NORTHWESTERN UNIVERSITY

# Modeling, Motion Planning, and Feedback Control for Dynamic, Graspless, and Hybrid Robotic Manipulation Tasks

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Mechanical Engineering

By

J. Zachary Woodruff

EVANSTON, ILLINOIS

December 2020

# ABSTRACT

Modeling, Motion Planning, and Feedback Control for Dynamic, Graspless, and Hybrid Robotic Manipulation Tasks

J. Zachary Woodruff

This thesis presents methods to improve the manipulation capabilities of robots. People and animals can effectively handle objects of many shapes, sizes, weights, and materials using a variety of manipulation primitives such as grasping, pushing, sliding, tipping, rolling, and throwing. In contrast, most robots manipulate objects by pick-and-place. Restricting robots to only grasp objects artificially limits the set of tasks that they can accomplish, and leveraging a larger set of manipulation primitives is crucial for robots to reach their full potential in applications such as flexible manufacturing, agricultural automation, service industries, and disaster response.

We first outline a high-level framework for planning and control for dynamic, graspless, and hybrid manipulation tasks. "Dynamic" means that the momentum of the objects cannot be ignored. "Graspless" means that the objects are not grasped in a traditional sense, but rather manipulated with unilateral friction forces at the contacts. "Hybrid" means that multiple manipulation primitives are used that each have their own constraints

and dynamic equations. The main contributions of this work are the framework that outlines specific subproblems to solve dynamic, graspless, and hybrid manipulation tasks as well as an experimental implementation for an example task.

In the remaining chapters we focus specifically on modeling rolling contacts between two smooth bodies, designing motion planners and feedback controllers for rolling manipulation tasks, and testing them in simulation and experimentally. First-order kinematics addresses the rolling problem where the relative contact velocities are given. The second-order kinematics is a generalization of the first-order model where the relative accelerations at the contact are specified. The evolution of dynamic rolling systems is governed by forces and torques at the contact. We address both first- and second-order kinematics and dynamic rolling in our work. The main contributions of the first-order rolling work are a robust motion planner that can handle general smooth geometries, a method to test the controllability of linearized rolling trajectories, and a method to stabilize trajectories from initial state perturbations. The main contributions of the second-order rolling work are corrections to previous work that derived the second-order kinematics equations. The main contributions of the dynamic rolling work are that it is the first work we know of to formulate the rolling dynamics of a rigid body rolling on a six degree-of-freedom motion-controlled manipulator for general manipulator and object shapes, it provides a compact form that outputs the dynamics and contact forces that can be used for trajectory optimization, and the coordinate-based representation allows the dynamics to be linearized to generate feedback controllers that stabilize planned trajectories. We apply the method to model, plan, and stabilize dynamic, graspless, and hybrid rolling experiments that include the first known implementation of the rolling pendulum swing up.

## Acknowledgments

I want to thank my advisor Kevin Lynch for all the mentorship he has given me throughout my PhD. His advice has been invaluable in helping me formulate and solve research problems, and to effectively communicate my findings in publications and presentations. Thanks to Paul Umbanhowar for all of his guidance over the years ranging from brainstorming research topics, debugging low-level electronics issues, editing papers, serving on my thesis committee, to giving car repair advice for my minivan. I also want to thank my two other committee members Randy Freeman and Todd Murphey for their advice throughout my graduate studies, on my thesis, and during my defense.

Thanks to my undergraduate advisor David Go for connecting me with my first research position, introducing me to the research process, and encouraging me to publish my work and continue on to graduate school. Thank you to my undergraduate robotics professor Jim Schmiedeler for first introducing me to robotics and encouraging me to pursue a PhD as well!

As the unofficial greeter to the lab, my desk was conveniently located next to the main entrance, the lunch table, and the hallway to the mail and coffee room. There was rarely a dull moment, and I am so glad to have been part of such a vibrant community of students, researchers, and professors. I want to thank all of my lab mates, but specifically Adam, Matt, Tito, Bill, Roman, Dan, Ian, Giorgos, Shufeng, Nelson, Jarvis, Jian, and Huan for all their friendship and collaboration over the years. The same to Becca and

Ahalya, and for sticking it out with me through the end so we could reminisce about the early days.

To all my friends and the seemingly endless source of distractions they provided. Thank you for the concerts, bike rides through the city, late night guitar sessions, beach days, distant climbing adventures, art projects, sailing outings, and everything else that filled my time in grad school with cherished memories!

To my parents. Without you I would not exist, and without your care, guidance, and support I would not be where I am today, so thank you from all my heart! To my sister Amy and brother Fletcher, thanks for all you have done over the years as well!

It is a strange thing to complete my PhD during a global pandemic. Finishing experiments in an empty lab, defending my thesis virtually, packing up my apartment, and moving across the country, unable to say goodbyes or share my appreciation in person. It is an anticlimactic end to such a formative time in my life, but I look forward to reconnecting when this is all behind us to celebrate together!

-Zack

# Table of Contents

# List of Tables

# List of Figures

CHAPTER 1

# Introduction

People and animals can effectively handle objects of many shapes, sizes, weights, and materials using a variety of manipulation primitives such as grasping, pushing, sliding, tipping, rolling, and throwing. In contrast, most robots manipulate objects by pick-and-place. There is good reason for this: once a firm grasp is established, the robot can reliably control the motion of the part without needing to continuously sense the state of the part or correct for modeling uncertainties. Most manipulation primitives mentioned above are more sensitive to uncertainties in part state, geometry, mass, friction, and restitution, in addition to the robot's own control errors. Nonetheless, restricting robots to only grasp objects artificially limits the set of tasks that they can accomplish. Leveraging a larger set of manipulation primitives is crucial for robots to reach their full potential in applications such as flexible manufacturing, agricultural automation, service industries, and disaster response.

While manipulation primitives exist for manipulating several objects simultaneously, we will focus on the case of a single rigid object. We define manipulation primitives according to the number and types of contacts the object makes with a robot and its rigid environment. Contacts are classified according to whether they are sliding (e.g., point-on-surface or surface-on-surface), fixed (e.g., point-on-surface), or rolling (e.g., surface-on-surface). Contacts with a robot are further classified according to the control law the robot

implements at that contact (e.g., position control, force control, combined position/force control, compliance control, etc.).

Consider the example in Figure 1.1 that outlines a dynamic, graspless, and hybrid manipulation plan. A simple robot arm without a gripper is clearing the cluttered table, and the current task is to move the orange, elliptical object into the bin. The goal is outside of the robot's workspace so the task cannot be completed quasistatically and will have to include at least one "dynamic" primitive (free flight). The robot does not have a fingered gripper so it cannot use pick-and-place. Instead the robot must use "graspless" manipulation primitives that move the object without a force-closure grasp. Lastly, the solution will require multiple contact modes between the object, the manipulator, and the environment which we refer to as a "hybrid" plan. An example set of manipulation primitives is shown that accomplish the desired task. The first motion is a "controlled roll" where the manipulator uses friction forces at the fixed point contact to roll the object on the table. This mode is controlled using compliant motion control, where the 2-DOF motion of the robot's endpoint is velocity controlled but with a finite stiffness. If we abstract away the details of the robot's control, this can be modeled using the kinematic rolling work described in Chapter 3. The second motion is a "free roll" where the manipulator releases the object and remains stationary while the object's momentum causes it to roll on the table and up the arm. This is modeled as a single object-environment rolling contact, followed by a single object-robot rolling contact with no robot control. The final motion is a "controlled throw" where the system tracks the state of the object and controls it to a desired release state which is addressed by our work in Chapter 5. The final motion is "free flight" where there are no contacts and the object follows a parabolic

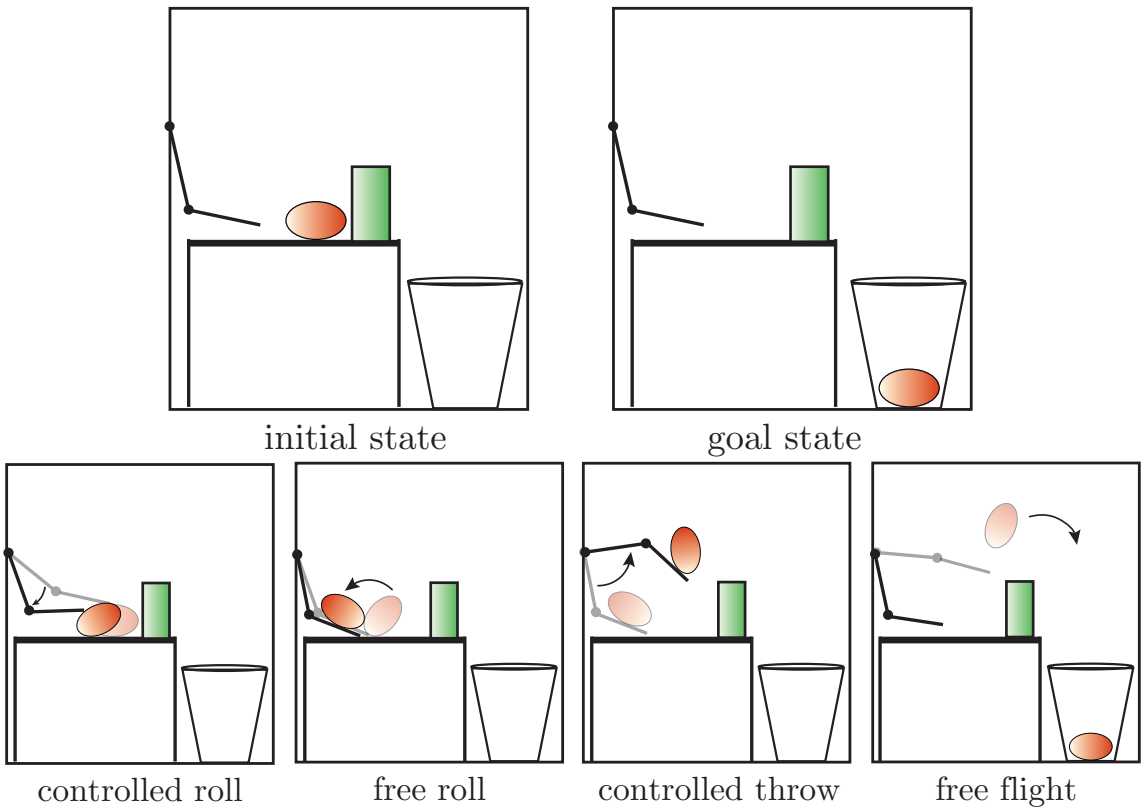



Figure 1.1. An example of a manipulation task of picking a smooth object up off a table and throwing it into a bin. This task involves multiple dynamic and graspless primitives.

trajectory under gravity that brings it to the goal state. Without leveraging, dynamic, graspless, and hybrid manipulation, the goal would not have been achievable, and this thesis presents methods to improve the manipulation capabilities of robots to accomplish such tasks.

## 1.1. Thesis Outline

Related work sections that summarize relevant research are included in each chapter. In Chapter 2 we first present a high-level framework for planning and control for dynamic, graspless (also called "nonprehensile"), and hybrid manipulation tasks. We outline five

subproblems to address this: determining a set of manipulation primitives, choosing a sequence of tasks, picking transition states, motion planning for each individual primitive, and stabilizing each mode using feedback control. We apply the framework to plan a sequence of motions for manipulating a block with a simple rectangular manipulator. The main contributions of this work are the framework that outlines specific subproblems to solve dynamic, graspless, and hybrid manipulation tasks, as well as an experimental implementation for an example task. In the remaining chapters we focus specifically on rolling contacts between two smooth bodies such as during the "free roll" and "controlled throw" modes of the hybrid manipulation task in Figure 1.1. Chapter 3 examines the problem of kinematic rolling where the relative velocities are directly controlled (first-order kinematics). It demonstrates planning and stabilizing the trajectory of one smooth body rolling on the surface of another. The main contributions of this work are a robust motion planner that can handle general smooth geometries, a method to test the controllability of linearized rolling trajectories, and a method to stabilize trajectories from initial state perturbations. Chapter 4 outlines the equations for the second-order kinematics where the relative accelerations are directly controlled. The main contributions of the second-order rolling work are corrections to previous work that derived the second-order kinematics equations. Chapter 5 address the problem of controlling the motion of objects that are in rolling contact with a robot manipulator or "hand" in three dimensions. We directly control the motion of the hand to indirectly control the motion of the rolling object. Our approach to dynamic rolling manipulation can be split into four subproblems: 1) calculating the first- and second-order rolling kinematic equations; 2) deriving the rolling dynamics; 3) planning rolling motions that satisfy the dynamics; and 4) feedback control

of rolling trajectories. The results are validated against examples with analytical solutions in simulation, and tested experimentally. The main contributions of the dynamic rolling work are that it is the first work we know of to formulate the rolling dynamics of a rigid body rolling on a six-DoF motion-controlled manipulator for general manipulator and object shapes, it provides a compact form that outputs the dynamics and contact forces that can be used for trajectory optimization, and the coordinate-based representation allows for the dynamics to be linearized to generate feedback controllers that stabilize planned trajectories. We further apply the method to model, plan, and stabilize dynamic, graspless, and hybrid rolling experiments that include the first-known implementation of the rolling pendulum swing up. Chapter 6 summarizes the results and outlines areas for future work. Appendix A includes detailed information about the experimental setup developed for this work.

This thesis includes four separate subdocuments that have been published or submitted to journals or conferences. Chapter 2 was presented at ICRA 2017 and published in the proceedings [**74**]. Chapter 3 was published in IEEE Access [**76**]. Chapter 4 was published in the Journal of Applied Mechanics [**75**]. Chapter 5 is being prepared for submission to IEEE Transactions on Robotics (T-RO). We present the papers in their original form, so notation is consistent within each chapter but not throughout the thesis.

CHAPTER 2

# Planning and Control for Dynamic, Nonprehensile, and Hybrid Manipulation Tasks

## 2.1. Abstract

In this chapter we propose a method for motion planning and feedback control of hybrid, dynamic, and nonprehensile (also called "graspless") manipulation tasks. We outline five subproblems to address this: determining a set of manipulation primitives, choosing a sequence of tasks, picking transition states, motion planning for each individual primitive, and stabilizing each mode using feedback control. We apply the framework to plan a sequence of motions for manipulating a block with a planar 3R manipulator. We demonstrate preliminary experimental results for a block resting on the manipulator with a desired goal state on a ledge outside of the robot's workspace. The planned primitives reorient the block using a series of fixed, rolling, and sliding contact modes, and throw it to the goal state.

The main contributions of this chapter are the framework that outlines specific subproblems to solve dynamic, graspless, and hybrid manipulation tasks as well as an experimental implementation for an example task. This chapter was initially published in the ICRA 2017 proceedings [**74**].

## 2.2. Introduction

People and animals can effectively manipulate objects of many shapes, sizes, weights, and materials using a variety of primitives such as grasping, pushing, sliding, tipping, rolling, and throwing. In contrast, most robots manipulate objects by pick-and-place. There is good reason for this: once a firm grasp is established, the robot can reliably control the motion of the part without needing to continuously sense the state of the part or correct for modeling uncertainties. Most manipulation primitives mentioned above are more sensitive to uncertainties in part state, geometry, mass, friction, and restitution, and to the robot's own control errors. Nonetheless, restricting robots to only grasp objects artificially limits the set of tasks that they can accomplish. Leveraging a larger set of manipulation primitives is crucial for robots to reach their full potential in industrial automation, exploration, home care, military, and space applications.

### 2.2.1. Background

While manipulation primitives exist for manipulating several objects simultaneously, for simplicity we will focus on the case of a single rigid object. We define manipulation primitives according to the number and types of contacts the object makes with a robot and its (rigid) environment. Contacts are classified according to whether they are sliding or fixed/rolling, and contacts with a robot are further classified according to the control law the robot implements at that contact (e.g., position control, force control, hybrid position/force control, compliance control, etc.).

Consider the planar example of a block and 3R manipulator shown in Figure 2.1. The block is initially at rest on the table with a desired goal state in the bin to the right.

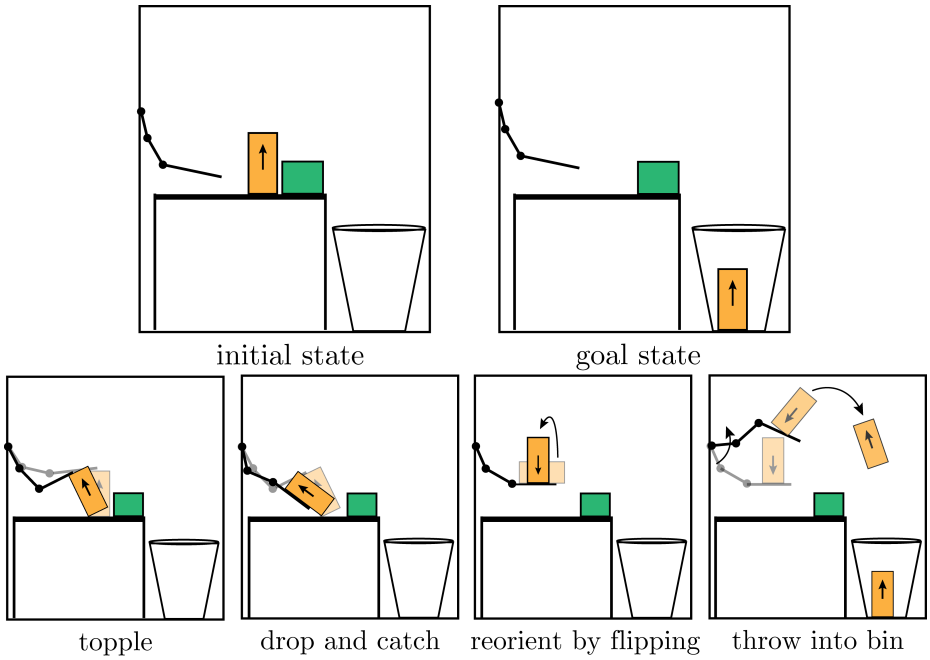



Figure 2.1. An example of a manipulation task of picking up a block off a table and throwing it into a bin. This task involves multiple dynamic nonprehensile primitives.

Another object, which should not be disturbed, is between the block's initial position and the goal configuration. The figure illustrates one possible solution to the manipulation task, consisting of a sequence of primitives. The controlled toppling primitive consists of one-point rolling between the block and the table while the robot applies a hybrid position-force controlled fixed contact to the top of the block, to control the internal force toward the rolling contact while controlling the orthogonal velocity. Once the block passes the unstable equilibrium point, the robot releases the block, letting it topple by gravity. The robot quickly moves underneath the block and "catches" it. The next primitive is a two-point, fixed-contact "dynamic grasp" carry, followed by a free-flight phase of the block (a throw). After catching the object, the robot executes a dynamic grasp carry,

followed by a phase where the object is in one-point rolling contact with the manipulator, followed by a free flight phase.

In summary, the manipulation sequence consists of a set of primitives punctuated by transitions:

controlled topple + free topple + catch + dynamic grasp + free flight + catch + dynamic grasp + rolling (controlled)+ free flight.

Each unique primitive is assigned an index $i$, and the dynamics governing each primitive are different, as the coupling of the manipulator controls to the object through the contacts, and the possible contact forces applied by the environment, are different. We define the manipulator controls to be $\mathbf{u} \in \mathbb{R}^{n_u}$. In coordinate parameterizations of the configurations of the object and the manipulator, the configuration of the object is $\mathbf{q}_{\text{obj}} \in \mathbb{R}^{n_{\text{obj}}}$, and the configuration of the manipulator is $\mathbf{q}_{\text{m}} \in \mathbb{R}^{n_{\text{m}}}$. The total system configuration is defined as $\mathbf{q} = [\mathbf{q}_{\text{m}}^{\mathsf{T}} \;\; \mathbf{q}_{\text{obj}}^{\mathsf{T}}]^{\mathsf{T}}$, and the state of the system as $\mathbf{x} = [\mathbf{q}^{\mathsf{T}} \;\; \dot{\mathbf{q}}^{\mathsf{T}}]^{\mathsf{T}}$. The dynamics during each primitive can then be written

$$\dot{\mathbf{x}} = \mathbf{f}_i(\mathbf{x}, \mathbf{u}).$$

In other words, manipulation is a hybrid system.

Each primitive $i$ has a domain of applicability $\mathcal{D}_i$, i.e., a region in the state-control space where the dynamics of that primitive describes the evolution of the system[1]. The state-control space has $(2n_{\text{m}} + 2n_{\text{obj}} + n_u) = d$ dimensions, and this space is partitioned by the different manipulation primitives. In the example in Figure 2.1, $n_m = n_u = 3$ and

[1]It is well known that problems in rigid-body frictional mechanics, such as those in this paper, are subject to ambiguity issues, i.e., the same system state $\mathbf{x}$ and controls $\mathbf{u}$ can result in more than one possible $\dot{\mathbf{x}}$. In this paper, we set this possibility aside.

$n_{\text{obj}} = 3$, so the state-control space has $d = 15$ dimensions. A full-dimensional subset of this state-control space corresponds to no contacts: the object is in free flight and the manipulator moves freely. We could call this the "free flight" primitive, a primitive that is always part of a throw. For other manipulation primitives, the state and control constraints implied by active contacts reduce the dimension of the domain of applicability. The "free topple" primitive in Figure 2.1 is twelve-dimensional assuming that the rolling vertex on the block is prespecified. It can be parameterized by the nine robot states and controls, the angle and rotational velocity of the block, and the position of the vertex on the table. The "controlled topple" primitive is nine-dimensional assuming the endpoint of the robot is in contact and that the rolling vertex is prespecified. This mode can be parameterized by the three control freedoms, the internal configuration and velocity of the robot, the angle and rotational velocity of the block, and the positions of the vertex on the table and the robot on the object. Of course there are also inequality constraints on the states and controls for each manipulation primitive, but they generally do not reduce the dimension of the domain of applicability.

In theory, the entire state-control space could be partitioned into manipulation primitives of different dimensions. Boundaries between these primitives, where a manipulation plan can transition from one primitive to the other, are described by one or more equations that are simultaneously satisfied. For example, the boundary between the twelve-dimensional "free topple" space and the nine-dimensional "controlled topple" space occurs where the conditions of both primitives are simultaneously satisfied, i.e., the controlled topple conditions are satisfied, but the contact force at the robot contact is zero.

Thus we can think of manipulation planning as planning a sequence of manipulation primitives, such that the initial state of the system is in the domain of applicability of the first primitive and the goal state of the system is in the domain of applicability of the last primitive. If the goal is given as $m$ constraints on the final state of the object, then the goal is specified by a $(d-m)$-dimensional subset of the state-control space, which may span more than one manipulation primitive. The manipulation problem can be broken into the following subproblems:

(1) **Primitive characterization.** Given descriptions of a robot, object, and the environment, derive a set of contact modes and determine contact constraints, dynamics, and the domain of applicability for each mode.

(2) **Primitive sequence planning.** Choose a sequence of $N$ primitives to transition through such that the first primitive contains the initial state and the final primitive contains the goal.

(3) **Transition state planning.** Choose a sequence of transition states $\{(\mathbf{x}_p, \mathbf{u}_p) \mid p = 1 \ldots (N-1)\}$, such that each transition lies on the boundary of sequential primitives.

(4) **Planning the individual manipulation primitives.** Derive $N$ motion plans, one for each primitive, so that they connect at their transitions in the state-control space, and such that the first primitive begins at the initial state of the system and the last primitive ends somewhere in the $(d-m)$-dimensional goal region.

(5) **Stabilizing the individual primitives.** Derive feedback controllers to stabilize the motion plans for different primitives.

In practice, it is not possible to explicitly construct the full partitioning of the state-control space. This problem is harder than explicitly calculating a mathematical representation of configuration-space obstacles, a problem that researchers in motion planning have purposely avoided for years. It may be possible, however, to define a (small) library of manipulation primitives, their domains of applicability, and the state-control transition equations between them, such that the domains of applicability cover a good percentage of the state space of interest.

In this paper, we first develop a set of steps to address the subproblems above for solving hybrid, dynamic, and nonprehensile manipulation problems. We begin to explore this approach to manipulation planning using the example of a three-degree-of-freedom manipulator and a planar block shown in Figure 5.11. This builds upon our past work developing individual manipulation primitives such as rolling and pushing using 1-joint dynamic robots [**41**]. We demonstrate the motion plans experimentally for a block resting on the manipulator with a desired goal state on a ledge outside of the robot's workspace. The planned primitives shown in Figure 2.6 manipulate the block using a series of rolling and sliding contacts and throw it to the goal state.

### 2.2.2. Paper Outline

Section 5.3 reviews related work on which this paper builds. Section 2.4 gives a general outline to plan for dynamic, nonprehensile, and hybrid manipulation tasks, and Section 2.5 applies the framework to a specific example of flipping up a block at rest on a manipulator and balancing it. Section 2.6 describes the experimental setup, and presents preliminary results of planning and executing a block-manipulation task.

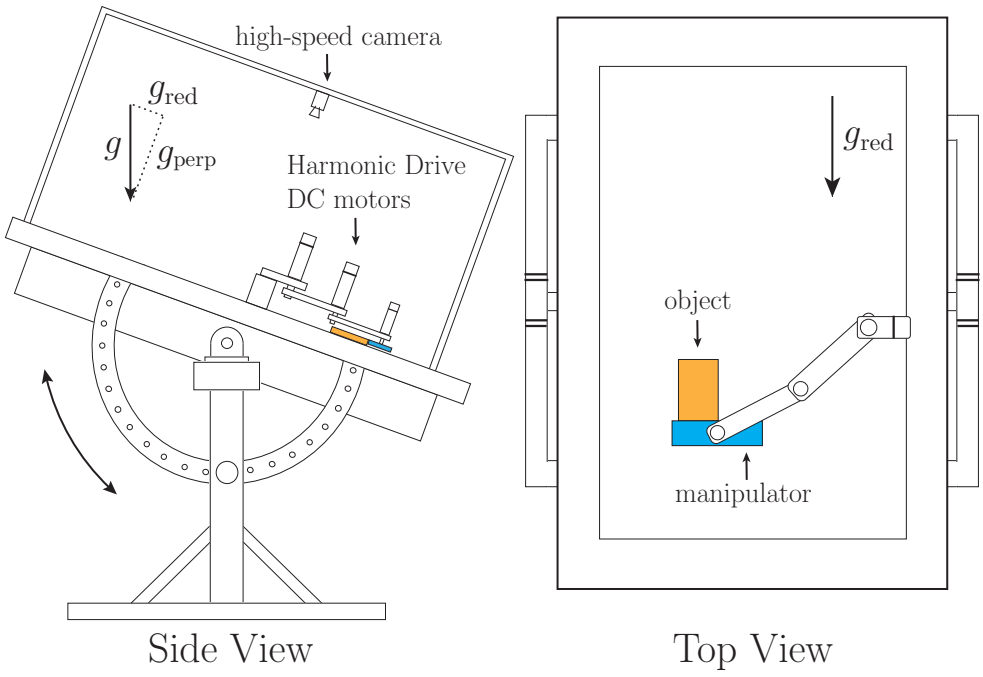



Figure 2.2. Our experimental setup consists of an inclined air hockey table with a planar 3R robot driven by brushed DC motors with harmonic drive gearing, and an OptiTrack s250e 250 Hz camera. The angle of the table allows 2D dynamic manipulation experiments in reduced gravity ($0.4g$), and the camera system gives feedback on object positions.

## 2.3. Related Work

### 2.3.1. Hybrid Manipulation Planning

We use the classical hybrid automata formulation for modeling the manipulation problem. Goebel et al. [**25**] provide a tutorial on modeling hybrid system dynamics, analyzing stability, and designing stabilizing controllers. Johnson et al. [**30**] present a hybrid dynamical system model that provides existence and uniqueness guarantees for systems with common approximations such as rigid bodies and plastic impacts.

Motion planners for hybrid systems must reason about trajectories that pass through different contact modes. Trinkle and Hunter [**71**] extend the dexterous manipulation

planning problem to consider rolling and slipping. Erdmann [**20**] analyzes the task of two palms manipulating a part. Given information about the object, and a desired start and end configuration, the planner determines a set of nonprehensile motions to reorient the part to a goal position. The hybrid planning problem is further developed by Yashima [**78**] and Miyazawa [**48**] using randomized motion planning to plan dexterous and graspless manipulation tasks, respectively. Additionally Maeda [**44**] uses graph-based methods for planning graspless manipulation.

Numerous works have addressed hybrid planning for dynamic manipulation tasks. Furukawa et al. demonstrate dynamic, prehensile, robotic manipulation by tossing a foam cylinder up and catching it [**23**]. Srinivasa et al. [**67**] address a dynamic flip-up problem to find motions that tip a block while maintaining a rolling contact with a flat manipulator that can move in a vertical plane. Pekarovskiy et al. [**55**] calculate optimal batting trajectories for a planar object on an air table and deform them online to send the object to desired goal states.

Some recent works have moved away from the hybrid automata model to formulations that do not treat modes separately. Tassa and Todorov [**69**] use the method of stochastic complementarity to smooth the discontinuous dynamics of hybrid systems allowing them to be solved by more classical optimization methods. Posa et al. [**56**] use direct methods and the complementarity formulation to plan motions for dynamic systems with impacts and Coulomb friction.

### 2.3.2. Trajectory Control

Trajectory control involves the design of feedback controllers to stabilize dynamic systems about desired trajectories. For linear systems this is often done with LQR control, and many nonlinear systems are stabilized using a time varying LQR controller about a linearized trajectory [**4**]. Cimen [**17**] surveys the State-Dependent Riccati Equation (SDRE) control method which parameterizes nonlinear dynamics into a linear structure with state-dependent coefficient matrices. Posa et al. [**57**] use sum-of-squares methods for computing Lyapunov certificates to show stability and design controllers for rigid-body systems with impacts and friction. Numerous methods and references for the control and stabilization of hybrid systems can be found in [**25**]. Our hybrid formulation has similarities to those in locomotion research, but we are not generally interested in periodic trajectories (like gaits) which prevents us from achieving cycle-wise stability.

## 2.4. Hybrid Planning and Control Formulation

The following is an outline of the hybrid, dynamic manipulation problem we address in this paper. We assume a manipulator and an object interacting in a 2D environment. Given a description of a manipulator and an object, the initial state of the system $\mathbf{x}_{\mathrm{o}}$, and the desired final state $\mathbf{x}_{\mathrm{f}}$, we find controls $\mathbf{u}(t)$ and stabilizing feedback controllers $\mathbf{u}_{\mathrm{fbk},i}(\mathbf{x}, t)$ that bring the system to the desired final state through a set of $N$ contact modes $\{i_p \mid i_p \in \mathcal{I},\ p = 1 \ldots N\}$ and transition state-control pairs $\{(\mathbf{x}_p, \mathbf{u}_p) \mid p = 1 \ldots (N-1)\}$, while satisfying system dynamics and contact constraints of each mode, transition constraints between modes, and state/control inequality constraints.

### 2.4.1. Primitive Characterization

The first step we take to solve the manipulation planning problem is to determine a set of manipulation primitives. Rather than enumerating all possible contact modes, we choose $\mathcal{I}$ desired contact modes from the full set of possible block/manipulator/environment contacts, and then determine the necessary contact and transition constraints for planning through and transitioning between them.

Each mode is defined by a set of contacts that either slide, roll, or remain fixed along the surface of the object. The configuration constraints can be expressed by the function $\phi(\mathbf{q}) = 0$, and the velocity constraints can be expressed as $c_i$ Pfaffian constraints of the form $\mathbf{A}_i(\mathbf{q})\dot{\mathbf{q}} = 0$ where $\mathbf{A}_i(\mathbf{q}) \in \mathbb{R}^{c_i \times (n_{\mathrm{m}}+n_{\mathrm{obj}})}$. These constraint forces and the friction properties determine the total force at the contact. The set of all $m_i$ constraints in each mode $i$ reduces the $d$ dimensional state-control space to $d_i = d - m_i$. The set of all state-control pairs that satisfy the constraints for a given mode is the domain of applicability $\mathcal{D}_i$.

Each contact mode $i \in \{1 \ldots \mathcal{I}\}$ has a set of corresponding dynamics $\dot{\mathbf{x}} = \mathbf{f}_i(\mathbf{x}, \mathbf{u})$. We define the boundaries between modes by the guard sets $\mathcal{G}_{jk}(\mathbf{x}, \mathbf{u}) = 0$ that represent a transition from mode $j$ to mode $k$. A guard set is empty if no feasible transitions exist between modes $(j, k) \in \mathcal{I} \times \mathcal{I}$. Transitions include a reset map $\mathbf{x}^+ = (\mathbf{q}, \dot{\mathbf{q}}^+) = \mathcal{R}_{jk}(\mathbf{q}, \dot{\mathbf{q}}^-)$ that maps the pre-transition state to the post-transition state. This reset map is the identity function for transitions that do not involve impacts. For transitions with impacts, it encodes the instantaneous velocity change.

