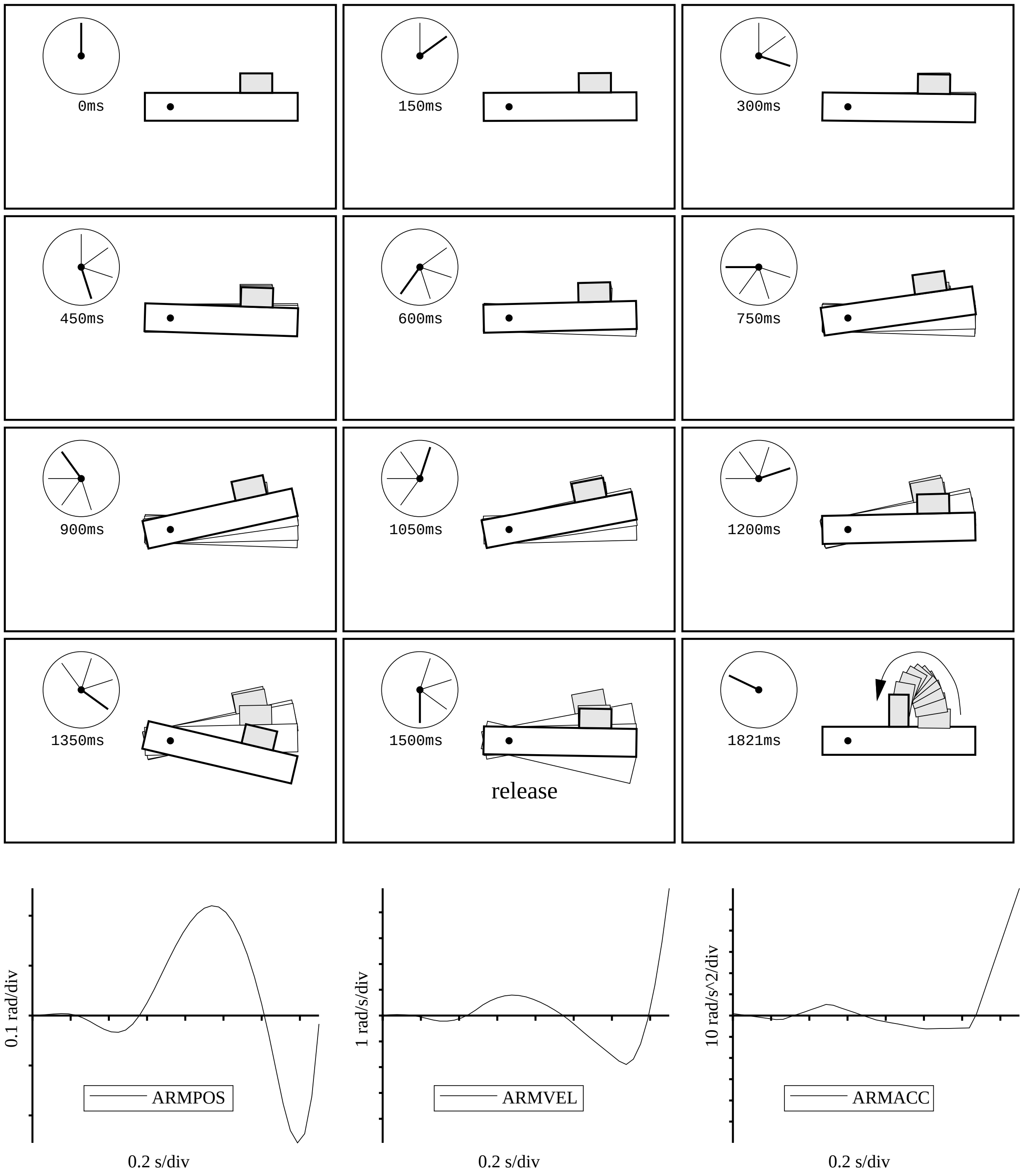


Figure 11.15: Throw example 2: the trajectory found by the optimization. Friction is 0.350.

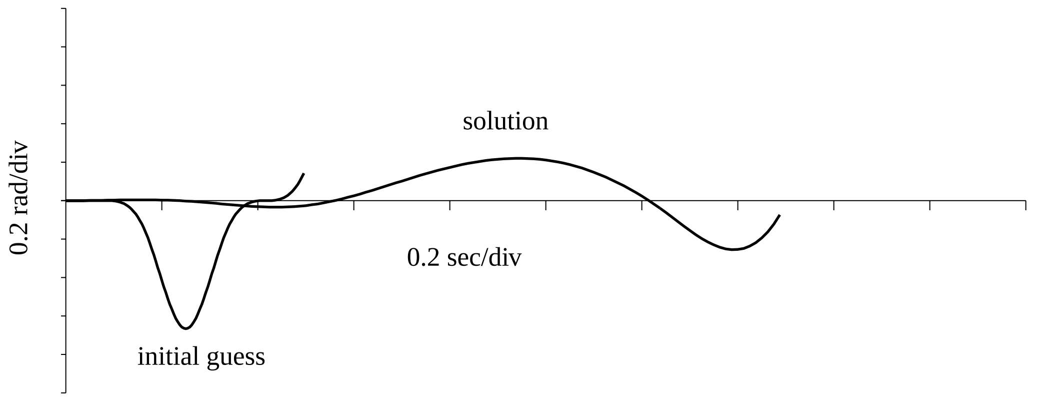


Figure 11.16: Throw example 2: the initial trajectory guess and the solution.

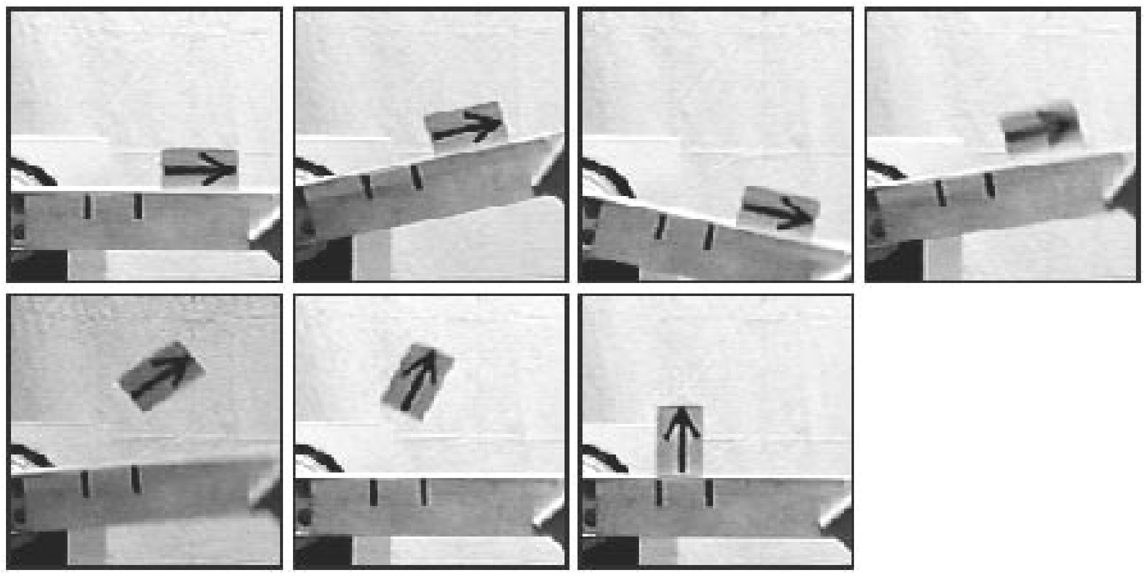
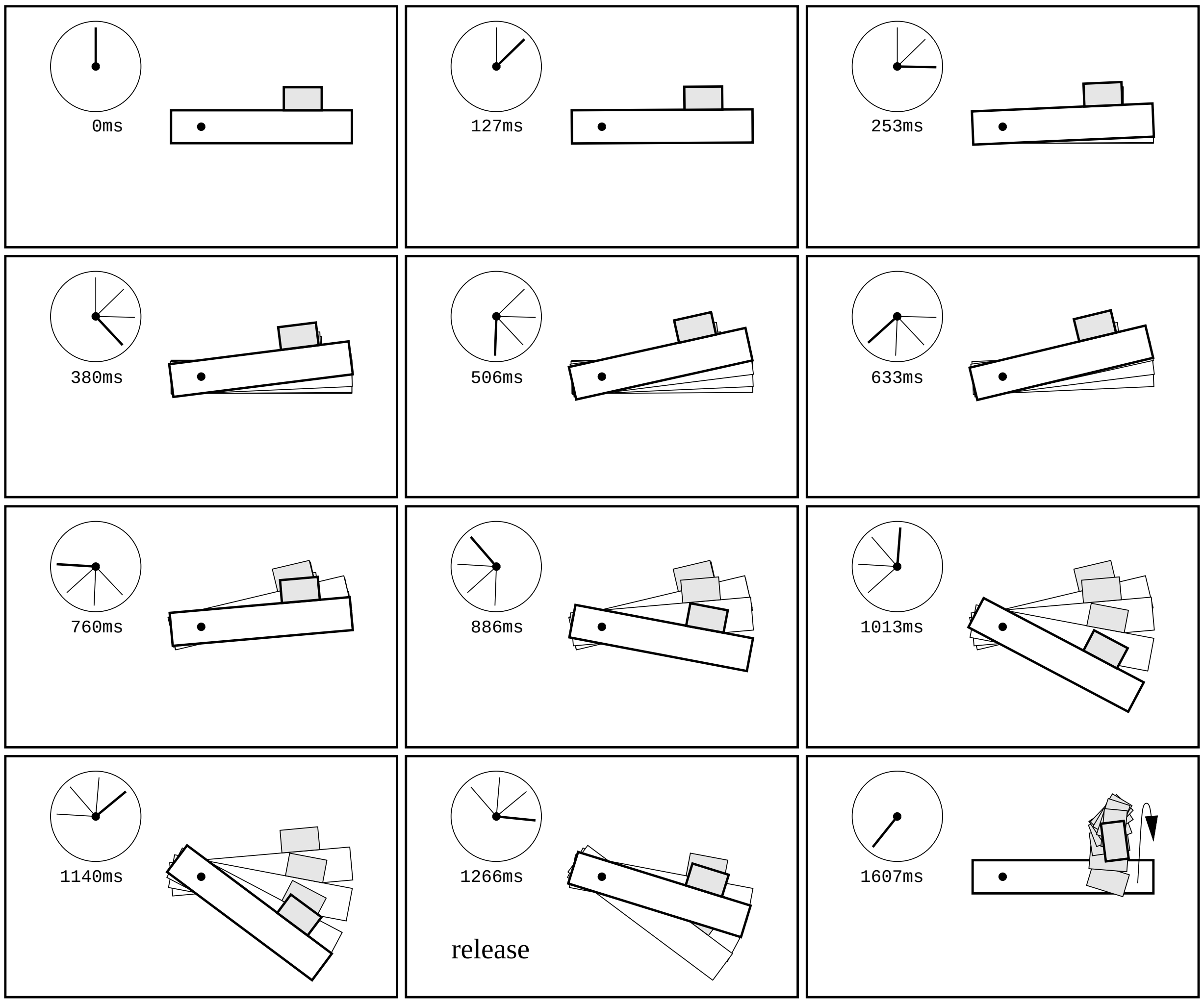


Figure 11.17: Throw example 2: a video record of the implementation on the robot.



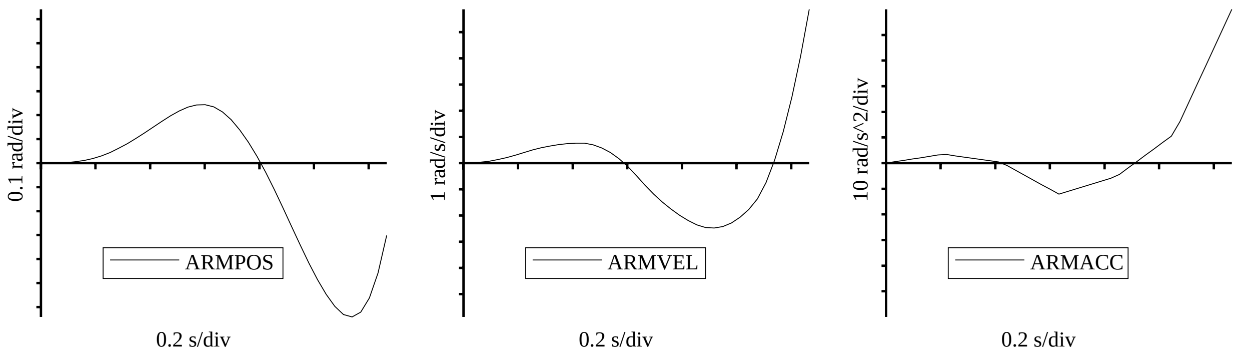


Figure 11.18: Throw example 3: the trajectory found by the optimization. Friction is 0.616.

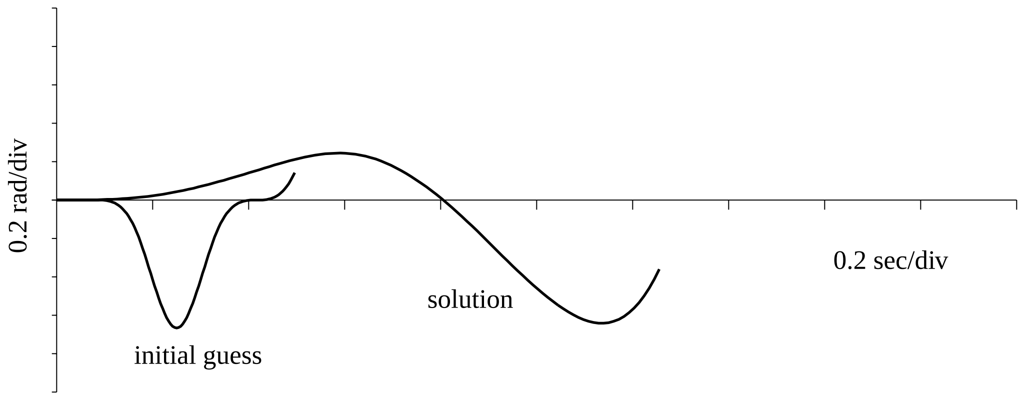


Figure 11.19: Throw example 3: the initial trajectory guess and the solution.

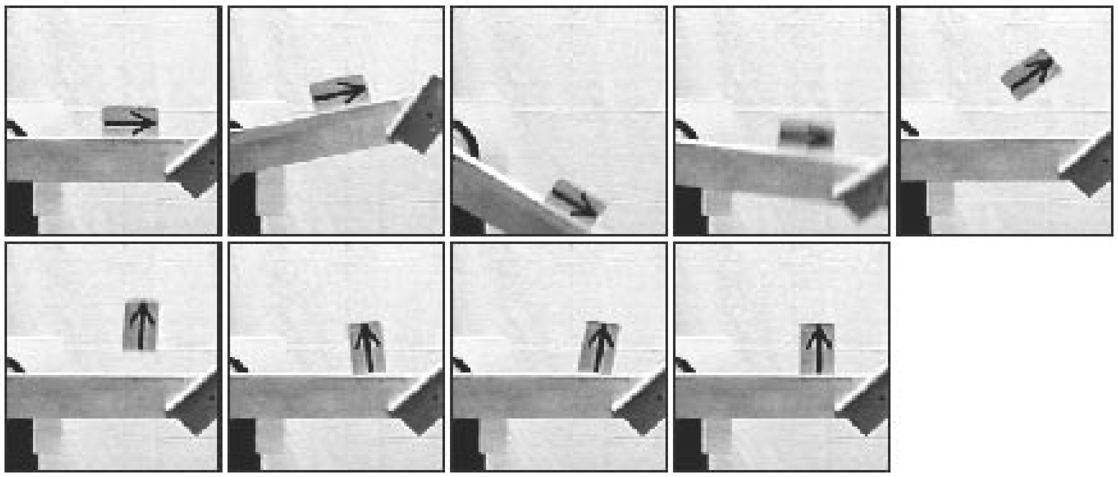
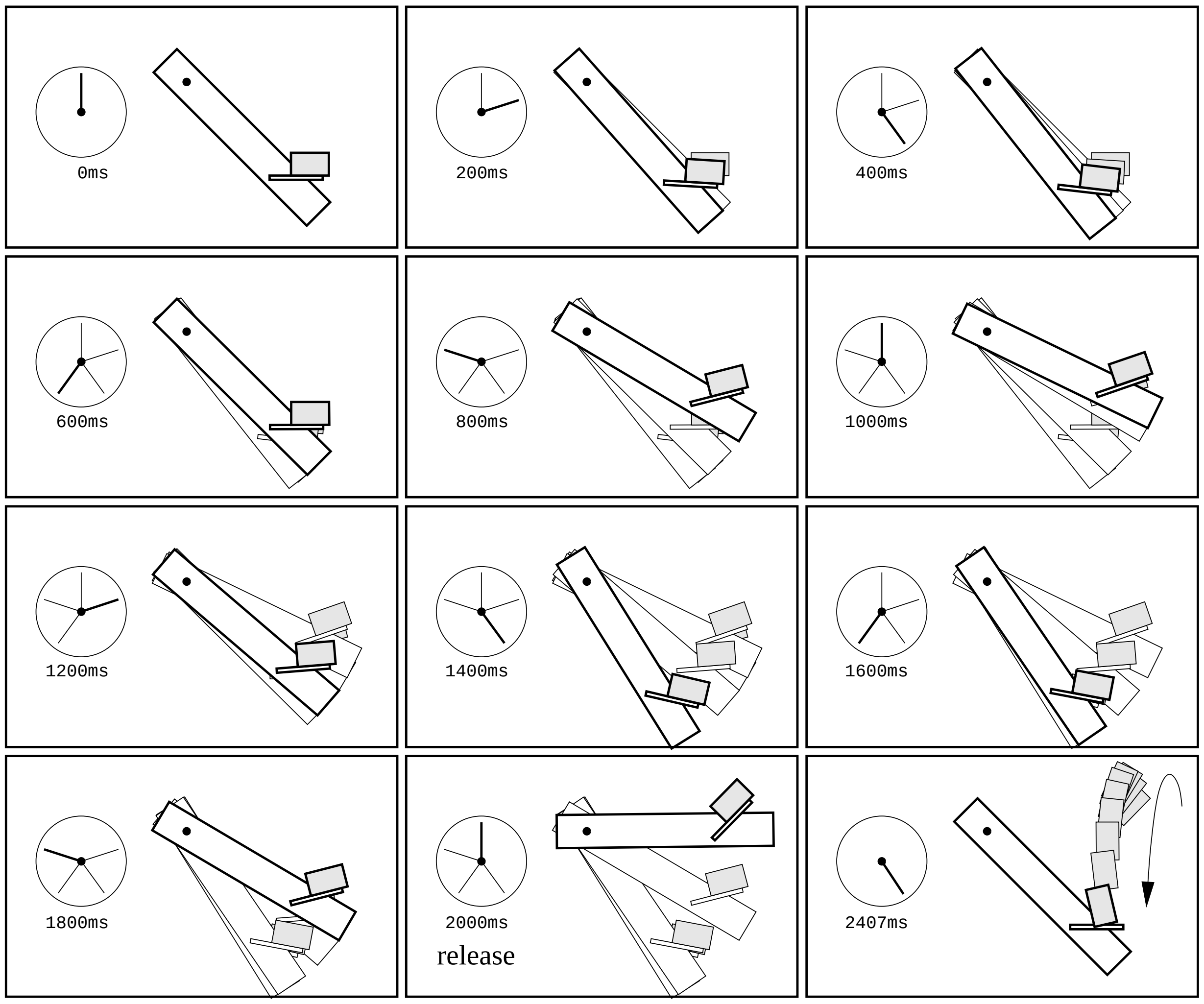


Figure 11.20: Throw example 3: a video record of the implementation on the robot.



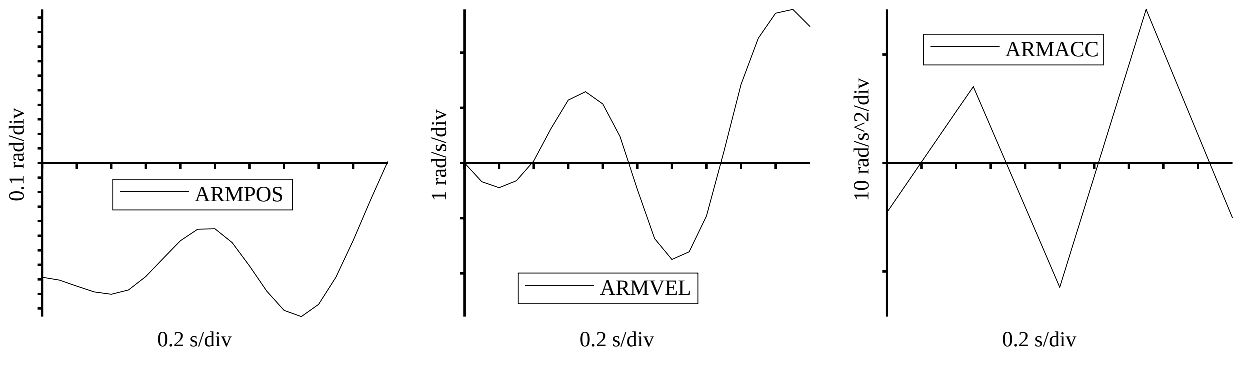


Figure 11.21: Throw example 4: the trajectory found by the optimization. Friction is 0.156.