### 2.4.2. Primitive Sequence Planning

The purpose of the primitive sequence planner is to choose the contact mode order for the motion plan. Given a mathematical description of a hybrid system, the initial state $\mathbf{x}_{\mathrm{o}}$, and desired final state $\mathbf{x}_{\mathrm{f}}$ (or $m$ constraints on the goal region), determine the number of contact modes $N$, and the mode order $\{i_p \mid i_p \in \mathcal{I},\ p = 1 \ldots N\}$ connecting the initial and final states.

We first compile a transition map using the information about the hybrid system derived in Section 2.4.1. This map represents the topology of the state-control space with the $\mathcal{I}$ modes as the nodes, and the feasible transitions $\mathcal{G}_{jk}$ as the edges. The map includes information on whether transitions are smooth or have impacts that cause state discontinuities according to the reset map $\mathcal{R}_{jk}$. It also has information on how many additional constraints are gained or lost when a transition occurs which can give insight into how robust a transition is to state uncertainty. An example of such a transition map is shown in Figure 2.3 for the block-manipulator system analyzed in Section 2.5.

Reaching a transition point by moving along the state axis depends on the continuous evolution of the system state, whereas transitions along the control axis can happen instantaneously. For example, we cannot catch an object in free flight until the object and manipulator are in contact, but we can instantaneously release a nonprehensile contact by accelerating away from an object. We can therefore classify transitions as controlled, partially controlled, or state determined, depending on the constraints that are active on the guard set[2].

[2]If control rate limits are applied to the manipulator, then all transitions must evolve over time, but we assume direct acceleration control.

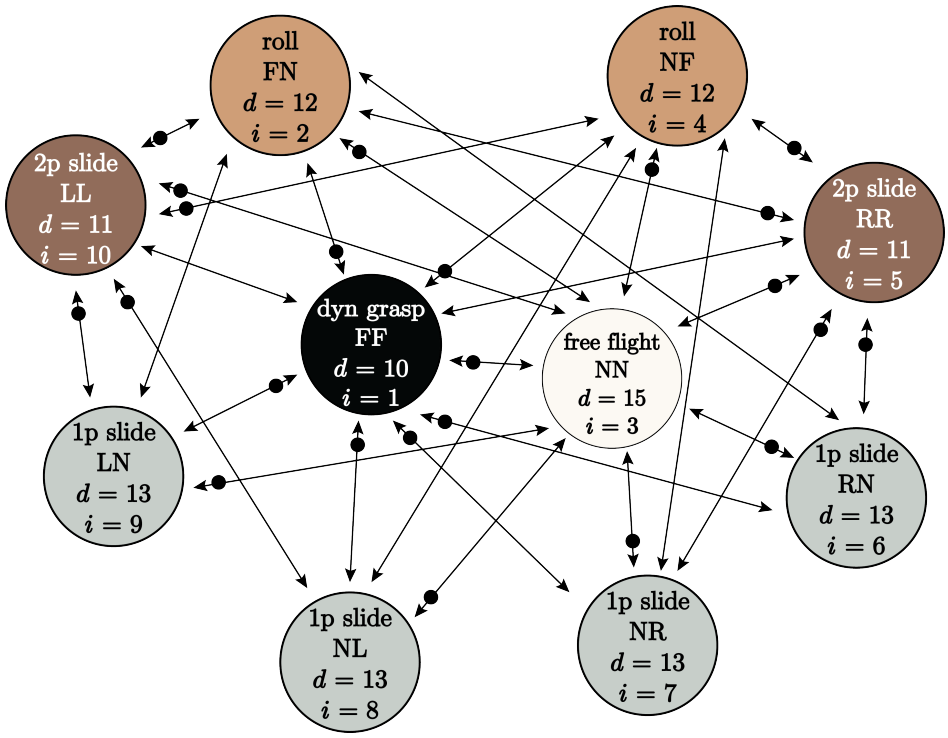



Figure 2.3. Diagram showing the transition map for the block-manipulator system in Figure 2.4. The map only considers two of the corners of the block in contact with one surface of the manipulator. This results in 10 of the 25 total contact modes for the block with one edge of the manipulator. Each node shows the mode name, the contact state at each of the two contacts, the number of free dimensions, and the mode number $i$. The two letters for each contact represent whether the left and right contact are fixed (F), not in contact (N), sliding left (L), or sliding right (R). The total number of dimensions are the six block states, six manipulator states, and the three controls resulting in a fifteen-dimensional system. Free flight is the only unconstrained mode, and all other modes must satisfy contact and velocity constraints which reduce the dimension. The black dots on some edges indicate that the transition requires an impact to occur.

To create the mode sequence, we first choose initial and final modes with domains of applicability $\mathcal{D}_i$ that contain the given initial and final states $\mathbf{x}_\mathrm{o}$ and $\mathbf{x}_\mathrm{f}$. These modes are not necessarily unique because a given state can be in more than one contact mode. In the simplest case, the desired motion can be achieved within a single contact mode,

and in that case $N = 1$ and the high-level mode planner is finished. For the more general case, a motion-planning algorithm can be used to plan a sequence of modes through the transition map that connects the initial and final states.

### 2.4.3. Transition-State Planning

Once a set of contact modes is chosen, the transition-state-control pairs between them must be determined. Given the hybrid system description and the contact mode order $\{i_p \mid i_p \in \mathcal{I},\ p = 1 \dots N\}$, pick transition states $\{(\mathbf{x}_p, \mathbf{u}_p) \mid p = 1 \dots (N-1)\}$, where $\mathcal{G}_{i_p i_{p+1}}(\mathbf{x}_p, \mathbf{u}_p) = 0$. For $p = N$, no transition occurs, but we set the desired "transition state" $(\mathbf{x}_p, \mathbf{u}_p)$ as the desired final state of the system $(\mathbf{x}_\mathrm{f}, \mathbf{u}_\mathrm{f})$.

Determining the transition state is difficult because the union of domains of applicability is a lower-dimensional subspace of the full state-control space. Therefore we must have a method to choose transitions that lie on the guard set $\mathcal{G}_{jk}(\mathbf{x}, \mathbf{u})$. This problem is related to sample-based motion planning for robots with pose constraints [**9**].

### 2.4.4. Single-Mode Motion Planning

With the transition points chosen, a motion planner determines a set of controls for each mode that brings the object and manipulator from the initial state in that mode to the desired transition state out of the mode. Given a hybrid system description, the mode order, and transition states $\{(i_p, \mathbf{x}_p, \mathbf{u}_p) \mid p = \{1 \dots N\}\}$, the task is to find a time $t_p$ and set of controls $\mathbf{u}(t)$ for $\{t \mid t_{p-1} \le t \le t_p\}$ that brings the system from $(\mathbf{x}_{p-1}, \mathbf{u}_{p-1})$ to $(\mathbf{x}_p, \mathbf{u}_p)$ while satisfying dynamics $\mathbf{f}_{i_p}$.

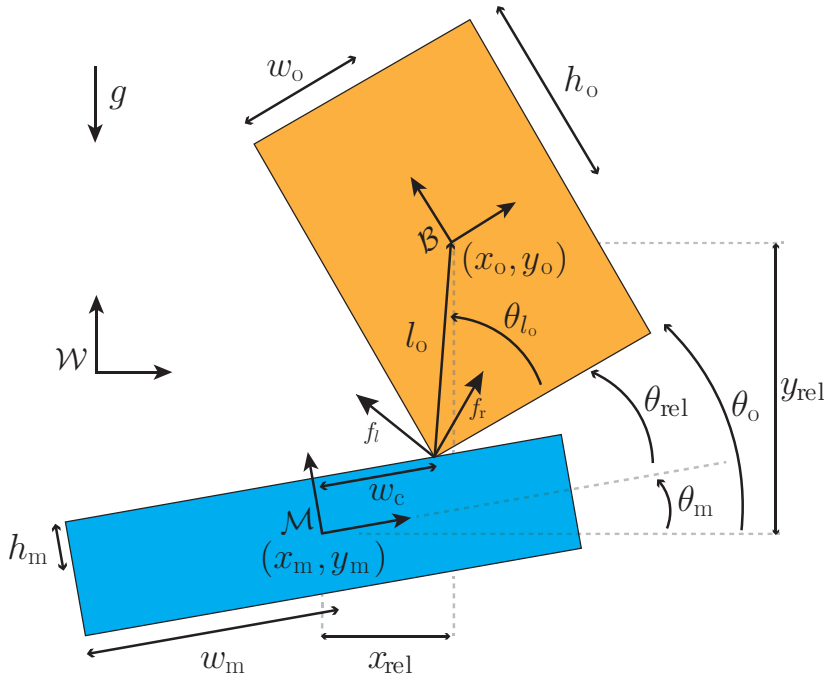



Figure 2.4. Diagram showing the manipulator (blue) and the block (orange), relevant measurements and parameters, and the world $\{\mathcal{W}\}$, body $\{\mathcal{B}\}$, and manipulator $\{\mathcal{M}\}$ frames.

This can be done using a variety of motion planners such as direct and indirect optimization or sample-based methods. The motion planner needs to account for the different system dynamics in each of the modes, and ensure that the desired trajectory does not result in undesired mode transitions. Some modes are underactuated which raises important issues in trajectory planning and control [**39**] [**40**].

The primitive-sequence planning, transition-state planning, and single-mode motion planning can be implemented as an iterative process. If the planner fails at any point, it can return to previous steps and calculate new transition states or mode sequences. This could allow algorithms that are guaranteed to approximately search the space of all possible solutions eventually, either for completeness or optimality. This iterative method

can be further extended to reason about hybrid trajectories that are more robust to uncertainty in system state, modeling parameters, and controls.

### 2.4.5. Primitive Stabilization

Once each of the individual mode plans has been determined, the output is a dynamically feasible trajectory $(\mathbf{x}(t), \mathbf{u}(t))$ for $\{t \mid t_\mathrm{o} \leq t \leq t_N\}$ from the initial state $\mathbf{x}_\mathrm{o}$ to the desired final state $\mathbf{x}_\mathrm{f}$. This trajectory passes through contact modes $\{i_p \mid i_p \in \mathcal{I},\ p = 1 \ldots N\}$ and transition states $\{(\mathbf{x}_p, \mathbf{u}_p) \mid p = 1 \ldots (N-1)\}$.

The final step is to develop a feedback controller $\mathbf{u}_{\mathrm{fbk},i}(\mathbf{x}, t)$ to stabilize the motion plan about deviations from the trajectory in each mode. Some examples are receding-horizon or direct-state-feedback controllers. The controller commands within each mode $i_p$ must be consistent with the constraints of the current contact mode to avoid causing an unwanted transition between modes. We define $\mathbf{u}_\mathrm{des}(t) = \mathbf{u}(t) + \mathbf{u}_\mathrm{fbk}(\mathbf{x}, t)$ as the desired control with feedback, and $\mathbf{u}_\mathrm{proj}(t) = \mathcal{P}_i(\mathbf{x}(t), \mathbf{u}_\mathrm{des}(t))$ as a projection that maps the desired control back to the set of controls consistent with maintaining the current contact mode $i$.

## 2.5. Block and Manipulator Example

The general hybrid planner and control system outlined in Section 2.4 will now be applied to the problem of a rectangular block and manipulator in contact and moving in a 2D plane. The block is initially at rest on the manipulator, and the desired final state has the block resting on a ledge rotated 180 degrees. We use a set of five primitives to

accomplish the goal and these are shown in Figure 2.6. We chose these primitives because they demonstrate controlled, partially controlled, and state-determined transitions between multiple contact modes, as well as feedback control within the rolling balance contact mode. We assume the manipulator's acceleration is directly controlled.

The block-manipulator system is shown in Figure 2.4. The specific contact mode in the figure is the block rolling about the left contact (mode $i = 2$), but the system description is valid for all modes. All angles and positions are measured with respect to the world frame $\{\mathcal{W}\}$ unless otherwise stated. A body frame $\{\mathcal{B}\}$ is attached to the center of the object, and a manipulator frame $\{\mathcal{M}\}$ is attached to the center of the manipulator. The manipulator's pose is represented by $\mathbf{q}_{\mathrm{m}} = [x_{\mathrm{m}}, y_{\mathrm{m}}, \theta_{\mathrm{m}}]^{\mathsf{T}}$, and the object's pose is represented by $\mathbf{q}_{\mathrm{o}} = [x_{\mathrm{o}}, y_{\mathrm{o}}, \theta_{\mathrm{o}}]^{\mathsf{T}}$. The configuration and velocity of the system are denoted as $\mathbf{q} = [\mathbf{q}_{\mathrm{m}}^{\mathsf{T}}, \mathbf{q}_{\mathrm{o}}^{\mathsf{T}}]^{\mathsf{T}}$ and $\dot{\mathbf{q}} = [\dot{\mathbf{q}}_{\mathrm{m}}^{\mathsf{T}}, \dot{\mathbf{q}}_{\mathrm{o}}^{\mathsf{T}}]^{\mathsf{T}}$, respectively. The full state of the system is defined as $\mathbf{x} = [\mathbf{q}^{\mathsf{T}}, \dot{\mathbf{q}}^{\mathsf{T}}]^{\mathsf{T}}$. The variable $\mathbf{q}_{\mathrm{rel}} = \mathbf{q}_{\mathrm{o}} - \mathbf{q}_{\mathrm{m}}$ describes the relative position and orientation between the manipulator frame and the object frame. The variable $\theta_{\mathrm{rel}} = \theta_{\mathrm{o}} - \theta_{\mathrm{m}}$ is the relative orientation between the object and manipulator. The side lengths of the object and manipulator are denoted by the half-widths and half-heights $w_{\mathrm{o}}, h_{\mathrm{o}}, w_{\mathrm{m}}$, and $h_{\mathrm{m}}$. We assume a Coulomb friction cone model at each contact represented by $f_\ell$ and $f_{\mathrm{r}}$, with a friction coefficient $\mu$.

### 2.5.1. Primitive Characterization

In this section we derive some of the $\mathcal{I}$ possible contact modes for the block-manipulator system, along with the mode dynamics and contact and transition constraints. Even for this relatively simple system there are 25 contact modes considering only the block and

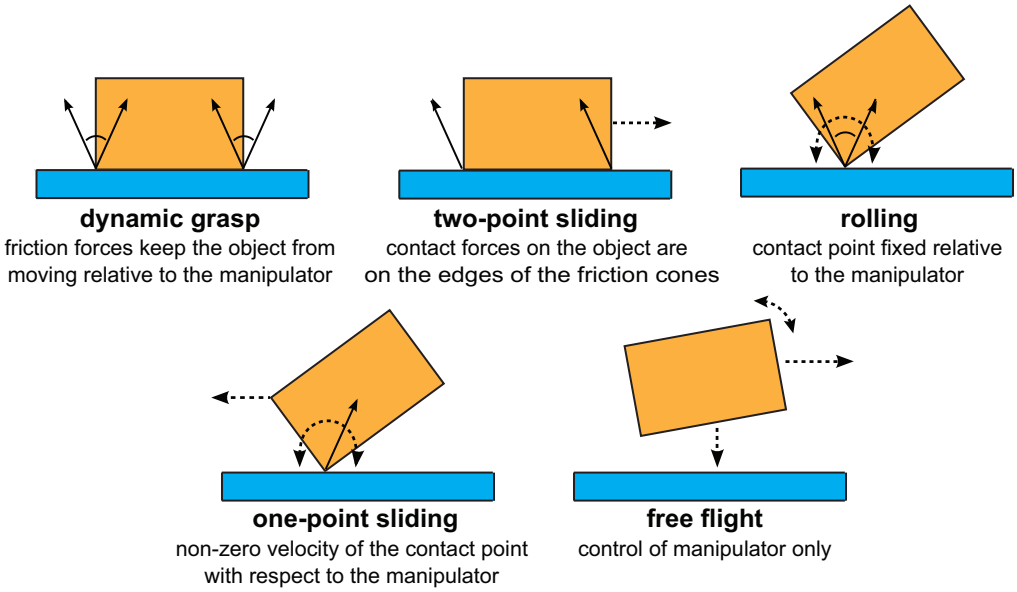



Figure 2.5. Diagram showing the different classes of contact modes of the block-manipulator system. The frictional forces are shown by solid arrows and relative motion is shown by dashed arrows.

the top surface of the manipulator. There are many similar modes where relative motion is in the opposite direction causing forces on the other edge of the friction cone, or the corner(s) in contact are different, so we define five classes of contact modes which are shown in Figure 2.5. A detailed transition map of the contact modes for two corners of the block with one edge of the manipulator is shown in Figure 2.3. In our motion plan we use dynamic grasp, sliding, free flight, and rolling primitives. For dynamic grasp ($i = 1$) the block follows the manipulator as long as the forces that must be applied to the block to have it follow the manipulator's trajectory are inside the wrench cone available from the contacts. For sliding regrasp ($i = 5$) the manipulator accelerates beyond this limit to cause relative motion between the part and the object [**66**] which is the focus of our previous work. Free flight dynamics ($i = 3$) involve no contact so the object follows a parabolic trajectory. We derive the rolling mode ($i = 2$) constraints and dynamics below.

We first define additional parameters shown in Figure 2.4 before deriving the constraints. We denote the contact location distance along the $x$ direction of the manipulator frame $\{\mathcal{M}\}$ as $w_{\mathrm{c}}$. The distance $\ell_{\mathrm{o}}$ represents the diagonal length from the contact corner of the object to its center, and $\theta_{\ell_{\mathrm{o}}}$ is the angle between the base of the object and $\ell_{\mathrm{o}}$. The following kinematic constraints keep the block and manipulator in contact.

$$\begin{aligned} x_{\mathrm{o}} - \ell_{\mathrm{o}}\cos(\theta_{\mathrm{o}} + \theta_{\ell_{\mathrm{o}}}) &= x_{\mathrm{m}} + w_{\mathrm{c}}\cos(\theta_{\mathrm{m}}) - h_{\mathrm{m}}\sin(\theta_{\mathrm{m}}) \\ y_{\mathrm{o}} - \ell_{\mathrm{o}}\sin(\theta_{\mathrm{o}} + \theta_{\ell_{\mathrm{o}}}) &= y_{\mathrm{m}} + w_{\mathrm{c}}\sin(\theta_{\mathrm{m}}) + h_{\mathrm{m}}\cos(\theta_{\mathrm{m}}). \end{aligned} \tag{2.1}$$

Taking the derivative of these constraints yields the Pfaffian constraints $A_2(\mathbf{q})\dot{\mathbf{q}} = 0$, where

$$A_2(\mathbf{q}) = \begin{bmatrix} 1 & 0 & -h_{\mathrm{m}}\cos(\theta_{\mathrm{m}}) - w_{\mathrm{c}}\sin(\theta_{\mathrm{m}}) & -1 & 0 & -\ell_{\mathrm{o}}\sin(\theta_{\mathrm{o}} + \theta_{\ell_{\mathrm{o}}}) \\ 0 & 1 & -h_{\mathrm{m}}\sin(\theta_{\mathrm{m}}) + w_{\mathrm{c}}\cos(\theta_{\mathrm{m}}) & 0 & -1 & \ell_{\mathrm{o}}\cos(\theta_{\mathrm{o}} + \theta_{\ell_{\mathrm{o}}}) \end{bmatrix} \tag{2.2}$$

We assume that the manipulator is directly acceleration controlled so we can choose desired $\mathbf{u} = [\ddot{x}_{\mathrm{m}}, \ddot{y}_{\mathrm{m}}, \ddot{\theta}_{\mathrm{m}}]^{\mathsf{T}}$. The motion of the manipulator results in frictional constraint forces at the contact which we denote as $f_\ell$ and $f_{\mathrm{r}}$ for the left and right edges of the friction cone as shown in Figure 2.4. Due to the contact constraints and the given contact location with the manipulator $w_{\mathrm{c}}$, the full fifteen-dimensional state-control space can be represented by an eleven-dimensional subspace. We choose to represent the system as $\mathbf{x}_{\mathrm{roll}} = [\mathbf{q}_{\mathrm{roll}}^{\mathsf{T}}, \dot{\mathbf{q}}_{\mathrm{roll}}^{\mathsf{T}}]^{\mathsf{T}}$, where $\mathbf{q}_{\mathrm{roll}} = [\mathbf{q}_{\mathrm{o}}^{\mathsf{T}}, \theta_{\mathrm{m}}]^{\mathsf{T}}$. We then use the constraints in (2.2) to derive expressions $f_\ell = g_\ell(\mathbf{x}_{\mathrm{roll}}, \mathbf{u}_{\mathrm{m}})$ and $f_{\mathrm{r}} = g_{\mathrm{r}}(\mathbf{x}_{\mathrm{roll}}, \mathbf{u}_{\mathrm{m}})$ which map given state-control pairs to friction cone forces.

We then change coordinates and assume we can directly apply the controls $\mathbf{u}_{\text{roll}} = [f_\ell, f_\text{r}, \ddot{\theta}_\text{m}]^\mathsf{T}$ which includes the two forces at the contact and the rotational acceleration of the manipulator. Summing the forces and moments acting on the block from $f_\ell$ and $f_\text{r}$ leads to the following control-affine dynamic equations for the object accelerations:

$$\ddot{\mathbf{q}}_\text{o} = \begin{bmatrix} -\frac{\cos(\theta_\mu - \theta_\text{m})}{m_\text{o}} & \frac{\cos(\theta_\mu + \theta_\text{m})}{m_\text{o}} \\ \frac{\sin(\theta_\mu - \theta_\text{m})}{m_\text{o}} & \frac{\sin(\theta_\mu + \theta_\text{m})}{m_\text{o}} \\ \frac{\ell_\text{o}\sin(\theta_\text{m} - \psi - \theta_\mu)}{j_\text{o}} & \frac{\ell_\text{o}\sin(\psi - \theta_m - \theta_\mu)}{j_\text{o}} \end{bmatrix} \begin{bmatrix} f_\ell \\ f_\text{r} \end{bmatrix} + \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix}, \tag{2.3}$$

where $\theta_\mu = \arctan(\mu)$ is the angle the friction cone makes with the contact normal, $\psi = \theta_\text{o} + \theta_{\ell_\text{o}}$ is the angle of the block centerline $\ell_\text{o}$ in the world frame, $m_\text{o}$ is the mass of the object, $j_\text{o}$ is the rotational inertia of the object, and $g$ is the gravity acting on the block.

For rolling dynamics we have analyzed a subspace $\mathbf{x}_{\text{roll}}$ of the total state-space $\mathbf{x}$. As long as the applied frictional forces at the contact $[f_\ell, f_\text{r}]^\mathsf{T}$ are greater than zero, the block is neither slipping on the manipulator nor breaking contact.

### 2.5.2. Primitive Sequence Planning

We now plan the sequence of modes to move the block from its initial state on the manipulator to the goal state on a ledge rotated 180 degrees. Because the ledge is outside of the manipulator's workspace it is clear that the plan will require a throw to get the object to the desired goal state. A transition map between the various contact modes for the block manipulator system is shown in Figure 2.3. We manually choose the following mode sequence to achieve the goal:

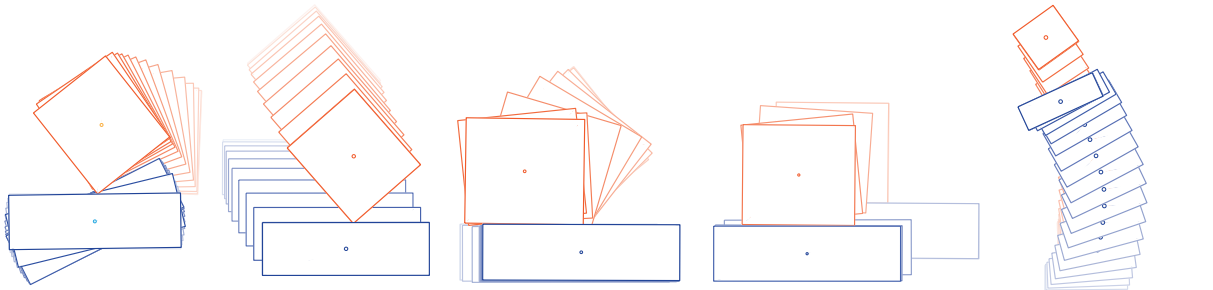



Figure 2.6. Diagrams showing the five motion primitives planned and used in the experiment. The figures are generated from experimental trajectories of a block (orange) and manipulator (blue) transitioning through multiple contact modes.

dynamic grasp + rolling (balance controller) + rolling (roll left controller) + catch +

sliding + catch + dynamic grasp + free flight.

In this paper the catch is not so much a primitive as it is a way to transition between primitives while allowing some uncertainty. Although not necessary to achieve this specific goal, the balanced-rolling mode is included to demonstrate an example of real-time feedback control during the motion.

### 2.5.3. Transition State Planning

The transition into rolling is chosen to have the block near its unstable equilibrium with the manipulator at rest to increase the chance of a successful balance. The transition out of the balance mode is chosen to have the block near the bottom of the workspace to allow more distance to accelerate the block during the throw. The sliding transition is chosen to quickly reposition the block on the manipulator, and the dynamic grasp to free flight transition is chosen so the block will reach the goal state along its free-flight trajectory.

### 2.5.4. Single-Mode Motion Planning

With the mode order and transition states chosen, the next step is to determine trajectories in each mode that join the initial state, the $(N-1)$ transition points, and the final state. In this paper we manually generated manipulator motions that followed fifth-order polynomials, and used simulation to verify the block trajectory and check that the contact constraints were not violated. To achieve the desired goal rotation of the object, we first rotate the block by 90 degrees before throwing it. The initial dynamic grasp motion rotates the block up to a goal angle for the rolling (balancing) mode. The rolling with the balance controller brings the object to a new position, then accelerates to the right causing the object to roll to the left. A catch allows the block to come to rest on the manipulator before sliding along the surface of the object to reposition it. Another catch allows the block to come to rest before using dynamic grasp and then free flight to throw the block to the goal state. Diagrams of these motion plans generated from experimental data are shown in Figure 2.6. We have implemented more automated planning methods such as sequential quadratic programming for optimizing trajectories in different modes but that method was not in the experiment in this paper.

### 2.5.5. Primitive Stabilization

Feedback can be used to account for sources of modeling error and increase the reliability of motion plans. For dynamic-manipulation tasks the set of possible feedback methods is constrained by computation time limitations. Control loops running at a high frequency (1000 Hz in this case) provide only a small window to perform calculations and adjust the open-loop motion plan based on system feedback. For this reason we chose to use a

linearized LQR controller. The nonlinear dynamics for the block can be approximated as a linear system in a small neighborhood of the trajectory. By linearizing the dynamics about the desired trajectory calculated in Section 2.5.4, we can create a time-varying state-feedback controller that stabilizes desired motion primitives about the nominal trajectory.

The output of the LQR feedback controller is a set of controls $\mathbf{u}_{\text{fbk}} = [f_{l,\text{fbk}}, f_{\text{r,fbk}}, \ddot{\theta}_{\text{m,fbk}}]^{\mathsf{T}}$. The desired controls are then $\mathbf{u}_{\text{des}}(t) = \mathbf{u}(t) + \mathbf{u}_{\text{fbk}}(\mathbf{x}, t)$. We use a projection method $\mathbf{u}_{\text{proj}} = \mathcal{P}_i(\mathbf{x}, \mathbf{u}_{\text{des}})$ that maps invalid commands to controls that will not cause an unwanted mode transition. For rolling mode ($i = 2$), the projection maps negative contact forces $f_\ell, f_{\text{r}}$ less than zero to zero to satisfy unilateral contact constraints, and saturates infeasible manipulator accelerations. If the nominal trajectory is far enough from the boundary of feasible controls, then small perturbations about the trajectory will be recoverable with the LQR control output.

## 2.6. Block Manipulation Experiment

### 2.6.1. Experimental Setup

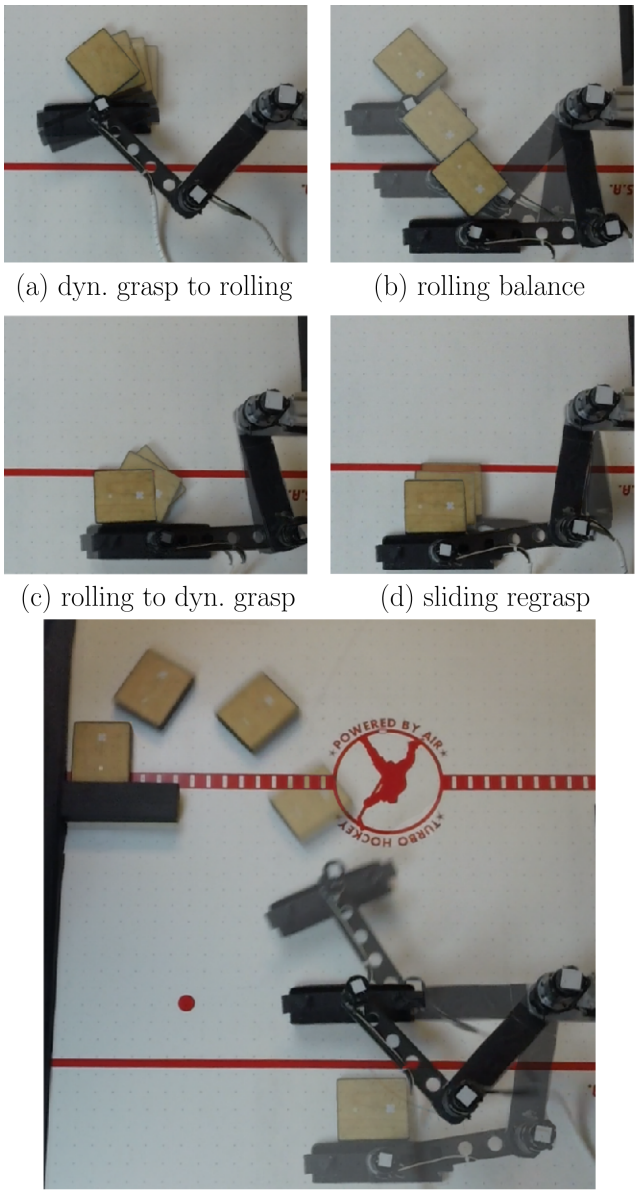
The experimental setup consists of a 3-DOF robot arm that moves in a plane parallel to the surface of an inclined air hockey table. A diagram of the experimental setup is shown in Figure 5.11. Experiments are conducted at 40% full gravity by inclining the table at 24 degrees with respect to horizontal. Each link is actuated by a brushed DC motor with harmonic-drive gearing and current controlled using Junus motor amplifiers. The 1000 Hz motion controller runs on a PC104 embedded computer running the QNX real-time operating system. Vision feedback is given by a 250 Hz IR Optitrack camera.

Desired trajectories and experimental results are transmitted between the PC104 and a PC running MATLAB using a TCP/IP connection.

### 2.6.2. Experiment

The block is initially at rest on the manipulator, and the desired final state has the block resting on a ledge rotated 180 degrees. We use a set of five primitives to accomplish the goal. The initial dynamic grasp motion rotates the block up to a goal angle for the balancing mode. The controlled-rolling primitive brings the object to a new position then accelerates to the right causing it to roll to the left. The manipulator then slides along the surface of the object to reposition it and then throws it to the goal state. Diagrams of these primitives generated from the actual experiment are shown in Figure 2.6. A set of images from the experiment are shown in Figure 2.7 and a video is attached in the supplemental media and at the following link: https://vimeo.com/206086422.

The experiment successfully uses different primitives to transition between contact modes during nonprehensile and dynamic manipulation tasks. It also demonstrates the use of a feedback controller during the manipulation task to stabilize the desired trajectory. Most of the planned motions were reliable over multiple experiments, but the basin of attraction of the balancing controller is relatively small. In the future we plan to improve the reliability of the rolling-mode controller, develop feedback controllers for additional modes, test the repeatability of the experimental results, and analyze tracking performance in individual modes and at the transitions.




(a) dyn. grasp to rolling (b) rolling balance

(c) rolling to dyn. grasp (d) sliding regrasp

(e) dynamic grasp throw

Figure 2.7. Images from the experiment showing the motion executions. The video can be found at the following link: https://vimeo.com/206086422

## 2.7. Conclusions

In this paper we proposed a method for motion planning and feedback control of hybrid, dynamic, and nonprehensile manipulation tasks. We outlined five subproblems—primitive characterization, sequence planning, picking transition states, planning individual motion primitives, and stabilizing individual modes—and then demonstrated the framework by manually planning a sequence of motions for manipulating a block. The motions were demonstrated experimentally with a planar 3R manipulator and block on an inclined air hockey table. Future work will focus on automating the process of generating primitives, choosing mode sequences, planning within single modes, and stabilizing them. We plan to implement additional real-time nonlinear feedback controllers such as sequential action control [**6**] to improve the reliability of planned motions. For contact modes where feedback control is impractical, we can develop methods to explicitly estimate and manage uncertainty. Future experiments testing these plans will allow more in-depth analysis of the framework.

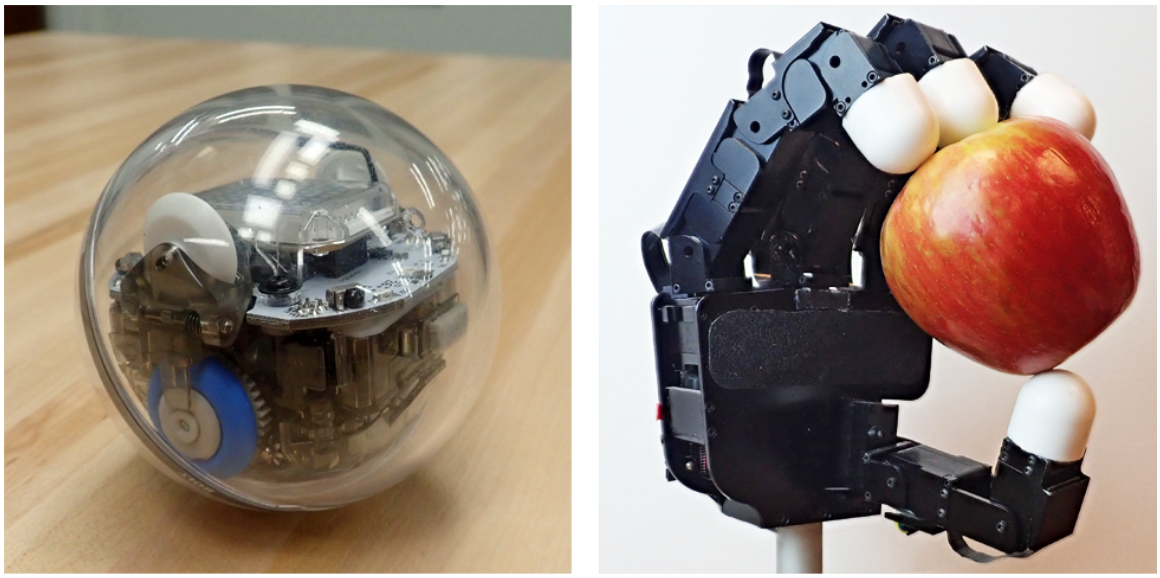
CHAPTER 3

# Kinematic Motion Planning and Feedback Control of Rolling Bodies

## 3.1. Abstract

We gave a high-level formulation in Chapter 2, and in the following chapters we focus specifically on the rolling primitive. This chapter examines the problem of planning and stabilizing the trajectory of one smooth body rolling on the surface of another. The two control inputs are the angular velocity of the moving body about two orthogonal axes in the contact tangent plane; spinning about the contact normal is not allowed. To achieve robustness and computational efficiency, our approach to trajectory planning is based on solving a series of optimization problems of increasing complexity. To stabilize the trajectory in the face of perturbations, we use a linear quadratic regulator. We apply the approach to examples of a sphere rolling on a sphere and an ellipsoid rolling on an ellipsoid. Finally, we explore the robustness and performance of the motion planner. Although the planner is based on non-convex optimization, in practice the planner finds solutions to nearly all randomly-generated tasks, and the solution trajectories are smoother and shorter than those found in previous work in the literature.

The main contributions of this chapter are a robust motion planner that can handle general smooth geometries, a method to test the controllability of linearized rolling



(a) (b)

Figure 3.1. Examples of robot tasks that can be modeled as objects in rolling contact. (a) A ball-type mobile robot on a smooth surface. (b) Robot fingers rolling over a smooth object.

trajectories, and a method to stabilize trajectories from initial-state perturbations. This chapter was published in IEEE Access [**76**]

## 3.2. Introduction

This paper examines the problem of planning and stabilizing the trajectory of one smooth body rolling on the surface of another. This is relevant for systems such as a ball-type mobile robot rolling over smooth terrain (Figure 3.1(a)) or a robot hand planning multi-finger rolling motions to reorient an object (Figure 3.1(b)). Much research on rolling motion planning has been limited to specialized geometries such as planes and spheres. In this work we present a method to generate motion plans and stabilizing feedback controllers for general, smooth, three-dimensional objects in rolling contact.

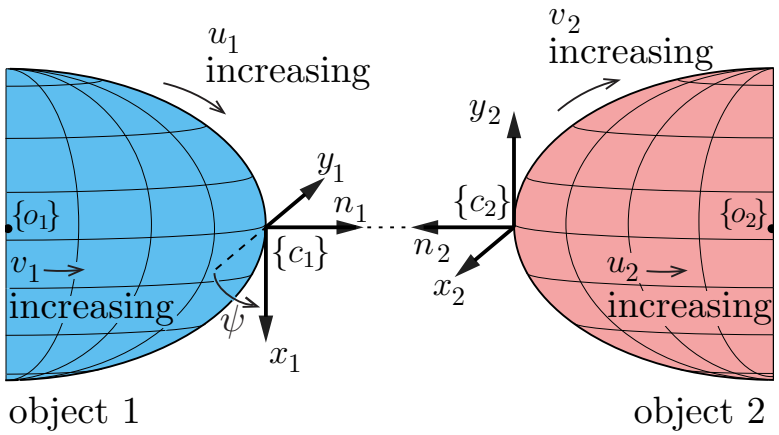



Figure 3.2. Objects 1 and 2 contact at the origin of frames $\{c_1\}$ and $\{c_2\}$, but are shown separated for clarity. Collectively, the contact configuration is written $q = (u_1, v_1, u_2, v_2, \psi)$. The surfaces of objects 1 and 2 are orthogonally parameterized by $(u_1, v_1)$ and $(u_2, v_2)$, respectively. At the point of contact, the unit $x_i$- and $y_i$-axes of the coordinate frame $\{c_i\}$ are in the direction of increasing $u_i$ (and constant $v_i$) and increasing $v_i$ (and constant $u_i$), respectively, and the contact normal $n_i$ is the cross product $x_i \times y_i$. Rotating frame $\{c_2\}$ by $\psi$ about the $n_1$-axis of frame $\{c_1\}$ aligns the $x_2$-axis of frame $\{c_2\}$ and the $x_1$-axis of frame $\{c_1\}$. The controls for pure rolling (no relative spin about the contact normal) are the relative angular velocities $\Omega = (\omega_x, \omega_y)$ about the $x_2$- and $y_2$-axes of the contact frame $\{c_2\}$ [**75**].

The two rolling objects are modeled as a well-known nonholonomic system with five degrees of freedom and two controls. Collectively the contact configuration is written $q = (u_1, v_1, u_2, v_2, \psi)$, which gives the contact location $(u_1, v_1)$ on object 1, $(u_2, v_2)$ on object 2, and the angle of "spin" $\psi$ between contact frames (Figure 5.2). The two control inputs are the angular velocities $\Omega = (\omega_x, \omega_y)$ of the moving body about two orthogonal axes in the contact tangent plane; spinning about the contact normal is not allowed. We refer to the no-spin assumption as "pure rolling." Since we assume the velocities are directly controlled, we refer to the equations of motion as the "first-order kinematics."

Our approach to trajectory planning is based on solving a series of optimization problems of increasing complexity. We first solve a convex problem that uses the two rolling velocity inputs to drive two of the five configuration variables directly to their desired values. This motion serves as the initial trajectory guess for direct-collocation constrained optimization. Using this initial guess, the optimization solves the full five-dimensional trajectory-planning problem. The optimization first solves for a trajectory history that is represented coarsely, using a small number of state and control segments. The solved-for controls are then simulated by a more accurate, higher-order numerical integration method than the integrator implicit in the constraints in the nonlinear optimization. If the simulated trajectory satisfies the error tolerances, the problem is solved. If not, the previous solution is used as an initial guess, the number of state and control segments is increased, and the optimization is called again.

The motion planner is structured this way to balance three goals: 1) increasing the likelihood of finding a solution; 2) decreasing the computation time required to find a solution; and 3) optimizing the quality of the solution, as measured by the trajectory length and control cost. In our tests, an initial optimization with a fine control discretization often takes an unnecessarily long time to converge or even fails to converge to a feasible solution. The coarse initial guess followed by successive refinement yields higher-quality solutions faster and more consistently. The iterative refinement process acts as a form of regularization.

To stabilize a planned trajectory to small perturbations, we use linear feedback control based on a linear quadratic regulator. For this to be successful, the linearized trajectory

must be controllable, so we examine the controllability of rolling trajectories and provide examples of uncontrollable trajectories.

Our primary contributions are a robust motion planner for generating rolling motions of general smooth objects and an approach to stabilize those trajectories.

### 3.2.1. Paper Outline

Section 5.3 reviews previous work related to this paper. Section 3.4 summarizes the rolling kinematics, and Section 4.3 formally states the problem we are solving. Section 3.6 outlines the design of the motion planner, and Section 3.7 analyzes the controllability of rolling trajectories. Section 3.8 demonstrates planning for a sphere rolling on a sphere and an ellipsoid on an ellipsoid, applying feedback controllers to stabilize the planned trajectories from perturbations to the initial configurations. Section 3.9 analyzes the robustness and performance of the planner for random goal states and different methods of generating initial trajectory guesses. Section 5.10 summarizes the results and describes planned future work. Some background on differential geometry and derivations of terms used in the kinematics expressions are given in the Appendix.

## 3.3. Related Work

### 3.3.1. Modeling of Rolling Surfaces

First-order kinematics addresses the rolling problem where the relative contact velocities are directly controlled. The second-order kinematics is a generalization of the first-order model where the relative accelerations at the contact are controlled. Dynamic rolling assumes that forces and torques are controlled.

Cai and Roth derive the first- and second-order contact kinematic equations for two objects in contact [**13**]. They focus on the special cases of pure translation and pure rotation about the contact point. They only consider the four-dimensional evolution of the contact points on the objects, not the full five-dimensional configuration.

Montana derives the first-order contact kinematics for two 3D objects in contact [**49**]. His method models the full five-dimensional configuration space, but it is not easily generalized to second-order kinematics or dynamics. First- and second-order contact equations were derived by Sarkar et al. in [**61**] and published again in later works [**62**, **63**]. Errors in the published equations for second-order contact kinematics in [**61**–**63**] were addressed and corrected in our recent work [**75**]. Each of [**49**, **61**–**63**, **75**] assumes an orthogonal parameterization, as shown in Figure 5.2. Chitour et al. survey results on the pure rolling problem for surfaces represented as manifolds and analyzed using Riemannian geometry and geometric control theory [**15**].

### 3.3.2. Motion Planning For Rolling Systems

First- and second-order roll-slide kinematics, as discussed above, allow sliding at the contact, but we focus on planning for first-order systems in pure-rolling contact. This is a well-known driftless nonholonomic system, where the rolling constraints do not necessarily translate to constraints on the achievable relative configuration.

Lafferriere and Sussmann give a method for motion planning and feedback control for nonholonomic systems without drift, and the methods are exact for systems with nilpotent, or feedback-nilpotentizable, Lie algebras [**34**]. Murray provides a method of finding a nilpotent basis for nonholonomic systems and shows how to generate plans for

systems including a disk rolling on a plane [**52**]. The general pure-rolling problem does not satisfy the nilpotent condition, so Oriolo and Vendittelli generalize the method and present an algorithm based on nilpotent approximation and iterative steering to achieve asymptotic stability for the plate-ball system [**53**].

Murray and Sastry outline a special class of nilpotentizable, nonholonomic systems called chained-form systems. They introduce sufficient conditions to check if a system can be converted to chained form, and they give a method to plan motions between arbitrary states for these systems (e.g., a kinematic car or a car pulling a trailer) [**51**]. Fliess et al. outline methods to represent nonlinear systems as differentially flat [**22**]. Such systems are amenable to simplified motion planning methods. Bicchi and Sorrentino show that the rolling system cannot be put into chained form and is not differentially flat, so those methods cannot be applied [**10**].

There are many works on motion planning for rolling systems that assume special geometries such as planes and spheres. Li and Canny derive the first-order contact equations for rolling objects parameterized by orthogonal coordinate systems, analyze the controllability properties, and provide a geometric motion planning algorithm for a sphere on a plane [**36**]. Marigo and Bicchi plan motions for general surfaces on a plane in the presence of obstacles using local approximations of Li and Canny's method [**46**]. Alouges et al. demonstrate the use of numerical continuation methods to generate open-loop motion plans for a general surface rolling on a plane without slipping [**2**]. Rehan and Reyhanoglu derive geometric planners for a sphere rolling on a smooth surface and demonstrate the method for a sphere rolling on a plane and on another sphere [**58**].

### 3.3.3. Rolling Controllability and Feedback

A pure-rolling system is locally controllable from a given initial configuration if the set of locally reachable configurations, using only the two controls, is five dimensional. Li and Canny study the controllability of rolling bodies by taking Lie brackets of the rolling vector fields to generate higher-order vector fields [**36**]. They conclude that a sphere can reach any contact configuration on the plane by pure rolling, and that a sphere can reach any contact configuration by pure rolling on another sphere if their radii are different. For more general body geometries, however, deriving symbolic Lie bracket vector fields is cumbersome.

Marigo and Bicchi study the controllability of rolling bodies with regular surfaces, derive admissibility conditions for rolling contacts, and define necessary conditions for the reachability of rolling contacts [**45**]. They provide an in-depth analysis of the types of surfaces and initial conditions that cause the reachable set to drop from five to two dimensions. Agrachev and Sachkov (Section 24.4 of [**1**]) show that two objects in rolling contact are controllable when their local Gaussian curvatures are not equal. When the local Gaussian curvatures are equal, the three nonholonomic constraints become integrable, reducing the reachable set to a two-dimensional subset of the full five-dimensional configuration space. Krakowski et al. provide examples of when the rolling system fails to be controllable [**33**]. Feedback stabilization of rolling is addressed by Walsh et al., who present a control law to exponentially stabilize linearized trajectories [**72**]; Sarkar et al., who demonstrate the use of feedback linearization to control dynamic rolling motions for two planes in contact with a sphere [**63**]; and Choudhury and Lynch, who stabilize

the orientation of a ball rolling in an ellipsoidal dish actuated along a single degree of freedom [**16**].

### 3.4. Rolling Kinematics

In this paper an object is a two-dimensional surface $S$ embedded in 3D space. An open, connected subset of a surface $S$ is defined as $S_k$. For a given $S_k$, the surface is parameterized by the coordinates $(u_k, v_k) \in U_k \subset \mathbb{R}^2$, and the shape is given by $f_k : U_k \to \mathbb{R}^3 : (u_k, v_k) \mapsto (x_k, y_k, z_k)$, where the $(x_k, y_k, z_k)$ coordinates are expressed in a frame fixed to the body. The triplet $(S_k, f_k, U_k)$ is called a coordinate chart, and a set of coordinate charts is called an atlas for $S$ if the union of the $S_k$ fully covers the surface $S$.

Throughout this paper we assume that rolling motion is confined to a single coordinate chart of object 1, $(S_1, f_1, U_1)$, and a single coordinate chart of object 2, $(S_2, f_2, U_2)$.

An example coordinate chart for a sphere covers all points of the sphere except for the poles, with

$$f : U \to \mathbb{R}^3 : (u, v) \mapsto \\ (\rho \sin(u) \cos(v), \rho \sin(u) \sin(v), \rho \cos(u)), \tag{3.1}$$

where $\rho$ is the radius of the sphere, $u$ satisfies $0 < u < \pi$, and $v$ satisfies $-\pi < v < \pi$.

It is assumed that $f$ is continuous up to the second derivative (class $C^2$) so that the local contact geometries (contact frames and curvature associated with the first and second derivatives of $f$, respectively) are uniquely defined. We also assume that coordinate charts are orthogonal ($\frac{\partial f}{\partial u} \cdot \frac{\partial f}{\partial v} = 0$). Any smooth, regular surface can be locally represented this way.

The configuration space of objects in rolling contact (see Figure 5.2) can be parameterized by $q = (u_1, v_1, u_2, v_2, \psi)$, where $U_1 = (u_1, v_1)$ describes the contact point on the surface of object 1, $U_2 = (u_2, v_2)$ describes the contact point on the surface of object 2, and $\psi$ describes the angle of "spin" between contact frames $\{c_1\}$ and $\{c_2\}$ about their common normal.

The linear velocity relating the relative motion between objects expressed in $\{c_2\}$ can be written as $V = (V_x, V_y, V_z)$. Similarly, the relative angular velocity at the contact expressed in $\{c_2\}$ can be written as $\omega = (\omega_x, \omega_y, \omega_z)$. The controls for pure rolling are the relative angular velocities $\Omega = (\omega_x, \omega_y)$ about the $x$- and $y$-axes of the contact frame at object 2, which defines the contact tangent plane. We use the first-order kinematics derived by Sarkar et al. in [**61**] with the pure-rolling assumptions applied ($V_x = V_y = V_z = \omega_z = 0$). These are equivalent to the equations in [**49**] for the first-order analysis, but are chosen to allow for direct extension to second-order planning in future work. The pure-rolling kinematics reproduced from [**61**] are

$$
\begin{aligned}
\dot{U}_1 &= (\sqrt{\mathbf{G}_1})^{-1}\mathbf{R}_\psi(\widetilde{\mathbf{H}}_1 + \mathbf{H}_2)^{-1}\begin{bmatrix} -\omega_y \\ \omega_x \end{bmatrix}, \\
\dot{U}_2 &= (\sqrt{\mathbf{G}_2})^{-1}(\widetilde{\mathbf{H}}_1 + \mathbf{H}_2)^{-1}\begin{bmatrix} -\omega_y \\ \omega_x \end{bmatrix}, \\
\dot{\psi} &= \sigma_1\Gamma_1\dot{U}_1 + \sigma_2\Gamma_2\dot{U}_2,
\end{aligned}
\tag{3.2}
$$

where $\mathbf{G}_i$ is the metric tensor of object $i$, the $2 \times 2$ rotation matrix $\mathbf{R}_\psi$ is defined as

$$\mathbf{R}_\psi = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ -\sin(\psi) & -\cos(\psi) \end{bmatrix},$$

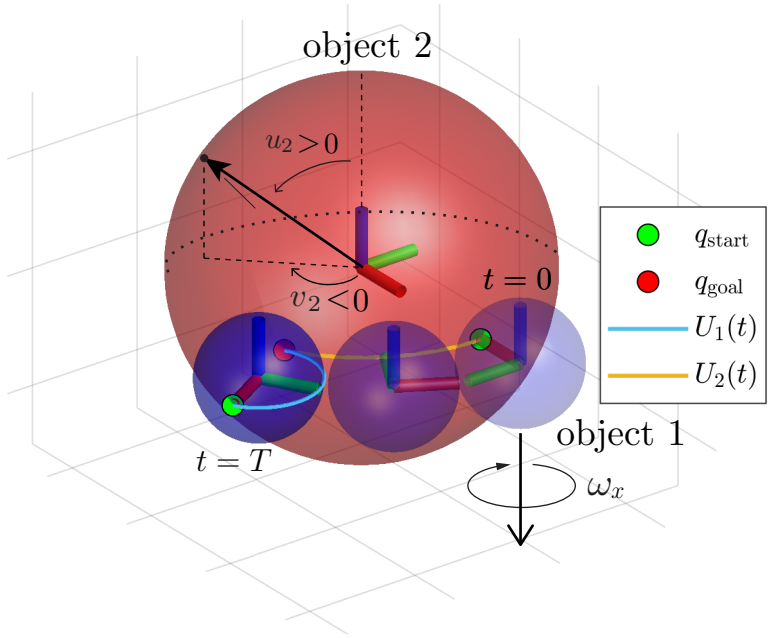
$\mathbf{H}_i$ is a $2\times2$ matrix that gives the curvature of the surface, $\widetilde{\mathbf{H}}_1$ is defined as $\widetilde{\mathbf{H}}_1 = \mathbf{R}_\psi \mathbf{H}_1 \mathbf{R}_\psi$, the scalar $\sigma_i$ is defined as $\sigma_i = \sqrt{g_{22,i}/g_{11,i}}$ where $g_{11,i}$ and $g_{22,i}$ are the diagonal entries of the metric tensor $\mathbf{G}_i$, and $\Gamma_i$ is a $1 \times 2$ matrix of the Christoffel symbols of the second kind. Bold, capitalized letters indicate matrices, and derivations of these expressions are given in the Appendix and in [**61**].

Eq. (3.2) can be expressed in control-affine form as:

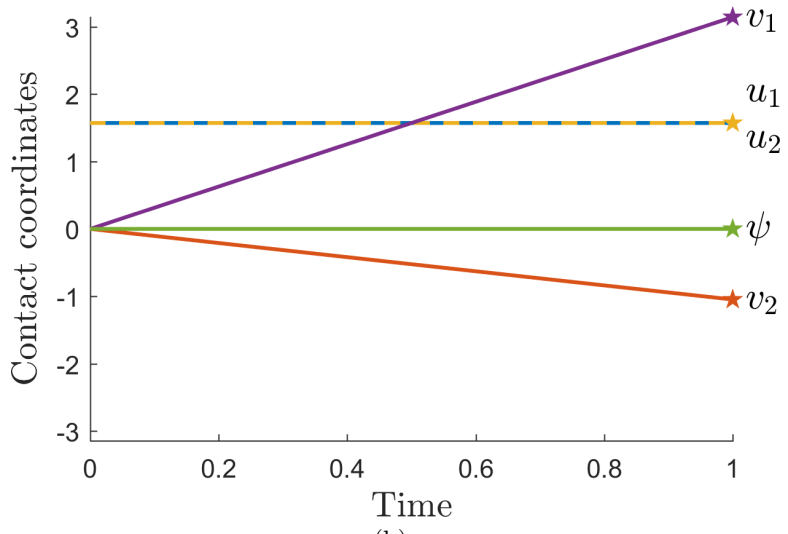
$$\dot{q} = \mathcal{F}(q)\Omega = \begin{bmatrix} (\sqrt{\mathbf{G}_1})^{-1}\mathbf{R}_\psi \\ (\sqrt{\mathbf{G}_2})^{-1} \\ \mathbf{T}_1\mathbf{R}_\psi + \mathbf{T}_2 \end{bmatrix} \mathbf{H}_{\text{rel}}^{-1} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \Omega, \tag{3.3}$$

where $q = [u_1\ v_1\ u_2\ v_2\ \psi]^\mathsf{T}$ are the states, $\Omega = [\omega_x\ \omega_y]^\mathsf{T}$ are the controls, $\mathbf{T}_i = \sigma_i\Gamma_i(\sqrt{\mathbf{G}_i})^{-1}$, and $\mathbf{H}_{\text{rel}} = (\widetilde{\mathbf{H}}_1 + \mathbf{H}_2)$ represents the relative curvature at the contact.

An example of two spheres rolling with constant relative rotational velocity $\Omega = (\omega_x, 0) = (\frac{4\pi}{3}, 0)$ along their respective "equators" is shown in Figure 3.3. Figure 3.3(a) shows a visualization of the spheres, the contact paths on the surface of each object $U_i(t)$, and the coordinate chart for object 2. Figure 3.3(b) shows the values of the contact coordinates during the rolling motion.




(a)




(b)

Figure 3.3. Example of object 1, the blue sphere of radius $\rho_1 = 1$, rolling on the equator of object 2, the red sphere of radius $\rho_2 = 3$. The coordinate charts are given by Eq. (3.1), the initial conditions are $q(0) = (\frac{\pi}{2}, 0, \frac{\pi}{2}, 0, 0)$, and the constant relative rotational velocity is $\Omega = (\omega_x, 0) = (\frac{4\pi}{3}, 0)$. A visualization with the start and goal locations and the contact trajectories $U_i(t)$ is shown in (a). Note that the controls $\Omega$ are measured in the object 2 contact frame $\{c_2\}$ with the $x_2$-axis pointing downwards and the contact normal $n_2$ pointing out of object 2. A plot of the contact coordinates is shown in (b), with the desired goal states $q_{\text{goal}}$ represented by stars. Note that $u_1$ and $u_2$ are equal throughout the trajectory and therefore overlap.

### 3.5. Problem Statement

The goal is to find a pure-rolling trajectory and a stabilizing feedback controller from an initial state $q_{\text{start}}$ to a goal state $q_{\text{goal}}$ for two bodies in pure-rolling contact. We assume that a path between $q_{\text{start}}$ and $q_{\text{goal}}$ exists within a single pair of coordinate charts ($f_1$ on object 1 and $f_2$ on object 2). An "admissible" trajectory is defined as a set of states and controls $\xi(t) = (q(t), \Omega(t))$, from $t = 0$ to the final time $t = T$, that satisfies the first-order pure-rolling kinematics in Eq. (3.3). A "valid" trajectory is defined as an admissible trajectory that also satisfies $q_{\text{error}}(T) < \eta$, where $\eta$ is the tolerance on the final state error and $q_{\text{error}}(T) = ||q(T) - q_{\text{goal}}||$, where $|| \cdot ||$ corresponds to a weighted norm that puts contact parameter errors and spin angle errors in common units. (Throughout the rest of this paper, we use the Euclidean norm.) A stabilizing state-feedback controller about a trajectory is defined as $\Omega_{\text{fbk}}(q, t)$. With these definitions, the problem can be stated as follows:

**Given:** The surface parameterizations $(f_1, f_2)$, the states $(q_{\text{start}}, q_{\text{goal}})$, and the rolling time $T$,

**find:** (1) a valid rolling trajectory $\xi_{\text{sol}}(t)$ for $t \in [0, T]$ that brings the system from $q(0) = q_{\text{start}}$ to $q(T) = q_{\text{goal}}$ and (2) a feedback controller $\Omega_{\text{fbk}}(q, t)$ that stabilizes that trajectory.

In the following section we outline the multi-step process we developed to solve the pure-rolling motion planning and control problem.

## 3.6. Motion Planning

A multi-step algorithm is established to solve the motion planning problem presented in Section 4.3. Given the model parameters, the start state $q_{\text{start}}$, and goal state $q_{\text{goal}}$, the motion planner solves multiple problems of increasing complexity to find a valid trajectory.

The first step is a two-state control (TSC) method that solves a simplified motion planning problem that directly controls two of the five configuration variables. The output of the TSC planner is the trajectory $\xi_{\text{TSC}}(t) = (q_{\text{TSC}}(t), \Omega_{\text{TSC}}(t))$ consisting of the configurations and controls as a function of time. The second step is the iterative direct collocation (iDC) method that takes $\xi_{\text{TSC}}$ as the initial trajectory guess and runs an optimization to find a coarse rolling trajectory (large time steps, simple integration method) for the full five-dimensional configuration. The control output from the optimization $\Omega_{\text{iDC}}(t)$ is then used to numerically integrate the first-order kinematics using a higher-order integrator (MATLAB's `ode45`), and the goal error $q_{\text{error}}(T)$ is calculated. If the trajectory is valid ($q_{\text{error}}(T) < \eta$), then the optimization is stopped and the trajectory is returned. If the terminal condition is not satisfied, the previous trajectory serves as the initial trajectory guess for a finer direct collocation optimization with twice as many collocation segments ($N \to 2N$). This is repeated until a valid trajectory is found, the maximum number of iDC iterations is reached, or the optimization converges to an invalid point. A flowchart of the algorithm is shown in Figure 3.4, and the details of the individual methods are given below.

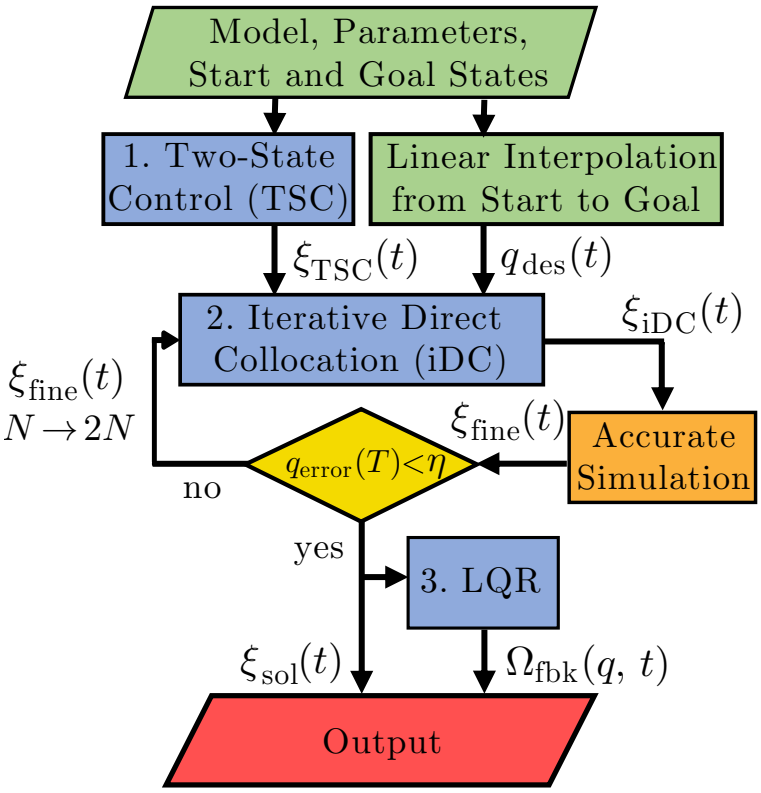



Figure 3.4. Flowchart of the multi-step motion planning algorithm and stabilization method outlined in Sections 3.6 and 3.7, respectively. The inputs are the start/goal states, the model, and the parameters. Two-state control (see Section 3.6.1) is used to generate the initial trajectory guess $\xi_{\text{TSC}}(t) = (q_{\text{TSC}}(t), \Omega_{\text{TSC}}(t))$ for the iterative direct collocation, and $q_{\text{des}}(t)$ is the straight-line desired path for the cost function in Eq. (5.47). Each output trajectory $\xi_{\text{iDC}}(t) = (q_{\text{iDC}}(t), \Omega_{\text{iDC}}(t))$ is recalculated using a higher-order integration method and the goal error tolerance is checked ($q_{\text{error}}(T) < \eta$). If the trajectory is not valid, it is used as the initial trajectory guess for the next iteration of the direct-collocation method with twice as many segments ($N \to 2N$). This is repeated until a valid trajectory is found, the maximum number of iDC iterations is reached, or the optimization converges to an invalid point. The linear quadratic regulator (LQR) step outputs a feedback control law $\Omega_{\text{fbk}}(q, t)$ that stabilizes the solution trajectory $\xi_{\text{sol}}(t)$.

### 3.6.1. Two-State Control (TSC)

The purpose of the two-state control method is to obtain the control input that moves two states along the shortest path to the goal state. The choice of which coordinates to control will bias the optimization solutions toward different paths in the state-space. In this paper we choose to control the coordinates $U_2(t) = [u_2(t)\ v_2(t)]^\mathsf{T}$. An analysis of the planner performance for different initial trajectory guess methods is included in Section 3.9.2.

We first linearly interpolate the object 2 coordinates from $U_2(0)$ to $U_2(T)$. With the trajectory $U_2(t)$ known, we calculate the control input $\Omega(t)$ from the second expression in Eq. (3.2), which results in:

$$\begin{bmatrix} \omega_x(t) \\ \omega_y(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\widetilde{\mathbf{H}}_1(t) + \mathbf{H}_2(t)) \sqrt{\mathbf{G}_2}(t) \dot{U}_2(t). \tag{3.4}$$

We then use $q_{\text{start}}$, $\Omega(t)$, and the other two kinematics expressions of Eq. (3.2) to calculate the trajectories of the remaining three states $u_1(t)$, $v_1(t)$, and $\psi(t)$. This method generates a trajectory $\xi_{\text{TSC}}(t) = (q_{\text{TSC}}(t), \Omega_{\text{TSC}}(t))$ that tracks the straight-line path between the start and goal contact locations on object 2. Examples are shown in Figures 3.3(b) and 3.8(a).

### 3.6.2. Iterative Direct Collocation (iDC)

The initial trajectory $\xi_{\text{TSC}}(t)$ is admissible, but only two of the five states reach the desired goal states. We therefore perturb the input trajectory from the TSC method so

that $q(T) = q_{\text{goal}}$, and to do this we use direct collocation. We first describe the details of the direct collocation method, and then outline our iterative version.

Direct collocation is a method for trajectory optimization that optimizes an objective function $J(\xi(t)) = J(q(t), \Omega(t))$ using polynomial spline approximations of the continuous states and controls. We chose to use trapezoidal collocation where the control trajectory $\Omega(t)$ is represented by piecewise-linear splines, the state trajectory $q(t)$ is represented by a quadratic spline, and the trapezoidal rule is used for integration. Higher-order representations such as Hermite-Simpson collocation can also be used but with increased computational cost [**32**]. We define the objective function $J(q(t), \Omega(t))$ as the sum of the terminal cost and the running cost and omit the dependence on $t$ for clarity:

$$
\begin{aligned}
J(q, \Omega) &= m(q(T)) + \int_0^T l(q, \Omega) dt, \\
m(q(T)) &= \frac{1}{2}(q(T) - q_{\text{goal}})^\mathsf{T} \mathbf{P}_1 (q(T) - q_{\text{goal}}), \\
l(q, \Omega) &= \frac{1}{2}(q - q_{\text{des}})^\mathsf{T} \mathbf{Q} (q - q_{\text{des}}) + \frac{1}{2} \Omega^\mathsf{T} \mathbf{R} \Omega,
\end{aligned}
\tag{3.5}
$$

where $\mathbf{P}_1$, $\mathbf{Q}$, and $\mathbf{R}$, penalize goal-state error, desired trajectory deviation, and control cost respectively, and $q_{\text{des}}(t)$ is a nominal trajectory. The path $q_{\text{des}}(t)$ is chosen as the linear interpolation from $q_{\text{start}}$ to $q_{\text{goal}}$, which penalizes motions that do not move $q$ towards the goal. Note that $q_{\text{des}}(t)$ is not admissible in general.