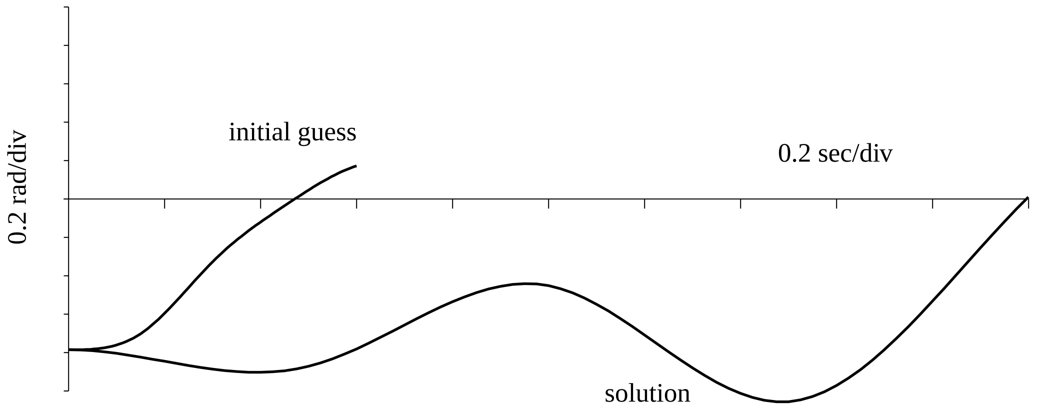


Figure 11.22: Throw example 4: the initial trajectory guess and the solution.

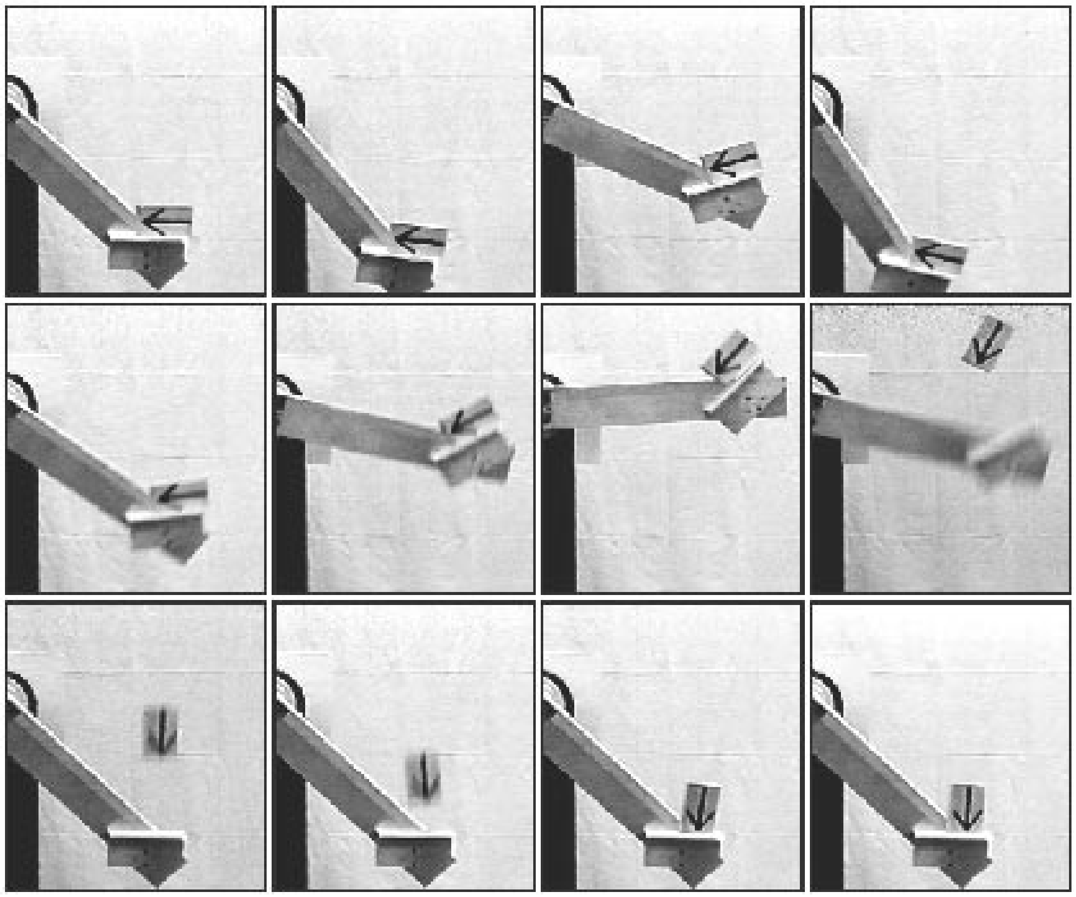


Figure 11.23: Throw example 4: a video record of the implementation on the robot.

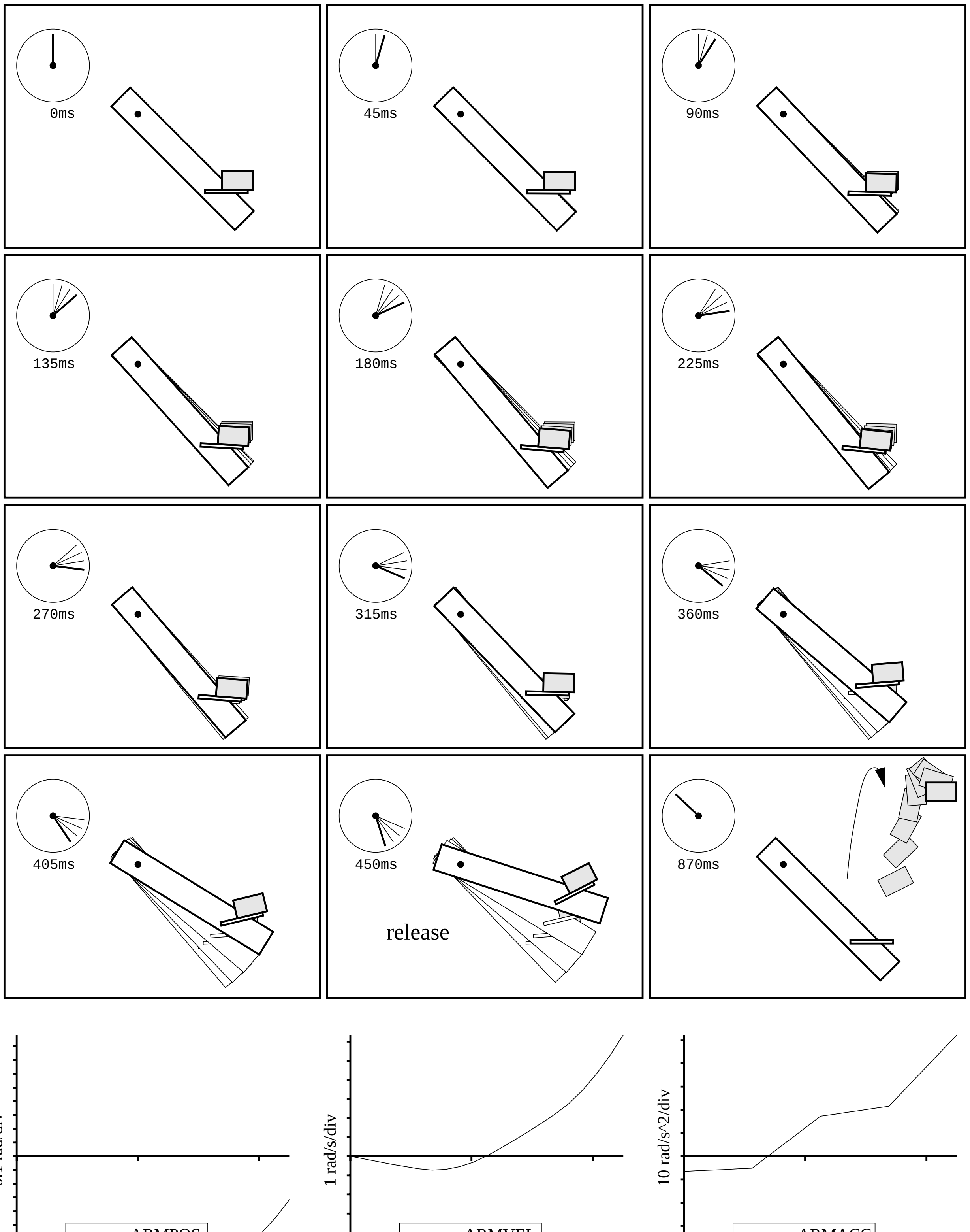


Figure 11.24: Throw example 5: the trajectory found by the optimization. Friction is 0.243.

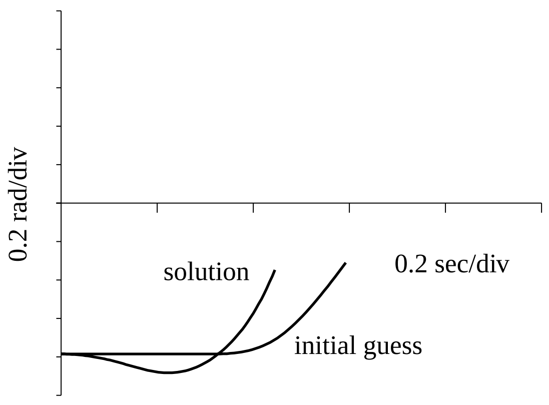


Figure 11.25: Throw example 5: the initial trajectory guess and the solution.

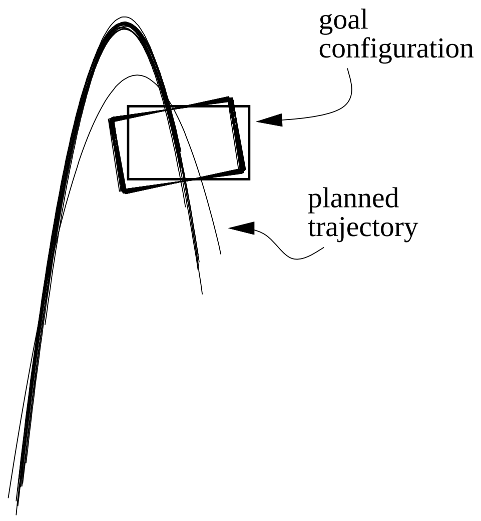


Figure 11.26: Throw example 5: parabolic fits to the data of 20 consecutive throws. Also shown are the block configurations that are closest to the goal (x, y) configuration for each throw. The repeatability of the throw is very good, but there is noticeable systematic error.

11.2.4 Encoding Impact Constraints: Catching at the "Sweet Spot"

The throwing experiments until now have focused on throwing objects to desired configurations. For convenience, I have chosen many of the goal configurations to correspond to robust catches on the arm. However, the optimization itself has no notion of catching.

To plan for a catch, we must account for impact. Although solving the forward impact problem (simulation) in the planar case is not difficult, in general it is very difficult to solve the inverse problem: given a desired goal configuration, find a state before the impact sequence that results in the goal configuration.

Because the arm is covered with a soft foam, however, we can employ a simplified model of impact that allows us to encode catching constraints in the optimization. The planner can then solve explicitly for a throw and catch motion.

I assume the initial impact vertex does not rebound and does not slip. This is usually true for the actual system; the impact vertex tends to dig into the foam. With this simplified model, the motion of the object immediately after the impact is simply the motion that corresponds to the impulse through the impact vertex that keeps this vertex stationary.

Of particular interest is arranging an initial impact that exactly cancels all linear and angular velocity of the object. This is "catching at the sweet spot." Formally, given a reference point O and a line L through O and the center of mass, the sweet spot or *center of percussion* (Symon [208]) is the point O' on L where an impulse perpendicular to L transmits no impulse to O. If O is a distance d from the center of mass, then O' lies a distance ρ^2/d from the center of mass on the other side of the center of mass.

Baseball provides a well-known example of the center of percussion: the batter attempts to hit the ball at the sweet spot of the bat with respect to the grip, so no impulse is felt at the hands.

To stop the motion of a moving object, define O to be the point in the object's frame that has no linear velocity. There is always such a point if the object has nonzero angular velocity. Then an impulse of the proper magnitude through the center of percussion O' will instantaneously stop the angular velocity about O, while leaving the linear velocity of O zero. (If the object has zero angular velocity, the impulse must act through the center of mass of the object and opposite the linear velocity of the object.)

I assume that the catcher is fixed, so it is the job of the thrower to yield a flight trajectory that results in a sweet spot catch. Two of the three release constraints are used to place the impact vertex at the desired location at T_{flight}, and the remaining constraint is used to encode the sweet spot condition

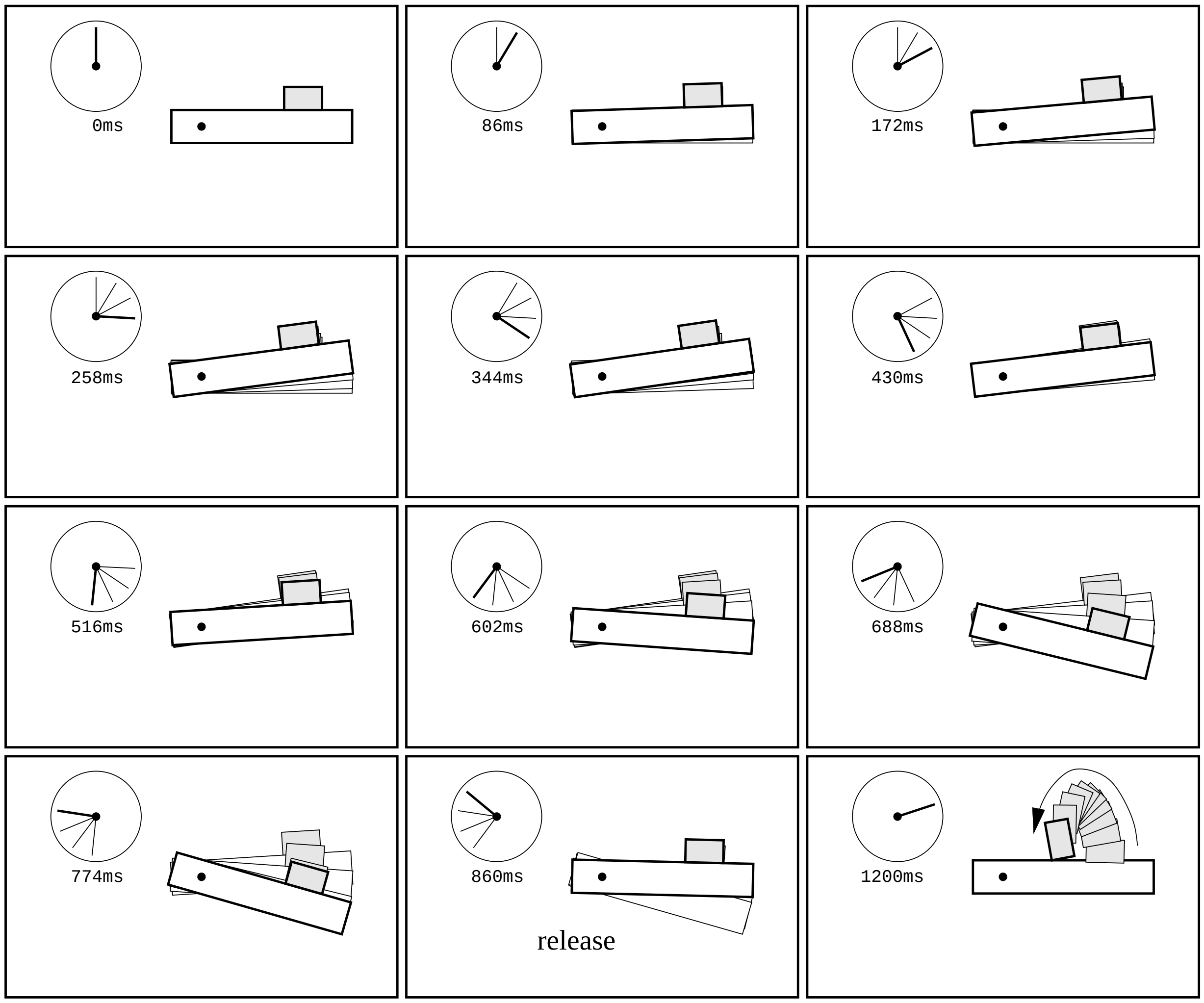
$$r_x\dot{y} - r_y\dot{x} - \dot{\phi}\rho^2 = 0,$$

where $(\dot{x}, \dot{y}, \dot{\phi})^T$ is the object velocity at impact, and (r_x, r_y) is the vector from the center of mass of the object to the impact vertex, expressed in the world frame. This constraint ensures that if the impulse causes the impact vertex to remain fixed, then the impulse will immediately cancel all velocity of the object.

Other issues in catching should be addressed, such as how to choose an impact vertex that causes the object to fall into the desired line contact.

Throw and Catch Example The throw and catch in this example is the same as that in throw example 2. Instead of specifying the goal configuration of the object, however, I specified the goal position for the impact vertex and a sweet spot constraint. The nine-knot trajectory shown in Figures 11.27 and 11.28 was obtained after 192 iterations and 63 seconds. The friction coefficient is 0.440. At impact, the block has overrotated to 1.75 radians to cancel its pre-impact velocity. The block then falls into a static grasp.

Despite small throwing errors in the implementation, the sweet spot effect was very apparent; the block seemed to come to a standstill before falling onto the desired face. The implementation on the robot is shown in Figure 11.29.



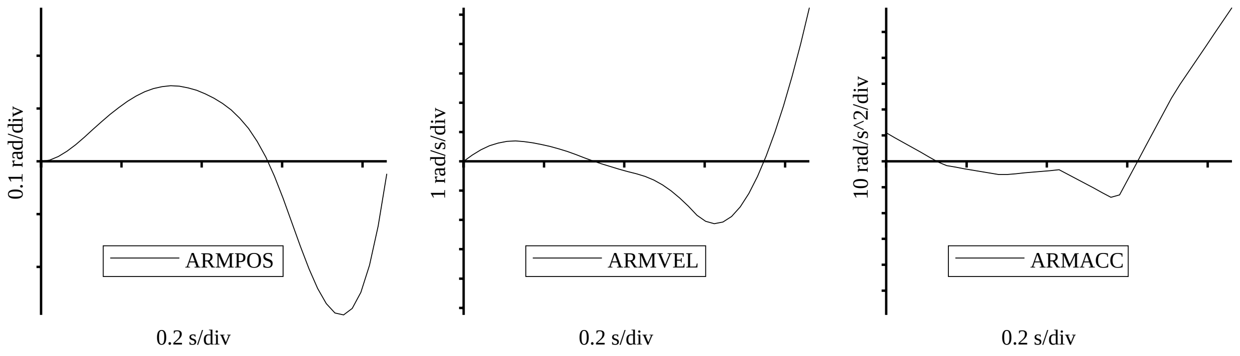


Figure 11.27: Throwing and catching at the sweet spot: the trajectory found by the optimization. Friction is 0.440.

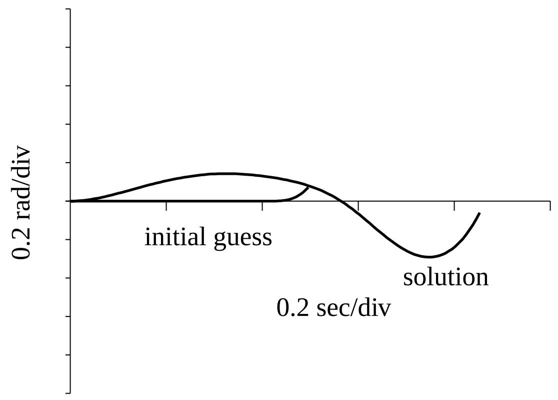


Figure 11.28: Throwing and catching at the sweet spot: the initial trajectory guess and the solution.

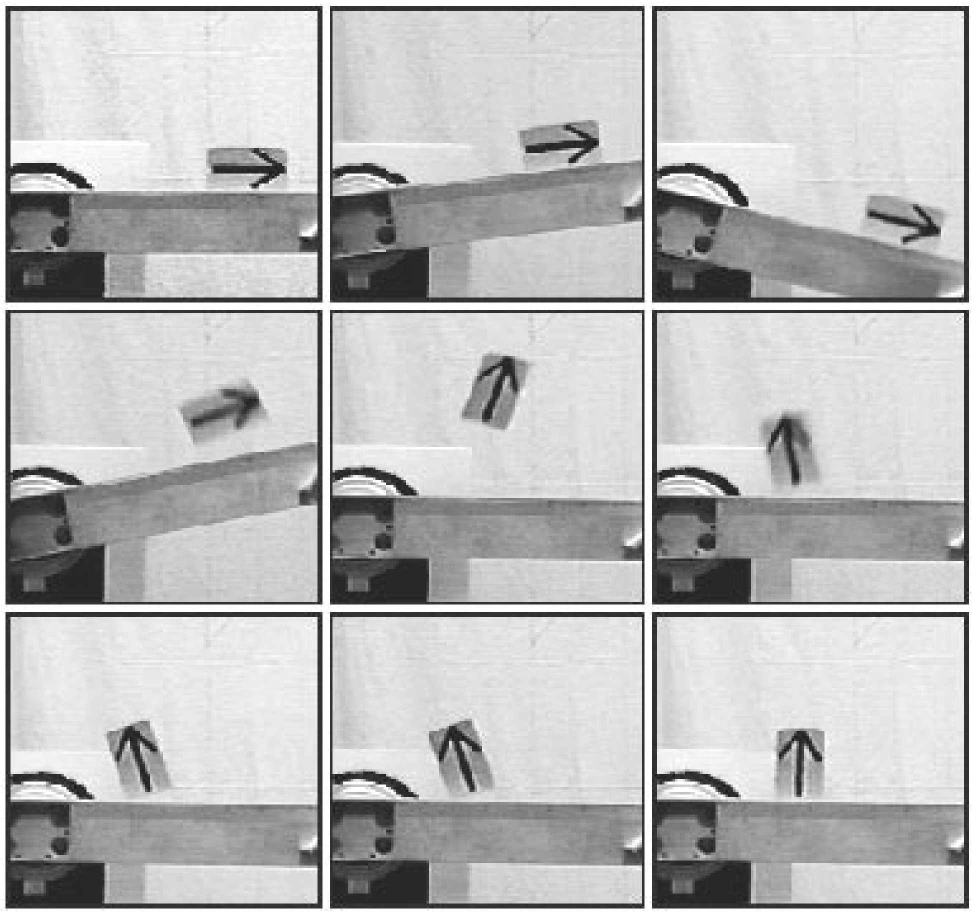


Figure 11.29: Throwing and catching at the sweet spot: a video record of the implementation on the robot.

11.2.5 Rolling and Throwing

If the object is allowed to roll before the release, in principle up to $2n+3$ independent goal state constraints can be specified, where the extra three freedoms come from the rolling state and the time of flight. For the one-degree-of-freedom arm, up to five constraints can be specified, leaving a one-dimensional goal manifold. In practice, however, the accessible state space may be rather "thin." In the three examples that follow, rolling is used before the throw to obtain a CW angular velocity during flight, which is impossible without a roll for an upward throw on the portion of the arm $x > 0$. All of the examples use the cube B.

The objective function in each of these examples was the friction coefficient. In each case, the required friction coefficient was approximately 1.

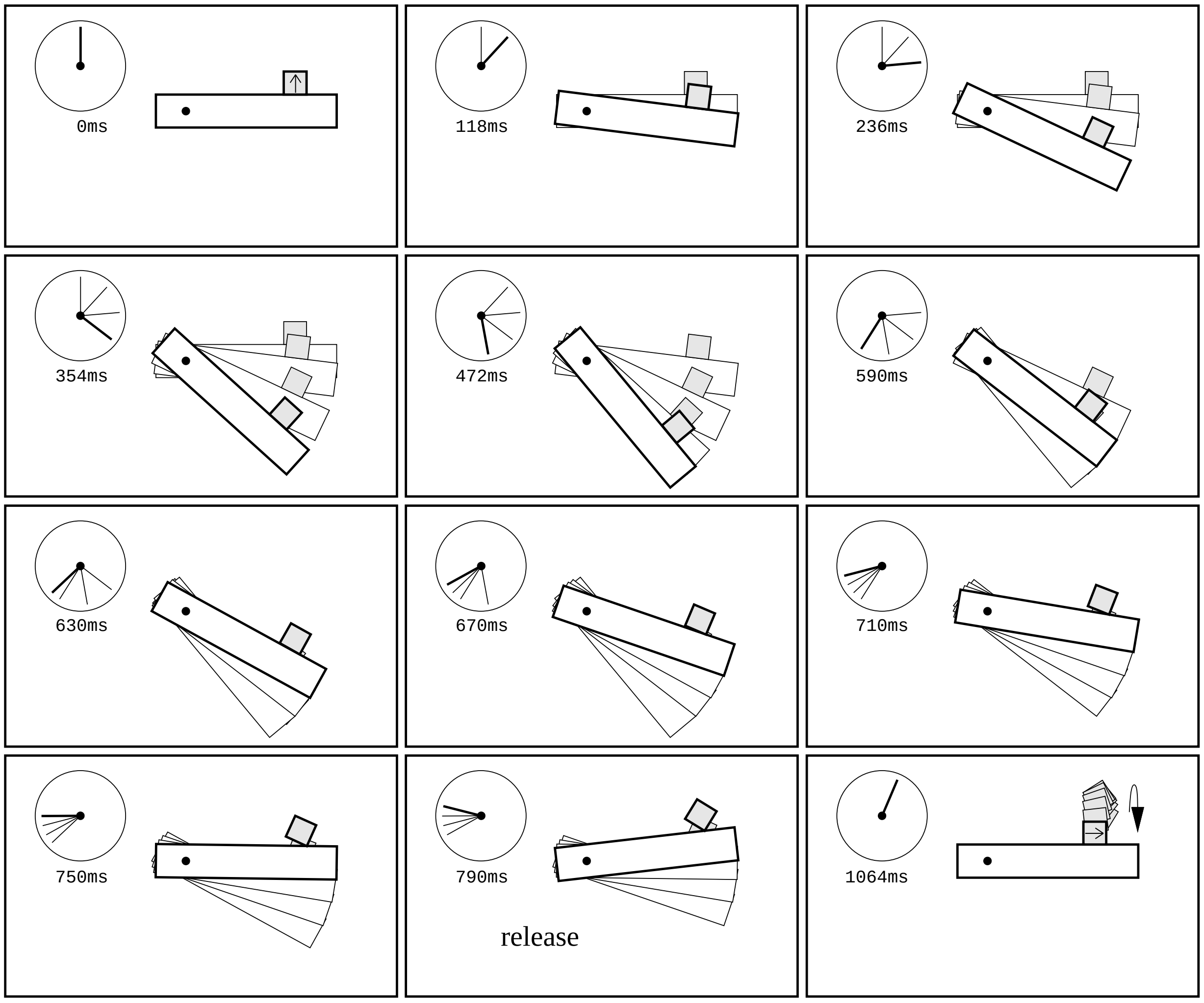
Implementing rolling and throwing trajectories is a bit of a challenge, because there is no convergence in the mechanics. Errors during the roll cannot be corrected. In particular, the slight compression of the foam introduced errors not accounted for in the planner. To roughly account for this rounding effect, I adjusted the effective rolling corner slightly. The resulting throws appear repeatable, but the accuracy of a rolling throw is generally less than that of a throw.

Rolling Throw Example 1 The objective is to rotate the object 1/4 revolution CW in place, where the far (pivot) vertex is initially at $x = 40$ cm. The nine-knot trajectory shown in Figures 11.30 and 11.31 was obtained after 54 iterations and 40 seconds. Notice that the effective centrifugal force, combined with gravity, causes the object to begin to roll out, not simply gravity when the arm is at its nadir. Ten experiments (Figure 11.32) with this throw yielded an average final position of the far vertex at $x = 39.3$ cm, with all final positions within 0.3 cm.

Rolling Throw Example 2 This example is similar to the previous, except the goal is to rotate the object 1/2 revolution CW. The nine-knot trajectory shown in Figures 11.33 and 11.34 is found after 54 iterations and 35 seconds. The implementation on the robot is shown in Figure 11.35.

Rolling Throw Example 3 The goal in this example is to throw the object to straddling the $x = 10$ cm mark with a 1/4 revolution CW. I found empirically that it was necessary to underrotate the object to counteract its large x velocity on impact. When I tested this throw ten times, the result was an indentation in the foam where the edge impacted. The impact caused some rebound and settling, sometimes giving out of plane motions.

The optimization took 50 iterations and 37 seconds to find the nine-knot trajectory shown in Figures 11.36 and 11.37. The implementation is shown in Figure 11.38.



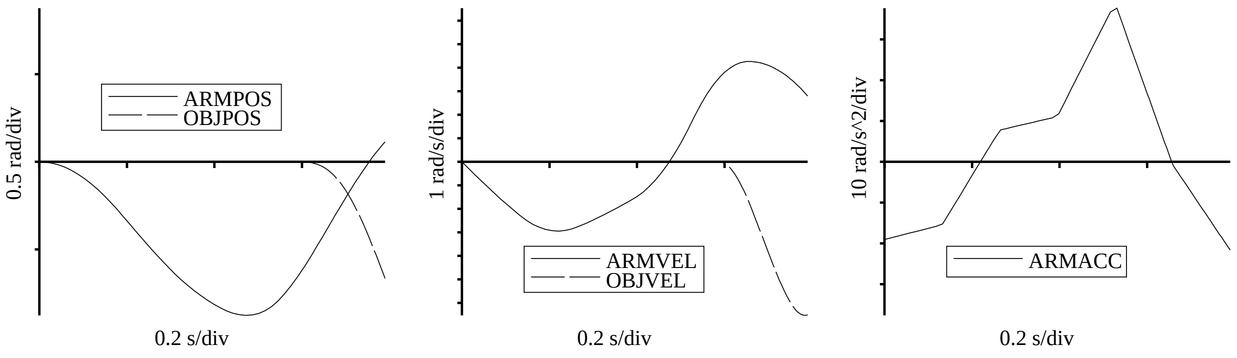


Figure 11.30: Rolling throw example 1: the trajectory found by the optimization.

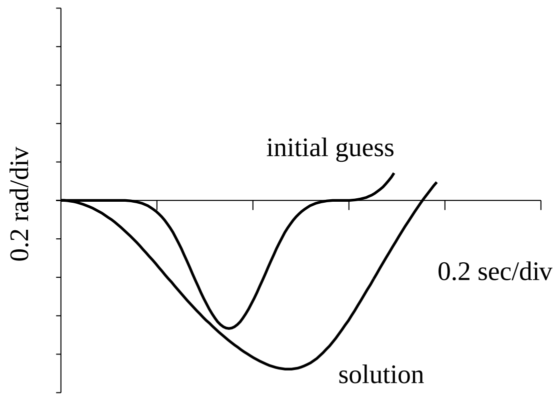


Figure 11.31: Rolling throw example 1: the initial trajectory guess and the solution.

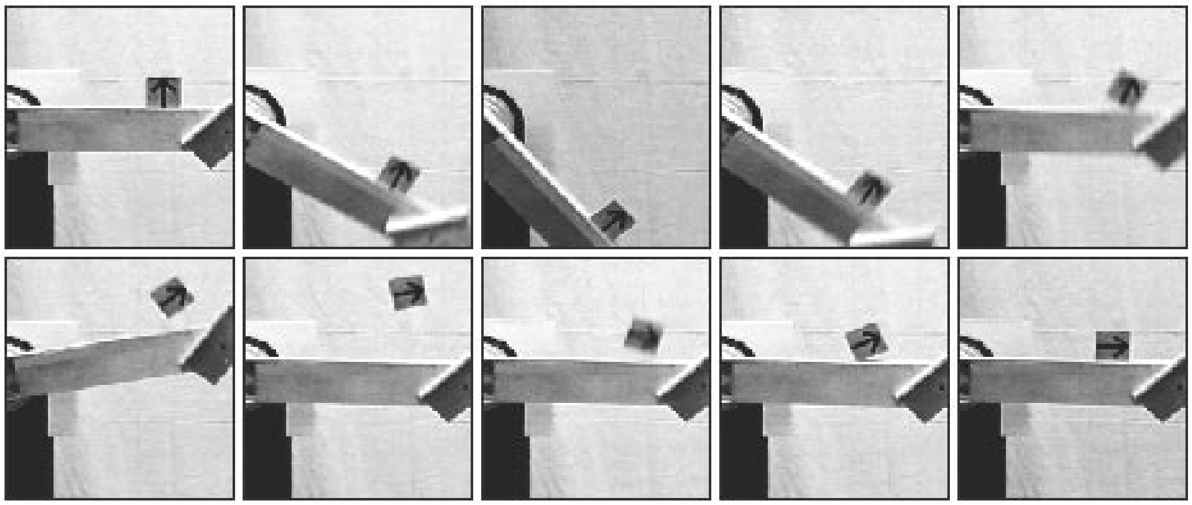


Figure 11.32: Rolling throw example 1: a video record of the implementation on the robot.

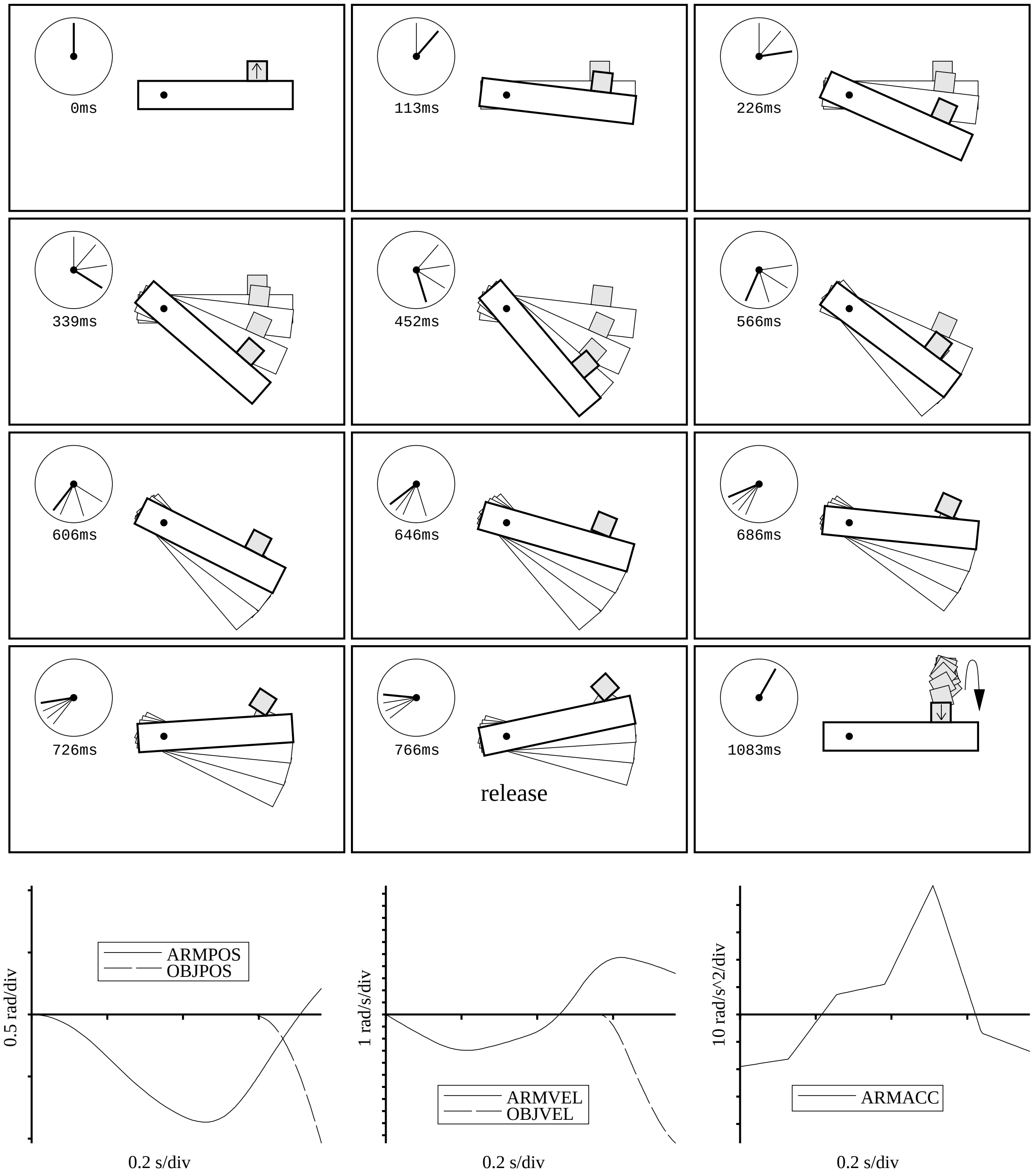


Figure 11.33: Rolling throw example 2: the trajectory found by the optimization.

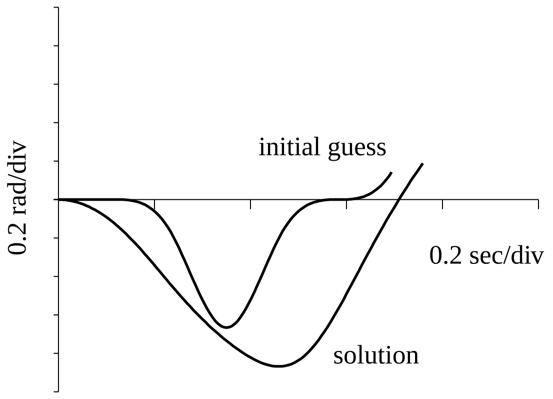


Figure 11.34: Rolling throw example 2: the initial trajectory guess and the solution.

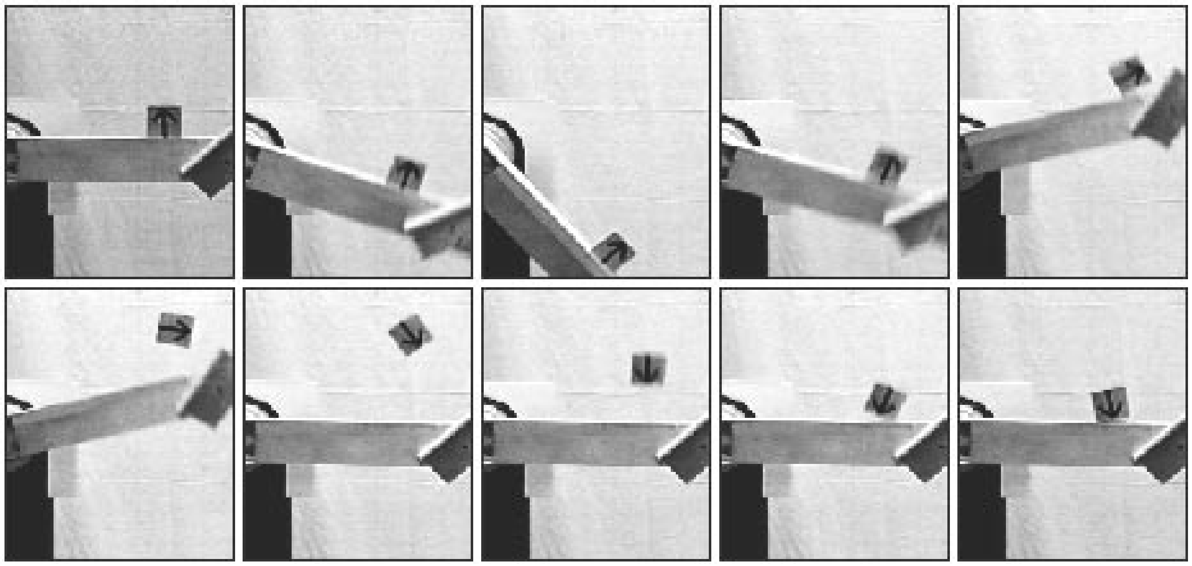
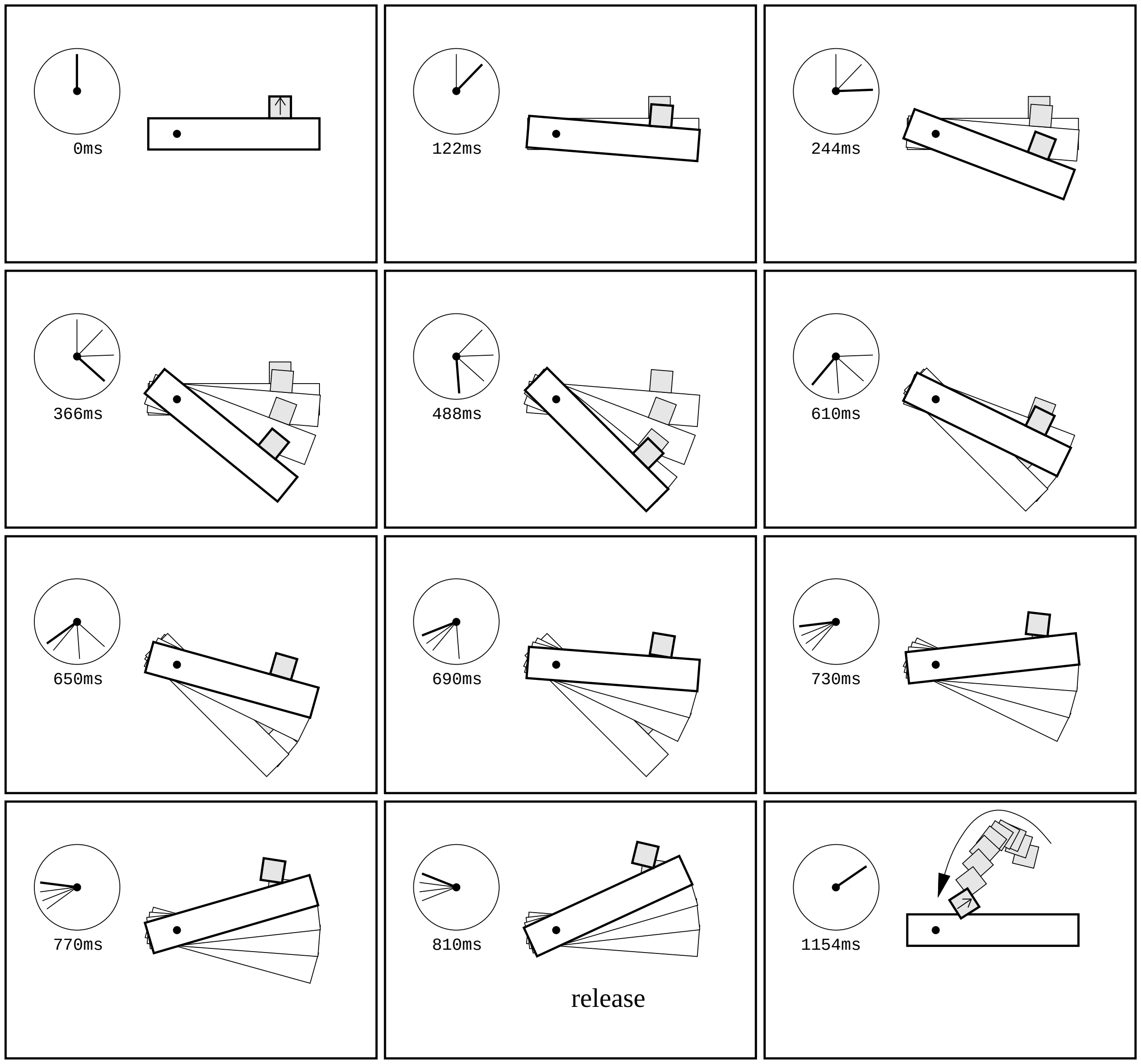


Figure 11.35: Rolling throw example 2: a video record of the implementation on the robot.