The collocation method divides the trajectory $\xi(t)$ into $N$ segments, and the $N + 1$ nodes at the ends of each segment are called collocation points. Each collocation point is expressed as $\xi_k(t) = (q(t_k), \Omega(t_k))$ for $k \in [0, \ldots, N]$. For systems with $m$ state variables and $n$ control variables there are a total of $(N + 1)(m + n)$ collocation points. The dynamics between each pair of sequential collocation points are enforced by the following

condition:

$$q_{k+1} - q_k = \frac{1}{2}\Delta t_k(\mathcal{F}(q_{k+1})\Omega_{k+1} + \mathcal{F}(q_k)\Omega_k), \quad k \in [0, \ldots, N-1], \tag{3.6}$$

where $\Delta t_k = (t_{k+1} - t_k)$ indicates the interval duration and $\mathcal{F}(q)\Omega$ is the first-order kinematics function from Eq. (3.3). Equation (5.48) is unique to the choice of trapezoidal collocation, and other integration methods require a different constraint [**32**].

The optimal control problem can be represented as the following nonlinear programming problem:

$$\begin{aligned} \underset{q(t_k),\ \Omega(t_k)}{\arg\min} \quad & m(q(T)) + \sum_{i=0}^{N} l(q(t_k), \Omega(t_k))\Delta t_k \\ \text{such that} \quad & h(q(t_0) : q(t_N); \Omega(t_0) : \Omega(t_{N-1})) = 0, \\ & g(q(t_0) : q(t_N); \Omega(t_0) : \Omega(t_{N-1})) \leq 0, \end{aligned} \tag{3.7}$$

where $h(\cdot)$ gives the equality constraints $q(0) = q_{\text{start}}$ and $q(T) = q_{\text{goal}}$ and enforces the first-order kinematics in Eq. (5.48). The expression $g(\cdot)$ gives the inequality constraints which constrain the controls ($\Omega_{\min} \leq \Omega \leq \Omega_{\max}$) and enforces any constraints on the configurations (e.g., due to singularities in the coordinate chart). Equation (5.49) is a finite-dimensional nonlinear optimization problem, and a solution $\xi_{\text{iDC}}(t)$ can be found using nonlinear optimizers such as SNOPT, IPOPT, or MATLAB's `fmincon`.

The integration error can be determined by comparing the trajectory $q_{\text{iDC}}(t)$ from the direct collocation method with the trajectory $q_{\text{fine}}(t)$, where $q_{\text{fine}}(t)$ is obtained by integrating the initial state over the interval $t = [0, T]$ using Eq. (3.3), the piecewise-linear

output controls $\Omega_{\text{iDC}}(t)$, and a higher-order integrator with small time steps ($dt \leq 0.001$). With fewer segments $N$, the integration error is larger, but there are fewer constraints for the nonlinear solver. This means that the optimizer is more likely to find a solution, and with less computational cost. The choice of $N$ is therefore a trade-off between computational cost/optimizer convergence and integration error. We implemented the iterative direct collocation (iDC) method to address this.

We first run the nonlinear optimization method using MATLAB's `fmincon` for a small number of segments ($N = 25$) to find a trajectory $\xi_{\text{iDC}}(t)$. The recalculated path $q_{\text{fine}}(t)$ is found using smaller integration timesteps and a higher-order integrator (`ode45`), and the planner is terminated if the goal-state tolerance of the fine trajectory is satisfied ($q_{\text{error}}(T) < \eta$). If the goal-state error is too large, the previous output trajectory serves as the initial trajectory guess for the next iteration with twice as many segments ($N \to 2N$). This is repeated until a valid trajectory $\xi_{\text{sol}}(t)$ is found, the maximum number of iDC iterations is reached, or the optimization converges to an invalid point. Small values of $N$ result in good solutions but may require many iDC iterations to converge to a solution, and large values will increase $N$ too quickly resulting in slower computation time and convergence to invalid points. We chose to double the segments between each iteration so the exact solution from the previous iteration could be used, but $N$ could be increased by a fixed value $\Delta N$ between each iteration to add an additional tuning parameter for the planning method.

## 3.7. Feedback Control of Rolling Surfaces

To stabilize the rolling trajectory to state disturbances, we construct a linear time-varying feedback controller based on a linearization of the rolling kinematics about the planned trajectory $q_{\text{sol}}(t)$. For this approach to be appropriate, the system should be linearly controllable about the planned trajectory, which may not always be the case, even if the system is nonlinearly controllable.

### 3.7.1. Linearization About a Trajectory

Given a nominal trajectory $\xi_{\text{nom}}(t) = (q_{\text{nom}}(t), \Omega_{\text{nom}}(t))$, we define perturbations about the trajectory as:

$$\widetilde{q}(t) = q(t) - q_{\text{nom}}(t), \tag{3.8}$$

$$\widetilde{\Omega}(t) = \Omega(t) - \Omega_{\text{nom}}(t). \tag{3.9}$$

The perturbed version of the dynamics in Eq. (3.3) can be written using a first-order Taylor expansion (and omitting the dependence on $t$) as:

$$\begin{aligned} \dot{q}_{\text{nom}} + \dot{\widetilde{q}} = \mathcal{F}(q_{\text{nom}})\Omega_{\text{nom}} + \\ \left[\frac{\partial(\mathcal{F}(q)\Omega)}{\partial q}\right]_{\text{nom}} \widetilde{q} + \left[\frac{\partial(\mathcal{F}(q)\Omega)}{\partial \Omega}\right]_{\text{nom}} \widetilde{\Omega} + \text{h.o.t.}, \end{aligned} \tag{3.10}$$

where $[\cdot]_{\text{nom}}$ means the enclosed expressions are evaluated along the nominal trajectory, and h.o.t. represents higher-order terms. Because $\dot{q}_{\text{nom}} = \mathcal{F}(q_{\text{nom}})\Omega_{\text{nom}}$ and h.o.t. are

negligible for nearby trajectories, Eq. (3.10) simplifies to:

$$\dot{\widetilde{q}} = \underbrace{\left[\frac{\partial(\mathcal{F}(q)\Omega)}{\partial q}\right]_{\text{nom}}}_{\widetilde{\mathbf{A}}(t)_{m\times m}} \widetilde{q} + \underbrace{[\mathcal{F}(q)]_{\text{nom}}}_{\widetilde{\mathbf{B}}(t)_{m\times n}} \widetilde{\Omega}, \tag{3.11}$$

where $m$ is the number of state variables and $n$ is the number of controls. We analyze the controllability properties of the linear time-varying (LTV) system $(\widetilde{\mathbf{A}}(t), \widetilde{\mathbf{B}}(t))$ to determine whether the nominal trajectory error ($\widetilde{q}(t)$, $\widetilde{\Omega}(t)$) can be stabilized to zero by a simple linear controller.

### 3.7.2. Controllability of Linear Time-Varying (LTV) Systems

The controllability of an LTV system along a nominal trajectory can be checked using the controllability gramian (e.g., Ch. 11.6 of [**12**]),

$$\mathbf{W_c}(t_1, t_0) = \int_{t_0}^{t_1} \mathbf{\Phi}(t_1, \tau)\widetilde{\mathbf{B}}(\tau)\widetilde{\mathbf{B}}(\tau)^{\mathsf{T}}\mathbf{\Phi}(t_1, \tau)^{\mathsf{T}} d\tau, \tag{3.12}$$

where the state-transition matrix[1] $\mathbf{\Phi}(t_1, \tau)$ and $\widetilde{\mathbf{B}}$ come from Eq. (3.11) and correspond to the linearization about the nominal trajectory. If the square matrix $\mathbf{W_c}(t_1, t_0)$ is non-singular on the interval $t \in [t_0, t_1]$, then the linearized trajectory can be stabilized by an appropriate controller. For a system with $m$ state variables, any system with $\text{rank}(\mathbf{W_c}(t_1, t_0)) < m$ is rank deficient, and therefore uncontrollable.

[1]The state-transition matrix $\mathbf{\Phi}(t_1, \tau)$ can be calculated for the LTV system using methods in Section 9.5-9.6 of [**12**] such as the the Peano-Baker Series or the fundamental solution matrix.

### 3.7.3. Controllability of Rolling Trajectories

The state $q$ and controls $\Omega$ for the rolling system are five-dimensional and two-dimensional, respectively, so $m = 5$ and $n = 2$. Therefore $\widetilde{\mathbf{A}}(t)$ is a $5 \times 5$ matrix, $\widetilde{\mathbf{B}}(t)$ is a $5 \times 2$ matrix, and full rank of the controllability gramian is 5.

While the linearized rolling dynamics of "generic" objects are controllable about "generic" rolling trajectories, there are at least three degenerate situations where the controllability gramian fails to achieve full rank, as outlined below.

**3.7.3.1. Degenerate Geometry.** An example of this case is two spheres of equal radius. As shown by Li and Canny [**36**], the relative configuration of two spheres of equal radius is uncontrollable by pure rolling, and therefore the linearization of the kinematics about any rolling trajectory is also uncontrollable. This conclusion is independent of the initial contact configuration of the bodies.

**3.7.3.2. Degenerate Initial Configuration.** In other cases, the relative configuration of two objects may be controllable by rolling from most configurations but not from others. An example is shown in Figure 3.5, where two identical ellipsoids, each with two equal principal semi-axes and one longer principal semi-axis, make initial contact such that the body geometries are symmetric about the contact tangent plane. Regardless of the rolling motion chosen from this initial configuration, the system will be confined to a lower-dimensional subset of its five-dimensional configuration space. More details can be found in [**45**] and Section 24.4 of [**1**].

**3.7.3.3. Degenerate Trajectory.** Finally, even if the two bodies are not geometrically degenerate and their initial configuration is not degenerate, a rolling trajectory may be chosen such that the rank of the controllability gramian of the linearized dynamics about

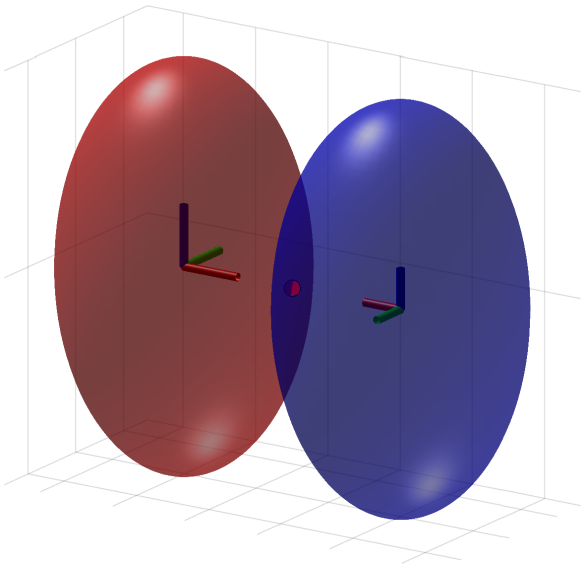


Figure 3.5. Example of uncontrollable initial condition for two identical ellipsoids.

the rolling trajectory never exceeds four. A trivial example is a stationary trajectory. For stationary trajectories ($\dot{q}(t) = 0 \ \forall \ t \in [0, T]$), the matrix $\widetilde{\mathbf{A}}(t)$ is zero and the matrix $\widetilde{\mathbf{B}}(t) = \widetilde{\mathbf{B}}(t_0)$ is constant. $\widetilde{\mathbf{B}}(t_0)$ is always full rank because rolling is allowed in two directions, so the controllability gramian is rank two for all stationary trajectories. The linearized rolling system is not controllable about stationary trajectories.

Another example is shown in the sphere-on-sphere trajectory of Figure 3.3. This trajectory illustrates constant $\Omega$, $u_1$, $u_2$, and $\psi$. The linearized dynamics are governed by

the linear time invariant (LTI) matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$:

$$\widetilde{\mathbf{A}} = \begin{bmatrix} 0 & 0 & 0 & 0 & \pi \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\pi & 0 & \frac{\pi}{3} & 0 & 0 \end{bmatrix}, \widetilde{\mathbf{B}} = \begin{bmatrix} 0 & \frac{3}{4} \\ \frac{3}{4} & 0 \\ 0 & \frac{1}{4} \\ -\frac{1}{4} & 0 \\ 0 & 0 \end{bmatrix}. \tag{3.13}$$

The Kalman controllability matrix $[\widetilde{\mathbf{B}}\ \widetilde{\mathbf{A}}\widetilde{\mathbf{B}}\ \widetilde{\mathbf{A}}^2\widetilde{\mathbf{B}}\ \widetilde{\mathbf{A}}^3\widetilde{\mathbf{B}}\ \widetilde{\mathbf{A}}^4\widetilde{\mathbf{B}}]$ is only rank four.

Even a trajectory about which the linearized system is controllable can be problematic to stabilize if the controllability gramian is ill-conditioned; large controls may be required to recover from small errors in certain state directions. Various metrics on the controllability gramian can be used to quantify controllability, such as the minimum eigenvalue ($\lambda_{\min}(\mathbf{W_c}(t_1, t_0))$), the trace of the inverse, ($\mathrm{tr}(\mathbf{W_c}^{-1}(t_1, t_0))$), and the determinant ($\det(\mathbf{W_c}(t_1, t_0))$) [**50**, **54**]. One of these measures could be included in the objective function in Eq. (5.47) to bias the trajectory optimization away from nearly-degenerate trajectories.

### 3.7.4. Stabilization of Rolling Trajectories

We use the linear quadratic regulator (LQR) to stabilize the linearized dynamics in Eq. (3.11). LQR computes a time-varying gain matrix $\mathbf{K}(t)$ that optimally reduces the total cost for small perturbations about the nominal trajectory. LQR requires a cost function, and we use the one given in Eq. (5.47). We solve the matrix Riccati equation

to find the time varying feedback control matrix $\mathbf{K}(t)$ (see Section 2.3 of [**4**]).

$$\begin{aligned} -\dot{\mathbf{P}}(t) &= \mathbf{P}(t)\widetilde{\mathbf{A}}(t) + \widetilde{\mathbf{A}}(t)^{\mathsf{T}}\mathbf{P}(t) - \\ &\qquad \mathbf{P}(t)\widetilde{\mathbf{B}}(t)\mathbf{R}_{\mathrm{LQR}}^{-1}\widetilde{\mathbf{B}}(t)^{\mathsf{T}}\mathbf{P}(t) + \mathbf{Q}_{\mathrm{LQR}}, \\ \mathbf{P}(T) &= \mathbf{P}_{1,\mathrm{LQR}} \\ \mathbf{K}(t) &= \mathbf{R}_{\mathrm{LQR}}^{-1}\widetilde{\mathbf{B}}(t)^{\mathsf{T}}\mathbf{P}(t). \end{aligned} \tag{3.14}$$

The matrix $\mathbf{K}(t)$ is then used in the feedback control law

$$\Omega_{\mathrm{fbk}}(q,t) = \Omega_{\mathrm{nom}}(t) - \mathbf{K}(t)(q(t) - q_{\mathrm{nom}}(t)) \tag{3.15}$$

to stabilize the nominal trajectory. The performance of this feedback controller depends on the controllability properties of the linearized dynamics.

## 3.8. Simulation Examples

We now demonstrate the motion planning and feedback control method for a sphere rolling on a sphere and an ellipsoid rolling on an ellipsoid. The sphere example demonstrates the ability to generate shorter paths for nontrivial trajectories when compared to the recent geometric planner for a sphere rolling on a sphere in [**58**]. The ellipsoid example demonstrates motion planning and feedback control for shapes with spatially-varying curvature.

We use the SQP algorithm of MATLAB's `fmincon` as our nonlinear optimization solver, and a list of the parameters used is included in Table 3.1. The code was run on an i7-4700MQ CPU @ 2.40 GHz with 16 GB of RAM. For each example we present the

Table 3.1. Parameters used in Section 3.8 for the iterative direct collocation algorithm.

| Description | Value |
|---|---|
| Trajectory execution time $T$ | 1 s |
| Initial # of segments $N$ | 25 ($\Delta t = 0.04$ s) |
| Goal error tolerance $\eta$ | 0.01 |
| Max iDC iterations | 4 |
| Max `fmincon` func evals/iteration | 25,000 |
| Control limits | $\|\|\omega_x\|\| \leq 30, \|\|\omega_y\|\| \leq 30$ |
| Constraint integration method | trapezoidal |
| $\mathbf{P}_1$ (terminal state weight) | $\text{diag}(100, 100, 100, 100, 100)$ |
| $\mathbf{Q}$ (tracking weight) | $\text{diag}(1, 1, 1, 1, 1)$ |
| $\mathbf{R}$ (control weight) | $\text{diag}(0.1, 0.1)$ |

number of iDC iterations, computation time, final state error ($q_{\text{error}}(T) = ||q(T) - q_{\text{goal}}||$), and trajectory cost from the cost function in Eq. (5.47).

### 3.8.1. Sphere on Sphere

We compare the results of our planner to a geometric trajectory planner for a sphere rolling on a sphere presented in Section 4.2 of [**58**]. Geometric trajectory planners use properties of the surface geometries to derive analytic expressions for the motion plans between start and goal states. While they are guaranteed to find exact solutions, such approaches are limited to specific geometries and solution paths are often unnecessarily long. Convergence is not guaranteed for our method, but in practice it works well and finds shorter paths.

Eq. (3.1) gives the parametric model of the spheres, and their radii are $\rho_1 = 2$ and $\rho_2 = 10$. The start and goal states are chosen as $q_{\text{start}} = (\frac{\pi}{2}, \frac{\pi}{4}, \frac{\pi}{2}, 0, 0)$ and $q_{\text{goal}} = (2.19, \frac{-3\pi}{4}, 0.96, \frac{\pi}{4}, 0)$ to match the sphere-on-sphere example in Section 4.2 of [**58**].

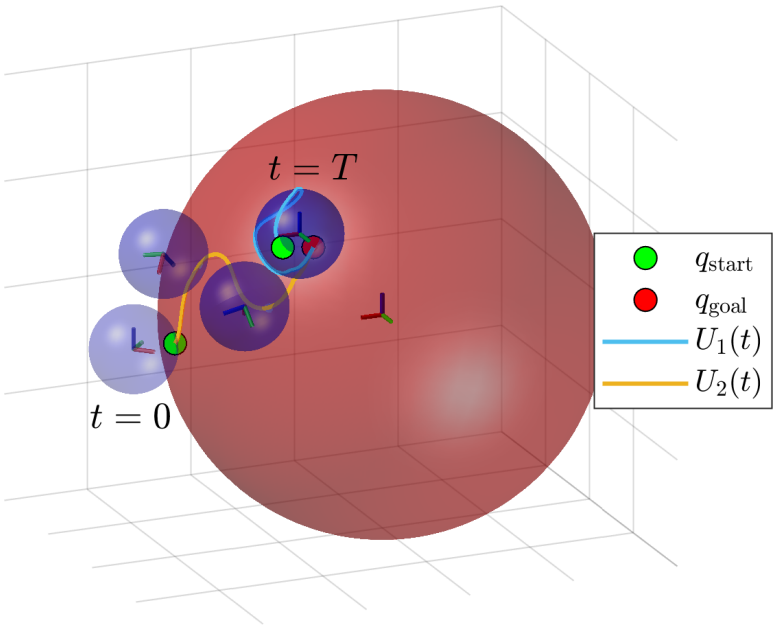



Figure 3.6. The sphere-on-sphere solution trajectory from Section 3.8.1. The smaller blue object 1 is rolling from bottom left to top right on the larger red object 2, and is shown at times $t = (0, \frac{T}{3}, \frac{2T}{3}, T)$. The contact path $U_1(t)$ is shown on the object at $t = T$, and the contact path $U_2(t)$ is shown on object 2. The initial and goal states were chosen to compare to the results from Rehan et al. [**58**], and our planned path $U_2(t)$ on object 2 is approximately three times shorter than the solution from the geometric planning method in that paper.

A solution was found after four iterations of the iDC method, with a total computation time of 47 seconds, a final state error of $q_{\text{error}}(T) = 0.002$, and a trajectory cost of 5.3. A visualization of the resulting motion plan is shown in Figure 3.6. The length of the path $U_2(t)$ (the path on the larger sphere) is approximately three times shorter than the solution presented in [**58**]. An animation of the initial guess and final trajectory is in the attached supplemental media.

### 3.8.2. Ellipsoid on Ellipsoid

We now demonstrate the planner on the more complex example of an ellipsoid rolling on an ellipsoid. Eq. (3.16) gives the parametric model of the ellipsoids

$$f_i : U_i \to \mathbb{R}^3 : (u_i, v_i) \mapsto (\rho_{ia} \sin(u_i) \cos(v_i), \rho_{ib} \sin(u_i) \sin(v_i), \rho_{ic} \cos(u_i)), \tag{3.16}$$

where $u_i$ satisfies $0 < u_i < \pi$, $v_i$ satisfies $-\pi < v_i < \pi$, and the principal semi-axes are chosen as $(\rho_{1a}, \rho_{1b}, \rho_{1c}) = (1, 1, 1.5)$ and $(\rho_{2a}, \rho_{2b}, \rho_{2c}) = (3, 3, 5)$. The coordinate systems are orthogonal because $\rho_{1a} = \rho_{1b}$ and $\rho_{2a} = \rho_{2b}$. The start and goal states are chosen as $q_{\text{start}} = (\frac{\pi}{2}, 0, \frac{\pi}{2}, 0, 0)$ and $q_{\text{goal}} = (\frac{\pi}{2}, 0, \frac{\pi}{4}, -\frac{\pi}{2}, -\frac{\pi}{4})$. A solution was found after four iterations of the iDC method, with a total computation time of 61 seconds, a final state error of $q_{\text{error}}(T) = 0.003$, and a trajectory cost of 12.8. The resulting path is shown in Figure 3.7. The contact coordinates and controls for the initial trajectory guesses, the results from the first iDC iteration, and the final trajectory are shown in Figure 3.8. An animation of the initial guess and final trajectory is in the attached supplemental media.

### 3.8.3. Feedback Control

This section demonstrates the performance of the LQR controller in stabilizing trajectories with initial state perturbations. The weighting parameters for the LQR feedback controller are given in Table 3.2. The weights $\mathbf{Q}_{\text{LQR}}$ and $\mathbf{P}_{1,\text{LQR}}$ were both increased from the direct collocation optimization weights to improve tracking performance and decrease the final state error.

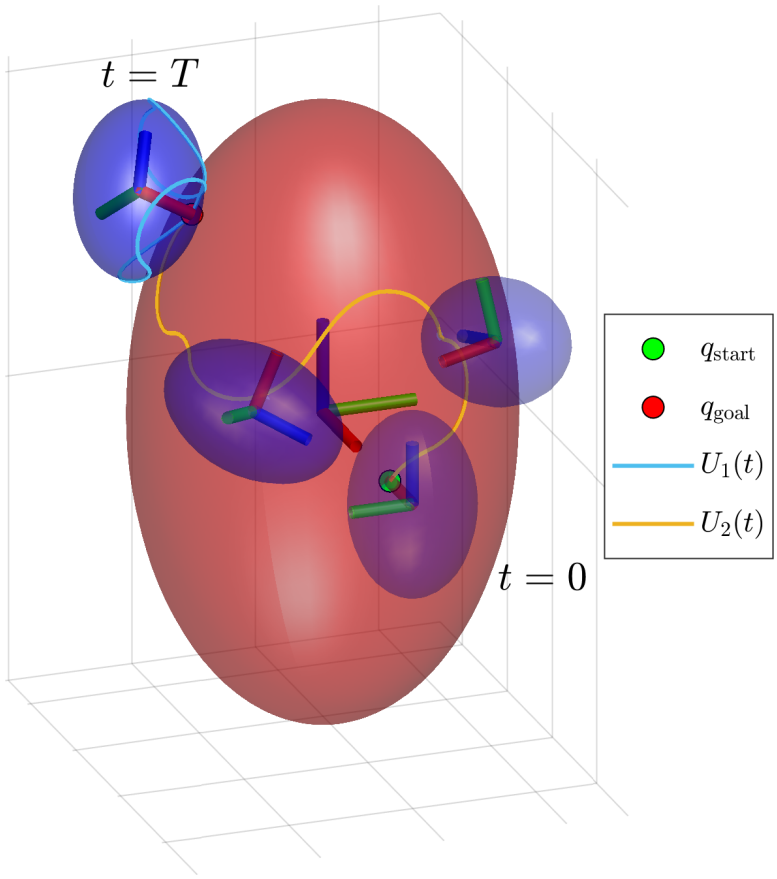



Figure 3.7. Ellipsoid-on-ellipsoid visualization for the motion plan in Section 3.8.2 and Figure 3.8. The smaller blue object 1 rolls from bottom right to top left on the larger red object 2, and is shown at times $t = (0, \frac{T}{3}, \frac{2T}{3}, T)$. The contact path $U_1(t)$ is shown on object 1 at $t = T$, and the contact path $U_2(t)$ is shown on object 2. The $U_1(t)$ and $U_2(t)$ trajectories are shown in Figure 3.8 (c).

Table 3.2. Parameters used in Section 3.8.3 for the LQR feedback control of rolling trajectories.

| Description | Value |
|---|---|
| $\mathbf{P}_{1,\text{LQR}}$ (terminal state weight) | $\text{diag}(1e5, 1e5, 1e5, 1e5, 1e5)$ |
| $\mathbf{Q}_{\text{LQR}}$ (tracking weight) | $\text{diag}(100, 100, 100, 100, 100)$ |
| $\mathbf{R}_{\text{LQR}}$ (control weight) | $\text{diag}(0.1, 0.1)$ |

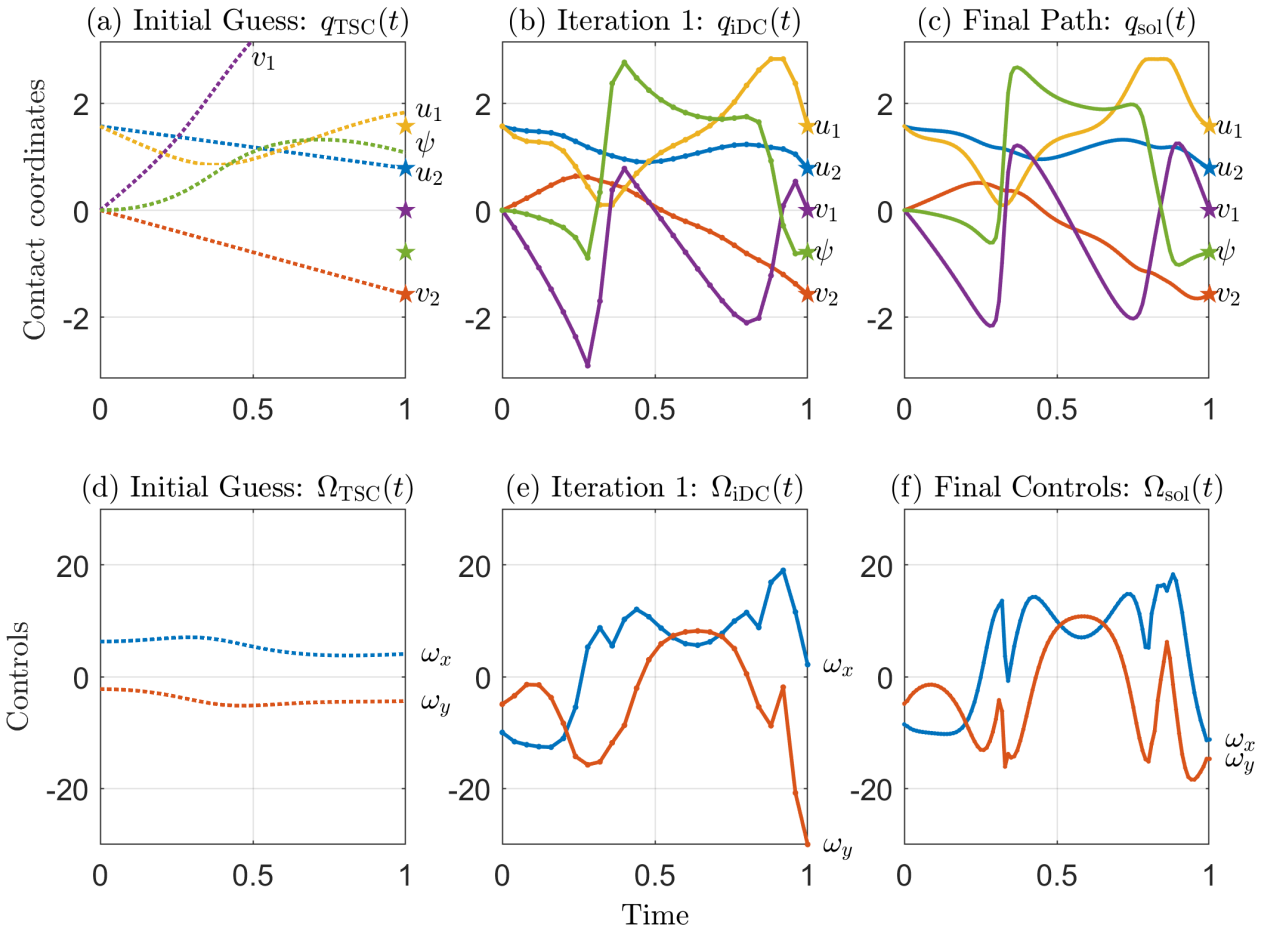



Figure 3.8. Contact coordinate plots and control plots for the ellipsoid-on-ellipsoid rolling plan in Figure 3.7. Column one shows the initial trajectory guess from the two-state control method, column two shows the output trajectory from the first iteration of the iterative direct-collocation method (which fails to satisfy the tolerance criterion after accurate simulation), and column three shows the solution trajectory. The stars in (a)-(c) show the desired goal states $q_{\text{goal}}$.

**3.8.3.1. Sphere-on-Sphere.** We first demonstrate feedback control on the simple sphere-on-sphere equator trajectory in Figure 3.3 with an initial state perturbation of $\epsilon(q(0)) = (0.1, 0.05, -0.05, -0.1, 0)$, where $\epsilon(q(t))$ is the difference between the current and nominal

reference trajectory and is defined as $\epsilon(q(t)) = q(t) - q_{\text{nom}}(t)$. We set the nominal trajectory to $\xi_{\text{nom}}(t) = \xi_{\text{sol}}(t)$, and as mentioned in Section 3.7.3, the linearized dynamics are not controllable about this degenerate trajectory.

Figure 3.9(a) shows the individual and total coordinate error over time. The norm of the initial state error is $||\epsilon(q(0))|| = 0.16$ and the norm of the final state error is $||\epsilon(q(T))|| = 0.08$. LQR is ineffective at eliminating the error because the linearized dynamics are uncontrollable about the nominal trajectory. Animations of the open- and closed-loop performance for the perturbed sphere-on-sphere trajectories in Figures 3.3 and 3.6 can be seen in the supplemental media. The controllability gramian of the linearized dynamics about the sphere-on-sphere trajectory in Figure 3.6 is full rank, and therefore LQR eliminates the state error.

**3.8.3.2. Ellipsoid-on-Ellipsoid.** Figure 3.9(b) shows the individual and total coordinate error over time for the nominal trajectory of Figure 3.7 and an initial perturbation $\epsilon(q(0)) = (0.1, 0.05, -0.05, -0.1, 0)$. The norm of the initial state error is $||\epsilon(q(0))|| = 0.16$ and the norm of the final state error is $||\epsilon(q(T))|| = 0.0004$. We see that the feedback controller effectively recovers from the initial error. An animation of the open and closed-loop performance for the perturbed ellipsoid-on-ellipsoid trajectory can be seen in the supplemental media.