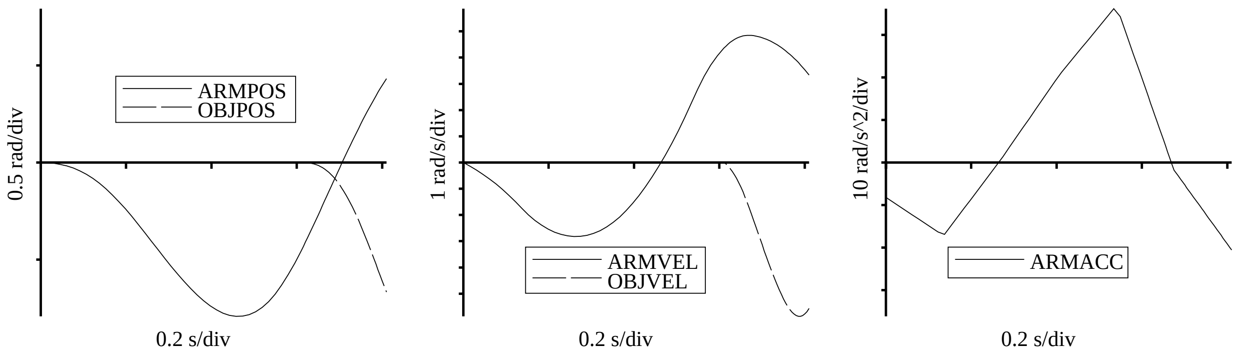


Figure 11.36: Rolling throw example 3: the trajectory found by the optimization.

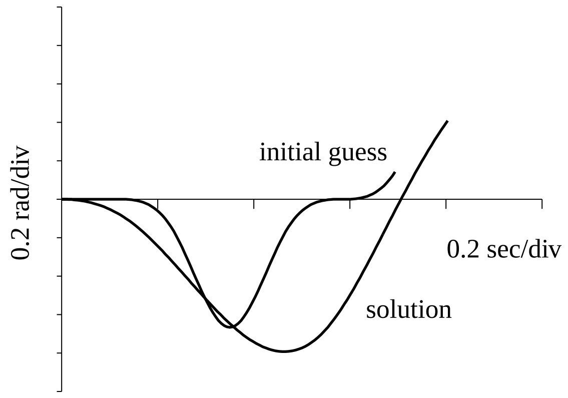


Figure 11.37: Rolling throw example 3: the initial trajectory guess and the solution.

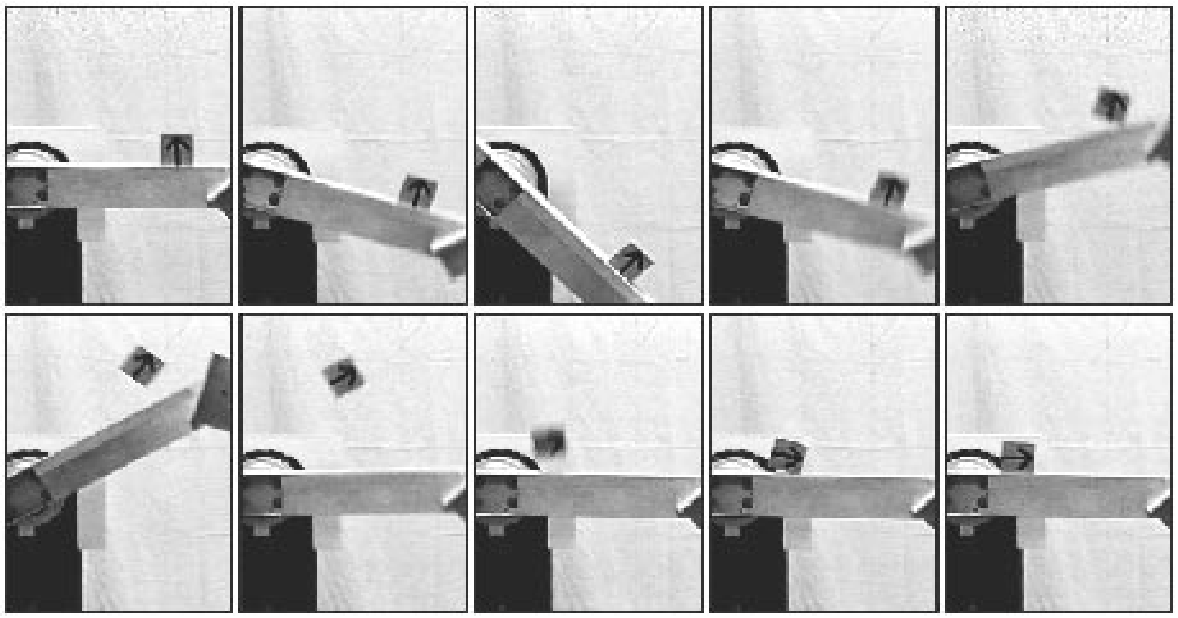


Figure 11.38: Rolling throw example 3: a video record of the implementation on the robot.

11.2.6 Flipping with Feedback

Inspired by previous work on robotic juggling (Aboaf *et al.* [3]; Bühler and Koditschek [46]; Rizzi and Koditschek [175, 176]; Sakaguchi *et al.* [179, 178]; Schaal and Atkeson [185]), I decided to try a different form of juggling: repeatedly flipping a block on the arm. The cube B was used. The goal of each throw was to rotate the block 1/4 revolution CCW and land with the far support vertex at $x = 40$ cm.

Five different throwing motions were found by the optimization, all with a goal position with the far vertex at $x = 40$ cm, but with starting positions at 35 cm, 37.5 cm, 40 cm, 42.5 cm, and 45 cm. After each flip, the current position of the block was measured, and the plan with the nearest starting position was used. (I provided the feedback by eyeballing the object's position.)

After 9 runs, the average number of successful consecutive flips was 15, with a low of 4 and a high of 32. In all but two runs, the cause for failure was the block falling off the arm due to accumulation of out-of-plane motions after impact. Two times the block was flipped and continued on to the next face.

In contrast, flipping the block open-loop rarely succeeds more than twice in a row due to accumulation of error on the order of a couple centimeters. And performance with a coarser resolution for the feedback (using only the throws starting from 35 cm, 40 cm, and 45 cm) is not much more successful. Apparently the throws are very sensitive to the initial configuration.

Chapter 12

Discussion

The experiments of Chapter 11 demonstrate the feasibility of dynamic nonprehensile manipulation. By carefully calibrating the response of the motor, it is possible to automatically generate solutions to dynamic tasks that can be downloaded directly to the arm. There are several limitations, however, and more work to be done.

Gradient descent for motion planning SQP is a general method for handling trajectory planning problems, simultaneously handling linear and nonlinear kinematic, dynamic, and friction constraints. It is, however, inherently a local method: it performs well near the goal, but not necessarily at points far away from the goal. It is better suited as a finishing step that "tweaks" a rough plan.

The results reported in this thesis were obtained by manually giving the optimization an initial guess. Often the initial guess was sufficient to find a solution. If the optimization failed to converge, I provided it with another guess. In some cases, even after a few tries, it failed to find a solution. (As discussed below, it is generally impossible to know if the specified goal state is even feasible.) One possibility to partially automate this process would be to have a layer on top of the optimization which provides it with random guesses or guesses chosen from a grid of points in the design variable space. Another way to assist the optimization is to first make the number of constraint checks coarse, and provide the solution to this simpler problem as the initial guess to a problem with constraint checks at a finer resolution.

In short, SQP as a motion planner suffers from the same local problems inherent to any potential field method. (Notable exceptions are the navigation functions constructed by Koditschek [108] and Rimon and Koditschek [174], which are guaranteed to be almost globally convergent based on prior knowledge of the topology of the space.) Randomly changing the iterate in the optimization is similar to the random jumps used to escape potential field minima in randomized path planning (Latombe [115]).

Objective function Recognizing that friction between the arm and the object is limited and possibly varying, I chose to minimize the required friction for most of the examples.

The friction coefficient may only be critical at a single point of the trajectory, however. A hybrid energy-friction objective function could be used to choose between minimum-friction trajectories.

Accessible state space The geometry of the accessible state space is a very complex function of gravity, actuator constraints, and friction. In general, answering the question "Is the goal state reachable?" appears to be as difficult as the problem of finding a manipulator trajectory that takes the object to the goal state. In my experiments, I was guided by intuition.

Uncertainty Because of the nonholonomic nature of the manipulator/object system, there is no smooth feedback control law that will take the system to the goal state (Brockett [39]). For this reason, this thesis has studied open-loop controls. A serious limitation of the open-loop approach presented here is that there is no robustness to uncertainties and errors during the execution. Ideally there would be some feedback component to reject errors during execution of the feedforward control specified by the optimization. There are two ways to address this problem: active feedback control and passively convergent mechanics.

Active feedback has previously been used to control an unactuated joint (Arai and Tachi [13]; Arai and Tachi [12]; Arai *et al.* [14]; Bergerman *et al.* [23]) using dynamic coupling with an actuated joint. This is similar to controlling rolling motion of an object. Straightforward application of active feedback is difficult, however, since the object is usually not instrumented to provide angular feedback. Vision feedback is impractical in a gravity field, because rolls typically last only a few frames. Another possibility is to use force feedback to estimate the state of the object being manipulated. This would require a precise dynamic model of the system and very accurate force information.

An even more interesting approach is to design the trajectory so that the object's motion is passively convergent. The stable push of Part I is an example of passively convergent mechanics: any pushing motion in the interior of $\hat{\mathcal{V}}_{stable}$ is also stable for nearby pushing motions. In the same way, a dynamic grasp is usually stable for a set of accelerations with nonempty interior. A dynamic grasp may be made robust to uncertainty in the location of the object's center of mass, its radius of gyration, the contact segment, and errors in the arm's acceleration by shrinking the dynamic grasp accelerations.

The same cannot be said for rolling motions. Whereas a dynamic grasp places inequality constraints on the motion of the manipulator, the rolling motion of an object is governed by equations including the geometry of the object, the mass parameters of the object, and the motion of the manipulator. Any uncertainty in any of these will result in uncertainty in the rolling motion.

On the other hand, if we allow nonprehensile contacts other than line contacts and fixed point contacts, it might be possible to design rolling and sliding motions that are convergent. This raises the question of the tradeoff between accessibility and robustness. In short, passively convergent mechanics tends to "prefer" a trajectory to others, perhaps hindering the accessibility of the total system. The key is to retain the accessibility of the system while choosing a trajectory that is passively convergent.

Computational complexity There appears to be two competing approaches to trajectory planning problems: direct search and iterative gradient-based optimization. Kinodynamic planning is an example of the former. Kinodynamic planning has the advantage that solutions are guaranteed to be close to optimal. It suffers from high computational complexity, however, and I am not aware of any implementations for robots with more than two degrees-of-freedom (Xavier [220]).

Iterative gradient-based approaches take advantage of smoothness to obtain directional information to guide the search for a solution. This information can potentially lead to relatively fast searching of high-dimensional spaces. Unless the space and the objective function has special structure (i.e., convexity), however, these methods are doomed to find local optima.

Testbed Many of the trajectories reported in this thesis require large friction coefficients. As we add more joints to the robot, it is better able to control the angle of the contact normal, reducing the required friction.

The motor used in the experiments is a high torque direct-drive motor. Large accelerations are needed for nonprehensile manipulation in a strong gravity field. Garth Zeglin and Matt Mason are developing a planar dynamic manipulation testbed, dubbed "flatland," where the manipulators and objects float on an air table. The gravity vector is adjustable by adjusting the tilt angle of the air table. By making gravity small, the motors can be scaled down, resulting in cheap, easy-to-build manipulators. The time scale of rolling and throwing manipulation increases, making it easier to observe dynamic behaviors and debug problems with the system. At these low speeds, vision feedback becomes feasible.

To extend the trajectory planner to three-dimensional problems, only the dynamic grasp and rolling friction constraints need to be modified. The spatial friction cones must be approximated by friction pyramids, so the dynamic grasp and rolling constraints are no longer exact.

Motion planning for underactuated systems When the one-degree-of-freedom arm rolls an object from one face to another, it acts as a two-degree-of-freedom arm with an unactuated joint. Viewed this way, the system is extremely rich and flexible, because the location of that "joint" on the arm can be changed by simply moving the object. (Of course there is no built-in encoder for this joint as there is for most underactuated arms.) We need only constrain the manipulator motion so this virtual joint acts as a real joint: friction constraints must be satisfied at all times.

Although this thesis focuses on nonprehensile manipulation, the approach to trajectory planning is general enough to be applicable to other underactuated manipulators and nonholonomic systems in general. If we can utilize the structure of a particular system, however, we may be able to find more efficient solution techniques. Future work on dynamic nonprehensile manipulation should focus on understanding the structure of the accessible state space in an effort to derive more efficient planning algorithms.

Chapter 13

Conclusion

13.1 Contributions

This thesis has demonstrated that even simple robots can perform interesting tasks by exploiting mechanics. Some of the main results are:

- A two-degree-of-freedom robot can push almost any object arbitrarily closely along any path in its three-dimensional configuration space.
- A one-degree-of-freedom robot, operating above a fixed-speed conveyor, can move any planar polygonal part from a random initial configuration to a specific goal configuration.
- Considering dynamics, a one-degree-of-freedom robot (described by two state variables) can take a planar object to a full six-dimensional subset of its state space.

To find manipulator controls that take the object to the goal state, this thesis has presented

- a procedure that finds stable pushing motions for line pushing contact,
- a planner that uses stable pushing motions to find pushing paths among obstacles,
- a planner for feeding polygonal parts on a conveyor by pushing with a rotating fence, and
- a dynamic manipulation planner that solves problems such as snatching an object, rolling an object, and throwing and catching.

Finally, to demonstrate the feasibility of nonprehensile manipulation, the plans have been implemented on real robot systems.

This work represents a first step toward understanding the capabilities of simple robots using nonprehensile manipulation. Future work in nonprehensile manipulation should address uncertainty and feedback; contacts other than point and line contact; spatial objects;

tractable models for analyzing dynamic systems with repeated, complex contacts; and applications to industrial automation and space.

13.2 Closing Thought

A model of the mechanics of a task is a resource for a robot, just as actuators and sensors are resources. The effective use of frictional, gravitational, and dynamic forces can substitute for extra actuators; the expectation derived from a good model can minimize sensing requirements.

As the mechanics model becomes more detailed, however, it becomes more challenging to assess the capabilities of a robot. The set of accessible configurations of the manipulated object is no longer just the set of configurations the end-effector can reach. A question of great interest is "Given a robot and a task, is there a valid control input that accomplishes the task?" Unfortunately, there is little hope of answering such a question in a general way. We must instead confine our inquiry to particular strategies and incomplete models of the world. This thesis has explored such questions for nonprehensile manipulation with quasistatic and dynamic models, and it has proposed solutions to the attendant control synthesis problems which have been successfully implemented on real robot systems.

Appendix A

Pushing with Point Contact

To determine bounds on the possible quasistatic motions of a pushed object, we need models describing the friction between the object and the support plane and between the pusher and the object. Models of support and pushing friction include the following:

1. Support friction
    (a) A geometric model of the region of possible contact between the object and the support.
    (b) A known centroid of support friction.
    (c) A completely specified support friction distribution.
2. Pushing contact friction
    (a) An unknown coefficient of friction.
    (b) A known coefficient of friction:
        i. Zero friction.
        ii. Nonzero friction.
        iii. Perfectly rough (no slip).[1]

To get useful results for robotic manipulation planning, this thesis uses a known center of friction for the pushed object. This appendix addresses techniques for bounding the motion of a pushed object when the center of friction is not known.

Mason [139] studied several models for support and pushing friction, and he utilized theoretical results for case (1.b) to synthesize robot manipulation plans. Building on this work, Mason and Brost [142] and Peshkin and Sanderson [162] studied case (1.b), and Goyal *et al.* [84] investigated case (1.c). Alexander and Maddocks [9] argue that cases

[1]Note that this is not equivalent to an infinite friction contact, because there may be slip even if the friction coefficient is infinite (Lynch and Mason [132]).

(1.b) and (1.c) require information that may not be available or assumptions that may not be met. They present new results for cases (1.a, 2.b.i) and (1.a, 2.b.iii), which require only a geometric model of the possible support region. For an object pushed at a single point with zero friction, Alexander and Maddocks describe the set of all possible resulting motions of the object. This set is described by the possible locations of the object's instantaneous center of rotation (COR). A modification of the analysis allows treatment of the case where the pushing contact is perfectly rough.

Alexander and Maddocks find the motion of a pushed object by minimizing a dissipation functional describing the power dissipated by support friction. This appendix describes a simple reinterpretation of the circle criteria of their Propositions 4 and 15 (derived by a power minimization analysis) in terms of force-moment balance. This interpretation may aid in understanding these circle criteria. I use this interpretation to find the set of all possible CORs for case (1.a, 2.a). (Extension to the case (1.a, 2.b.iii) is trivial.) This represents the weakest of the listed models of single-point pushing: only the location of the pushing contact and the region of possible contact between the object and the support are known.

If the pusher applies a force to the sliding object through a point $\mathbf{p}$ and causes the object to rotate around a COR at a point $\mathbf{c}$, I define the $\mathbf{pc}$-*circle* to be the circle whose diameter is defined by the points $\mathbf{p}$ and $\mathbf{c}$. (The points $\mathbf{p}$ and $\mathbf{c}$ can never be coincident.) The friction force opposing the motion of the object at each point of support on the $\mathbf{pc}$-circle makes zero moment about $\mathbf{p}$. If the sense of rotation of the object is clockwise (respectively counterclockwise), all support points inside the $\mathbf{pc}$-circle contribute negative (resp. positive) moment about $\mathbf{p}$, and all support points outside the circle contribute positive (resp. negative) moment about $\mathbf{p}$. (See Figure A.1.) For quasistatic moment balance, the moment about $\mathbf{p}$ due to support friction must sum to zero. Therefore, a point $\mathbf{c}$ is a possible COR if and only if the $\mathbf{pc}$-circle intersects $\mathcal{B}$, the region of contact between the object and the support.

For the case of a unipod at $\mathbf{y}$, the set of valid CORs $\mathcal{J}(\mathbf{y})$ is the line through $\mathbf{y}$ and perpendicular to the line through $\mathbf{p}$ and $\mathbf{y}$. This line includes a point at infinity ($|\mathbf{c}| = \infty$), which corresponds to translation. Any COR $\mathbf{c}$ chosen on this line forms a $\mathbf{pc}$-circle which passes through $\mathbf{y}$. If $\mathbf{p} = \mathbf{y}$, $\mathcal{J}(\mathbf{y})$ is the entire extended plane (including all translations), excluding $\mathbf{p}$.

For a line segment of support $\mathcal{S}$, the valid COR set $\mathcal{J}(\mathcal{S})$ is the union of $\mathcal{J}(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{S}$. This set has a simple closed-form expression. Without loss of generality, assume that the line segment $\mathcal{S}$ between $\mathbf{y}$ and $\mathbf{z}$ is an interval $[y_1, z_1]$ on the x_1-axis, with a pushing point $\mathbf{p}$ on the x_2-axis at $(0, p_2)$, $p_2 \geq 0$. If $p_2 = 0$ and $y_1 \leq 0 \leq z_1$, the valid COR set is the entire extended plane, excluding $\mathbf{p}$. If $p_2 = 0$ and $y_1 > 0$ or $z_1 < 0$, the valid COR set is the infinite strip $y_1 \leq c_1 \leq z_1$. Otherwise the valid COR set is all points $\mathbf{c} = (c_1, c_2)$ that simultaneously satisfy the following constraints:

$$c_2 \geq \begin{cases} z_1(c_1 - z_1)/p_2 & \text{if } c_1 \leq y_1 + z_1 \\ y_1(c_1 - y_1)/p_2 & \text{if } c_1 > y_1 + z_1 \end{cases}$$

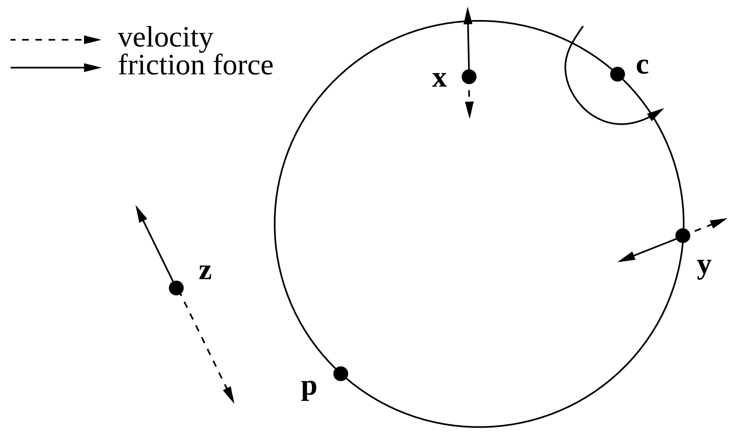


Figure A.1: A **pc**-circle for a counterclockwise COR **c** and three support points, **x**, **y**, and **z**. Support points inside the circle (e.g., **x**) contribute positive moment about **p**, points on the circle (e.g., **y**) contribute zero moment, and points outside the circle (e.g., **z**) contribute negative moment.

and

$$c_2 \le \begin{cases} y_1(c_1 - y_1)/p_2 & \text{if } c_1 \le 2y_1 \\ c_1^2/4p_2 & \text{if } 2y_1 < c_1 \le 2z_1 \\ z_1(c_1 - z_1)/p_2 & \text{if } c_1 > 2z_1. \end{cases}$$

If the first constraint on c_2 is violated, the **pc**-circle completely encloses $\mathcal{S}$. If the second constraint is violated, the **pc**-circle completely excludes $\mathcal{S}$. The parabolic portion of the boundary results from choices of **c** such that the **pc**-circle is tangent to the line segment. The rest of the boundary is defined by $\mathcal{L}(\mathbf{y})$ and $\mathcal{L}(\mathbf{z})$. Figure A.2 shows examples of the three cases described above.

These constructions can be used to find the valid COR set for any polygonal support region $\mathcal{B}$ and pushing point **p**. If $\mathbf{p} \in \mathcal{B}$, $\mathcal{J}(\mathcal{B})$ is the entire extended plane, excluding **p**. If $\mathbf{p} \notin \mathcal{B}$, $\mathcal{J}(\mathcal{B})$ is the union of the valid COR sets for each edge, since the **pc**-circle must intersect an edge for **c** to be a valid COR. Examples are given in Figure A.3.

One case of particular interest is a straight edge pushing a polygonal object at a vertex, as in a fence-pushing alignment operation. If **p** is on the boundary of $\mathcal{B}$ and the support region is the closed set $\mathcal{B}$, then any COR (excluding **p**) is possible. If, however, the support region is restricted to the open set which is the interior of $\mathcal{B}$, some of these CORs can be eliminated. Consider a horizontal pushing edge with $\mathcal{B}$ wholly above the edge. To find the

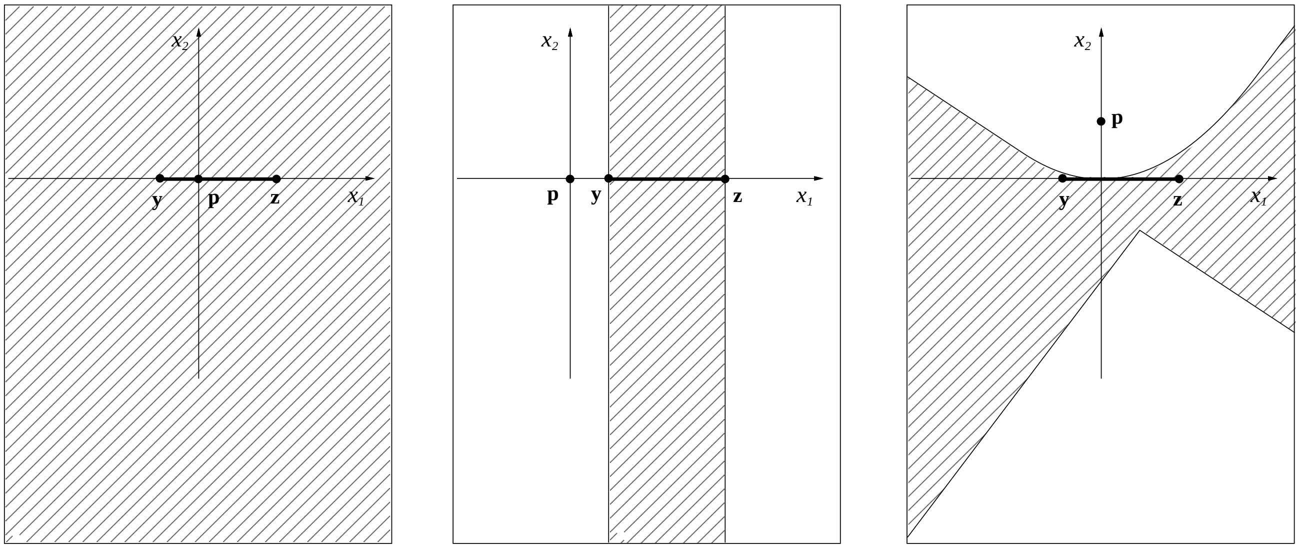


Figure A.2: COR sets (hatched) for line segments.

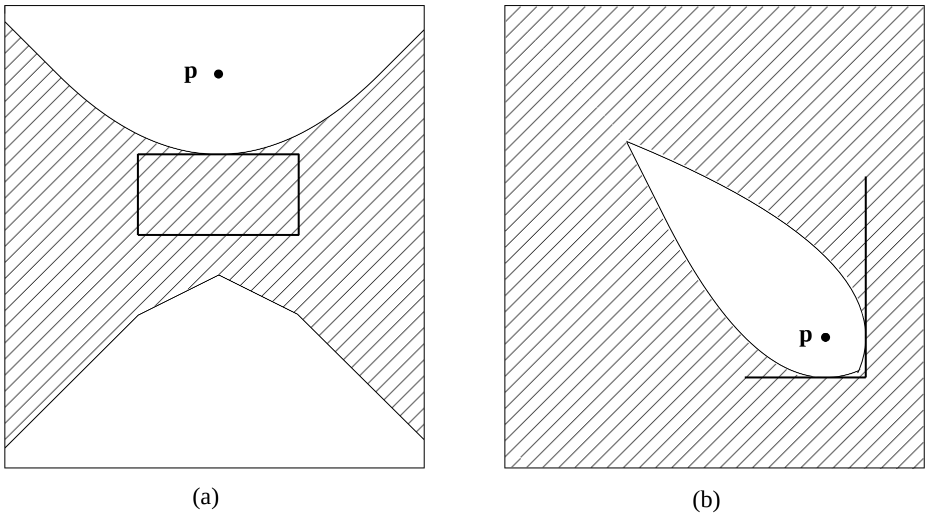


Figure A.3: COR sets (hatched) for (a) a rectangle and (b) an Allen wrench. If the pushing point is on or inside the convex hull of the support region, as in (b), all translation directions ($|\mathbf{c}| = \infty$) are included in the COR set.

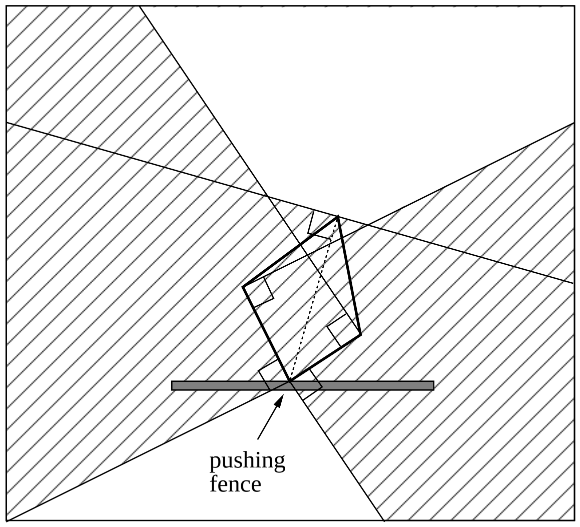


Figure A.4: COR set (hatched) for a quadrilateral pushed by a fence at a vertex. The possible support region is the interior of the polygon, and the COR set does not include its bounding lines.

valid COR set,

1. Find the convex hull of the polygon $\mathcal{B}$.

2. Draw two lines through $\mathbf{p}$, one perpendicular to each edge adjacent to $\mathbf{p}$. The COR must lie above at least one of these lines (otherwise the $\mathbf{pc}$-circle completely excludes $\mathcal{B}$).

3. At all other vertices on the convex hull, draw a line perpendicular to the line connecting that vertex to $\mathbf{p}$. The COR must lie below at least one of these lines (otherwise the $\mathbf{pc}$-circle completely encloses $\mathcal{B}$).

An example is shown in Figure A.4.

All of these bounds are exact, in the sense that any COR $\mathbf{c}$ (with either rotation sense) in the derived set could occur for some support friction distribution in $\mathcal{B}$ and some applied force through $\mathbf{p}$. By kinematic arguments, CORs along the line of pushing are impossible, and CORs to the left (respectively right) of the pushing line must be counterclockwise (resp. clockwise). If there is no slip at the pushing contact (case (1.a, 2.b.iii)), the valid COR set is the set found above intersected with a line perpendicular to the pushing direction.

The method described above for finding the COR set for a perfectly rough contact is exactly equivalent to Alexander and Maddocks' circle criterion for a perfectly rough pusher (Proposition 15, case (1.a, 2.b.iii)). They demonstrate the circle criterion by showing that the dissipation functional cannot be minimal unless the $\mathbf{pc}$-circle intersects $\mathcal{B}$.

With a little more work, the circle criterion for bipods (Proposition 4) can be seen to be a special case of the $\mathbf{pc}$-circle. The circle C described by Alexander and Maddocks for

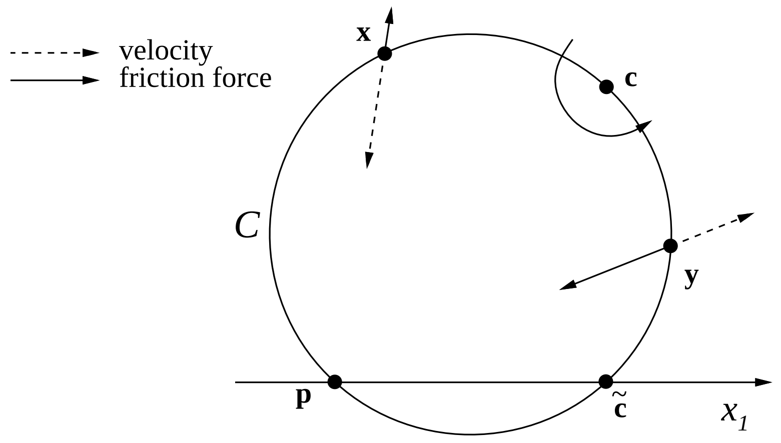


Figure A.5: The circle C for a bipod with supports at $\mathbf{x}$ and $\mathbf{y}$, a counterclockwise COR $\mathbf{c}$, and a pushing force along the x_1-axis.

the case of the frictionless pusher (1.a, 2.b.i) must intersect the line of pushing force (the x_1-axis) at either one point (if C is tangent to the x_1-axis) or two points. (See Figure A.5.) One point of intersection is denoted $\tilde{\mathbf{c}} = (c_1, 0)$, and the other point is diametrically opposed to the COR $\mathbf{c} = (c_1, c_2)$. In fact, this other point of intersection is equivalent to the pushing point $\mathbf{p}$, and C is equivalent to the $\mathbf{pc}$-circle. The friction forces from support points $\mathbf{x}$ and $\mathbf{y}$, which lie on C but are distinct from $\mathbf{c}$, each pass through $\mathbf{p}$, thus creating zero moment about $\mathbf{p}$. The moment balance condition is therefore satisfied. In order for the total force to lie along the x_1-axis, however, the points $\mathbf{x}$ and $\mathbf{y}$ must lie on opposite sides of the chord $\overline{\mathbf{c}\tilde{\mathbf{c}}}$. Only in this case can we choose friction force magnitudes for $\mathbf{x}$ and $\mathbf{y}$ to achieve a force balance. Thus the circle criterion of Proposition 4 is explained simply using force-moment balance and the $\mathbf{pc}$-circle.

The point $\tilde{\mathbf{c}}$ of the circle C is equivalent to the "effective center of friction" (Lynch [129]). In other words, for the COR $\mathbf{c}$, the bipod distribution acts like a single point at $\tilde{\mathbf{c}}$: the total support frictional force acts through $\tilde{\mathbf{c}}$ in a direction opposite its motion.

Appendix B

Modeling and Controlling the NSK Arm

Precise control of the acceleration of our one-degree-of-freedom manipulator is needed for dynamic manipulation. Conventional PD servoing along a trajectory may be insufficient, as the acceleration of the motor along the path may vary greatly. For this reason, I use feedforward control, where the torque trajectory is calculated offline. This requires a precise model of the response of the motor, but ensures a smooth torque (and therefore acceleration) profile. In practice, I implement a feedback servo around this feedforward control, but the feedback corrections are small. I am also interested in updating the feedforward signal ("learning" control), so that a desired acceleration history can be followed more closely by practicing the motion a few times.

B.1 Setup

The one-degree-of-freedom arm is driven by a direct-drive motor donated by NSK. The motor is a 0608 Megatorque variable reluctance motor, model type `R S 0608 F N 001 E A 05 C`, running on 220 V. The control computer is a 386 PC. A serial line to the RS-232C port of the motor driver chooses the operating mode for the motor. The quadrature resolver feedback of 153,600 counts/rev is fed into a decoder board on the PC. Analog commands (usually in torque mode) in the ± 10 V range are sent to the driver by a D/A card on the PC. The analog command may also be issued by a dial. The motor operates in the vertical plane, mounted about 1.22 m above the floor. The load in the experiments is a 1.22 m aluminum beam weighing 2.64 kg, with an inertia of 0.331 kg-m^2 when mounted in the center and 0.708 kg-m^2 when mounted at one end. According to the literature, the rotor inertia of the motor is 0.0075 kg-m^2, making the total inertia 0.3385 and 0.7155 kg-m^2, respectively. The servo rate is 1 kHz.

B.2 Model

The equation I have chosen to model the one-degree-of-freedom arm is

$$f(V_{command}, \dot{\theta}) = a\,\mathrm{sgn}(\dot{\theta}) + b\,\dot{\theta} + c\sin(\theta) + d\,\ddot{\theta}.$$

Each side of the equation is an actual torque. $V_{command}$ is the voltage (in analog torque mode) sent to the driver, and $f(V_{command}, \dot{\theta})$ is the function which converts this voltage to a torque for the current speed. a is the static coefficient of friction and b is the viscous coefficient of friction. It is assumed that friction increases linearly with the motor speed. c is the gravity component of the torque, and it is zero when the load is balanced. d is the inertia of the load.

From inspection of the speed-torque curve included with the NSK literature, it appears that the torque available is fairly constant at low speeds, and decreases approximately linearly at higher speeds. According to the manual, the torque delivered at a given velocity is a linear function of the voltage applied, with max torque available at that velocity corresponding to the max voltage. Thus, a reasonable model of the speed-torque curve would seem to be given by three parameters: (1) the maximum torque available, (2) the speed at which the available torque begins to drop (the "knee"), and (3) the slope of this drop.

B.3 Identifying the Speed-Torque Relationship

I devised a simple test for calculating the speed-torque curve for our motor. (I did not use a dynamometer.) I first used velocity mode to give the motor a negative bias velocity. Then, in torque mode, a constant positive voltage $V_{command}$ is applied until the motor reached approximately 3 revs/s. Then zero voltage is briefly applied followed by $-V_{command}$ until the motor reached approximately -3 revs/s. Finally, zero voltage followed by $V_{command}$ is applied until the motor again began spinning in the positive direction. I filtered the resulting position data to get velocity and acceleration, and using the known inertia (0.3385 kg-m^2 in the balanced position), I calculated the torque due to acceleration during the motion.

I plotted the data as the torque magnitude vs. velocity. Because of friction, I expect some hysteresis in the curve. Assuming the torque available is independent of the direction of motion, the difference in the torque magnitude as the motor accelerates and decelerates at the same velocity should be twice the amount of friction torque at that speed. Thus the actual torque available at a given speed should lie halfway between the values found for that speed. Figure B.1 shows what the plots should are expected to look like.

B.3.1 Effect of Filtering

After each run, I used simple, acausal, central difference filters to calculate the velocity and acceleration from the position data. To reject noise, I also performed low-pass filtering. The

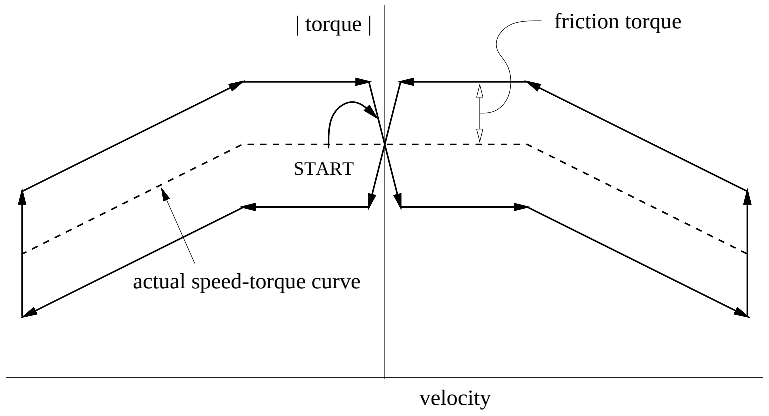


Figure B.1: The shape of the expected speed-torque curve with static friction and $b = 0$ (no viscous friction). Note that the hysteresis transition near the torque axis should actually lie exactly on the torque axis; it is exaggerated here.

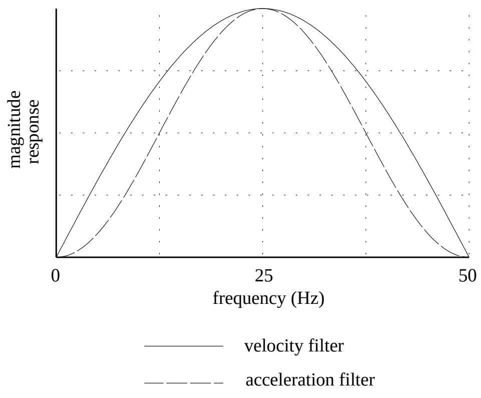


Figure B.2: The magnitude response of digital filters used to calculate velocity and acceleration.

filters are given by

$$\dot{\theta}(kT) = \frac{\theta((k+n)T) - \theta((k-n)T)}{2nT}$$

$$\ddot{\theta}(kT) = \frac{\theta((k+2n)T) - 2\theta(kT) + \theta((k-2n)T)}{(2nT)^2},$$

where T is the sample period (1 ms) and n is a nonzero positive integer. I set n to 10 to reject high frequency noise. The magnitude responses of the filters are shown in Figure B.2. Notice that the Nyquist frequency is 50 Hz for $n = 10$, and the acceleration filter is just the velocity filter convolved with itself (double differentiation with low pass).

To understand the effect of the filter, I filtered ideal simulated hysteresis data. The result is shown in Figure B.3. The low-pass effect of the filters demonstrates itself in the smoothness of the acceleration change at zero velocity. This effect is exaggerated at higher accelerations,

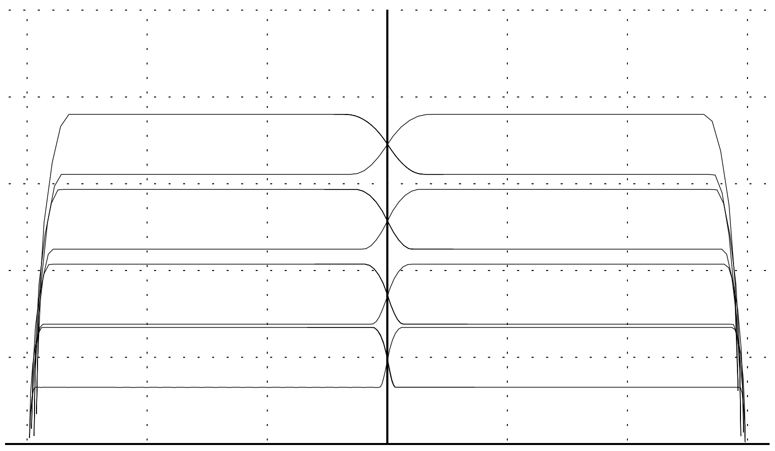


Figure B.3: Ideal data after filtering.

since there are fewer data points near zero velocity. Also notice that the filtered acceleration drops near zero when the actual acceleration reverses.

B.3.2 Results

Figure B.4 shows some speed-torque curves obtained at different commanded voltages. Figure B.5 shows ten runs using the maximum voltage (±10 V) superimposed on top of each other. These speed-torque curves are fairly repeatable, except for behavior near zero velocity (probably due to friction) and glitches around ±2 revs/s. I am not certain of the reason for the behavior near ±2 revs/s.

Some general observations:

- The curves are symmetric about the zero velocity axis and the crossover in the hysteresis takes place near the zero velocity axis, as expected.

- The shapes of the curves at different commanded voltages are not similar (compare 3 V to 10 V, for instance), which was unexpected.

- The "knee" of the speed-torque curve appears to be pushed out to higher velocities as the torque decreases. This suggests that the torque delivered is not actually a linear function of the torque available and the commanded voltage.

- The amount of hysteresis increases at lower commanded voltages (compare the the 10 V curve against the 3 V curve). This could indicate that more torque is available to slow down the motor than to speed it up at any given velocity.