## 3.9. Discussion

### 3.9.1. Robustness

To test the robustness of the proposed planner, we generated 100 random trajectory planning tasks for the sphere-on-sphere and ellipsoid-on-ellipsoid. The initial state was

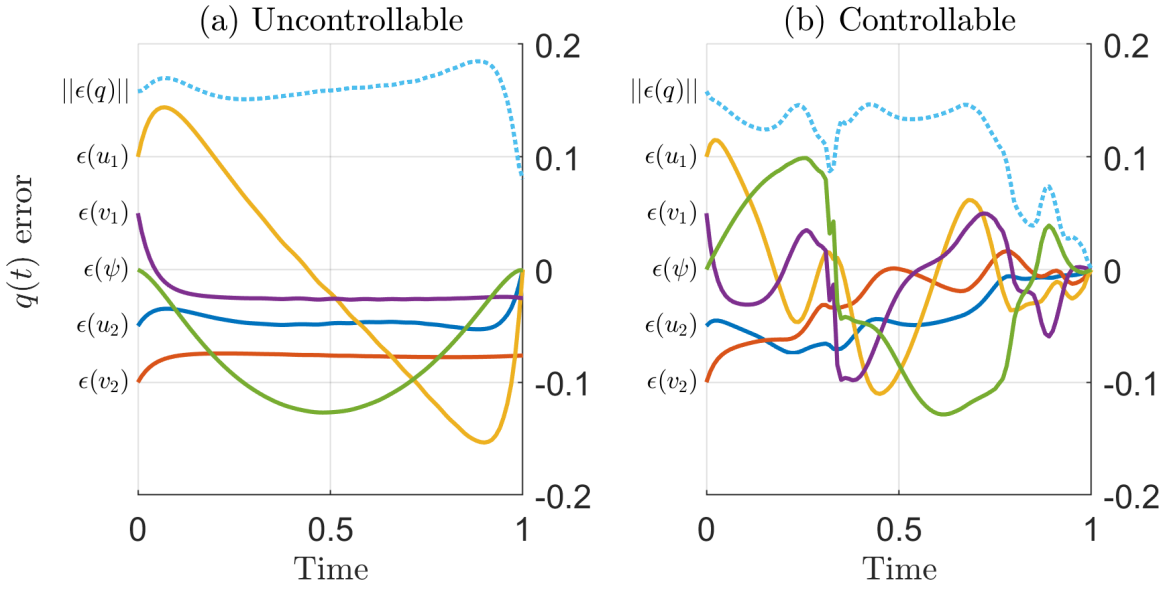



Figure 3.9. Error recovery of sphere-on-sphere (a) and ellipsoid-on-ellipsoid (b) under feedback control with an initial state perturbation of $\epsilon(q(0)) = (0.1, 0.05, -0.05, -0.1, 0)$. The function $\epsilon(\cdot)$ calculates the difference between input coordinate(s) and the reference coordinate(s) in $q_{\text{nom}}(t)$, and $||\epsilon(q)||$ is the norm of the total coordinate error. The sphere trajectory in (a) is the equator example given in Figure 3.3, where the controllability gramian is not full rank, and therefore LQR cannot eliminate the state error. The ellipsoid trajectory in (b) is for the ellipsoid example in Figure 3.7. The controllability gramian for this trajectory is full rank and therefore the controller is able to reduce the error to zero.

fixed at $q_{\text{start}} = (\frac{\pi}{2}, 0, \frac{\pi}{2}, 0, 0)$ and goal states were chosen in the range $(0, -\pi, 0, -\pi, -\pi) < q_{\text{goal}} < \pi$. The results are shown in Table 3.3. The planner used the parameters in Table 3.1, other than maximum `fmincon` function evaluations set to 15,000 and $\eta$ set to 0.1 to decrease computation time.

These results demonstrate the ability of the planner to reliably find solutions for random goal states. While the method is not guaranteed to find a solution, it works well in practice.

Table 3.3. Testing the planning method on 100 random goal states for sphere-on-sphere and ellipsoid-on-ellipsoid rolling. Results are given as mean (standard deviation).

| Geometry | Planning time (s) | $q_{\text{error}}(T)$ | Cost | % Success |
|---|---|---|---|---|
| Spheres | 16 (9) | 0.045 (0.027) | 13 (6) | 99 |
| Ellipsoids | 17 (8) | 0.04 (0.028) | 12 (5.7) | 99 |

### 3.9.2. Effect of the Initial Trajectory Guess

The initial trajectory guess to the iDC optimization is "two-state control," which drives the contact coordinates of object 2 directly to their desired final value. This trajectory is admissible, i.e., it satisfies the rolling conditions. Other initial trajectory guesses could be used, like two-state control for the contact coordinates of object 1, which is admissible; linear interpolation ($q(t) = q_{\text{des}}(t)$, $\Omega(t) = 0$), which is inadmissible; and the stationary trajectory ($\dot{q}(t) = 0$, $\Omega(t) = 0$), which is admissible. While all types of initial trajectory guesses led to solutions in most cases, the two-state control on object 2 performed the best with the shortest planning times, lowest costs, smallest errors, and highest success rate for random planning problems for the ellipsoid-on-ellipsoid (Table 3.4). The size difference between the objects could explain the difference between the performance of TSC on objects 1 and 2. Because object 1 is smaller than object 2, TSC1 generates short contact paths on object 2, so the initial guess for TSC1 more closely resembles the stationary initial guess.

### 3.9.3. Computation Time

The direct-collocation method in `fmincon` has stop conditions based on different tolerances (optimality, function, step, constraint), and the maximum number of times the objective

Table 3.4. Comparing the effect of the initial trajectory guess method for 100 random goal states for ellipsoid-on-ellipsoid rolling. The different initial guess methods were two-state control on object 1, two-state control on object 2, linear interpolation ($q(t) = q_{\text{des}}$, $\Omega(t) = 0$), and stationary ($\dot{q}(t) = 0$, $\Omega(t) = 0$). Results are given as mean (standard deviation).

| Geometry | Planning time (s) | $q_{\text{error}}(T)$ | Cost | % Success |
|---|---|---|---|---|
| TSC Object 1 | 17 (13) | 0.059 (0.18) | 13.6 (17) | 99 |
| TSC Object 2 | 17 (8) | 0.04 (0.028) | 12 (5.7) | 99 |
| Linear Interp. | 31 (54) | 0.24 (0.65) | 33 (96) | 88 |
| Stationary | 26 (29) | 0.21 (0.59) | 30 (85) | 91 |

function is called. We analyzed the effect that the max objective function evaluations has on the quality of the output trajectories and the computation time. For each step of `fmincon`, the cost function is called $(N + 1)(m + n)$ times, where $N$ is the number of collocation segments, $m$ is the number of states, and $n$ is the number of controls. The solver can take a long time to converge to a solution, and the marginal improvement of each step decreases over time. These small improvements have a negligible effect when the solution is used as the initial guess for another optimization problem. We therefore tested how limiting the number of function evaluations changes the computation time and effects the results of the planner. This allows us to terminate the optimizations sooner to save unnecessary computation time.

Table 3.5 demonstrates the results of planning for ten random goal states, for $N = 20$, and with six different limits on the number of function evaluations. Our system has $m = 5$ states and $n = 2$ controls, so the number of function evaluations per `fmincon` step is $(N + 1)(m + n) = (20 + 1)(5 + 2) = 147$.

Table 3.5. Testing the equation function count limit for ten random goal states each for an ellipsoid on an ellipsoid. results are given as mean(standard deviation)

| Max Evaluations | t (s) | Cost | $q_{\text{error}}(T)$ |
|---|---|---|---|
| 5,000 | 10(4) | 13.4(7.5) | 0.032(0.024) |
| 15,000 | 22(10) | 11.3(5.8) | 0.028(0.013) |
| 25,000 | 36(12) | 10.9(5.3) | 0.036(0.026) |
| 50,000 | 60(25) | 10.2(5.1) | 0.030(0.020) |
| 100,000 | 95(50) | 10.1(5.0) | 0.029(0.019) |
| 200,000 | 114(81) | 10.1(5.0) | 0.026(0.022) |

As expected, the results show a trend of increasing computation time and lower cost for more function evaluations. For our simulations we chose to set the limit at 15,000 evaluations because the speed increase justified the loss of accuracy.

## 3.10. Conclusions and Future Work

This paper presents a motion planner and feedback controller for pure-rolling motions between general smooth rigid bodies. The methods were demonstrated for a sphere rolling on a sphere and an ellipsoid rolling on an ellipsoid. Future work includes extending motion planning and control to second-order kinematic and dynamic rolling. We are also interested in methods that allow planning for more general object parameterizations (non-orthogonal) or smooth approximations of general surfaces represented by meshes.

## 3.11. Appendix: Local Geometry of Smooth Bodies

Below are some expressions for the geometry of a surface that are used to define the first-order kinematics in Section 3.4. References and derivations of these expressions can be found in [**61**].

We represent the surface of each body in contact as a mapping $f_i : (u_i, v_i) \mapsto (x_i, y_i, z_i)$ for objects $i \in [1, 2]$ (see Section 3.4). It is assumed that $f_i$ is continuous up to the second derivative (class $C^2$) so that the local contact geometries (contact frames and curvature associated with the first and second derivatives of $f_i$, respectively) are uniquely defined. The natural bases at a point on a body are given as $x_i = \partial f_i / \partial u_i$ and $y_i = \partial f_i / \partial v_i$. We also assume that coordinate charts are orthogonal ($x_i \cdot y_i = 0$), and note that $x_i$ and $y_i$ are not necessarily unit vectors. The normal is given as $n_i = (x_i \times y_i)/||x_i \times y_i||$.

The normalized Gauss frame at a point $U_i$ on object $i$ is defined as the coordinate frame $\{c_i\}$ with origin at $f_i(U_i)$ and coordinate axes given by

$$\mathbf{R}_{o_i c_i} = \left[ \frac{x_i}{||x_i||}, \frac{y_i}{||y_i||}, n_i \right], \tag{3.17}$$

where $\mathbf{R}_{o_i c_i}$ expresses the Gauss frame in the object $i$ frame $\{o_i\}$. The metric tensor $\mathbf{G}_i$ is a $2 \times 2$ positive-definite matrix defined as

$$\mathbf{G}_i = \begin{bmatrix} x_i \cdot x_i & x_i \cdot y_i \\ y_i \cdot x_i & y_i \cdot y_i \end{bmatrix}. \tag{3.18}$$

The coefficients $g_{jk,i}$ reference the indices of matrix $\mathbf{G}_i$, and $\mathbf{G}_i$ is diagonal ($g_{12,i} = g_{21,i} = 0$) when the coordinate chart $f_i$ is orthogonal. The $2 \times 2$ matrix $L_i$ is the second fundamental form given by the expression

$$L_i = \begin{bmatrix} \frac{\partial^2 f_i}{\partial u_i^2} \cdot n_i & \frac{\partial^2 f_i}{\partial u_i \partial v_i} \cdot n_i \\ \frac{\partial^2 f_i}{\partial v_i \partial u_i} \cdot n_i & \frac{\partial^2 f_i}{\partial v_i^2} \cdot n_i \end{bmatrix}. \tag{3.19}$$

$\mathbf{H}_i$ combines the metric tensor $\mathbf{G}_i$ with the second fundamental form $\mathbf{L}_i$ and is given by,

$$\mathbf{H}_i = (\sqrt{\mathbf{G}_i})^{-1}\mathbf{L}_i(\sqrt{\mathbf{G}_i})^{-1}. \tag{3.20}$$

The $1 \times 2$ array $\Gamma_i$ is given by the expression

$$\Gamma_i = \left[\Gamma^2_{11,i}\ \Gamma^2_{12,i}\right], \tag{3.21}$$

where $\Gamma^2_{11,i}$ and $\Gamma^2_{12,i}$ are christoffel symbols of the second kind given by

$$\begin{aligned} \Gamma^2_{11,i} &= \left(\frac{\partial x_i}{\partial u_i} \cdot x_i\right) g_i^{12} + \left(\frac{\partial x_i}{\partial u_i} \cdot y_i\right) g_i^{22}, \\ \Gamma^2_{12,i} &= \left(\frac{\partial x_i}{\partial v_i} \cdot x_i\right) g_i^{12} + \left(\frac{\partial x_i}{\partial v_i} \cdot y_i\right) g_i^{22}, \end{aligned} \tag{3.22}$$

where $g_i^{jk}$ are entries $(j, k)$ of the metric tensor inverse $(\mathbf{G}_i)^{-1}$.

CHAPTER 4

# Second-Order Contact Kinematics Between Three-Dimensional Rigid Bodies

## 4.1. Abstract

Chapter 3 focused on first-order kinematics, and we now focus on the more general second-order kinematics where the relative accelerations at the contact are given. In this chapter, we provide corrections to the second-order kinematic equations describing contact between three-dimensional rigid bodies, originally published in Sarkar et al. (1996) ["Velocity and Acceleration Analysis of Contact Between Three-Dimensional Rigid Bodies," ASME J. Appl. Mech., 63(4), pp. 974-984]. [DOI: 10.1115/1.4043547]

The main contributions of this chapter are the corrections to the second-order kinematics equations that are used to derive the rolling dynamics in Chapter 5. This chapter was published in the Journal of Applied Mechanics [**75**]

## 4.2. Introduction

When a three-dimensional rigid body (object 1) is in single-point contact with another rigid body (object 2), the configuration of object 1 relative to object 2 is five dimensional: the six degrees of freedom of object 1 subject to the single constraint that the distance to object 2 is zero. This five-dimensional configuration space can be parametrized by the two coordinates $U_1 = (u_1, v_1)$ describing the contact location on the surface of object 1, the

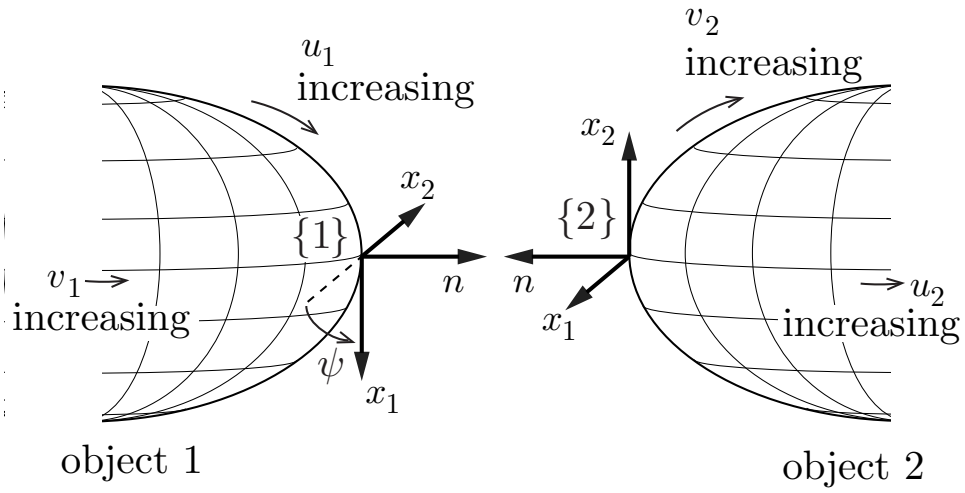



Figure 4.1. Objects 1 and 2 are in contact, but they are shown separated for clarity. The surfaces of objects 1 and 2 are parametrized by $(u_1, v_1)$ and $(u_2, v_2)$, respectively. At the point of contact, the unit $x_1$- and $x_2$-axes of the coordinate frame $\{i\}$ are in the direction of increasing $u_i$ (and constant $v_i$) and increasing $v_i$ (and constant $u_i$), respectively, and the contact normal $n$ is the cross product of $x_1$ and $x_2$. Rotating frame $\{2\}$ by $\psi$ about the $n$-axis of frame $\{1\}$ brings the $x_1$-axes of the frames $\{1\}$ and $\{2\}$ into alignment.

two coordinates $U_2 = (u_2, v_2)$ describing the contact location on the surface of object 2, and one coordinate $\psi$ describing the angle of "spin" between frames fixed to each body at the contact point. Collectively the contact configuration is written $q = (u_1, v_1, u_2, v_2, \psi)$. See Fig. 5.2.

"Contact kinematics" refers to equations relating the relative motion between the objects to the evolution of $q$. The velocity of one object relative to the other can be written in terms of the linear velocity $V = (V_x, V_y, V_z)$ and the angular velocity $\omega = (\omega_x, \omega_y, \omega_z)$ at a frame at the current contact, where $V_z = 0$ is required to maintain contact. The "first-order" contact equations relate $(V, \omega)$ (where $V_z = 0$) to $\dot{q}$. The "second-order" contact equations express $\ddot{q}$ in terms of the relative linear and angular accelerations $(a, \alpha)$, given

an initial state where the first-order contact condition $V_z = 0$ is satisfied and a choice of $a$ in the two-dimensional space of linear accelerations that maintain the contact.

These first- and second-order contact kinematics are fundamental to planning and control of robot motions in contact. Such motions are sometimes called "roll-slide" motions. The specialization of these equations to the case of no sliding is useful for manipulation tasks involving rolling.

Second-order contact equations were first published in [**61**], based on the work in Sarkar's PhD thesis [**60**]. These equations generalized Montana's first-order contact kinematics [**49**] and the partial results on second-order contact kinematics in [**13**]. Sarkar et al. then restated the second-order contact kinematics in [**62**, **63**], where they were used in the context of robotic manipulation.

Each of the statements of the second-order contact kinematics in [**60**–**63**] is slightly different, but each contains errors, including sign inversions. Other than a journal typesetting error, the most correct version of the equations is in [**61**], which contains only the sign inversions. Given the importance of these equations to robot motion planning and manipulation (the papers [**60**–**63**] have been cited hundreds of times according to Google Scholar) and our own work, we present the corrected equations.

## 4.3. Problem Statement

The contact coordinates for each object $i \in \{1, 2\}$ are parametrized by $f_i : U_i \to \mathbb{R}^3 : (u_i, v_i) \mapsto (x_i, y_i, z_i)$ expressed in a frame fixed to the body. It is assumed that $f_i$ is continuous up to the third derivative (class $C^3$), so that the local contact geometry (contact frames associated with the first derivative of $f_i$, curvature associated with the

second derivative, and derivative of the curvature associated with the third derivative) is uniquely defined. For details on other definitions, see [**61**].

With these definitions, the problem can be stated as the following: given the current state (the relative configurations of the objects and their relative velocity $(V, \omega)$ satisfying the first-order contact condition) and their relative acceleration $(a, \alpha)$ satisfying the second-order contact condition, find the contact accelerations $\ddot{q} = (\ddot{u}_1, \ddot{v}_1, \ddot{u}_2, \ddot{v}_2, \ddot{\psi})$.

### 4.4. Second-Order Contact Kinematics Derivation

Equations (4.1), (4.2), and (4.3) below correspond to Eqns. (39), (41), and (42), respectively, in [**61**]. We have rederived and verified these equations, which represent five equality constraints relating the evolution of $\ddot{q}$ and $(a, \alpha)$ when $a$ satisfies the second-order condition for rolling given by Eqn. (60) in [**61**].

$$\begin{aligned}\sqrt{G_2}(\ddot{U}_2 + \bar{\Gamma}_2 W_2) = R_\psi \sqrt{G_1}(\ddot{U}_1 + \bar{\Gamma}_1 W_1) \\ +2\omega_z E_1 R_\psi \sqrt{G_1}\dot{U}_1 + \begin{bmatrix} a_x \\ a_y \end{bmatrix},\end{aligned} \tag{4.1}$$

$$\begin{aligned}R_\psi E_1(\sqrt{G_1})^{-1}(\bar{\bar{L}}_1 W_1 - L_1\ddot{U}_1) - R_\psi(\sqrt{G_1})^{-1}L_1\dot{U}_1\omega_z \\ +\sigma_1\Gamma_1\dot{U}_1\begin{bmatrix} -\omega_y \\ \omega_x \end{bmatrix} + \begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix} = (\sqrt{G_2})^{-1}L_2\dot{U}_2\dot{\psi} \\ +E_1(\sqrt{G_2})^{-1}(\bar{\bar{L}}_2 W_2 - L_2\ddot{U}_2),\end{aligned} \tag{4.2}$$

$$\begin{aligned}-\sigma_1(\Gamma_1\ddot{U}_1 + \bar{\bar{\Gamma}}_1 W_1) - \begin{bmatrix} -\omega_y \\ \omega_x \end{bmatrix}^T R_\psi E_1(\sqrt{G_1})^{-1}L_1\dot{U}_1 + \alpha_z \\ = -\ddot{\psi} + \sigma_2(\Gamma_2\ddot{U}_2 + \bar{\bar{\Gamma}}_2 W_2).\end{aligned} \tag{4.3}$$

In these equations, $G_i$ is the metric tensor of object $i$, $\sigma_i = \sqrt{g_{22,i}/g_{11,i}}$ where $g_{11,i}$ and $g_{22,i}$ are the diagonal entries of $G_i$, and $W_i$ comprises the velocity product terms $[\dot{u}_i^2,\ \dot{u}_i\dot{v}_i,\ \dot{v}_i^2]^T$. The matrices $\Gamma_i$, $L_i$, $\bar{\Gamma}_i$, $\bar{L}_i$, $\bar{\bar{\Gamma}}_i$, and $\bar{\bar{L}}_i$ describe the local contact geometry as derived from $f_i$ in [**61**], and the matrices $E_1$ and $R_\psi$ are defined as

$$E_1 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad R_\psi = \begin{bmatrix} \cos\psi & -\sin\psi \\ -\sin\psi & -\cos\psi \end{bmatrix}.$$

The corrected derivation of the contact kinematics begins here. Rearranging Eqns. (4.1) and (4.2) yields

$$\begin{aligned} R_\psi\sqrt{G_1}\ddot{U}_1 - \sqrt{G_2}\ddot{U}_2 = \sqrt{G_2}\bar{\Gamma}_2 W_2 - R_\psi\sqrt{G_1}\bar{\Gamma}_1 W_1 \\ -2\omega_z E_1 R_\psi\sqrt{G_1}\dot{U}_1 - \begin{bmatrix} a_x \\ a_y \end{bmatrix}, \end{aligned} \tag{4.4}$$

$$\begin{aligned} R_\psi E_1(\sqrt{G_1})^{-1}L_1\ddot{U}_1 - E_1(\sqrt{G_2})^{-1}L_2\ddot{U}_2 = \\ R_\psi E_1(\sqrt{G_1})^{-1}\bar{\bar{L}}_1 W_1 - R_\psi(\sqrt{G_1})^{-1}L_1\dot{U}_1\omega_z \\ +\sigma_1\Gamma_1\dot{U}_1\begin{bmatrix} -\omega_y \\ \omega_x \end{bmatrix} + \begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix} - (\sqrt{G_2})^{-1}L_2\dot{U}_2\dot{\psi} \\ -E_1(\sqrt{G_2})^{-1}\bar{\bar{L}}_2 W_2. \end{aligned} \tag{4.5}$$

Combining Eqns. (4.4) and (4.5) into a single equation yields

$$
\begin{bmatrix} R_\psi\sqrt{G_1} & -\sqrt{G_2} \\ R_\psi E_1(\sqrt{G_1})^{-1}L_1 & -E_1(\sqrt{G_2})^{-1}L_2 \end{bmatrix}\begin{bmatrix} \ddot{U}_1 \\ \ddot{U}_2 \end{bmatrix} =
$$

$$
\begin{bmatrix} -R_\psi\sqrt{G_1}\bar{\Gamma}_1 \\ R_\psi E_1(\sqrt{G_1})^{-1}\bar{\bar{L}}_1 \end{bmatrix}W_1 + \begin{bmatrix} \sqrt{G_2}\bar{\Gamma}_2 \\ -E_1(\sqrt{G_2})^{-1}\bar{\bar{L}}_2 \end{bmatrix}W_2
$$

$$
+\begin{bmatrix} -2\omega_z E_1 R_\psi\sqrt{G_1} & 0 \\ -\omega_z R_\psi(\sqrt{G_1})^{-1}L_1 & -(\sqrt{G_2})^{-1}L_2\dot{\psi} \end{bmatrix}\begin{bmatrix} \dot{U}_1 \\ \dot{U}_2 \end{bmatrix} \tag{4.6}
$$

$$
-\begin{bmatrix} 0_{2\times1} \\ \sigma_1\Gamma_1\dot{U}_1\begin{pmatrix} \omega_y \\ -\omega_x \end{pmatrix} \end{bmatrix} + \begin{bmatrix} 0_{2\times1} \\ \begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix} \end{bmatrix} - \begin{bmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} \\ 0_{2\times1} \end{bmatrix}.
$$

Following [**61**], we define $H_i = (\sqrt{G_i})^{-1} L_i (\sqrt{G_i})^{-1}$, substitute into (4.6), and rearrange to get

$$
\begin{aligned}
\begin{bmatrix} \ddot{U}_1 \\ \ddot{U}_2 \end{bmatrix} &= \begin{bmatrix} R_\psi \sqrt{G_1} & -\sqrt{G_2} \\ R_\psi E_1 H_1 \sqrt{G_1} & -E_1 H_2 \sqrt{G_2} \end{bmatrix}^{-1} \\
&\left\{ \begin{bmatrix} \boxed{-R_\psi \sqrt{G_1} \bar{\Gamma}_1} \\ R_\psi E_1 (\sqrt{G_1})^{-1} \bar{\bar{L}}_1 \end{bmatrix} W_1 + \begin{bmatrix} \sqrt{G_2} \bar{\Gamma}_2 \\ \boxed{-E_1 (\sqrt{G_2})^{-1} \bar{\bar{L}}_2} \end{bmatrix} W_2 \right. \\
&\qquad + \begin{bmatrix} -2\omega_z E_1 R_\psi \sqrt{G_1} & 0 \\ -\omega_z R_\psi H_1 \sqrt{G_1} & -\dot{\psi} H_2 \sqrt{G_2} \end{bmatrix} \begin{bmatrix} \dot{U}_1 \\ \dot{U}_2 \end{bmatrix} \\
&\left. - \begin{bmatrix} 0_{2\times 1} \\ \sigma_1 \Gamma_1 \dot{U}_1 \begin{pmatrix} \omega_y \\ -\omega_x \end{pmatrix} \end{bmatrix} + \begin{bmatrix} 0_{2\times 1} \\ \begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix} \end{bmatrix} - \begin{bmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} \\ 0_{2\times 1} \end{bmatrix} \right\}.
\end{aligned}
\tag{4.7}
$$

This is equivalent to Eqn. (43) of [**61**] except the two boxed terms in red are preceded by a minus sign in the corrected equations. The fifth equation for $\ddot{\psi}$ is derived by rearranging Eqn. (4.3),

$$
\begin{aligned}
\ddot{\psi} = &\boxed{- \begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix}^T R_\psi E_1 (\sqrt{G_1})^{-1} L_1 \dot{U}_1} - \alpha_z \\
&+ \sigma_1 (\Gamma_1 \ddot{U}_1 + \bar{\bar{\Gamma}}_1 W_1) + \sigma_2 (\Gamma_2 \ddot{U}_2 + \bar{\bar{\Gamma}}_2 W_2),
\end{aligned}
\tag{4.8}
$$

where the minus sign on the left of the boxed red term did not appear in Eqn. (44) in [**61**].

## 4.5. Example: A Sphere Rolling on a Sphere

Consider the example of a small sphere (object 1) rolling without sliding on a larger sphere (object 2), as shown in Fig. 4.2. The spheres $i \in \{1, 2\}$ are parameterized by

$$\begin{aligned} &f_i : U_i \to \mathbb{R}^3 : (u_i, v_i) \mapsto \\ &(\rho_i \sin(u_i)\cos(v_i), \rho_i \sin(u_i)\sin(v_i), \rho_i \cos(u_i)), \end{aligned} \tag{4.9}$$

where the "latitude" $u_i$ satisfies $0 < u_i < \pi$ and the "longitude" $v_i$ satisfies $-\pi < v_i < \pi$.

Object 2 (the large red sphere) remains fixed in space while object 1 (the small blue sphere) rolls on it. To ensure rolling, the relative linear velocity at the contact satisfies $V = 0$ and the linear acceleration $a$ satisfies the three constraints given in Eqn. (60) of [**61**]: one constraint to maintain contact and two constraints that prevent slip.

Object 1 is made to roll at a constant speed along the "equator" of object 2 by choosing $(\omega_x, \omega_y, \omega_z) = (\omega, 0, 0)$: object 1 always rotates about the downward-pointing $x_1$-axis of frame $\{2\}$ in the tangent plane of the contact.

### 4.5.1. Instantaneous Solution

Consider the case where the initial configuration is $q_0 = (\pi/2, 0, \pi/2, 0, 0)$ as shown in Fig 4.2(a). From the first-order kinematics in [**61**] we can solve for the initial contact velocities

$$\dot{q}_0 = (0, \rho_2 \omega k_1, 0, -\rho_1 \omega k_1, 0), \tag{4.10}$$

where $k_1 = 1/(\rho_1 + \rho_2)$. Using the initial contact state $(q_0, \dot{q}_0)$, the controls $(\alpha_x, \alpha_y, \alpha_z) = (0, 0, 0)$, and the rolling assumptions $(v_x, v_y, v_z) = (0, 0, 0)$, we solve for $(a_x, a_y, a_z)$ to satisfy the second-order rolling conditions (Eqn. (60) of [**61**]) and the coordinate acceleration $\ddot{q}_0$. For the original equations (43) and (44) in [**61**] and the corrected equations (5.38) and (4.8), we get

$$\ddot{q}_0 = (\ddot{u}_1, \ddot{v}_1, \ddot{u}_2, \ddot{v}_2, \ddot{\psi}) = (0, 0, 0, 0, 0). \tag{4.11}$$

The $v_1$ and $v_2$ coordinates change linearly with time while all other contact coordinates remain constant, as would be expected for rolling along the equators. The original contact kinematic equations give correct answers when the boxed terms in Eqns. (5.38) and (4.8) are zero.

For the initial configuration in Fig. 4.2(b), however, everything is the same except object 1 is tilted by $\pi/4$, i.e., the initial configuration is $q_0 = (\pi/4, 0, \pi/2, 0, 0)$. According to the first-order kinematics in [**61**],

$$\dot{q}_0 = (0, \sqrt{2}\rho_2\omega k_1, 0, -\rho_1\omega k_1, \rho_2\omega k_1), \tag{4.12}$$

where $k_1 = 1/(\rho_1 + \rho_2)$. Solving the corrected equations (5.38) and (4.8) we obtain

$$(\ddot{u}_1, \ddot{v}_1, \ddot{u}_2, \ddot{v}_2, \ddot{\psi}) = (k_2, 0, 0, 0, 0), \tag{4.13}$$

where $k_2 = \frac{\rho_2^2\omega^2}{(\rho_1+\rho_2)^2}$. As expected, $u_2$ remains constant (contact remains on the "equator" of object 2) as $v_2$ (the "longitude") changes with time. On the other hand, Eqns. (43)

and (44) in [**61**] yield

$$(\ddot{u}_1, \ddot{v}_1, \ddot{u}_2, \ddot{v}_2, \ddot{\psi}) = \left(\left(1 - \frac{2\rho_1}{\rho_1 + \rho_2}\right) k_2, 0, \left(\frac{2\rho_1}{\rho_1 + \rho_2}\right) k_2, 0, 0\right). \tag{4.14}$$

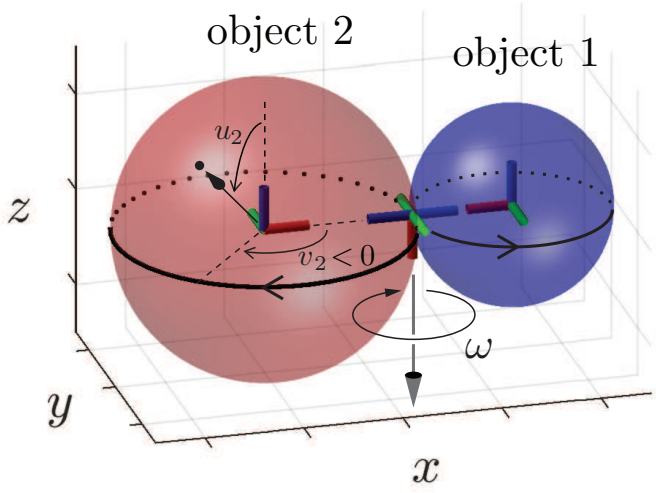
The nonzero value of $\ddot{u}_2$ shows that the contact point incorrectly accelerates away from the equator of object 2.

### 4.5.2. Simulation

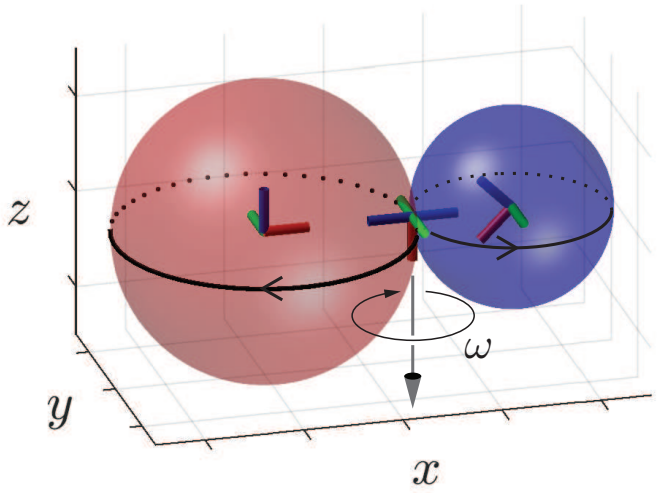
The errors in the original equations are clearly demonstrated by simulation. Figure 4.3 shows the results of numerical simulations of the original second-order kinematics from [**61**] and the corrected equations in this paper for $\omega = 2.5$, sphere radii $\rho_1 = 2$ and $\rho_2 = 3$, and the initial configurations shown in Fig. 4.2. A video of these simulations is available at https://youtu.be/HlMitXg09rg.