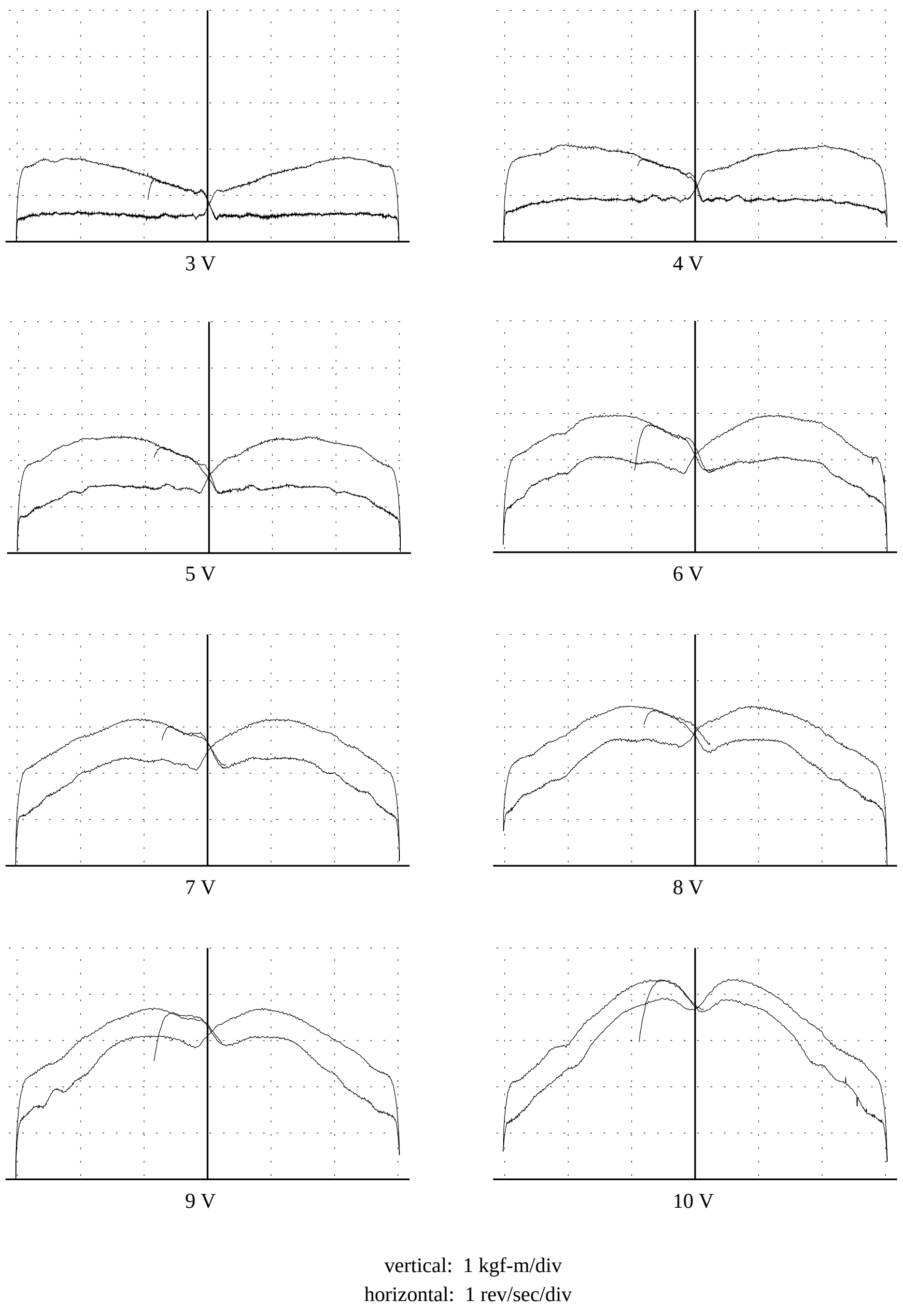


Figure B.4: Example speed-torque curves obtained for several different analog torque commands.

	0.00	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	
0.0	0.00	1.61	3.44	4.56	5.73	6.87	8.42	9.59	10.00	10.00	3.70
0.5	0.00	1.89	3.51	4.71	5.90	6.94	8.50	9.62	10.00	10.00	3.70
1.0	0.00	1.92	3.50	4.74	5.89	7.12	8.59	9.56	10.00	10.00	3.78
1.5	0.00	1.85	3.45	4.62	5.80	7.19	8.60	9.49	10.00	10.00	3.87
2.0	0.00	1.82	3.40	4.55	5.69	7.08	8.50	9.41	10.00	10.00	3.97
2.5	0.00	1.82	3.35	4.51	5.60	6.94	8.33	9.35	9.94	10.00	4.05
3.0	0.00	1.82	3.32	4.50	5.52	6.77	8.15	9.32	9.91	10.00	4.08
3.5	0.00	1.79	3.30	4.42	5.43	6.66	8.00	9.26	9.89	10.00	4.09
4.0	0.00	1.69	3.19	4.30	5.28	6.53	7.92	9.23	9.89	10.00	4.08
4.5	0.00	1.67	3.17	4.28	5.25	6.41	7.86	9.20	9.93	10.00	4.05
5.0	0.00	1.69	3.13	4.31	5.25	6.34	7.78	9.20	9.95	10.00	4.03
5.5	0.00	1.67	3.10	4.28	5.22	6.31	7.80	9.19	10.00	10.00	4.00
6.0	0.00	1.61	3.05	4.21	5.17	6.23	7.77	9.22	10.00	10.00	3.97
6.5	0.00	1.59	2.97	4.19	5.13	6.18	7.79	9.26	10.00	10.00	3.92
7.0	0.00	1.54	2.92	4.19	5.11	6.12	7.84	9.29	10.00	10.00	3.86
7.5	0.00	1.47	2.86	4.13	5.08	6.08	7.84	9.37	10.00	10.00	3.79
8.0	0.00	1.47	2.84	4.13	5.08	6.04	7.93	9.50	10.00	10.00	3.71
8.5	0.00	1.45	2.79	4.15	5.11	6.00	8.00	9.67	10.00	10.00	3.63
9.0	0.00	1.43	2.77	4.13	5.09	6.09	8.14	9.94	10.00	10.00	3.52
9.5	0.00	1.39	2.76	4.10	5.08	6.14	8.47	10.00	10.00	10.00	3.43
10.0	0.00	1.37	2.71	4.08	5.06	6.28	8.88	10.00	10.00	10.00	3.31
10.5	0.00	1.35	2.67	4.08	5.09	6.47	9.22	10.00	10.00	10.00	3.18
11.0	0.00	1.33	2.61	4.08	5.13	6.75	9.71	10.00	10.00	10.00	3.06
11.5	0.00	1.32	2.60	4.09	5.17	7.19	10.00	10.00	10.00	10.00	2.94
12.0	0.00	1.28	2.55	4.09	5.21	7.50	10.00	10.00	10.00	10.00	2.87
12.5	0.00	1.30	2.55	4.05	5.29	8.00	10.00	10.00	10.00	10.00	2.84
13.0	0.00	1.25	2.50	4.06	5.37	8.63	10.00	10.00	10.00	10.00	2.71
13.5	0.00	1.22	2.46	4.11	5.49	9.25	10.00	10.00	10.00	10.00	2.59
14.0	0.00	1.20	2.45	4.21	5.67	10.00	10.00	10.00	10.00	10.00	2.49
14.5	0.00	1.19	2.44	4.22	5.85	10.00	10.00	10.00	10.00	10.00	2.45
15.0	0.00	1.16	2.42	4.24	6.50	10.00	10.00	10.00	10.00	10.00	2.40
15.5	0.00	1.18	2.45	4.38	7.00	10.00	10.00	10.00	10.00	10.00	2.31
16.0	0.00	1.19	2.48	4.58	7.56	10.00	10.00	10.00	10.00	10.00	2.17

Table B.1: The derived motor model. The rows correspond to current velocity magnitude in radians/s, and the columns correspond to desired torque magnitude in kgf-m. The entries are the necessary voltage command magnitudes, in volts. The rightmost column is the maximum torque available at that speed.

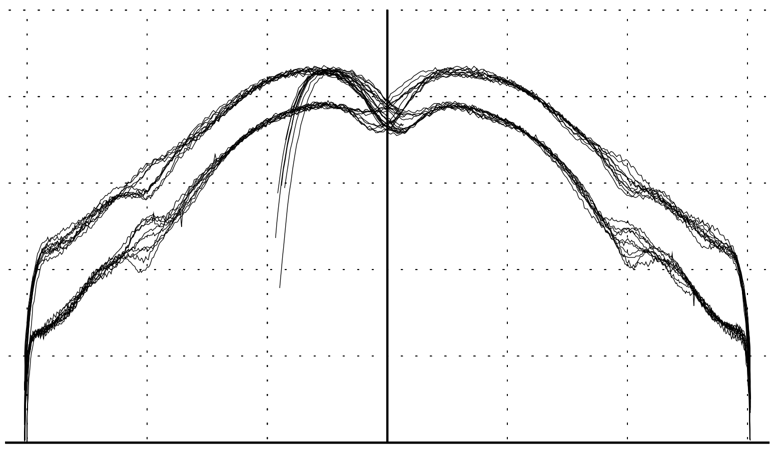


Figure B.5: The speed-torque curves from ten runs using maximum voltage (±10 V) superimposed on top of each other.

- It appears from the curves that the amount of torque available from the motor is not maximized at zero velocity. In fact, for the 3 V case, it appears that the torque is maximized at high velocity.
- I found that the analog output of the D/A card was being slightly loaded by the NSK driver, so I buffered it.
- The constant positive voltage is first applied while the motor is spinning with negative angular velocity. Looking at the data, the motor appears to take a little time to reach the acceleration curve, presumably due to response delay.

The lesson learned from this experiment is that the actual motor torque is a complex function of the voltage signal and its current velocity. I don't know how the driver CPU calculates the commutation of the motor, so I don't know what other factors (besides current velocity and voltage command) may be important in the model.

I chose to model the driver and motor essentially as a black box. Using the speed-torque curves obtained in Figure B.4, I derived a table (Table B.1) which is indexed by the current velocity magnitude and the desired torque magnitude. The entry at that position is the voltage magnitude necessary to obtain that torque magnitude. Simple interpolation is used to obtain command voltages at speed and torque values not on the grid. The resolution of the table is 0.5 rad/s along the speed direction, and 0.5 kgf-m along the torque direction. (It assumes, however, that the actual torque applied by the motor is independent of the direction the motor is spinning; we can see that this is not quite true by the speed-torque curves of Figure B.4.)

For the rest of this appendix, commands will be written in terms of torque in kgf-m. The actual voltage command is found using the table.

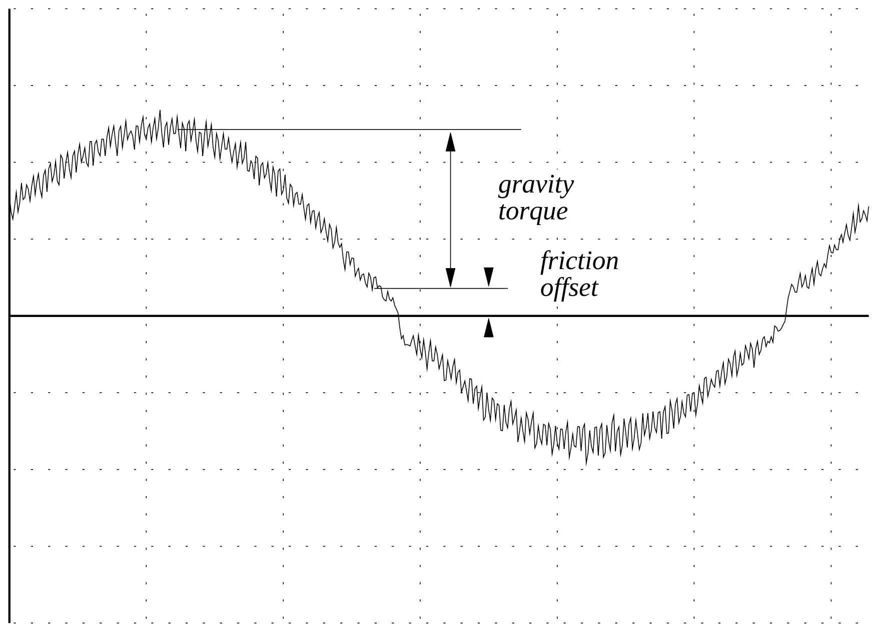


Figure B.6: The torque command for one revolution of the unbalanced load at a velocity of 0.5 radian/s.

B.4 Estimating Friction and Gravity Terms

Following a procedure outlined by Murakami and Onishi [153], I implemented a constant velocity servo to estimate the friction terms a and b and the gravity term c. With a balanced load, the torque necessary to maintain a constant velocity is exactly the friction torque at that speed. With an unbalanced load, the torque is the friction torque plus the torque to hold the load statically at the current configuration. The gravity component c is the magnitude of the sine curve the torque describes as the load moves through a full revolution (see Figure B.6).

With the load in both the balanced and unbalanced positions, I spin the load at a range of different speeds to determine the friction torque at those speeds. To calculate static friction, the commanded torque is increased until the arm begins to move. The results are shown in Figure B.7 and Table B.2.

Under the model of friction, a is measured to be approximately 0.23 kgf-m, and b is between 0.01 and 0.02 kgf-m-s/rad. The gravity component c is measured to be about 1.0 kgf-m when the load is in the unbalanced configuration. This coincides with the expected value of c, which is (2.64 kgf)(0.378 m) = 1.0 kgf-m, where 0.378 m is the distance from the center of the bolt circle to the center of mass of the beam.

velocity magnitude (rad/s)	*balanced*		*unbalanced*	
	neg torque (kgf-m)	*pos torque (kgf-m)*	*neg torque (kgf-m)*	*pos torque (kgf-m)*
0.0	-0.24	0.23	-0.23	0.23
0.1	-0.24	0.23	-0.16	0.16
0.2	-0.23	0.23	-0.17	0.15
0.3	-0.24	0.23	-0.20	0.17
0.4	-0.24	0.23	-0.20	0.19
0.5	-0.22	0.22	-0.21	0.19
0.6	-0.21	0.21	-0.24	0.21
0.7	-0.22	0.21	-0.24	0.23
0.8	-0.22	0.21	-0.24	0.24
0.9	-0.22	0.21	-0.24	0.24
1.0	-0.21	0.21	-0.25	0.24
1.5	-0.22	0.22	-0.25	0.25
2.0	-0.23	0.22	-0.28	0.27
2.5	-0.24	0.23	-0.30	0.28
3.0	-0.24	0.24	-0.31	0.30
3.5	-0.24	0.24	-0.32	0.32
4.0	-0.25	0.25	-0.30	0.31
4.5	-0.26	0.26	-0.30	0.32
5.0	-0.28	0.27	-0.36	0.36

Table B.2: Measured friction torque.

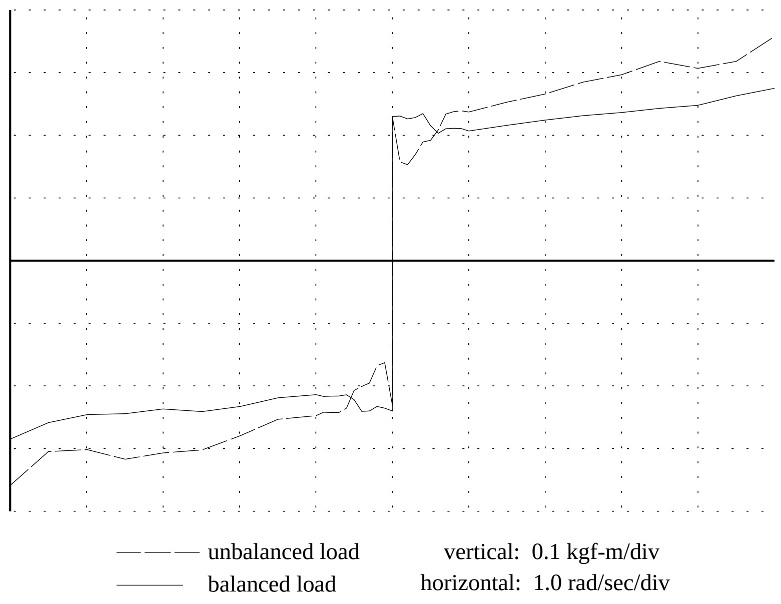


Figure B.7: The friction torque as a function of velocity for the balanced and unbalanced load.

B.5 Estimating the Inertial Term

I implemented a simple test to measure the inertial term. I define the zero angle for the load to be a point where the gravity term is zero (i.e., hanging straight down in the unbalanced case). The measurement routine then offsets the load slightly from zero, and accelerates through the zero point with a constant torque command. The position data is filtered to get velocity and acceleration (using the filters described earlier), and then I look at the data as the load passes through the zero angle. The torque due to friction is obtained from the friction torque vs. velocity curve and subtracted from the commanded torque. The remaining torque accelerates the load. By dividing this torque by the acceleration, we obtain the inertia.

This procedure was tested on the load in the centered configuration and in the unbalanced configuration. The test torque was ± 3 kgf-m. For the centered configuration, I obtained estimates of 0.340 and 0.328 kg-m^2 with positive and negative torque, respectively, giving an average of 0.334 kg-m^2. For the unbalanced configuration, I estimated 0.723 and 0.692 kg-m^2, for an average of 0.708 kg-m^2. These differ from the hand-measured values of 0.3385 and 0.7155 kg-m^2 by approximately 1 percent.

B.6 Learning Control

Even with a precise feedforward model, errors will accumulate during the course of a run. For this reason, I use a feedback loop around the feedforward control to provide small corrections.

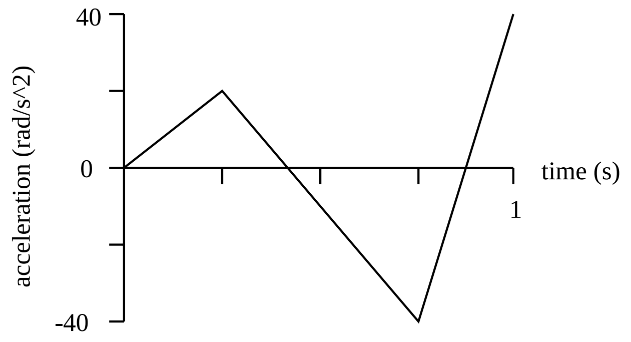


Figure B.8: The test trajectory, shown as acceleration.

I am also interested in learning control (Arimoto [15]), so trajectory following improves with practice.

I have implemented two simple learning rules. The first is to simply add the PD feedback used during the run to the feedforward control of the next run. The second is a learning rule described by Atkeson [16]. The estimated torque command error at each timestep is given by

$$\tilde{\tau}_{err} = \tilde{\tau}((\theta, \dot{\theta}, \ddot{\theta})_{measured}) - \tilde{\tau}((\theta, \dot{\theta}, \ddot{\theta})_{desired}),$$

where $\tilde{\tau}(\theta, \dot{\theta}, \ddot{\theta})$ is the torque necessary using the model of the arm. The estimated torque command error $\tilde{\tau}_{err}$ is subtracted from the previous commanded torque at this timestep.

B.7 Results

I have tested the feedforward and learning control on many different motion trajectories. The data shown here is using the 1 second trajectory of Figure B.8. The results of this experiment are shown in Figure B.9 for the centered load configuration and Figure B.10 for the unbalanced load configuration. In both cases the load starts with zero velocity, and in the latter case, it begins hanging straight down. Results are shown using Atkeson's learning rule and learning with velocity feedback. The two approaches can also be used in combination. The results are encouraging.

B.8 Questions and Lessons Learned

I don't understand how the CPU of the NSK driver computes the commutation based on the state of the motor and the analog voltage command sent to the driver. For this reason, I ended up treating the driver as a black box. With a better understanding of the commutation scheme, it might be possible to improve on the feedforward performance. For instance, what other state variables (besides motor velocity) are important? Currently, the speed-torque table assumes that if command voltage V at velocity $\dot{\theta}$ yields torque τ, then the command

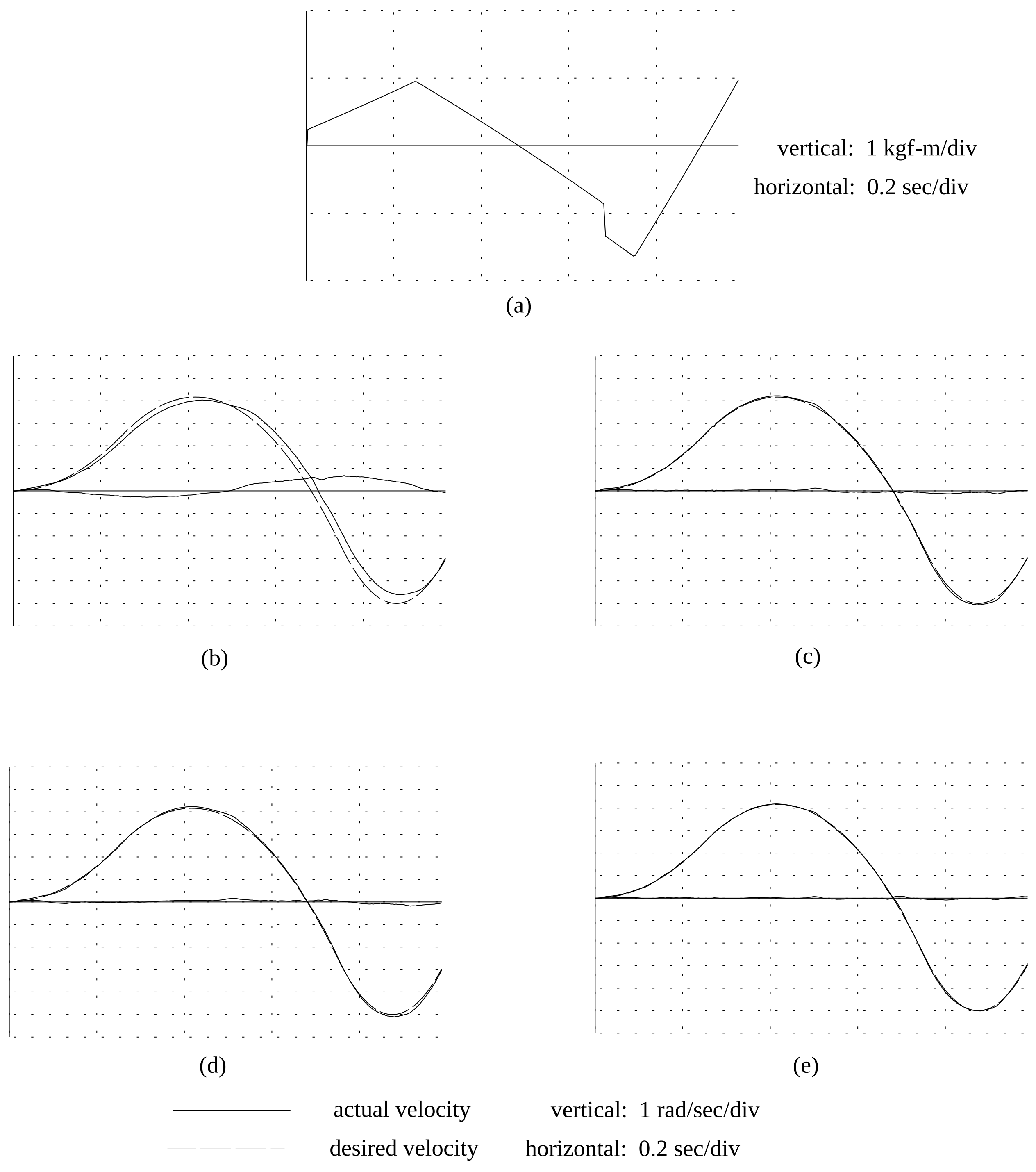


Figure B.9: The load is in the centered configuration. (a) The feedforward torque command. (b) The velocity and velocity error using feedforward control with no feedback. (c) After one iteration using Atkeson's learning rule (again no feedback). (d) The first run using velocity feedback of -1 kgf-m-s/rad. (e) The second run with velocity feedback. The feedforward command has been modified by the velocity feedback of the previous run.

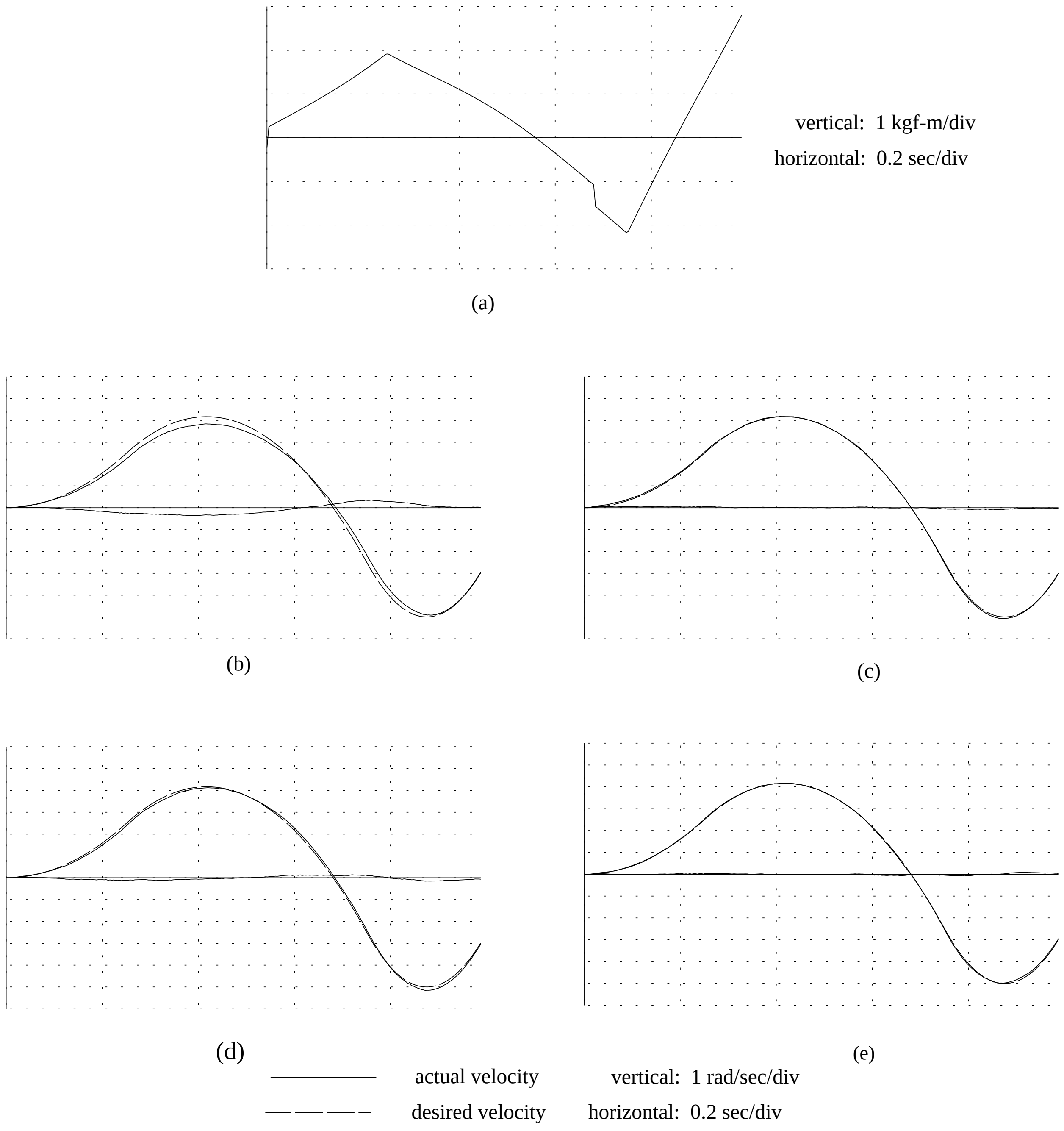


Figure B.10: The load is in the unbalanced configuration. (a) The feedforward torque command. (b) The velocity and velocity error using feedforward control with no feedback. (c) After one iteration using Atkeson's learning rule (again no feedback). (d) The first run using velocity feedback of -1 kgf-m-s/rad. (e) The fourth run with velocity feedback. The feedforward command has been modified by the velocity feedback of the three previous runs.

voltage $-V$ at the same speed would yield torque $-\tau$. The speed-torque curves show that this may not be quite accurate. There may be more torque to slow down than to speed up.

The model of friction in the bearing is a common friction model, but it uses no information about the motor's particular bearing.

I experimented with the higher resolver feedback setting (614,400 counts/rev). According to the manual, this reduces the maximum velocity in velocity mode to 1 rev/s. This setting also reduces the torque available at each speed in torque mode, however. I did not expect this.

For real-time velocity feedback, I generally used finite-differencing of the position signal over a 1 or 2 ms interval. I considered using the velocity monitor output of the driver (obtained by differencing by the driver CPU), but this signal was noisier than that obtained using my own differencing scheme.

I discovered that there is still some torque available at speeds of 5 or 6 rev/s.

B.9 Replies from NSK

On May 31, 1995, Matt Mason, Mike Erdmann, and I went to visit NSK in Takasaki, Japan, where we had the opportunity to talk to NSK engineers. The following are some of their responses to our questions. Our discussion was only about an hour, so the answers are not necessarily complete. Our host was Mr. Yoshiharu Yamaguchi.

"Why does the serial communication slow down while running a position or velocity servo on the driver, but not when operating the motor in torque mode?"

> Serial communication is the lowest priority task for the driver's CPU, so it slows down when the computation demand is high. In particular, minimize the number of filters (low pass and notch) for the CPU to calculate if you want to maintain the communication speed.

"Why isn't the motor torque a linear function of the voltage signal at a given speed?"

> The commutation is tuned to maximize the amount of torque available at a given speed, not to maintain the linear relationship. NSK typically only looks at the maximum voltage speed-torque curve. In addition, the commutation is tuned to maximize the accelerating torque available, while largely ignoring the decelerating torque. So the accelerating torque and decelerating torque at a given speed and voltage magnitude do not have equivalent magnitude. They also speculated that there may be regenerative torque to assist deceleration. We would expect the regeneration effect to be larger at higher speeds, but the torque difference appears to be more a function of the command voltage.

"Why is torque maximized at nonzero velocity?"

> At zero velocity, there is no torque ripple. As the speed increases, the commutation essentially uses torque ripple to increase the average torque available at that speed. So although the published speed-torque curve does not indicate this, this effect is not a surprise to them.

"Why does the maximum torque (in torque mode) decrease when we change the resolver setting to 614,400 counts/rev?"

> They did not believe this should happen, and asked me for data to prove this. I did not have any data, just my own anecdotal experience.

"Why is the velocity feedback so noisy?"

> The velocity feedback is provided as a convenience for the user, but it is not used in the driver's own velocity controller. This explains why the driver's velocity control appears much better than the velocity feedback. They acknowledge it is a relatively low quality signal, and suggest I continue to use my own finite-differencing scheme.

"What is a good model for the bearing friction?"

> They didn't have much to say about this, as they are the motor group, not the bearing group.

"How should the torque ripple be modeled?"

> They provided a drawing illustrating the torque ripple as a function of the motor's position. The shape of the torque ripple is fairly complicated. So far we have not worried much about torque ripple; most of our applications require large motions (a radian or so). Torque ripple is a greater concern when the motions are slow through a small range, where the torque variation is more noticeable.

Bibliography

[1] T. Abell and M. A. Erdmann. Stably supported rotations of a planar polygon with two frictionless contacts. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.

[2] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer. Task-level robot learning: Ball throwing. AI memo 1006, Massachusetts Institute of Technology, 1987.

[3] E. W. Aboaf, S. M. Drucker, and C. G. Atkeson. Task-level robot learning: Juggling a tennis ball more accurately. In *IEEE International Conference on Robotics and Automation*, pages 1290–1295, Scottsdale, AZ, 1989.

[4] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison Wesley, 1974.

[5] Y. Aiyama, M. Inaba, and H. Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 136–143, Yokohama, Japan, 1993.

[6] S. Akella, W. Huang, K. M. Lynch, and M. T. Mason. Planar manipulation on a conveyor with a one joint robot. In *International Symposium on Robotics Research*, 1995.

[7] S. Akella and M. T. Mason. Posing polygonal objects in the plane by pushing. In *IEEE International Conference on Robotics and Automation*, pages 2255–2262, Nice, France, 1992.

[8] R. Alami, T. Simeon, and J.-P. Laumond. A geometrical approach to planning manipulation tasks. The case of discrete placements and grasps. In *International Symposium on Robotics Research*, pages 453–459, Tokyo, Japan, Aug. 1989. Cambridge, Mass: MIT Press.

[9] J. C. Alexander and J. H. Maddocks. Bounds on the friction-dominated motion of a pushed object. *International Journal of Robotics Research*, 12(3):231–248, 1993.

[10] R. L. Andersson. Understanding and applying a robot ping-pong player's expert controller. In *IEEE International Conference on Robotics and Automation*, pages 1284–1289, Scottsdale, AZ, 1989.

[11] H. Arai and O. Khatib. Experiments with dynamic skills. In *1994 Japan–USA Symposium on Flexible Automation*, pages 81–84, 1994.

[12] H. Arai and S. Tachi. Position control of a manipulator with passive joints using dynamic coupling. *IEEE Transactions on Robotics and Automation*, 7(4):528–534, Aug. 1991.

[13] H. Arai and S. Tachi. Position control system of a two degree of freedom manipulator with a passive joint. *IEEE Transactions on Industrial Electronics*, 38(1):15–20, Feb. 1991.

[14] H. Arai, K. Tanie, and S. Tachi. Dynamic control of a manipulator with passive joints in operational space. *IEEE Transactions on Robotics and Automation*, 9(1):85–93, Feb. 1993.

[15] S. Arimoto. Learning control theory for robotic motion. *International Journal of Adaptive Control and Signal Processing*, 4:543–564, 1990.

[16] C. G. Atkeson. *Roles of Knowledge in Motor Learning.* PhD thesis, Massachusetts Institute of Technology, Sept. 1986. AI Laboratory TR 924.

[17] Z. Balorda. Reducing uncertainty of objects by robot pushing. In *IEEE International Conference on Robotics and Automation*, pages 1051–1056, Cincinnati, OH, 1990.

[18] Z. Balorda. Automatic planning of robot pushing operations. In *IEEE International Conference on Robotics and Automation*, pages 1:732–737, Atlanta, GA, 1993.

[19] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[20] J. Barraquand and J.-C. Latombe. On nonholonomic mobile robots and optimal maneuvering. In *International Symposium on Intelligent Control*, pages 340–347, 1989.

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

[22] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann, 1987.

[23] M. Bergerman, C. Lee, and Y. Xu. Experimental study of an underactuated manipulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2: 317–322, 1995.

[24] M. D. Berkemeier and R. S. Fearing. Control of a two-link robot to achieve sliding and hopping gaits. In *IEEE International Conference on Robotics and Automation*, pages 286–291, Nice, France, 1992.

[25] V. Bhatt and J. C. Koechling. Three dimensional frictional rigid body impact. Technical report, Department of Mechanical Engineering, Cornell University, 1993.

[26] A. Bicchi and R. Sorrentino. Dexterous manipulation through rolling. In *IEEE International Conference on Robotics and Automation*, pages 452–457, 1995.

[27] A. M. Bloch and P. E. Crouch. Nonholonomic control systems on riemannian manifolds. *SIAM Journal on Control and Optimization*, 33(1):126–148, 1995.

[28] A. M. Bloch and N. H. McClamroch. Control of mechanical systems with classical nonholonomic constraints. In *IEEE International Conference on Decision and Control*, pages 201–205, 1989.

[29] A. M. Bloch, M. Reyhanoglu, and N. H. McClamroch. Control and stabilization of nonholonomic dynamic systems. *IEEE Transactions on Automatic Control*, 37(11):1746–1757, 1992.

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

[31] K. Böhringer, R. Brown, B. Donald, J. Jennings, and D. Rus. Distributed robotic manipulation: Experiments in minimalism. In *International Symposium on Experimental Robotics*, 1995.

[32] K. F. Böhringer, V. Bhatt, and K. Y. Goldberg. Sensorless manipulation using transverse vibrations of a plate. In *IEEE International Conference on Robotics and Automation*, 1995.

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

[34] G. Boothroyd. *Assembly Automation and Product Design*. Marcel Dekker, 1992.

[35] G. Boothroyd, C. Poli, and L. E. Murch. *Automatic Assembly*. Marcel Dekker, 1982.

[36] F. P. Bowden and D. Tabor. *Friction: An Introduction to Tribology*. Anchor Press/Doubleday, 1973.

[37] D. L. Brock. Enhancing the dexterity of a robot hand using controlled slip. In *IEEE International Conference on Robotics and Automation*, pages 249–251, 1988.

[38] R. W. Brockett. Nonlinear systems and differential geometry. *Proceedings of the IEEE*, 64(1), Jan. 1976.

[39] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. W. Brockett, R. S. Millman, and H. J. Sussmann, editors, *Differential Geometric Control Theory*. Birkhauser, 1983.

[40] M. Brokowski, M. Peshkin, and K. Goldberg. Curved fences for part alignment. In *IEEE International Conference on Robotics and Automation*, pages 3:467–473, Atlanta, GA, 1993.

[41] R. C. Brost. Automatic grasp planning in the presence of uncertainty. *International Journal of Robotics Research*, 7(1):3–17, Feb. 1988.

[42] R. C. Brost. Computing the possible rest configurations of two interacting polygons. In *IEEE International Conference on Robotics and Automation*, Sacramento, CA, 1991.

[43] R. C. Brost. Dynamic analysis of planar manipulation tasks. In *IEEE International Conference on Robotics and Automation*, pages 2247–2254, 1992.

[44] R. C. Brost and M. T. Mason. Graphical analysis of planar rigid-body dynamics with multiple frictional contacts. In *International Symposium on Robotics Research*, pages 293–300, Tokyo, Japan, Aug. 1989. Cambridge, Mass: MIT Press.

[45] R. G. Brown and J. S. Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3: 562–568, 1995.

[46] M. Bühler and D. E. Koditschek. From stable to chaotic juggling: Theory, simulation, and experiments. In *IEEE International Conference on Robotics and Automation*, pages 1976–1981, Cincinnati, OH, 1990.

[47] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Toward a dynamical pick and place. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2: 292–297, 1995.

[48] M. Caine. The design of shape interactions using motion constraints. In *IEEE International Conference on Robotics and Automation*, 1994.

[49] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. In *ACM Symposium on Computational Geometry*, pages 271–280, Berkeley, CA, 1990.

[50] J. Canny, J. Reif, B. Donald, and P. Xavier. On the complexity of kinodynamic planning. In *IEEE Symposium on the Foundations of Computer Science*, pages 306–316, White Plains, NY, 1988.

[51] J. F. Canny and K. Y. Goldberg. "RISC" industrial robotics: Recent results and open problems. In *IEEE International Conference on Robotics and Automation*, pages 1951–1958, 1994.

[52] B. Carlisle, K. Goldberg, A. Rao, and J. Wiegley. A pivoting gripper for feeding industrial parts. In *IEEE International Conference on Robotics and Automation*, 1994.

[53] Y.-C. Chen. Solving robot trajectory planning problems with uniform cubic B-splines. *Optimal Control Applications and Methods*, 12:247–262, 1991.

[54] A. D. Christiansen. Manipulation planning from empirical backprojections. In *IEEE International Conference on Robotics and Automation*, pages 762–768, Sacramento, CA, 1990.

[55] A. D. Christiansen, A. D. Edwards, and C. A. Coello. Automated design of parts feeders using a genetic algorithm. Submitted to the 1996 IEEE International Conference on Robotics and Automation.

[56] M. F. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26:293–302, 1992.

[57] C. A. Coulomb. Théorie des machines simples en ayant égard au frottement de leurs parties et à la roideur des cordages. *Mémoires des mathématique et de physique présentés à l'Académie des Sciences*, 1781.

[58] P. E. Crouch. Spacecraft attitude control and stabilization: Applications of geometric control theory to rigid body models. *IEEE Transactions on Automatic Control*, 29(4):321–331, Apr. 1984.

[59] B. Dacre-Wright, J.-P. Laumond, and R. Alami. Motion planning for a robot and a movable object amidst polygonal obstacles. In *IEEE International Conference on Robotics and Automation*, pages 2474–2480, Nice, France, 1992.

[60] A. De Luca, L. Lanari, and G. Oriolo. A sensitivity approach to optimal spline robot trajectories. *Automatica*, 27(3):535–539, 1991.

[61] A. W. Divelbiss and J. Wen. Nonholonomic path planning with inequality constraints. In *IEEE International Conference on Decision and Control*, pages 2712–2717, 1993.

[62] G. Dodd and L. Rossol, editors. *Computer Vision and Sensor-Based Robots*. New York: Plenum Press, 1979.

[63] B. Donald and P. Xavier. A provably good approximation algorithm for optimal-time trajectory planning. In *IEEE International Conference on Robotics and Automation*, pages 958–963, Scottsdale, AZ, 1989.

[64] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for Cartesian robots and open chain manipulators. In *ACM Symposium on Compuational Geometry*, pages 290–300, Berkeley, CA, 1990.

[65] B. R. Donald, J. Jennings, and D. Rus. Information invariants for cooperating autonomous mobile robots. In *International Symposium on Robotics Research*, Hidden Valley, PA, 1993. Cambridge, Mass: MIT Press.

[66] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[67] P. E. Dupont. The effect of Coulomb friction on the existence and uniqueness of the forward dynamics problem. In *IEEE International Conference on Robotics and Automation*, pages 1442–1447, Nice, France, 1992.

[68] M. A. Erdmann. On motion planning with uncertainty. Master's thesis, Massachusetts Institute of Technology, Aug. 1984.

[69] M. A. Erdmann. A configuration space friction cone. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 455–460, Osaka, Japan, 1991.

[70] M. A. Erdmann. Multiple-point contact with friction: Computing forces and motions in configuration space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 163–170, Yokohama, Japan, 1993.

[71] M. A. Erdmann. On a representation of friction in configuration space. *International Journal of Robotics Research*, 13(3):240–271, 1994.

[72] M. A. Erdmann. An exploration of nonprehensile two-palm manipulation: Planning and execution. In *International Symposium on Robotics Research*, 1995.

[73] M. A. Erdmann. Understanding action and sensing by designing action-based sensors. *International Journal of Robotics Research*, 14(5):483–509, Oct. 1995.

[74] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics and Automation*, 4(4):369–379, Aug. 1988.

[75] M. A. Erdmann, M. T. Mason, and G. Vaněček, Jr. Mechanical parts orienting: The case of a polyhedron on a table. *Algorithmica*, 10:226–247, 1993.

[76] C. Fernandes, L. Gurvits, and Z. Li. Attitude control of a space platform/manipulator system using internal motion. *International Journal of Robotics Research*, 13(4):289–304, 1994.

[77] C. Fernandes, L. Gurvits, and Z. Li. Near-optimal nonholonomic motion planning for a system of coupled rigid bodies. *IEEE Transactions on Automatic Control*, 30(3):450–463, Mar. 1994.

[78] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the applications of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971.

[79] S. Fortune and G. Wilfong. Planning constrained motion. In *STOCS*, pages 445–457. ACM, 1988.

[80] F. Gandolfo, M. Tistarelli, and G. Sandini. Visual monitoring of robot actions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 269–275, Osaka, Japan, 1991.

[81] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, New York, 1981.

[82] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[83] K. Y. Goldberg. Completeness in robot motion planning. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 419–429. A. K. Peters, Boston, MA, 1995.

[84] S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction. Part 1. Limit surface and moment function. *Wear*, 143:307–330, 1991.