## 4.6. Conclusions

This paper presents a corrected version of the second-order contact kinematics for two smooth, rigid bodies in point contact initially derived in [**60**], and later published in [**61**–**63**]. The corrected equations derived in Eqns. (5.38) and (4.8) allow for the accurate simulation of second-order motion of two bodies in contact.

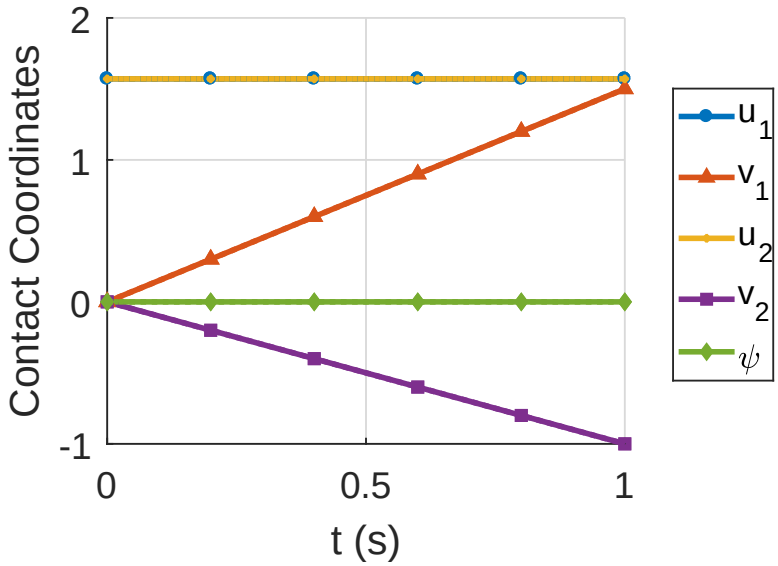



(a) Initial configuration $q_o = (\pi/2, 0, \pi/2, 0, 0)$. The $(u_2, v_2)$ representation of an example point on the surface of object 2 is shown.

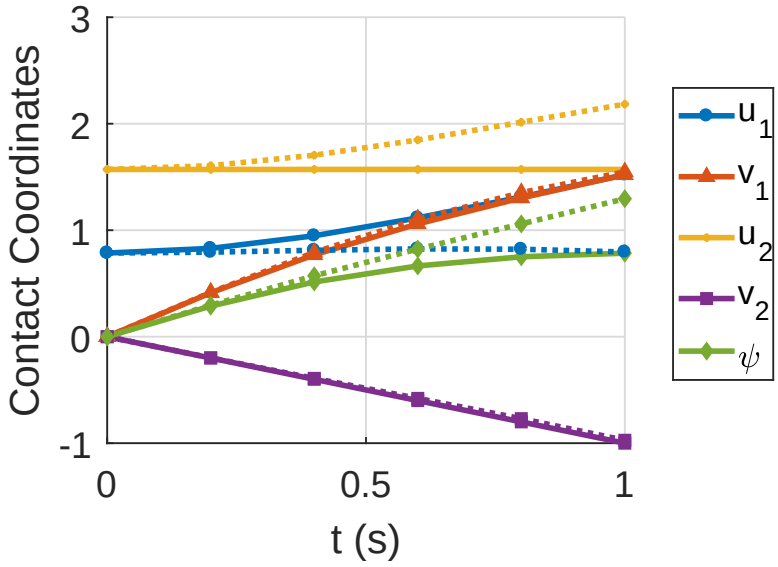



(b) Initial configuration $q_0 = (\pi/4, 0, \pi/2, 0, 0)$.

Figure 4.2. Small blue sphere: the rolling object 1. Large red sphere: the stationary object 2. From both initial configurations, the blue sphere is made to roll on the equator of the red sphere by rotating about the downward-pointing axis in the contact tangent plane at all times.




(a) Simulation starting from $q_o = (\pi/2, 0, \pi/2, 0, 0)$.




(b) Simulation starting from $q_0 = (\pi/4, 0, \pi/2, 0, 0)$.

Figure 4.3. Simulated second-order rolling trajectories with the original kinematics [**61**] shown by the dotted lines and the corrected kinematics by the solid lines. Note that $u_2$ and $\dot{v}_2$ should be constant for both (a) and (b). The original and corrected kinematics yield the same results in (a) because the incorrect terms in the original kinematics are zero, just as they are in the corrected kinematics. The errors in the original kinematics become clear in the simulation in (b).

CHAPTER 5

# Robotic Contact Juggling

## 5.1. Abstract

Chapter 4 focused on second-order kinematics, and we now focus on the more general dynamic rolling. We define "robotic contact juggling" to be the purposeful control of the motion of a three-dimensional smooth object as it rolls freely on a motion-controlled robot manipulator, or "hand." While specific examples of robotic contact juggling have been studied before, in this paper we provide the first general formulation and solution method for the case of an arbitrary smooth object in single-point rolling contact on an arbitrary smooth hand. Our formulation splits the problem into four subproblems: (1) deriving the second-order rolling kinematics; (2) deriving the three-dimensional rolling dynamics; (3) planning rolling motions that satisfy the rolling dynamics; and (4) feedback stabilization of planned rolling trajectories. The theoretical results are demonstrated in simulation and experiment using feedback from a high-speed vision system.

The main contributions of this chapter are that it is the first work we know of to formulate the rolling dynamics of a rigid body rolling on a six-DoF motion-controlled manipulator for general manipulator and object shapes, it provides a compact form that outputs the dynamics and contact wrench that can be used for trajectory optimization, and the coordinate-based representation allows for the dynamics to be linearized to generate feedback controllers that stabilize planned trajectories. We also apply the method to

model, plan, and stabilize dynamic, graspless, and hybrid rolling experiments that include the first known implementation of the rolling pendulum swing up.

## 5.2. Introduction

Contact juggling is a form of object manipulation where the juggler controls the motion of an object, often a crystal ball, as it rolls on the juggler's arms, hands, torso, or even shaved head. The manipulation is typically nonprehensile (no form- or force-closure grasp) and dynamic, i.e., momentum plays a crucial role. An example is shown in Figure 5.1. This is a variation of a contact juggling skill called "the butterfly," and robotic implementations of the butterfly have been described in [**14**,**43**,**68**]. The object (typically a ball) is initially at rest on the palm, and the goal state is rest on the back of the hand. The hand is accelerated to cause the object to roll up and over the fingers to the other side of the hand.

We define "robotic contact juggling" to be the purposeful control of the motion of a three-dimensional smooth object as it rolls freely on a motion-controlled robot manipulator, or "hand." Specific examples of robotic contact juggling have been studied before, such as the butterfly example mentioned above and specific geometries such as a sphere rolling on a motion-controlled flat plate. This paper extends previous work by providing the first general formulation and solution method for the case of an arbitrary smooth object in single-point rolling contact on an arbitrary smooth hand. Our formulation splits the problem into four subproblems: (1) deriving the second-order rolling kinematics; (2) deriving the three-dimensional rolling dynamics; (3) planning rolling motions that satisfy the rolling dynamics and achieve the desired goal state; and (4) feedback stabilization of

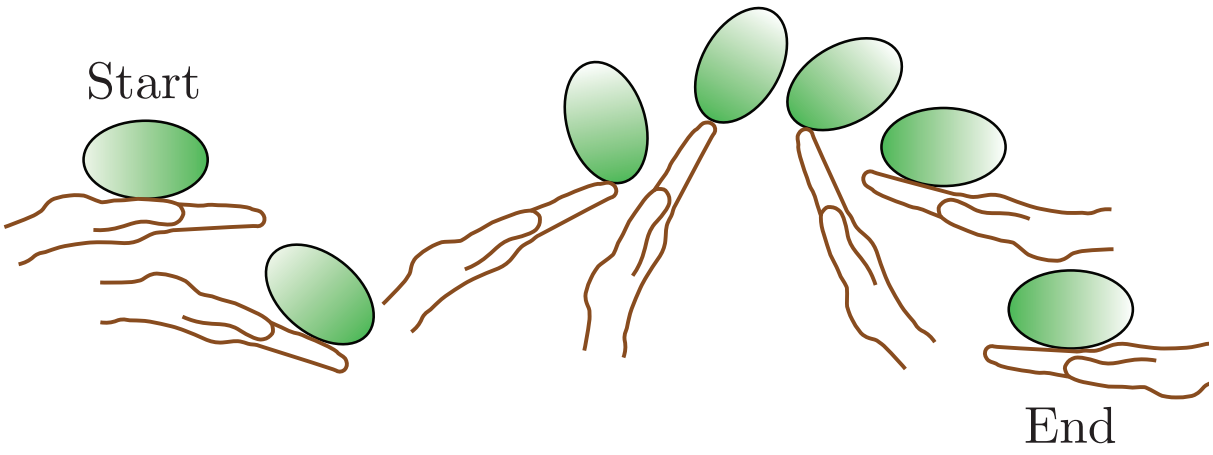



Figure 5.1. Example of a dynamic rolling manipulation task known as "the butterfly". A smooth object is initially at rest in the palm of the hand, and is regrasped to the back of the hand using dynamic rolling without slip or breaking contact.

planned rolling trajectories. The theoretical results are demonstrated in simulation and experiment using feedback from a high-speed vision system.

### 5.2.1. Background

When a three-dimensional rigid body (the object) is in single-point contact with another rigid body (the hand), the configuration of the object relative to the hand is five dimensional: the six degrees of freedom of the object subject to the single constraint that the distance to the hand is zero. This five-dimensional configuration space can be parameterized by two coordinates $\mathbf{u}_o = (u_o, v_o)$ describing the contact location on the surface of the object, two coordinates $\mathbf{u}_h = (u_h, v_h)$ describing the contact location on the surface of the hand, and one coordinate $\psi$ describing the angle of "spin" between frames fixed to each body at the contact point. Collectively the contact configuration is written $\mathbf{q} = (u_o, v_o, u_h, v_h, \psi)$ (Figure 5.2).

Rolling contact is maintained when there is no relative linear velocity at the contact $\mathbf{v}_{\text{rel}} = (v_x, v_y, v_z) = \mathbf{0}$ (i.e. no slipping or separation). For rolling bodies modeled with a point contact, no torques are transmitted through the contact, and relative spin about the contact normal is allowed. We refer to this as "rolling." For rolling bodies modeled with a soft contact, torques can be transmitted and no relative spin about the contact normal is allowed ($\omega_{\text{rel},z} = \omega_z = 0$). We refer to this as "pure rolling."

### 5.2.2. Paper Outline

As mentioned above, our approach to contact juggling divides the problem into four subproblems: (1) calculating the second-order rolling kinematics; (2) deriving the rolling dynamics; (3) planning rolling motions that satisfy the dynamics; and (4) feedback control of rolling trajectories. An outline of each subproblem is given below.

**5.2.2.1. Rolling Kinematics.** First-order kinematics models the evolution of contact coordinates $\mathbf{q}$ between two rigid bodies when the relative contact velocities are directly controlled. The second-order kinematics is a generalization of the first-order model where the relative accelerations at the contact are controlled. These models are used in our derivation of the rolling dynamics to describe the evolution of the contact coordinates during rolling motions. In Section 5.5 we outline the first- and second-order kinematics which follows from work by Sarkar. et al. [**61**], but we also derive a new expression for the acceleration constraints at the contact that enforce pure rolling.

**5.2.2.2. Rolling Dynamics.** The rolling dynamics equations give the accelerations of the object, and contact wrench that results from known accelerations of the manipulator. In Section 5.6 we define a state representation for the dynamic-rolling system and combine

the rolling kinematics with the Newton-Euler dynamics equations to derive an expression for the rolling dynamics. This formulation also characterizes the forces and torques at the contact which allows the tangential and rotational friction limits and normal force constraints to be checked. We validate the simulation of rolling dynamics with an example of a sphere rolling on a rotating plate which has an analytical solution.

**5.2.2.3. Rolling Motion Planning.** In Section 5.7 we use direct collocation methods and the rolling dynamics equations to plan dynamic rolling motions for an object rolling on a manipulator. Given an initial state, we find a set of manipulator controls that brings the system to the goal state.

**5.2.2.4. Feedback Control.** In Section 5.8 we demonstrate the use of a Linear Quadratic Regulator (LQR) feedback controller to stabilize a nominal rolling trajectory.

The motion planning and feedback control are validated experimentally in Section 5.9.

### 5.2.3. Statement of Contributions

To our knowledge this is the first paper to formulate the rolling dynamics of a rigid body rolling on a six-DoF motion-controlled manipulator for general manipulator and object shapes. Our derivation of the rolling dynamics provides a compact form for the rolling equations and contact wrench constraints which can be used in open-loop simulations or nonlinear-constrained trajectory optimization. The coordinate-based representation also allows for the dynamics to be linearized to generate feedback controllers that stabilize planned trajectories. We apply the method to model, plan, and stabilize dynamic, grasspless, and hybrid rolling experiments that include the first known implementation of the rolling pendulum swing up.

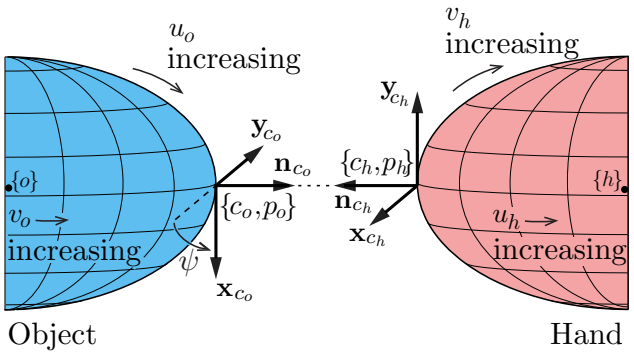



Figure 5.2. The object and hand are in contact at the origin of frames $\{c_o\}$ and $\{c_h\}$, but are shown separated for clarity. Two coincident contact frames $\{p_i\}$ and $\{c_i\}$ for $i \in [o, h]$ are given for each body at the contact, where $\{p_i\}$ is fixed to the object and $\{c_i\}$ is fixed in the inertial frame $\{s\}$. The surfaces of the object and hand are orthogonally parameterized by $(u_o, v_o)$ and $(u_h, v_h)$, respectively. At the point of contact, the $\mathbf{x}_{c_i}$- and $\mathbf{y}_{c_i}$-axes of the coordinate frames ($\{c_i\}$,$\{p_i\}$) are in the direction of increasing $u_i$ (and constant $v_i$) and increasing $v_i$ (and constant $u_i$), respectively, and the contact normal $\mathbf{n}_{c_i}$ is in the direction $\mathbf{x}_{c_i} \times \mathbf{y}_{c_i}$. Rotating frame $\{c_h\}$ by $\psi$ about the $\mathbf{n}_{c_o}$-axis of frame $\{c_o\}$ aligns the $\mathbf{x}_{c_h}$-axis of frame $\{c_h\}$ and the $\mathbf{x}_{c_o}$-axis of frame $\{c_o\}$.

## 5.3. Related Work

### 5.3.1. Kinematic Rolling

The rolling-kinematics equations describe the evolution of the contact coordinates during rolling motions. First-order kinematics addresses the rolling problem where the relative contact velocities are given. The second-order kinematics is a generalization of the first-order model where the relative accelerations at the contact are specified.

Montana derives the first-order contact kinematics for two 3D objects in contact [**49**]. His method models the full five-dimensional configuration space, but it is not easily generalized to second-order kinematics. Harada et al. define the concept of neighborhood equilibrium to perform quasistatic regrasps of an object rolling on a flat manipulator

using Montana's kinematics equations [**26**]. First- and second-order contact equations were derived by Sarkar et al. in [**61**] and republished in later works [**62**, **63**]. Errors in the published equations for second-order contact kinematics in [**61**–**63**] were corrected in our recent work [**75**]. Another of our recent works covers motion planning and feedback control for first-order-kinematic rolling between two surfaces [**76**].

Each of [**26**, **49**, **61**–**63**, **75**, **76**] assumes an orthogonal parameterization, as shown in Figure 5.2. Recent work by Xiao and Ding derives the second order kinematics equations for non-orthogonal surface parameterizations [**77**].

### 5.3.2. Dynamic Rolling

The evolution of dynamic rolling systems is governed by forces and torques at the contact. We review planar and spatial rolling for two- and three-dimensional models. Planar rolling is holonomic whereas spatial rolling is nonholonomic, but both have interesting motion planning and control results, and spatial rolling problems can be locally modeled using planar assumptions.

**5.3.2.1. Planar Rolling.** The butterfly example shown in Figure 5.1 was first introduced by Lynch et al. in [**43**]. Cefalo et al. demonstrate energy-based control of the butterfly robot to move a ball to an unstable equilibrium and stabilize it using a linear approximation of the system. Surov et al. plan and stabilize rolling motions for the butterfly robot using virtual-holonomic-constraints-based planning and transverse-linearization-based orbital stabilization [**68**].

Taylor and Rodriguez perform shape optimization of manipulators and motion planning for dynamic planar manipulation tasks [**70**]. Ryu and Lynch derive a feedback

controller that enables a planar, disk-shaped manipulator to balance a disk while tracking trajectories [**59**]. Erumalla et al. perform throwing, catching, and balancing for an experimental setup of a disk on a disk-shaped manipulator [**21**]. Lippiello et al. develop a control framework for nonprehensile planar rolling dynamic manipulation and validate it experimentally for a disk on a rotating disk [**37**]. Serra et al. apply a passivity-based approach to the same problem in [**65**].

**5.3.2.2. General Spatial Rolling.** General methods exist for simulating rigid bodies in contact, some of which explicitly handle rolling contacts. Anitescu et al. develop a general method for contact dynamic simulations [**5**]. It uses a complementary formulation that allows simulation of multiple rigid bodies, and uses the first-order-rolling equations from Montana [**49**] to solve for contact constraints. Liu and Wang develop a time-stepping method for rigid-body dynamics that specifically addresses rolling contacts [**38**]. Duindam et al. model the kinematics and dynamics of compliant contact between bodies moving in Euclidean space [**19**]. There are many software packages for performing dynamic simulations [**28**], and a subset of those can explicitly handle rolling constraints. Because each object state is represented as an independent six-DoF rigid body, these methods allow for non-zero contact distances (separation or penetration) and are less focused on modeling the rolling interaction between objects.

**5.3.2.3. Prehensile Spatial Rolling.** There are many works that address prehensile motion planning for a ball (sphere) that is in contact with a stationary plate but actuated by a second plate [**8**, **18**, **27**, **31**, **53**]. Sarkar et al. expand the second-order kinematics equations to generate a dynamic model for an object caged between two surfaces [**63**].

They use feedback linearization to control dynamic rolling motions for two planes in contact with a sphere.

**5.3.2.4. Nonprehensile Spatial Rolling.** Jia and Erdmann derive dynamic equations and show the observability of an object with an orthogonal parameterization on a flat plate equipped with a contact position sensor [**29**]. Choudhury and Lynch build off of this work and stabilize the orientation of a ball rolling in an ellipsoidal dish actuated along a single degree of freedom [**16**]. Both these works assume no rotational motion of the hand which simplifies the modeling and control problem.

Gahleitner models a sphere with three rotation inputs balancing another sphere, designs a stabilizing controller, and validates the results experimentally [**24**]. Additionally there are numerous papers on control for an object in nonprehensile contact with a plate that control the contact location on the hand but do not consider the orientation of the object [**7**,**35**]. To the extent of my knowledge, only one paper does dynamic, nonprehensile planning for a ball on a plate while considering the full position and pose of the ball [**64**]. Works by Milne (part III.XV of [**47**]) and Weltner [**73**] derive analytical solutions for a sphere rolling on a rotating plate with a constant angular velocity that we use to validate our rolling dynamics equations in Section 5.6.

As mentioned in Section 5.2.3, to our knowledge this paper is the first to formulate the rolling dynamics of a rigid body rolling on a six-DoF motion-controlled manipulator for general manipulator and object shapes.

## 5.4. Notation

For general variables, matrices and vectors are bold uppercase and bold lowercase letters respectively. Scalars are non-bold italic, and coordinate frames are expressed as lowercase letters in curly brackets. Notations for variables and operators (mappings between spaces) are shown in Table 5.1, and selected operator expressions are given in Appendix 5.11.

The space frame (i.e., inertial reference) is defined as $\{s\}$, and the hand frame is define as $\{h\}$. The object frame $\{o\}$ is located at its center of mass and aligned with the principal axes. Two coincident contact frames $\{p_i\}$ and $\{c_i\}$ for $i \in [o, h]$ are given for each body at the contact, where $\{p_i\}$ is fixed to the body and $\{c_i\}$ is fixed in $\{s\}$ (see Figure 5.2).

Double subscripts indicate a comparison between two frames expressed in the frame of the first subscript. For example, $\mathbf{R}_{so}$ gives the rotation matrix relating frame $\{o\}$ relative to frame $\{s\}$ expressed in $\{s\}$, and $\mathcal{V}_{sh}$ gives the twist of the hand relative to the space frame $\{s\}$ expressed in $\{s\}$. In deriving the kinematics and dynamics equations, we often compare the relative velocities between different frames expressed in a specific frame which we denote with a preceding superscript. For example, ${}^{c_h}\mathcal{V}_{p_h p_o}$ gives the twist of frame $\{p_o\}$ relative to $\{p_h\}$ expressed in the $\{c_h\}$ frame and is equivalent to $[\mathrm{Ad}_{\mathbf{T}_{c_h s}}](\mathcal{V}_{sp_o} - \mathcal{V}_{sp_h})$, where $[\mathrm{Ad}_{\mathbf{T}}]$ is the adjoint map operator associated with the transformation matrix $\mathbf{T}$ that maps variables between coordinate frames. "Body twists" are defined as ${}^{i}\mathcal{V}_{si}$, and represent the twists of the body relative to the space frame, expressed in its own coordinate frame $\{i\}$. Variables must be expressed in the same frame to compare, and we resolve the kinematics and dynamics expressions in the contact frame of the hand $\{c_h\}$.

Table 5.1. Notation

| Variable | Description | Dimensions |
|---|---|---|
| $\{\cdot\}$ | Coordinate frame | - |
| $\mathbf{r}$ | Position vector $\mathbf{r} = (x, y, z)$ | $3 \times 1$ |
| $\Phi$ | **xyz** Euler angles $\Phi = (\theta, \beta, \gamma)$ | $3 \times 1$ |
| $\mathbf{R}$ | Rotation matrix $\mathbf{R} \in SO(3)$ | $3 \times 3$ |
| $\mathbf{T}$ | Transformation matrix $\mathbf{T} \in SE(3)$ | $4 \times 4$ |
| $\omega$ | Rotational velocity $\omega = (\omega_x, \omega_y, \omega_z)$ | $3 \times 1$ |
| $\mathbf{v}$ | Linear velocity $\mathbf{v} = (v_x, v_y, v_z)$ | $3 \times 1$ |
| $\mathcal{V}$ | Twist $\mathcal{V} = (\omega, \mathbf{v})$ | $6 \times 1$ |
| $\alpha$ | Rotational acceleration $\alpha = \dot{\omega} = (\alpha_x, \alpha_y, \alpha_z)$ | $3 \times 1$ |
| $\mathbf{a}$ | Linear acceleration $\mathbf{a} = \dot{\mathbf{v}} = (a_x, a_y, a_z)$ | $3 \times 1$ |
| $\dot{\mathcal{V}}$ | Change in twist $\dot{\mathcal{V}} = (\alpha, \mathbf{a})$ | $6 \times 1$ |
| $\mathbf{q}$ | Contact coordinates $\mathbf{q} = (u_o, v_o, u_h, v_h, \psi)$ | $5 \times 1$ |
| $\mathcal{G}$ | Spatial inertia matrix | $6 \times 6$ |
| $\mathcal{F}$ | Wrench $\mathcal{F} = (\tau_x, \tau_y, \tau_z, f_x, f_y, f_z)$ | $6 \times 1$ |
| $\mathbf{s}$ | Dynamic rolling states | $22 \times 1$ |
| $\xi$ | State and control pair $(\mathbf{s}, {}^h\dot{\mathcal{V}}_{sh})$ | $28 \times 1$ |

| Operator | Description (expressions in Appendix 5.11) | Mapping |
|---|---|---|
| $[\cdot]$ | Vector to skew-symmetric form | $\mathbb{R}^3 \mapsto so(3)$ |
| $[\mathrm{Ad}_{\mathbf{T}}]$ | Adjoint map associated with $\mathbf{T}$ | $SE(3) \mapsto \mathbb{R}^{6\times6}$ |
| $[\mathrm{ad}_{\mathcal{V}}]$ | Lie bracket matrix form of $\mathcal{V}$ | $\mathbb{R}^6 \mapsto \mathbb{R}^{6\times6}$ |
| $\mathbb{F}(\mathbf{u})$ | Surface parameterization | $(u, v) \mapsto (x, y, z)$ |

The surface of each body is represented by an orthogonal parameterization: $\mathbb{F}_i : \mathbf{u}_i \to \mathbb{R}^3 : (u_i, v_i) \mapsto (x_i, y_i, z_i)$, where the coordinates $(x_i, y_i, z_i)$ are expressed in the $\{i\}$ frame. We assume that $\mathbb{F}_i$ is continuous up to the third derivative (class $C^3$), so that the local contact geometry (contact frames associated with the first derivative of $\mathbb{F}_i$, curvature associated with the second derivative, and derivative of the curvature associated with the third derivative) are uniquely defined. Standard expressions for the local geometry of smooth bodies are given in Appendix 5.12.1.

## 5.5. Rolling Kinematics

### 5.5.1. First-Order-Rolling Kinematics

The contact configuration of two bodies in rolling contact can be parameterized by $\mathbf{q} = (u_o, v_o, u_h, v_h, \psi)$ (see Figure 5.2). First-order kinematics models the evolution of contact coordinates between two rigid bodies when the relative contact velocities are known. The relative twist at the contact in $\{c_h\}$ is given by ${}^{c_h}\mathcal{V}_{p_h p_o} = \mathcal{V}_{\text{rel}} = [\omega_x\ \omega_y\ \omega_z\ v_x\ v_y\ v_z]^{\mathsf{T}}$. To maintain contact, the non-separation constraint $v_z = 0$ must be satisfied. Enforcing the constraints $v_x = v_y = 0$ ensures rolling without linear slip in the contact tangent plane (rolling). Further, enforcing the constraint $\omega_z = 0$ ensures no relative spinning about the contact normal (pure rolling). We refer to these as the first-order-rolling and first-order-pure-rolling constraints respectively. The first-order-rolling kinematics ($v_x = v_y = v_z = 0$) from [**61**] can be expressed in matrix form as:

$$\dot{\mathbf{q}} = \mathbf{K}_1(\mathbf{q})\omega_{\text{rel}}, \tag{5.1}$$

where $\omega_{\text{rel}} = {}^{c_h}\omega_{p_h p_o} = [\omega_x\ \omega_y\ \omega_z]^{\mathsf{T}}$ is the relative rotational velocity at the contact expressed in $\{c_h\}$. The matrix $\mathbf{K}_1(\mathbf{q})$, given in Appendix 5.12.2, maps the relative rotational velocity at the contact to the change in contact coordinates $\dot{\mathbf{q}}$. The dimension of valid contact velocities for rolling and pure rolling are three and two respectively, so the five-dimensional $\dot{\mathbf{q}}$ is subject to constraints given at the end of Appendix 5.12.2.

An expression for the body twist of the object given the body twist of the hand ${}^{h}\mathcal{V}_{sh}$ and the relative twist at the contact $\mathcal{V}_{\text{rel}}$ is used in the dynamics derivation in Section 5.6.

The equation is defined as:

$$^{o}\mathcal{V}_{so} = [\mathrm{Ad}_{\mathbf{T}_{oh}}]^{h}\mathcal{V}_{sh} + [\mathrm{Ad}_{\mathbf{T}_{oc_h}}]\mathcal{V}_{\mathrm{rel}}, \tag{5.2}$$

where $\mathcal{V}_{\mathrm{rel}} = {}^{c_h}\mathcal{V}_{p_h p_o}$, and $[\mathrm{Ad}_{\mathbf{T}}]$ is the adjoint map associated with the transformation matrix $\mathbf{T}$.

### 5.5.2. Second-Order-Rolling Kinematics

The general form of the second-order kinematics gives $\ddot{\mathbf{q}}$ as a function of the current state and $\dot{\mathcal{V}}_{\mathrm{rel}} = {}^{c_h}\dot{\mathcal{V}}_{p_h p_o} = [\alpha_x \ \alpha_y \ \alpha_z \ a_x \ a_y \ a_z]^{\mathsf{T}}$ (the derivative of the relative twist at the contact expressed in $\{c_h\}$). This expression includes the relative rotational accelerations $\alpha_{\mathrm{rel}} = [\alpha_x \ \alpha_y \ \alpha_z]^{\mathsf{T}}$ and the relative linear accelerations $\mathbf{a}_{\mathrm{rel}} = [a_x \ a_y \ a_z]^{\mathsf{T}}$. The second-order kinematics from [**61**] can be expressed in matrix form as:

$$\ddot{\mathbf{q}} = \mathbf{K}_2(\mathbf{q}, \omega_{\mathrm{rel}}) + \mathbf{K}_3(\mathbf{q})\dot{\mathcal{V}}_{\mathrm{rel}}, \tag{5.3}$$

where the terms $\mathbf{K}_2(\mathbf{q}, \omega_{\mathrm{rel}})$ and $\mathbf{K}_3(\mathbf{q})$ are given in Appendix 5.12.3.

This general form allows relative sliding at the contact. To maintain rolling, $\dot{\mathcal{V}}_{\mathrm{rel}}$ must lie in a three-dimensional subspace satisfying

$$\mathbf{a}_{\mathrm{rel}} = \mathbf{a}_{\mathrm{roll}} = [a_x \ a_y \ a_z]^{\mathsf{T}}_{\mathrm{roll}} = -\omega_{\mathrm{rel}} \times {}^{c_h}\mathbf{v}_{c_o o}, \tag{5.4}$$

as derived in Eq. (60) of [**61**]. To maintain pure rolling, $\dot{\mathcal{V}}_{\mathrm{rel}}$ must lie in a two-dimensional subspace additionally satisfying the constraint $\alpha_z = \alpha_{z,\mathrm{pr}}$, which is different from the result found in [**61**], and is derived for the first time in Appendix 5.12.3.1. For the case of

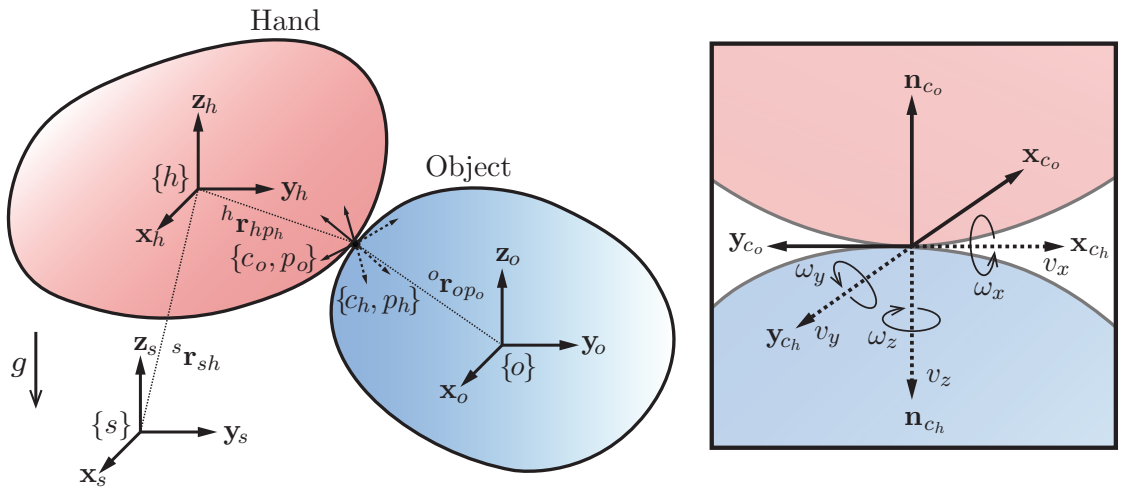



Figure 5.3. Rolling rigid bodies in space. The contact cordinate frames $\{c_o, p_o\}$ and $\{c_h, p_h\}$ are shown by solid and dotted coordinate axes respectively. The box shows a zoomed view of the frames at the contact and the relative rotational and linear velocity directions.

pure rolling $\omega_z = 0$, and the relative linear acceleration constraints $[a_x \; a_y]^\mathsf{T}_{\text{roll}}$ in Eq. (5.4) simplify to 0.