[85] S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction. Part 2. Dynamics of motion. *Wear*, 143:331–352, 1991.

[86] B. Grünbaum. *Convex Polytopes.* London: Interscience, 1967.

[87] J. Hauser and R. M. Murray. Nonlinear controllers for non-integrable systems: The acrobot example. In *American Control Conference*, pages 669–671, 1990.

[88] G. W. Haynes and H. Hermes. Nonlinear controllability via Lie theory. *SIAM Journal on Control*, 8(4):450–460, Nov. 1970.

[89] R. Hermann and A. J. Krener. Nonlinear controllability and observability. *IEEE Transactions on Automatic Control*, AC-22(5):728–740, Oct. 1977.

[90] D. T. Higdon and R. H. Cannon, Jr. On the control of unstable multiple-output mechanical systems. In *ASME Winter Annual Meeting*, 1963.

[91] T. Higuchi. Application of electromagnetic impulsive force to precise positioning tools in robot systems. In *International Symposium on Robotics Research*, pages 281–285. Cambridge, Mass: MIT Press, 1985.

[92] H. Hitakawa. Advanced parts orientation system has wide application. *Assembly Automation*, 8(3):147–150, 1988.

[93] J. K. Hodgins and M. H. Raibert. Biped gymnastics. *International Journal of Robotics Research*, 9(2):115–132, 1990.

[94] J. M. Hollerbach. Dynamic scaling of manipulator trajectories. *ASME Journal of Dynamic Systems, Measurement, and Control*, 106:102–106, 1984.

[95] J. Hong, G. Lafferiere, B. Mishra, and X. Tan. Fine manipulation with multifinger hands. In *IEEE International Conference on Robotics and Automation*, pages 1568–1573, Cincinnati, OH, 1990.

[96] W. Huang, E. P. Krotkov, and M. T. Mason. Impulsive manipulation. In *IEEE International Conference on Robotics and Automation*, 1995.

[97] M. Inaba and H. Inoue. Vision-based robot programming. In *International Symposium on Robotics Research*, Tokyo, Japan, 1989. Cambridge, Mass: MIT Press.

[98] A. Isidori. *Nonlinear Control Systems: An Introduction.* Springer-Verlag, 1985.

[99] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 2–7, Scottsdale, AZ, 1989.

[100] A. Jain and G. Rodriguez. Kinematics and dynamics of under-actuated manipulators. In *IEEE International Conference on Robotics and Automation*, pages 1754–1759, Sacramento, CA, 1991.

[101] Y.-B. Jia and M. A. Erdmann. Pose from pushing. *1996 IEEE International Conference on Robotics and Automation*, to appear.

[102] V. Jurdjevic. Certain controllability properties of analytic control systems. *SIAM Journal on Control*, 10(2):354–360, May 1972.

[103] V. Jurdjevic and H. J. Sussmann. Control systems on Lie groups. *Journal of Differential Equations*, 12:313–329, 1972.

[104] M. E. Kahn and B. Roth. The near-minimum-time control of open-loop articulated kinematic chains. *ASME Journal of Dynamic Systems, Measurement, and Control*, 93(3):164–172, Sept. 1971.

[105] D. E. Koditschek. Robot assembly: Another source of nonholonomic control problems. In *American Control Conference*, pages 1627–1632, 1991.

[106] D. E. Koditschek. Task encoding: Toward a scientific paradigm for robot planning and control. *Robotics and Autonomous Systems*, 9:5–39, 1992.

[107] D. E. Koditschek. Dynamically dexterous robots. In M. W. Spong, F. L. Lewis, and C. T. Abdallah, editors, *Robot Control: Dynamics, Motion Planning and Analysis.* New York: IEEE Press, 1993.

[108] D. E. Koditschek and E. Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11:412–442, 1990.

[109] Y. Koga. *On Computing Multi-Arm Manipulation Trajectories.* PhD thesis, Stanford University, Computer Science Department, Oct. 1994.

[110] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. In *IEEE International Conference on Robotics and Automation*, pages 945–952, 1994.

[111] P. S. Krishnaprasad, personal communication. 1996.

[112] M. Kurisu and T. Yoshikawa. Trajectory planning for an object in pushing operation. In *1994 Japan–USA Symposium on Flexible Automation*, pages 1009–1016, 1994.

[113] G. Lafferiere and H. Sussmann. Motion planning for controllable systems without drift. In *IEEE International Conference on Robotics and Automation*, pages 1148–1153, Sacramento, CA, 1991.

[114] J.-C. Latombe. A fast path planner for a car-like indoor mobile robot. In *National Conference on Artificial Intelligence*, pages 659–665, 1991.

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

[116] J.-P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *International Conference on Intelligent Autonomous Systems*, pages 346–354, 1986.

[117] J.-P. Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *International Joint Conference on Artificial Intelligence*, pages 1120–1123, 1987.

[118] J.-P. Laumond. Nonholonomic motion planning versus controllability via the multibody car system example. Technical Report STAN-CS-90-1345, Stanford University, 1990.

[119] J.-P. Laumond, P. E. Jacobs, M. Taïx, and R. M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5):577–593, Oct. 1994.

[120] C. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for CFSQP version 2.3. Institute for Systems Research 94-16, University of Maryland, 1994.

[121] N. E. Leonard and P. S. Krishnaprasad. Motion control of drift-free, left-invariant systems on lie groups. *IEEE Transactions on Automatic Control*, 40(9):1539–1554, Sept. 1995.

[122] A. D. Lewis and R. M. Murray. Configuration controllability of simple mechanical control systems. Technical Report CIT/CDS95-015, California Institute of Technology, June 1995.

[123] Z. Li and J. Canny. Motion of two rigid bodies with rolling constraint. *IEEE Transactions on Robotics and Automation*, 6(1):62–72, Feb. 1990.

[124] Z. Li and J. Canny. *Nonholonomic Motion Planning.* Kluwer Academic, 1993.

[125] J. Liebman, L. Lasdon, L. Schrage, and A. Waren. *Modeling and Optimization with GINO.* The Scientific Press, 1986.

[126] P. Lötstedt. Coulomb friction in two-dimensional rigid body systems. *Zeitschrift für Angewandte Mathematik und Mechanik*, 61:605–615, 1981.

[127] T. Lozano-Pérez, J. L. Jones, E. Mazer, and P. A. O'Donnell. Task-level planning of pick-and-place robot motions. *Computer*, 22(3):21–29, Mar. 1989.

[128] T. Lozano-Pérez, J. L. Jones, E. Mazer, and P. A. O'Donnell. *HANDEY: A Robot Task Planner.* The MIT Press, 1992.

[129] K. M. Lynch. The mechanics of fine manipulation by pushing. In *IEEE International Conference on Robotics and Automation*, pages 2269–2276, Nice, France, 1992.

[130] K. M. Lynch. Estimating the friction parameters of pushed objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 186–193, Yokohama, Japan, 1993.

[131] K. M. Lynch, H. Maekawa, and K. Tanie. Manipulation and active sensing by pushing using tactile feedback. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 416–421, Raleigh, NC, 1992.

[132] K. M. Lynch and M. T. Mason. Pulling by pushing, slip with infinite friction, and perfectly rough surfaces. *International Journal of Robotics Research*, 14(2):174–183, Apr. 1995.

[133] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. In *Algorithmic Foundations of Robotics*, pages 239–262. A. K. Peters, Boston, MA, 1995.

[134] W. D. MacMillan. *Dynamics of Rigid Bodies.* Dover, New York, 1936.

[135] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. Technical report, IBM Research Division, T.J. Watson Research Center, 1990.

[136] M. Mani and W. Wilson. A programmable orienting system for flat parts. In *North American Manufacturing Research Institute Conference XIII*, 1985.

[137] X. Markenscoff, L. Ni, and C. H. Papadimitriou. The geometry of grasping. *International Journal of Robotics Research*, 9(1):61–74, Feb. 1990.

[138] L. S. Martin and P. E. Crouch. Controllability on principal fibre bundles with compact structure group. *Systems and Control Letters*, 5:35–40, 1984.

[139] M. T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, Fall 1986.

[140] M. T. Mason. Compliant sliding of a block along a wall. In *International Symposium on Experimental Robotics*, pages 568–578. Springer-Verlag, June 1989.

[141] M. T. Mason. Two graphical methods for planar contact problems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 443–448, Osaka, Japan, Nov. 1991.

[142] M. T. Mason and R. C. Brost. Automatic grasp planning: An operation space approach. In *Sixth Symposium on Theory and Practice of Robots and Manipulators*, pages 321–328, Cracow, Poland, Sept. 1986. Alma-Press.

[143] M. T. Mason and K. M. Lynch. Dynamic manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 152–159, Yokohama, Japan, 1993.

[144] M. T. Mason and K. M. Lynch. Throwing a club: Early results. In *International Symposium on Robotics Research*, Hidden Valley, PA, Oct. 1993. Cambridge, Mass: MIT Press.

[145] M. T. Mason and J. K. Salisbury, Jr. *Robot Hands and the Mechanics of Manipulation.* The MIT Press, 1985.

[146] M. J. Matarić, M. Nilsson, and K. T. Simsarian. Cooperative multi-robot box-pushing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3: 556–561, 1995.

[147] T. McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9(2):62–82, 1990.

[148] T. McGeer. Passive walking with knees. In *IEEE International Conference on Robotics and Automation*, pages 1640–1645, 1990.

[149] B. Mishra, J. T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2(4):541–558, 1987.

[150] J. Miura. Integration of problem solving and learning in intelligent robots. In *International Symposium on Robotics Research*, pages 13–20, Tokyo, Japan, 1989. Cambridge, Mass: MIT Press.

[151] J. J. Moré and S. J. Wright. *Optimization Software Guide.* SIAM, Philadelphia, PA, 1993.

[152] H. Moseley. *Illustrations of Mechanics.* London, 1839.

[153] T. Murakami and K. Ohnishi. Robust and adaptive control strategies. In *IFAC Workshop on Motion Control for Intelligent Automation*, pages 367–372, 1992.

[154] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[155] R. M. Murray and S. S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38(5):700–716, 1993.

[156] S. Narasimhan. *Task-level Strategies for Robot Tasks.* PhD thesis, Department of Computer Science and Electrical Engineering, Massachusetts Institute of Technology, 1995.

[157] H. Nijmeijer and A. J. van der Schaft. *Nonlinear Dynamical Control Systems.* Springer-Verlag, 1990.

[158] Y. Okawa and K. Yokoyama. Control of a mobile robot for the push-a-box operation. In *IEEE International Conference on Robotics and Automation*, pages 761–766, Nice, France, 1992.

[159] G. Oriolo and Y. Nakamura. Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators. In *Conference on Decision and Control*, pages 2398–2403, 1991.

[160] J. Ostrowski, J. Burdick, A. D. Lewis, and R. M. Murray. The mechanics of undulatory locomotion: The mixed kinematic and dynamic case. In *IEEE International Conference on Robotics and Automation*, pages 1945–1951, 1995.

[161] J. Ostrowski, A. Lewis, R. Murray, and J. Burdick. Nonholonomic mechanics and locomotion: The snakeboard example. In *IEEE International Conference on Robotics and Automation*, pages 2391–2397, 1994.

[162] M. A. Peshkin and A. C. Sanderson. The motion of a pushed, sliding workpiece. *IEEE Journal of Robotics and Automation*, 4(6):569–598, Dec. 1988.

[163] M. A. Peshkin and A. C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4(5):524–531, Oct. 1988.

[164] M. A. Peshkin and A. C. Sanderson. Minimization of energy in quasi-static manipulation. *IEEE Transactions on Robotics and Automation*, 5(1):53–60, Feb. 1989.

[165] D. T. Pham, K. C. Cheung, and S. H. Yeo. Initial motion of a rectangular object being pushed or pulled. In *IEEE International Conference on Robotics and Automation*, pages 1046–1050, Cincinnati, OH, 1990.

[166] P. Pignon, Nomadics, Inc. Personal communication, 1995.

[167] J. Prescott. *Mechanics of Particles and Rigid Bodies.* Longmans, Green, and Co., London, 1923.

[168] M. H. Raibert. *Legged Robots That Balance.* Cambridge: MIT Press, 1986.

[169] V. T. Rajan. Minimum time trajectory planning. In *IEEE International Conference on Robotics and Automation*, pages 759–764, 1985.

[170] V. T. Rajan, R. Burridge, and J. T. Schwartz. Dynamics of a rigid body in frictional contact with rigid walls. In *IEEE International Conference on Robotics and Automation*, pages 671–677, 1987.

[171] A. Rao, D. Kriegman, and K. Y. Goldberg. Complete algorithms for reorienting polyhedral parts using a pivoting gripper. In *IEEE International Conference on Robotics and Automation*, pages 2242–2248, 1995.

[172] V. J. Rayward-Smith. *A First Course in Formal Language Theory.* Blackwell Scientific Publications, 1983.

[173] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[174] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, Oct. 1992.

[175] A. A. Rizzi and D. E. Koditschek. Progress in spatial robot juggling. In *IEEE International Conference on Robotics and Automation*, pages 775–780, Nice, France, 1992.

[176] A. A. Rizzi and D. E. Koditschek. Further progress in robot juggling: The spatial two-juggle. In *IEEE International Conference on Robotics and Automation*, pages 3:919–924, Atlanta, GA, 1993.

[177] G. Sahar and J. M. Hollerbach. Planning of minimum-time trajectories for robot arms. *International Journal of Robotics Research*, 5(3):90–100, 1986.

[178] T. Sakaguchi, M. Fujita, H. Watanabe, and F. Miyazaki. Motion planning and control for a robot performer. In *IEEE International Conference on Robotics and Automation*, pages 3:925–931, Atlanta, GA, 1993.

[179] T. Sakaguchi, Y. Masutani, and F. Miyazaki. A study on juggling task. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1418–1423, Osaka, Japan, 1991.

[180] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini. A direct approach to vision guided manipulation. In *International Conference on Advanced Robotics*, Tokyo, Japan, 1993.

[181] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini. A vision-based learning method for pushing manipulation. In *AAAI Fall Symposium on Machine Learning in Computer Vision*, Raleigh, NC, 1993.

[182] J. K. Salisbury. Whole arm manipulation. In *International Symposium on Robotics Research*, Santa Cruz, CA, Aug. 1987. Cambridge, Mass: MIT Press.

[183] K. Salisbury, B. Eberman, M. Levin, and W. Townsend. The design and control of an experimental whole-arm manipulator. In *International Symposium on Robotics Research*, pages 233–241, Tokyo, Japan, Aug. 1989. Cambridge, Mass: MIT Press.

[184] N. Sawasaki, M. Inaba, and H. Inoue. Tumbling objects using a multi-fingered robot. In *Proceedings of the 20th International Symposium on Industrial Robots and Robot Exhibition*, pages 609–616, Tokyo, Japan, 1989.

[185] S. Schaal and C. G. Atkeson. Open loop stable control strategies for robot juggling. In *IEEE International Conference on Robotics and Automation*, pages 3:913–918, Atlanta, GA, 1993.

[186] S. Schaal, C. G. Atkeson, and S. Botros. What should be learned? In *Seventh Yale Workshop on Adaptive and Learning Systems*, pages 199–204, 1992.

[187] K. Schittkowski. *QLD: A Fortran Code for Quadratic Programming, User's Guide.* Mathematisches Institut, Universität Bayreuth, Germany, 1986.

[188] J. G. Schneider and C. M. Brown. Robot skill learning, basis functions, and control regimes. In *IEEE International Conference on Robotics and Automation*, pages 1:403–410, Atlanta, GA, 1993.

[189] J.-P. Serre. *Lie Algebras and Lie Groups.* W. A. Benjamin, New York, 1965.

[190] Z. Shiller and S. Dubowsky. On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6):785–797, Dec. 1991.

[191] K. G. Shin and N. D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, June 1985.

[192] N. C. Singer and W. P. Seering. Utilizing dynamic stability to orient parts. *Journal of Applied Mechanics*, 54:961–966, Dec. 1987.

[193] P. R. Sinha. A contact stress model for determining forces in an equilibrium grasp. Technical Report MS-CIS-90-19, Department of Computer and Information Science, University of Pennsylvania, Mar. 1990.

[194] J.-J. E. Slotine, 1992. Vision Guided Motion Control (videotape).

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

[196] E. Sontag. Gradient techniques for systems with no drift: A classical idea revisited. In *IEEE International Conference on Decision and Control*, pages 2706–2711, 1993.

[197] O. J. Sørdalen, Y. Nakamura, and W. J. Chung. Design of a nonholonomic manipulator. In *IEEE International Conference on Robotics and Automation*, pages 8–13, 1994.

[198] M. W. Spong. Swing up control of the acrobot. In *IEEE International Conference on Robotics and Automation*, pages 2356–2361, 1994.

[199] G. Strang. *Linear Algebra and Its Applications.* Orlando: Academic Press, 1980.

[200] W. J. Stronge. Rigid body collisions with friction. *Proceedings of the Royal Society, London*, A431:169–181, 1990.

[201] W. J. Stronge. Unraveling paradoxical theories for rigid body collisions. *ASME Journal of Applied Mechanics*, 58:1049–1055, Dec. 1991.

[202] H. Sussmann. A continuation method for nonholonomic path-finding problems. In *IEEE International Conference on Decision and Control*, pages 2718–2723, 1993.

[203] H. J. Sussmann. A sufficient condition for local controllability. *SIAM Journal on Control and Optimization*, 16(5):790–802, Sept. 1978.

[204] H. J. Sussmann. Lie brackets, real analyticity and geometric control. In R. W. Brockett, R. S. Millman, and H. J. Sussmann, editors, *Differential Geometric Control Theory*. Birkhauser, 1983.

[205] H. J. Sussmann. A general theorem on local controllability. *SIAM Journal on Control and Optimization*, 25(1):158–194, Jan. 1987.

[206] H. J. Sussmann and V. Jurdjevic. Controllability of nonlinear systems. *Journal of Differential Equations*, 12:95–116, 1972.

[207] P. J. Swanson, R. R. Burridge, and D. E. Koditschek. Global asymptotic stability of a passive juggler: A parts feeding strategy. In *IEEE International Conference on Robotics and Automation*, pages 1983–1988, 1995.

[208] K. R. Symon. *Mechanics*. Addison-Wesley, 1971.

[209] S. Takashima. Control of gymnast on a high bar. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1424–1429, Osaka, Japan, 1991.

[210] H. H. Tan and R. B. Potts. Minimum time trajectory planner for the discrete dynamic robot model with dynamic constraints. *IEEE Journal of Robotics and Automation*, 4(2):174–185, Apr. 1988.

[211] J. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo. On dynamic multi-rigid-body contact problems with coulomb friction. Technical Report TAMU-CS 95-003, Texas A&M University, Computer Science Department, Jan. 1995.

[212] M. Vukobratović and M. Kirćanski. A method for optimal synthesis of manipulation robot trajectories. *ASME Journal of Dynamic Systems, Measurement, and Control*, 104:188–193, June 1982.

[213] Y. Wang. Dynamics and planning of collisions in robotic manipulation. In *IEEE International Conference on Robotics and Automation*, pages 478–483, Scottsdale, AZ, 1989.

[214] Y. Wang and M. T. Mason. Two-dimensional rigid-body collisions with friction. *ASME Journal of Applied Mechanics*, 59:635–641, Sept. 1992.

[215] Y.-T. Wang, V. Kumar, and J. Abel. Dynamics of rigid bodies undergoing multiple frictional contacts. In *IEEE International Conference on Robotics and Automation*, pages 2764–2769, Nice, France, 1992.

[216] F. W. Warner. *Foundations of Differentiable Manifolds and Lie Groups*. Springer-Verlag, 1983.

[217] D. E. Whitney. Quasi-static assembly of compliantly supported rigid parts. *ASME Journal of Dynamic Systems, Measurement, and Control*, 104:65–77, Mar. 1983.

[218] G. Wilfong. Motion planning in the presence of movable obstacles. In *ACM Symposium on Computational Geometry*, pages 279–288, 1988.

[219] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, 1988.

[220] P. G. Xavier. *Provably-Good Approximation Algorithms for Optimal Kinodynamic Robot Motion Plans*. PhD thesis, Department of Computer Science, Cornell University, Apr. 1992.

[221] V. Yen and M. L. Nagurka. A suboptimal trajectory planning problem for robotic manipulators. *ISA Transactions*, 27(1):51–59, 1988.

[222] M. H. Yim, personal communication. 1993.

[223] S. Yoshikawa and K. Yamada. Impact estimation for capturing a target by a space robot. In *International Conference on Advanced Mechatronics*, pages 836–841, Tokyo, Japan, 1993.

[224] T. Yoshikawa and M. Kurisu. Identification of the center of friction from pushing an object by a mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 449–454, Osaka, Japan, 1991.

[225] K.-H. Yu, T. Takahashi, and H. Inooka. Dynamics and motion control of a two-link robot manipulator with a passive joint. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2: 311–316, 1995.

[226] M. Žefran and V. Kumar. Optimal control of systems with unilateral constraints. In *IEEE International Conference on Robotics and Automation*, pages 2695–2700, 1995.

[227] T. Zrimec. *Towards Autonomous Learning of Behavior by a Robot*. PhD thesis, University of Ljubljana, Computer and Information Science, 1990.

[228] N. B. Zumel. Analysis and planning for nonprehensile two palm manipulation. Thesis Proposal, Carnegie Mellon University Robotics Institute, Apr. 1995.

[229] N. B. Zumel and M. A. Erdmann. Balancing of a planar bouncing object. In *IEEE International Conference on Robotics and Automation*, pages 2949–2954, San Diego, CA, 1994.