An expression for the body accelerations of the object ${}^o\dot{\mathcal{V}}_{so}$ as a function of the hand accelerations ${}^h\dot{\mathcal{V}}_{sh}$ and the relative accelerations $\dot{\mathcal{V}}_{\text{rel}}$ is used in the dynamics derivation in Section 5.6. Taking the derivative of Eq. (5.2) in frame $\{c_h\}$ (following the derivative rule for expressions in different frames from Appendix 5.11.2) gives:

$$\begin{aligned} {}^o\dot{\mathcal{V}}_{so} =& [\text{Ad}_{\mathbf{T}_{oh}}]\,{}^h\dot{\mathcal{V}}_{sh} + [\text{Ad}_{\mathbf{T}_{oc_h}}]\dot{\mathcal{V}}_{\text{rel}} \\ &+ \mathbf{K}_4(\mathbf{q}, \omega_{\text{rel}}, {}^h\omega_{sh}), \end{aligned} \tag{5.5}$$

where $\mathbf{K}_4(\mathbf{q}, \omega_{\text{rel}}, {}^h\omega_{sh})$ contains the velocity-product terms and is given in Appendix 5.12.3.2.

## 5.6. Rolling Dynamics

A diagram of the object and hand in rolling contact is shown in Figure 5.3. The full six-dimensional orientation and position of each body $i \in [o, h]$ is expressed by the transformation matrix $\mathbf{T}_{si} \in SE(3)$ that consists of the rotation matrix $\mathbf{R}_{si} \in SO(3)$ and vector $\mathbf{r}_{si} = (x_{si}, y_{si}, z_{si})$ that give the orientation and position of $\{i\}$ relative to the space frame $\{s\}$. The orientation of each body can be minimally represented by the roll-pitch-yaw Euler angles $\Phi_{si} = (\theta_i, \beta_i, \gamma_i)$. These map to a rotation matrix by combining rotations about the $x$-, $y$-, and $z$-axes of the space frame: $\mathbf{R}_{si}(\Phi_{si}) = \text{Rot}(\mathbf{z}_s, \gamma_i)\text{Rot}(\mathbf{y}_s, \beta_i)\text{Rot}(\mathbf{x}_s, \theta_i)$, where $\text{Rot}(\mathbf{z}_s, \gamma_i)$ is the rotation matrix representing a rotation of angle $\gamma_i$ about the axis $\mathbf{z}_s$. The six-dimensional velocity vector for each body is represented by the body twist ${}^i\mathcal{V}_{si} = ({}^i\omega_{si}, {}^i\mathbf{v}_{si})$ expressed in the $\{i\}$ frame for $i \in [o, h]$.

The combined configurations of the object and hand are 12-dimensional, subject to one constraint that the distance between the two bodies is zero. The velocities of the object and hand are 12-dimensional, with three velocity constraints for rolling ($v_x = v_y = v_z = 0$), and four velocity constraints for pure rolling ($v_x = v_y = v_z = \omega_z = 0$). A minimal representation of the state of the system therefore requires 20 states for rolling, or 19 states for pure rolling. The configuration of the hand can be minimally represented by the pair $(\Phi_{sh}, \mathbf{r}_{sh})$, and therefore $\mathbf{T}_{so}$, the configuration of the object, is fully specified by the hand configuration $(\Phi_{sh}, \mathbf{r}_{sh})$ and the contact configuration variables $\mathbf{q}$. The body twist of the hand is represented by ${}^h\mathcal{V}_{sh}$, and the body twist of the object ${}^o\mathcal{V}_{so}$ is fully specified by Eq. (5.2) using the state of the hand $(\Phi_{sh}, \mathbf{r}_{sh}, {}^h\mathcal{V}_{sh})$, the contact configuration $\mathbf{q}$, and the relative rolling velocity $\omega_{\text{rel}}$. We represent the relative rolling velocity as the five-dimensional vector $\dot{\mathbf{q}}$ from Eq. (5.1) instead of the three-dimensional $\omega_{\text{rel}}$. This adds

two variables to the state representation, but allows the evolution of the contact velocities to be integrated using Eq. (5.3). We therefore define the state of the dynamic rolling system as $\mathbf{s} = (\Phi_{sh}, \mathbf{r}_{sh}, \mathbf{q}, {}^{h}\mathcal{V}_{sh}, \dot{\mathbf{q}}) \in \mathbb{R}^{22}$ with two constraints on $\dot{\mathbf{q}}$ for rolling, and three constraints on $\dot{\mathbf{q}}$ for pure rolling given in Appendix 5.12.2.

We assume that the hand is directly controlled by acceleration inputs ${}^{h}\dot{\mathcal{V}}_{sh}$. The model parameters include contact friction model (rolling vs. pure rolling) and friction coefficient(s), the surface parameterizations $\mathbb{F}_i(\mathbf{u}_i)$, and the spatial inertia matrix of the object: $\mathcal{G}_o = \text{blockdiag}(\mathbf{J}_o, m_o\mathbf{I}_3)$, where $\mathbf{J}_o$ is the rotational inertia matrix for the object, $m_o$ is the mass of the object, and $\mathbf{I}_3$ is the $3\times3$ identity matrix. We then solve for dynamic equations describing the motion of the object as follows:

**Given:**

(1) Model parameters $\mathbb{F}_i(\mathbf{u}_i)$, $\mathcal{G}_o$, and friction parameters

(2) States: $\mathbf{s} = (\Phi_{sh}, \mathbf{r}_{sh}, \mathbf{q}, {}^{h}\mathcal{V}_{sh}, \dot{\mathbf{q}})$

(3) Acceleration of the hand: ${}^{h}\dot{\mathcal{V}}_{sh} = ({}^{h}\alpha_{sh}, {}^{h}\mathbf{a}_{sh})$

**Find:**

(1) Relative rotational accelerations at the contact: $\alpha_{\text{rel}}$

(2) Contact wrench ${}^{c_h}\mathcal{F}_{\text{contact}}$ (to test friction limits)

### 5.6.1. Rolling Dynamics Derivation

The six body accelerations of the object ${}^{o}\dot{\mathcal{V}}_{so}$ are governed by the Newton-Euler equations (see Ch. 8.2 of [**42**]) and can be expressed as:

$$\mathcal{G}_o {}^{o}\dot{\mathcal{V}}_{so} = [\text{ad}_{{}^{o}\mathcal{V}_{so}}]^{\mathsf{T}} \mathcal{G}_o {}^{o}\mathcal{V}_{so} + {}^{o}\mathcal{F}_{g_o} + {}^{o}\mathcal{F}_{\text{contact}}, \tag{5.6}$$

where $\mathcal{G}_o$ is the body inertia matrix of the object, and ${}^o\mathcal{F}_{g_o}$ is the gravitational wrench on the object. The contact wrench ${}^o\mathcal{F}_{\text{contact}}$ is given by ${}^o\mathcal{F}_{\text{contact}} = [\text{Ad}_{\mathbf{T}_{c_h o}}]^{\mathsf{T}\,c_h}\mathcal{F}_{\text{contact}}$, where ${}^{c_h}\mathcal{F}_{\text{contact}} = [0\ 0\ 0\ f_x\ f_y\ f_z]^{\mathsf{T}}$ for rolling and ${}^{c_h}\mathcal{F}_{\text{contact}} = [0\ 0\ \tau_z\ f_x\ f_y\ f_z]^{\mathsf{T}}$ for pure rolling. Eq. (5.5) can be substituted into Eq. (5.6) to obtain:

$$
\begin{aligned}
&[\text{Ad}_{\mathbf{T}_{oc_h}}]\dot{\mathcal{V}}_{\text{rel}} - {\mathcal{G}_o}^{-1}[\text{Ad}_{\mathbf{T}_{c_h o}}]^{\mathsf{T}\,c_h}\mathcal{F}_{\text{contact}} = \\
&{\mathcal{G}_o}^{-1}([\text{ad}_{{}^o\mathcal{V}_{so}}]^{\mathsf{T}}\,\mathcal{G}_o\,{}^o\mathcal{V}_{so} + {}^o\mathcal{F}_{g_o}) \\
&-\mathbf{K}_4(\mathbf{q}, \omega_{\text{rel}}, {}^h\omega_{sh}) - [\text{Ad}_{\mathbf{T}_{oh}}]^h\dot{\mathcal{V}}_{sh},
\end{aligned}
\tag{5.7}
$$

where the only unknowns are in $\dot{\mathcal{V}}_{\text{rel}}$ and ${}^{c_h}\mathcal{F}_{\text{contact}}$. We substitute the second-order-rolling constraints $\mathbf{a}_{\text{rel}} = \mathbf{a}_{\text{roll}}$ in Eq. (5.4) for rolling, and the additional constraint $\alpha_z = \alpha_{z,\text{pr}}$ in Eq. (5.43) for pure rolling. Rearranging gives us the following forms for rolling and pure rolling, respectively:

$$
\mathbf{K}_5(\mathbf{s})[\alpha_x\ \alpha_y\ \alpha_z\ f_x\ f_y\ f_z]^{\mathsf{T}} = \mathbf{K}_6(\mathbf{s}) - [\text{Ad}_{\mathbf{T}_{oh}}]^h\dot{\mathcal{V}}_{sh}, \tag{5.8}
$$

$$
\mathbf{K}_{5_{\text{pr}}}(\mathbf{s})[\alpha_x\ \alpha_y\ \tau_z\ f_x\ f_y\ f_z]^{\mathsf{T}} = \mathbf{K}_{6_{\text{pr}}}(\mathbf{s}) - [\text{Ad}_{\mathbf{T}_{oh}}]^h\dot{\mathcal{V}}_{sh}. \tag{5.9}
$$

The expressions for $\mathbf{K}_5(\mathbf{s})$, $\mathbf{K}_6(\mathbf{s})$, $\mathbf{K}_{5_{\text{pr}}}(\mathbf{s})$, $\mathbf{K}_{6_{\text{pr}}}(\mathbf{s})$ are omitted for brevity, but are the result of straightforward linear algebra on equation (5.7). An example derivation can be found in the supplemental code. For the rolling assumption, Eq. (5.8) can be solved to find the relative rotational acceleration at the contact $\alpha_{\text{rel}} = [\alpha_x\ \alpha_y\ \alpha_z]$. For the pure-rolling assumption, Eq. (5.9) can be solved for $[\alpha_x\ \alpha_y]$ and combined with the pure-rolling constraint on $\alpha_z$ from Eq. (5.43) to find $\alpha_{\text{rel}}$.

The contact wrenches for rolling and pure rolling can be extracted from Eq. (5.8) and Eq. (5.9) respectively:

$$ {}^{c_h}\mathcal{F}_{\text{contact,roll}} = [0\ 0\ 0\ f_x\ f_y\ f_z]^{\mathsf{T}}, \tag{5.10} $$

$$ {}^{c_h}\mathcal{F}_{\text{contact,pr}} = [0\ 0\ \tau_z\ f_x\ f_y\ f_z]^{\mathsf{T}}. \tag{5.11} $$

The dynamics equations allow positive and negative normal force and infinite tangential force at the contact (and spinning torque for pure rolling). The contact wrench expressions can be used to define nonlinear constraints that enforce non-negative normal force and bounded contact wrenches. Positive normal force is enforced by $f_z \geq 0$, and the tangential friction force constraint can be expressed as $[f_x^2\ f_y^2]^{\mathsf{T}} \leq f_z^2\mu_s$, where $\mu_s$ is the coefficient of static friction. For pure rolling, the rotational torque constraint can be additionally enforced as $\tau_z^2 \leq f_z^2\mu_{\text{spin}}$, where $\mu_{\text{spin}}$ is the coefficient of spinning friction at the contact. These constraints can be used during nonlinear trajectory optimization to enforce the friction limits.

### 5.6.2. Simulating the Rolling Dynamics

The state of the dynamic rolling system is defined as $\mathbf{s} = (\Phi_{sh}, \mathbf{r}_{sh}, \mathbf{q}, {}^h\mathcal{V}_{sh}, \dot{\mathbf{q}}) \in \mathbb{R}^{22}$. The state evolution of the hand is directly controlled by the change in body twist ${}^h\dot{\mathcal{V}}_{sh}$. The state of the object is represented by the state of the hand, the contact coordinates $\mathbf{q}$, and coordinate velocities $\dot{\mathbf{q}}$. The contact coordinate accelerations $\ddot{\mathbf{q}}$ are needed to integrate the contact coordinates over time. An expression for $\ddot{\mathbf{q}}$ is given by the second-order kinematics in Eq. (5.3), which takes the relative rotational accelerations $\alpha_{\text{rel}}$ as an input. The expression for $\alpha_{\text{rel}}$ is found by solving Eq. (5.8) and Eq. (5.9) for rolling and

pure rolling respectively. For both rolling and pure rolling the dynamics equations can be rearranged into control-affine form:

$$\dot{\mathbf{s}} = \mathbf{K}_7(\mathbf{s}) + \mathbf{K}_8(\mathbf{s})\,{}^h\dot{\mathcal{V}}_{sh}, \tag{5.12}$$

where the expressions for $\mathbf{K}_7(\mathbf{s})$ and $\mathbf{K}_8(\mathbf{s})$ are large symbolic expressions that are omitted for brevity (an example derivation can be found in the supplemental code). The evolution of the state $\mathbf{s}$ can be simulated using a numerical integrator such as MATLAB's ode45.

In implementation one can avoid large, symbolic matrix inversions needed to solve for Eq. (5.12) by numerically evaluating $\mathbf{K}_5$ and $\mathbf{K}_6$ from Eq. (5.8) for rolling or $\mathbf{K}_{5_{\text{pr}}}$ and $\mathbf{K}_{6_{\text{pr}}}$ from Eq. (5.9) for pure rolling at each time step, solving for $\alpha_{\text{rel}}$, and then solving Eq. (5.3) numerically for $\ddot{\mathbf{q}}$. The state $\mathbf{s}$, controls ${}^h\dot{\mathcal{V}}_{sh}$, and $\ddot{\mathbf{q}}$ can then be combined into the vector $\dot{\mathbf{s}}$ from Eq. (5.12).

### 5.6.3. Example: Ball on a Rotating Plate

To validate our dynamic equations (Eq. (5.12)) we consider a solid, homogeneous sphere rolling without slipping on a plate spinning at a constant speed about an axis perpendicular to the plate. The plate may be perpendicular to gravity (horizontal) or inclined. This is a well-studied problem (see [**73**] and Part III.XV of [**47**]) with analytical solutions for the motion of the sphere. For a horizontal plate, the contact point of the ball rolls in a circular orbit on the plate (Figure 5.4(a)), and if the plate is inclined in gravity, the motion is the circular orbit plus a constant drift in a direction perpendicular to the gravitational component in the plane of the plate (Figure 5.4(b)). The circle radius and center point are determined analytically from the initial conditions of the ball. Simulations of the

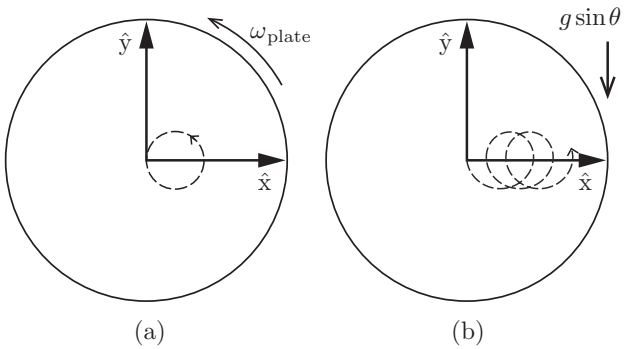



Figure 5.4. The $x$- and $y$-axes shown are fixed in the inertial frame and aligned with the plate. The plate spins about the $z$-axis at an angular velocity $\omega_{\text{plate}}$. A sphere is initially in contact at the origin and rolling in the $-y$ direction without slipping. In (a) the plane of the plate is in the $x$-$y$ plane of the inertial frame and gravity acts in the $-z$ direction. The contact point of the sphere follows a circular orbit. In (b) the plate is tilted by angle $\theta$ about the $x$-axis of the inertial frame, so has a component $g \sin\theta$ in the $-y$ direction. The contact point motion is the sum of the circular orbit and a constant drift in the $+x$ direction.

dynamic rolling equations from Section 5.6.1 are consistent with the analytical solutions, as demonstrated in the following two examples.

**5.6.3.1. Horizontal Rotating Plate.** Consider a horizontal plate coincident with the origin of the inertial frame with a constant rotational velocity about its $z$-axis, ${}^{h}\omega_{sh,z} = \omega_{\text{plate}}$ (Figure 5.4(a)). A ball with radius $\rho_o$ is initially in contact with the plate at ${}^{h}\mathbf{r}_{hc_h}(0)$, where $\{c_h\}$ is the contact frame on the plate. The ball has an initial linear velocity ${}^{h}\mathbf{v}_{ho}$ and no rotational velocity. From [**73**], the ball follows a circular trajectory with the following properties:

$$\omega_c = \frac{2}{7}\omega_{\text{plate}}, \tag{5.13}$$

$${}^h\mathbf{r}_{hc} = {}^h\mathbf{r}_{hc_h}(0) - {}^h\mathbf{v}_{ho} \times [0\ 0\ 1/\omega_c]^\mathsf{T}, \tag{5.14}$$

$$\rho_c = ||{}^h\mathbf{r}_{hc_h}(0) - {}^h\mathbf{r}_{hc}||, \tag{5.15}$$

where ${}^h\mathbf{r}_{hc}$ and $\rho_c$ are the center and radius of the circle trajectory, respectively, and $\omega_c$ is the angular velocity of the contact point about the center of the circle.

For this validation, we choose the following parameters and initial conditions for a ball in contact at the origin that is initially rolling without slipping in the $-y$ direction: $\omega_{\text{plate}} = 7$ rad/s, $\rho_o = 2$ m, ${}^h\mathbf{r}_{hc_h}(0) = [0\ 0\ 0]^\mathsf{T}$ m, ${}^h\mathbf{v}_{ho} = [0\ -2\ 0]^\mathsf{T}$ m/s. Evaluating at the initial conditions gives $\omega_c = 2$ rad/s, ${}^h\mathbf{r}_{hc} = [1\ 0\ 0]^\mathsf{T}$ m, and $\rho_c = 1$ m.

We now simulate the rolling sphere using the dynamic rolling method derived in Section 5.6.1. The surface of the ball is parameterized by the sphere equation $\mathbb{F}_o : \mathbf{u}_o \to \mathbb{R}^3 : (u_o, v_o) \mapsto (\rho_o \sin(u_o)\cos(v_o), \rho_o \sin(u_o)\sin(v_o), \rho_o \cos(u_o))$, where the "latitude" $u_o$ satisfies $0 < u_o < \pi$. The plate is parameterized as a plane $\mathbb{F}_h : \mathbf{u}_h \to \mathbb{R}^3 : (u_h, v_h) \mapsto (u_h, v_h, 0)$. The spatial inertia matrix for the sphere is given by $\mathcal{G}_o = \text{blockdiag}(\mathbf{J}_o,\ m_o\mathbf{I}_3)$, where $\mathbf{J}_o = \frac{2}{5}m_o\rho_o^2\mathbf{I}_3$, $m_o = 1$ kg, and $\mathbf{I}_3$ is the $3 \times 3$ identity matrix. The friction model is rolling (relative spin at the contact allowed), and the static friction $\mu_s$ is large enough so slip does not occur.

The state of the sphere-on-plane system can be represented by the state vector $\mathbf{s} = (\Phi_{sh}, \mathbf{r}_{sh}, \mathbf{q}, {}^h\mathcal{V}_{sh}, \omega_{\text{rel}})$, where $\Phi_{sh}(0) = (0,0,0)$, $\mathbf{r}_{sh}(0) = (0,0,0)$, $q(0) = (\pi/2, 0, 0, 0, 0)$,

${}^h\mathcal{V}_{sh}(0) = (0, 0, 7, 0, 0, 0)$, $\omega_{\text{rel}}(0) = (1, 0, -7)$ gives $\dot{\mathbf{q}} = (0, 1, 0, -2, -7)$ from Eq. (5.1), and the control input is given as ${}^h\dot{\mathcal{V}}_{sh}(t) = (0, 0, 0, 0, 0, 0)$.

The first- and second-order kinematics equations for the sphere-on-plane system are found using the expressions in Appendix 5.12. The state of the system is then simulated using the kinematics equations and the rolling dynamics in Eq. (5.12). The simulated rolling trajectory matches the analytical solution by tracing a circular trajectory on the plane of radius $\rho_c = 1$ m, centered at ${}^h\mathbf{r}_{hc} = [1\ 0\ 0]^\mathsf{T}$ m, and in $t_f = \pi$ ($\omega_c = 2$ rad/s). A visualization of the trajectory is shown in Figure 5.5(a) and a 10 second video of the simulation is included here: https://youtu.be/EZggYdh3F2M. We tested the numerical accuracy of our method by simulating the system for 120 seconds. The divergence of the radius from the analytical circular trajectory was less than 0.01%.

To compare the accuracy of our approach to a commonly used physics simulator we implemented the rolling example using Bullet Physics C++ version 2.89, and found comparable results except that the object diverged from the circular trajectory over time as shown in Figure 5.6. A video of the bullet simulation is included here: https://youtu.be/1eovapg_Ako. This method had much larger errors, and the divergence of the radius from the analytical circular trajectory was 40% after only 10 seconds.

**5.6.3.2. Tilted Rotating Plate.** The motion of a ball on a tilted plate with a constant rotational velocity is a cycloidal orbit with the same rotational velocity $\omega_c$ from the previous section but with an additional uniform linear drift velocity perpendicular to the force of gravity given by:

$$v_{\text{drift}} = \frac{5}{2}\frac{g}{\omega_{\text{plate}}}\sin\theta, \tag{5.16}$$

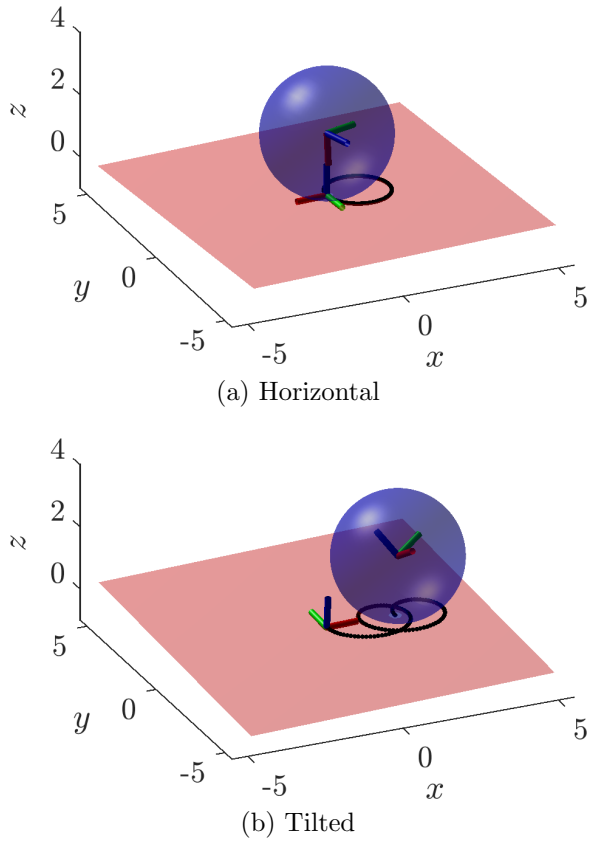



Figure 5.5. Visualizations of the sphere-on-plane rolling trajectories for a plane with a constant rotational body velocity ${}^{h}\omega_{sh,z} = \omega_{\text{plate}} = 7$ rad/$s$. The paths are shown by the black, dotted lines for (a) a horizontal plane and (b) a plane tilted by 0.1 rad about the $x$-axis of the inertial frame (see Figure 5.4(b)). The spheres move in a counter-clockwise motion along the trajectories with a period of $\pi$ seconds, and (b) drifts in the $x$ direction at a velocity given by Eq. (5.16). These results are consistent with the analytical solutions shown in Figure 5.4 and derived in [**73**].

where $g$ is gravity, $\omega_{\text{plate}} = {}^{h}\omega_{sh,z}$ is the rotational velocity of the plate about its $z$-axis and $\theta$ is the tilt of the plate creating a gravitational component in the $-y$ direction (see Figure 5.4(b)).

We simulate the system for 10 seconds using the same initial conditions and controls for the horizontal case except except with a tilt of 0.1 rad about the $x$-axis of the inertial

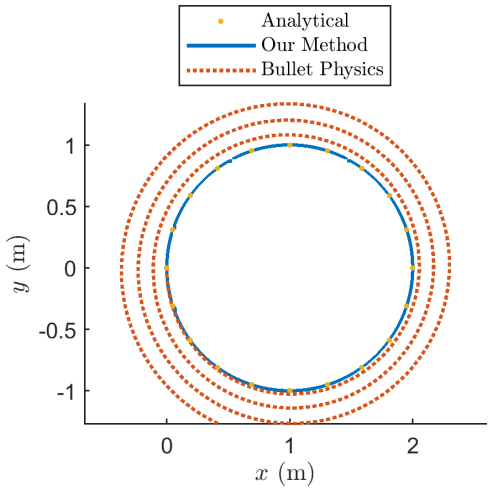



Figure 5.6. Comparison of the horizontal rolling trajectory using our method (blue) and using the Bullet physics simulator (red) with the same initial conditions and simulated for 10 s. Our method matches the circular analytical trajectory of radius 1 m centered at (1,0). The Bullet simulation has the same initial conditions but drifts from of the circular trajectory.

frame $\Phi_{sh}(0) = (0.1, 0, 0)$ rad. The sphere follows a circular motion of period $\pi$ seconds combined with a constant drift velocity of 0.35 m/s which is consistent with Eq. (5.16). A visualization of the rolling trajectory is shown in Figure 5.5(b) and a video of the simulation is included here: https://youtu.be/O72dWKWqFh0.

### 5.6.4. Open Loop 3D-Rolling Example

Our dynamics formulation applies to arbitrary smooth geometries of the object and hand, unlike previous work restricted to specific geometries (e.g., a sphere on a plane) or the approximations built into standard physics engine simulations (e.g., Bullet) of rolling contact. As one example, Figure 5.7 illustrates an ellipsoid rolling in an ellipsoidal shaped bowl. Videos of dynamic simulations with the rolling and pure-rolling friction assumptions applied can be seen in the supplemental media and at the following links: dynamic rolling

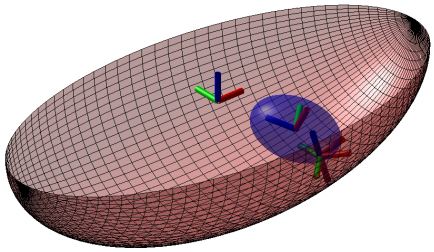


Figure 5.7. Dynamic rolling example of an ellipsoid rolling in an ellipsoidal dish. Videos of open loop dynamic simulations with the rolling and pure-rolling friction assumptions applied can be seen in the supplemental media and at the following links: dynamic rolling (rigid contact): https://youtu.be/zroDTij17JU, dynamic pure-rolling (soft contact): https://youtu.be/wV4II7uxtMk.

(rigid contact): https://youtu.be/zroDTij17JU, dynamic pure-rolling (soft contact): https://youtu.be/wV4II7uxtMk.

## 5.7. Contact Juggling Motion Planning

In this section the rolling dynamics equations are used to plan rolling motions that bring the hand and object from an initial state $\mathbf{s}_{\text{start}}$ to a goal state $\mathbf{s}_{\text{goal}}$. This builds on our previous work in [**76**] where we demonstrate an iterative planning method for kinematic rolling between smooth objects (the relative rolling velocities are directly controlled). The two primary differences in this implementation are: (1) we replace the kinematic equations of motion with the higher-dimensional dynamic-rolling equations; (2) we incorporate nonlinear constraints to enforce positive normal forces and bounded frictional wrenches.

An "admissible" trajectory is defined as a set of states and controls $\xi(t) = (\mathbf{s}(t), {}^{h}\dot{\mathcal{V}}_{sh}(t))$, from $t = 0$ to the final time $t = t_f$, that satisfies the rolling dynamics equation in Eq. (5.12) and the contact wrench limits. A "valid" trajectory is defined as an admissible trajectory that also satisfies $\mathbf{s}_{\text{error}}(t_f) < \eta$, where $\eta$ is the tolerance on the final state error and

$\mathbf{s}_{\text{error}}(t_f) = ||\mathbf{s}(t_f) - \mathbf{s}_{\text{goal}}||$, where $||\cdot||$ corresponds to a weighted norm that puts state errors in common units. (Throughout the rest of this paper, we use the Euclidean norm.) The motion-planning problem can be stated as:

**Given:** The rolling dynamics equations and contact wrench expressions from Section 5.6, the states $(\mathbf{s}_{\text{start}}, \mathbf{s}_{\text{goal}})$, and the rolling time $t_f$,
**find:** a valid rolling trajectory $\xi(t) = (\mathbf{s}(t), {}^h\dot{\mathcal{V}}_{sh}(t))$ for $t \in [0, t_f]$ that brings the system from $\mathbf{s}(0) = \mathbf{s}_{\text{start}}$ to $\mathbf{s}(t_f) = \mathbf{s}_{\text{goal}}$.

We plan rolling motions using an iterative direct collocation (iDC) nonlinear optimization method described in our recent work [**76**]. The optimization first solves for a trajectory history that is represented coarsely, using a small number of state and control segments. The solved-for controls are then simulated by a more accurate, higher-order numerical integration method than the integrator implicit in the constraints in the nonlinear optimization. If the simulated trajectory satisfies the error tolerances, the problem is solved. If not, the previous solution is used as an initial guess, the number of state and control segments is increased, and the optimization is called again. This is repeated until a valid trajectory is found, the maximum number of iDC iterations is reached, or the optimization converges to an invalid point. The details are included in Appendix 5.13.

We focus on plans for cases where the object and hand are stationary at $\mathbf{s}_{\text{start}}$. We refer to plans where $\mathbf{s}_{\text{goal}}$ is also stationary as "stationary-to-stationary" motions. We refer to plans where $\mathbf{s}_{\text{goal}}$ is moving as "stationary-to-rolling" motions, with the special case "stationary-to-freeflight" when the object and hand separate ($f_z = 0$, $a_{z,\text{rel}} > a_{z,\text{roll}}$)

at the final time $t_f$. The planning problem can also consider "full control" of all six hand accelerations ${}^h\dot{\mathcal{V}}_{sh}(t)$, or subsets such as "rotational control" of the rotational accelerations ${}^h\alpha_{sh}$, or "linear control" of the linear accelerations ${}^h\mathbf{a}_{sh}$.

### 5.7.1. Planning for a Ball on Plate Reconfiguration

Recent work by [**64**] addresses planning and control for dynamic, nonprehensile repositioning and reorientation for a ball on a horizontal plate (the $z$-axis is aligned with gravity). The ball is initially at rest on the plate, and a goal position and orientation for the ball is given. The controls are the two rotations about the $x$- and $y$-axes of the plate. The desired path is generated by time-scaling a geometric planning method from [**8**], and then tracked using a feedback controller. An example from the paper is reproduced in Figure 5.8; The stationary initial and goal configurations for the sphere on the plate are shown in Figure 5.8(a) and (b), respectively. Visualizations of the contact locations on the hand for the geometric solution are shown by the dashed black lines in Figure 5.8(c) and (d).

Our method for trajectory planning utilizes the iterative direct collocation method detailed in Appendix 5.13. To match the method above we constrain the hand to have two rotational degrees of freedom which decreases the number of controls from six to two. We then simplify the dynamics equation from Eq. (5.12) by substituting the constant values $\mathbf{r}_{sh} = {}^h\mathbf{v}_{sh} = {}^h\mathbf{a}_{sh} = \mathbf{0}$, and the constraint on the hand accelerations expressed in the world frame ${}^s\alpha_{sh,z} = 0$. The number of states is therefore decreased from 22 to 16.

We initialize the planner with a stationary trajectory where the ball stays in place and no controls are applied. The iDC algorithm was then run using the parameters given

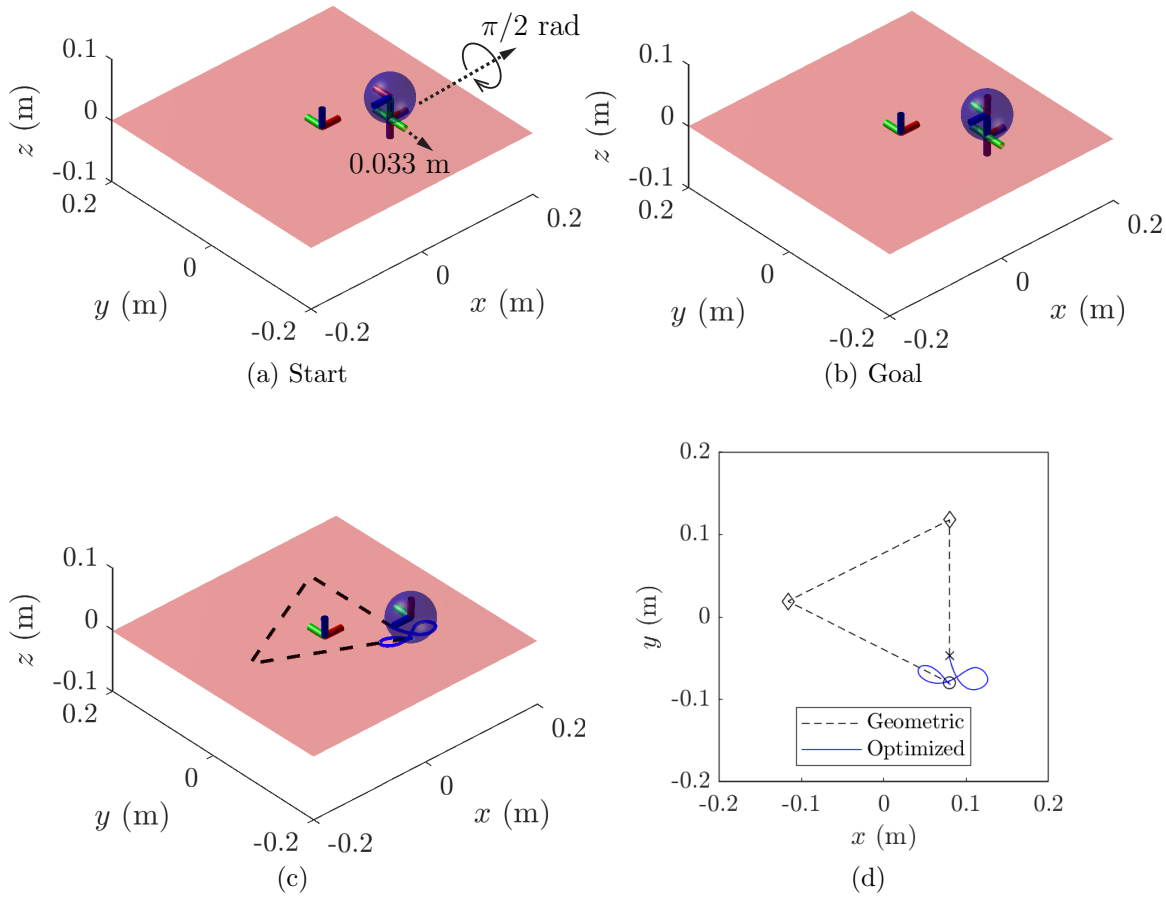



Figure 5.8. Initial and goal states for reorienting a sphere on a plate are shown in (a) and (b) respectively. The goal state is 0.033 m from the start state in the -$y$ direction, with the object rotated $\pi/2$ rad about the $x$-axis. A visualization of the sphere rolling trajectory from the geometric plan from [**64**] is shown by the black dashed lines in (c) and (d), and the optimized plan from the iterative direct collection method is shown by the solid blue lines. The start position is shown by the "×", and the goal position is shown by the "○" An animation of the optimized solution can be seen in the supplemental media.

in Table 5.2. The weight $\mathbf{Q}$ was reduced to zero after the first iteration because the inadmissible straight-line trajectory is no longer needed to bias the trajectory towards a solution.

Table 5.2. Trajectory optimization parameters for ball-on-plate example

| Description | Value |
|---|---|
| Trajectory time $t_f$ | 2 s |
| Initial # of segments $N$ | 50 ($\Delta t = 0.04$ s) |
| Goal error tolerance $\eta$ | 0.1 |
| Max iDC iterations | 4 |
| Max fmincon func evals/iter | $10^5$ |
| Control limits | $\|\|\alpha_x\|\| \leq 50, \|\|\alpha_y\|\| \leq 50$ |
| Constraint integration method | trapezoidal |
| Initial guess | stationary |
| $\mathbf{P}_1$ (terminal state weight) | blkdiag($\mathbf{I}_3, 10\mathbf{I}_2, 1000\mathbf{I}_2, 10, \mathbf{I}_3, 10\mathbf{I}_5$)1000 |
| $\mathbf{Q}$ (tracking weight) | blkdiag($\mathbf{I}_3, 10\mathbf{I}_2, 1000\mathbf{I}_2, 10, \mathbf{I}_3, 10\mathbf{I}_5$)/100 |
| $\mathbf{R}$ (control weight) | diag(0.001, 0.001) |

A valid trajectory was found after three iterations of (50, 100, 200) segments with planning times (171 s, 346 s, 946 s) for a total planning time of 24 minutes (run on an i7-4700MQ CPU @ 2.40 GHz with 16 GB of RAM). A visualization and plot of the optimized trajectory are shown in Figure 5.8 (c) and (d) respectively. A comparison of the two trajectories is shown in Table 5.3. The optimized trajectory is preferable because it has a shorter path length, and a smaller bounding box around the contact trajectory. The planning method can also be generalized to objects besides spheres on plates. The geometric method, however, guarantees completeness assuming the plate is large enough, has no final state error, and the trajectory is generated by simple algebraic expressions which makes the computation time negligible compared to the finite planning time for the iDC method. A video of the simulation is included here: https://youtu.be/WAX76SR8UOo

Table 5.3. Comparison of the geometric and optimization motion planning results for reorienting a sphere on a plate

| | Geometric [**64**] | Optimized (Sec. 5.7) |
|---|---|---|
| Path length on hand | 0.60 m | 0.24 m |
| Bounding box area on hand | 0.039 $m^2$ | 0.0037 $m^2$ |
| Final state error: $\mathbf{s}_{\text{error}}(t_f)$ | 0.0 | 0.07 |
| Handles general shapes? | sphere-plane only | yes |
| Planning time | Negligible | 24 minutes |
| Completeness | Complete | No guarantees |

## 5.8. Feedback Control for Dynamic Rolling

We apply the same feedback controller outlined in our recent work [**76**] to stabilize the rolling trajectories. The method uses a linear quadratic regulator (LQR) to stabilize the linearized dynamics along a known trajectory. LQR computes a time-varying gain matrix $\mathbf{K}_{\text{LQR}}(t)$ that optimally reduces the total cost for small perturbations about the nominal trajectory. LQR requires a cost function, and we use the one given in Eq. (5.47), where $\mathbf{s}_{\text{des}}(t)$ is set to the nominal trajectory we are tracking $\mathbf{s}_{\text{nom}}(t)$. We solve the matrix Riccati equation to find the time varying feedback control matrix $\mathbf{K}_{\text{LQR}}(t)$ (see Section 2.3 of [**3**]).

$$
\begin{aligned}
-\dot{\mathbf{P}}(t) &= \mathbf{P}(t)\widetilde{\mathbf{A}}(t) + \widetilde{\mathbf{A}}(t)^{\mathsf{T}}\mathbf{P}(t) - \\
&\qquad \mathbf{P}(t)\widetilde{\mathbf{B}}(t)\mathbf{R}_{\text{LQR}}^{-1}\widetilde{\mathbf{B}}(t)^{\mathsf{T}}\mathbf{P}(t) + \mathbf{Q}_{\text{LQR}}, \\
\mathbf{P}(t_f) &= \mathbf{P}_{1,\text{LQR}} \\
\mathbf{K}_{\text{LQR}}(t) &= \mathbf{R}_{\text{LQR}}^{-1}\widetilde{\mathbf{B}}(t)^{\mathsf{T}}\mathbf{P}(t).
\end{aligned}
\tag{5.17}
$$

The time-varying matrices $\widetilde{\mathbf{A}}(t)$ and $\widetilde{\mathbf{B}}(t)$ come from linearizing about the nominal trajectory $\mathbf{s}_{\text{nom}}(t)$ (see [**76**]), and $\mathbf{P}_{1,\text{LQR}}$, $\mathbf{Q}_{\text{LQR}}$, and $\mathbf{R}_{\text{LQR}}$ are the controller gain matrices

weighting the goal-state error, desired trajectory deviation, and control cost respectively. The time-varying gain matrix $\mathbf{K}_{\mathrm{LQR}}(t)$ from Eq. (5.17) can be expressed as the function:

$$\mathbf{K}_{\mathrm{LQR}}(t) = \mathcal{K}(\mathbf{s}_{\mathrm{nom}}(t), \mathbf{P}_{1,\mathrm{LQR}}, \mathbf{Q}_{\mathrm{LQR}}, \mathbf{R}_{\mathrm{LQR}}). \tag{5.18}$$

The matrix $\mathbf{K}_{\mathrm{LQR}}(t)$ is then used in the feedback control law

$${}^{h}\dot{\mathcal{V}}_{sh,\mathrm{fbk}}(\mathbf{s}, t) = {}^{h}\dot{\mathcal{V}}_{sh,\mathrm{nom}}(t) - \mathbf{K}_{\mathrm{LQR}}(t)(\mathbf{s}(t) - \mathbf{s}_{\mathrm{nom}}(t)) \tag{5.19}$$

to stabilize the nominal trajectory. The linearized dynamics are controllable about almost all trajectories (i.e., generic trajectories). Specially chosen trajectories can be uncontrollable such as those with symmetry properties as demonstrated for kinematic rolling in [**76**].

### 5.8.1. Feedback Control Example

Consider the optimized open loop rolling trajectory shown in Fig. 5.8 (c) and (d). The feedback controller in Eq. (5.19) can be used to recover from: (1) initial state error; (2) perturbations along the trajectory; and (3) error due to planning rolling motions using a coarse approximation of the dynamics. In this example we demonstrate how the feedback controller can be used to decrease error from the the third source, the coarse approximation of the dynamics.

As discussed in Section 5.7, the optimization first solves for a trajectory history that is represented coarsely, using a small number of state and control segments. The solved-for controls are then simulated by a more accurate, higher-order numerical integration method than the integrator implicit in the constraints in the nonlinear optimization. If the final state error in the fine trajectory is too large (i.e., greater than the goal error

tolerance $\eta$), we perform another iteration of the motion plan with additional state and control segments. The goal error tolerance for this motion plan was given by $\eta = 0.1$ from Table 5.2. To further decrease this error, we stabilize the planned trajectory with a feedback controller.

We generate the time-varying gain matrix $\mathbf{K}_{\text{LQR}}(t)$ by plugging the planned trajectory and gain matrices into Eq. (5.18), and then stabilize the trajectory using the feedback control law from Eq. (5.19). The open-loop simulation using the higher-order numerical integration method (MATLAB's ode45) gives us the state trajectory $\mathbf{s}_{\text{fine}}(t)$. The closed-loop simulation gives us the state trajectory $\mathbf{s}_{\text{fbk}}(t)$. Plots of the distance errors between the output trajectories and the nominal trajectory $\mathbf{s}_{\text{nom}}(t)$ are shown in Figure 5.9. The final state error ($\mathbf{s}_{\text{error}}(t_f)$) for the open-loop solution is 0.07, while the final state error for the closed-loop solution is 0.002.

## 5.9. Experiments

This section outlines a series of experiments in planning and feedback control for rolling manipulation tasks. The experiments are in two dimensions, but the model is a full three dimensional model as shown in Figure 5.10.

### 5.9.1. Experimental Setup

The experimental setup consists of a 3-DOF robot arm that moves in a plane parallel to the surface of an inclined air table. A diagram of the experimental setup is shown in Figure 5.11. Each link is actuated by a brushed DC motor with harmonic drive gearing and current controlled using Junus motor amplifiers. The 1000 Hz motion controller runs

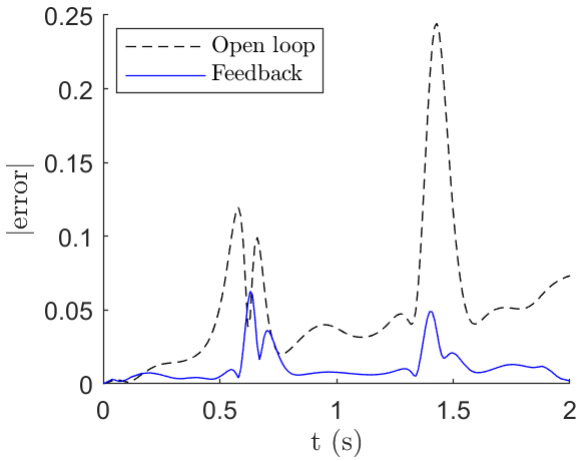
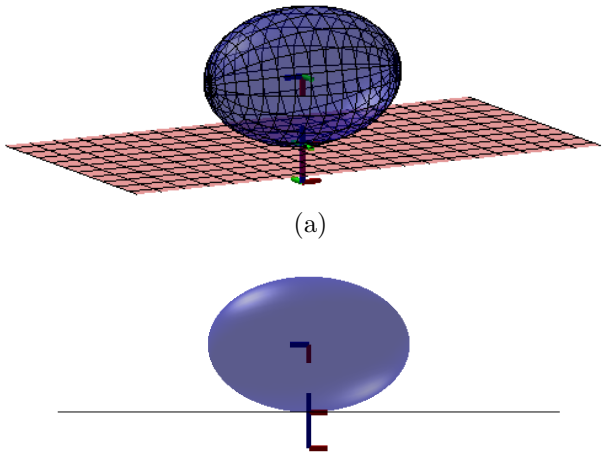



Figure 5.9. Comparison of trajectory error over time for open-loop and closed-loop trajectories. The open loop trajectory uses the coarse set of controls found by the trajectory optimization method which leads to error during the simulation with the finer integration method. The closed-loop trajectory stabilizes the state to the planned trajectory. The final state error ($\mathbf{s}_{\text{error}}(t_f)$) for the open-loop solution is 0.07, while the final state error for the closed-loop solution is 0.002. The two spikes occur at the corners of the trajectory where the ball changes direction because the system is more sensitive to integration errors at these points. The feedback method is weighted to stabilize the trajectory to the goal state which is why some error is not eliminated in the middle of the trajectory.

on a PC104 embedded computer running the QNX real-time operating system. Vision feedback is given by a 250 Hz IR Optitrack camera and reflective markers attached to the object. The manipulator is a flat plate of width 0.375 m, and the object is an ellipse of mass 0.0553 kg, and major and minor axes given by 0.0754 m and 0.0504 m, respectively. Experiments are conducted at 40% full gravity by inclining the table at 24 degrees with respect to horizontal.



(a)

(b)

Figure 5.10. The full three-dimensional model used for planning rolling motions is shown in (a) and the 2D projection is shown in (b).

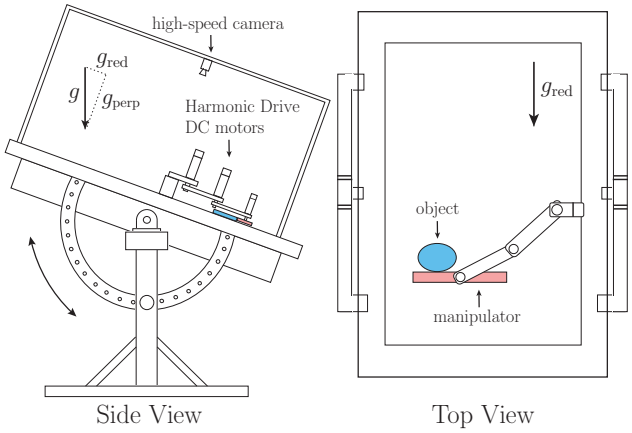



Figure 5.11. Diagram of the experimental setup

### 5.9.2. 2D Rolling Model

We model the system in full 3D, but restrict object and hand motions to the $xz$-plane so all the motion is planar. All rotations occur about the $y$-axis, and the full system state $\mathbf{s} = (\Phi_{sh}, \mathbf{r}_{sh}, \mathbf{q}, {}^{h}\mathcal{V}_{sh}, \dot{\mathbf{q}}) \in \mathbb{R}^{22}$ simplifies to $\mathbf{s}_{\text{2D}} = (\beta_{sh}, x_{sh}, z_{sh}, u_o, u_h, \omega_{sh,y}, v_{sh,x}, v_{sh,y}, \dot{u}_o, \dot{u}_h) \in$

$\mathbb{R}^{10}$ and all other states are zero. The controls are also limited to rotational accelerations about the $y$-axis, and linear accelerations in the $xz$-plane.

Knowledge of the system states is necessary to run feedback controllers based on our rolling dynamics equations. The six hand states $(\beta_{sh}, x_{sh}, z_{sh}, \omega_{sh,y}, v_{sh,x}, v_{sh,y})$ can be determined by the robot encoders, and the object states are estimated using the IR camera that tracks reflective markers on the object. We use the object and hand states to estimate the contact coordinates and velocities on the object and hand $(u_o, u_h, \dot{u}_o, \dot{u}_h)$. We analytically solve for the closest points on the object and plate using equations defining the ellipse and a line. The method would need to be generalized for other shapes, but works well for our implementation.

### 5.9.3. Rolling Damping Controller

The first experiment demonstrates a damping controller that stabilizes an ellipse in rolling contact with a plate in a neighborhood of the stable equilibrium $\mathbf{s}_{2\mathrm{D}} = (0, 0, 0, \pi/2, 0, 0, 0, 0, 0, 0)$ shown in Figure 5.10(b). Perturbations of the object cause it to wobble back and forth until dissipation brings it back to rest. We generate the time-varying gain matrix $\mathbf{K}_{\mathrm{LQR}}(t)$ by plugging the constant nominal trajectory $\mathbf{s}_{2\mathrm{D}}(t) = (0, 0, 0, \pi/2, 0, 0, 0, 0, 0, 0)$ and gain matrices into Eq. (5.18), and then stabilize the object using the feedback control law from Eq. (5.19). We then demonstrate the performance by perturbing the ellipse with and without the controller enabled. The object has large rotational motions with the controller off, but barely rotates with the controller enabled. A video of the experiment is shown in the supplemental media and here: https://youtu.be/HXDRVQUnDw8

### 5.9.4. Flip Up to Balance

The second experiment is a rolling version of the classic inverted pendulum swing-up to balance problem. We use the rolling dynamics to plan a flip-up motion, and derive a stabilizing feedback controller about the trajectory. The workflow is as follows: (1) given the initial state and goal state (balanced on end point), plan rotations of the manipulator that flip the object up to balance using iterative direct collocation outlined in Appendix 5.13, (2) stabilize the trajectory using a linearized LQR controller outlined in Section 5.8, (3) stabilize the final state using the same method, (4) test experimentally.

All of the initial conditions are zero except that $u_o = \pi/2$ and $u_h = -c_{\text{ellipse}}/4$ where $c_{\text{ellipse}}$ is the circumference of the ellipse, and the goal state is all zero except $u_o = \pi$. The goal state is a singularity of the surface parameterization, so we switch to a different coordinate chart to derive the balancing controller.

The contact location and velocity on the object for the planned, open-loop and closed-loop trajectories is shown in Figure 5.12. The open-loop execution of the planned trajectories consistently overshot the goal and the object rolled off the edge of the manipulator. The closed-loop execution was able to stabilize the trajectory and successfully balance the object in 12/12 trials. Snapshots from a successful closed-loop trial are shown in Figure 5.13, and a video of the 12 closed-loop trials can be seen here: https://youtu.be/9NSYPrjQbXk

### 5.9.5. Throwing and Catch

We now demonstrate a motion planner for throwing and catching an object in rolling contact with a manipulator. The planner is a shooting method trajectory optimizer that

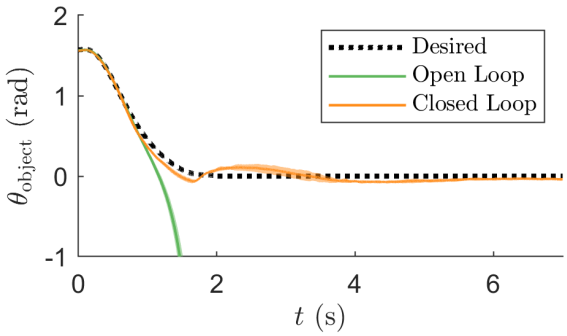



Figure 5.12. A plot showing the object angle in the world frame during the flip-up motion. It includes open and closed-loop results, with +/- two standard deviations of the 12 trials shown by the shaded region surrounding the lines. All closed-loop trials successfully flipped up the object to the balance position and kept it balanced until shutting off at seven seconds. Snapshots from one trial are shown in Figure 5.13.

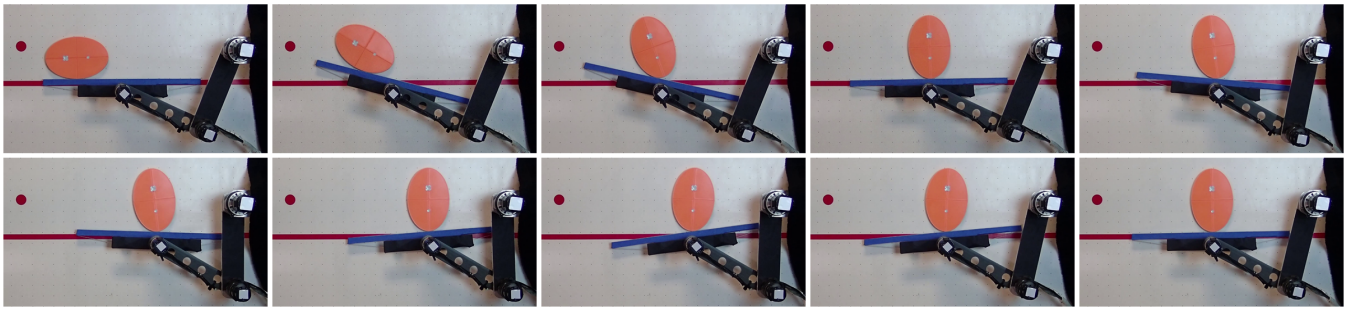


Figure 5.13. Demonstration of the closed-loop flip up to balance with LQR stabilization about the trajectory with snapshots taken every 0.5 seconds.

chooses a low-dimensional representation of hand accelerations at each iteration and then simulates the object using the rolling dynamics. The method finds a set of accelerations that satisfy release constraints that define a successful throw. We used this planner rather than the general iDC method outlined in Section 5.7 because the low-dimensional representation resulted in smoother throwing motions and the optimizer converged in

seconds rather than minutes/hours. The steps to generate a throwing trajectory are as follows:

(1) Give the initial equilibrium state with the object resting on the hand
(2) Give a desired $x$ and $z$ location for the impact with a stationary, flat hand after the throw
(3) A nonlinear optimization finds a time $t_{\text{throw}}$, and trapezoidal acceleration profile for $a_{sh,x}$ and $\alpha_{sh,y}$ that when integrated bring the object to a release state at $t_{\text{throw}}$ that causes it to rotate in the air and impact the hand at the desired location
(4) At the release time the hand accelerates away from the object and moves to the impact location
(5) The trajectory is then tested on the experimental setup

We plan a throwing motion for the object initially at rest on the manipulator with contact coordinates $u_o = \pi/2$ and $u_h = -0.1$ m. The goal impact configuration has the object rotated by $\pi$ radians at the same $x$ and $z$ location, and the trajectory optimizer successfully converged to a solution. We tested the planned throwing motion experimentally, and the object over rotates but still settles in roughly the desired location. We set the feedback control gains to zero for the stabilizing controller because the short throw gave little time to recover from error, and the impact location and post-impact stabilization can be used to account for error in the throw. The start and goal positions are the same besides a $\pi$ rad offset, and ten consecutive throws were completed successfully without manually repositioning the object. Snapshots of the experiment are shown in Figure 5.14, and a video of the ten throws can be found here: https://youtu.be/9xkVWfB2Q8c

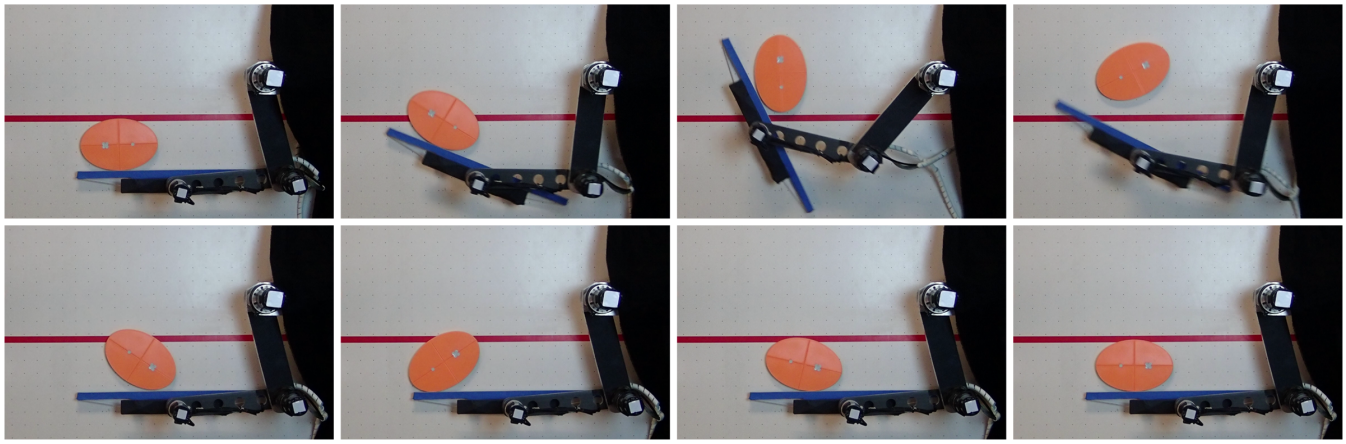


Figure 5.14. Snapshots from an experimental rolling throw that flips the object 180 degrees and catches it in the same position.

In the previous example we chose a catch location with a flat, stationary hand and horizontal object, but catch locations can also be chosen that meet specific properties. Brescianini et al. demonstrated quadcopters throwing, catching, and balancing a pole [**11**]. Assuming a stationary catch with high-friction, inelastic impact, they derived an expression for object states that would map to a post impact velocity with just enough kinetic energy to bring the pole up to balance with zero final velocity.

We demonstrate this on our system by planning a throw, choosing a catch location that satisfies the impact to balance constraint, and balancing the object after the catch. We rederived Eq. (42) from [**11**] using the moment of inertia of the ellipsoid and the contact location between the rotating ellipse and a flat hand. Because of the error between the planned and actual free-flight trajectories, we chose a catch location using the experimental data from the previous experiment. After the impact, the system waited until the object was near the balance point and initiated an LQR-based balancing controller. Snapshots from a successful throw, catch, and balance are shown in Figure 5.15, and a

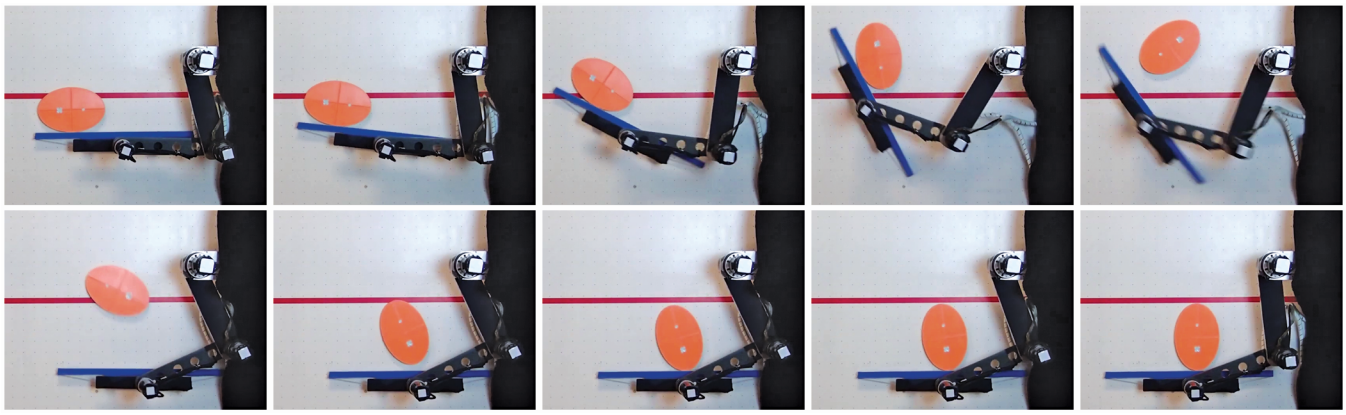


Figure 5.15. Snapshots from an experimental rolling throw that flips the object, moves to a catch position where an inelastic, high-friction impact would result in a post-impact velocity that brings the object upright (see [**11**]), and then balances it about the unstable equilibrium.

video can be found here: https://youtu.be/BMkrikiaLGU. Future versions could use feedback during the throwing and/or catching phases to improve robustness to control and modeling errors.

## 5.10. Conclusions

This paper demonstrates modeling, motion planning, and feedback control for robotic contact juggling. We first derived the rolling dynamics equations and friction constraints, then validated the results in simulation against known analytical solutions. We then used direct collocation methods and the rolling dynamics equations to plan robotic contact juggling motions, and demonstrated the use of an LQR feedback controller to stabilize planned trajectories. The three-dimensional rolling model was then used to plan and execute tasks with feedback control for a 2D ellipse in contact with a flat hand of a planar 3R robot.

### 5.10.1. Future Work

There are many interesting areas for future work that extend this paper.

**5.10.1.1. Surface Parameterization.** This paper requires orthogonal parameterizations of surfaces, and while any smooth surface can be locally represented as such, the ability to use non-orthogonal parameterizations would improve the utility of this method. We have used non-orthogonal parameterizations for simulating the second-order kinematics by defining local orthogonal frames at each point on the surface, but the dynamics derivation in Section 5.6 would need to be modified to relax the orthogonality assumption.

Moving beyond explicit parameterizations would also improve the scope of this work such as using smooth objects represented by point clouds. There is some relevant work in the computer graphics field that uses conformal (angle-preserving) mappings to map smooth objects to a sphere, and it could be interesting to explore how that could be applied to rolling manipulation.

Future research could process an object defined by a point-cloud, automatically generate an atlas of orthogonal coordinate charts that cover the surface, and plan motions with stabilizing feedback controllers through multiple coordinate charts.

**5.10.1.2. Feedback Control.** We showed one example of feedback control but there are many areas for extending feedback control for rolling objects. Our method utilizes a linearized LQR controller to stabilize planned trajectories and balance states which results in a simple feedback law given by Eq. (5.19) that can easily run at high speeds (1000 Hz in our implementation). This method requires knowledge of the contact coordinates which are difficult to estimate for general, 3D rolling motions. Hardware such as manipulators equipped with contact location sensing could help address this, especially if combined

with an observability model such as the one developed in [**29**]. Other feedback methods such as energy-based feedback controllers could be used to stabilize trajectories as well, and may avoid the need for estimating specific contact states [**14**].

**5.10.1.3. Hybrid Rolling, Sliding, and Free Flight Dynamics.** This work derives dynamics for rolling and pure-rolling but does not consider roll-slide dynamics where there is relative linear velocity at the contact. This would allow more general modeling where the contact mode is determined based on the state and contact constraints, and the appropriate dynamics equations are applied. This is then a hybrid dynamics problem, where the state evolves within a single contact mode until a mode transition occurs. Our past work [**74**] outlines a framework for hybrid manipulation planning and there are many interesting applications of dynamic manipulation that combine contact modes such as rolling, pure-rolling, rolling + sliding, and free flight.

## 5.11. Appendix A: Background

### 5.11.1. Operator Definitions

The skew-symmetric matrix form of a vector $\omega \in \mathbb{R}^3$ is given by

$$[\omega] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{5.20}$$

The operator $[\mathrm{Ad}_{\mathbf{T}}]$ is the adjoint map associated with $\mathbf{T} = (\mathbf{R}, \mathbf{r})$, where

$$[\mathrm{Ad}_{\mathbf{T}}] = \begin{bmatrix} \mathbf{R} & 0 \\ [\mathbf{r}]\mathbf{R} & \mathbf{R} \end{bmatrix}. \tag{5.21}$$

The operator $[\mathrm{ad}_{\mathcal{V}}]$ is the matrix form of a Lie bracket. The Lie bracket of two twists $\mathcal{V}_1$ and $\mathcal{V}_2$ can be represented as $[\mathrm{ad}_{\mathcal{V}_1}]\mathcal{V}_2$, where

$$[\mathrm{ad}_{\mathcal{V}}] = \begin{bmatrix} [\omega] & 0 \\ [\mathbf{v}] & [\omega] \end{bmatrix}. \tag{5.22}$$

### 5.11.2. Derivatives of expressions in multiple frames

The forumula for the derivative in frame $\{p_h\}$ of an expression represented in frame $\{o\}$ is:

$$^{p_h}d/dt(^o\mathbf{v}_{so}) = {}^o\mathbf{a}_{so} + {}^o\omega_{p_h o} \times {}^o\mathbf{v}_{so} \tag{5.23}$$

## 5.12. Appendix B: Kinematics Expressions

### 5.12.1. Local Geometry of Smooth Bodies

Below are some standard expressions for the geometry of a surface that are used to define the first- and second-order kinematics in the following sections. References and derivations of these expressions can be found in [**61**].

The surface of each rigid body is represented by an orthogonal parameterization: $\mathbb{F}_i : \mathbf{u}_i \to \mathbb{R}^3 : (u_i, v_i) \mapsto (x_i, y_i, z_i)$ for $i \in [o, h]$, where the coordinates $(x_i, y_i, z_i)$ are expressed in the $\{i\}$ frame. It is assumed that $\mathbb{F}_i$ is continuous up to the third derivative (class $C^3$), so that the local contact geometry (contact frames associated with the first derivative of $\mathbb{F}_i$, curvature associated with the second derivative, and derivative of the curvature associated with the third derivative) are uniquely defined.

The natural bases at a point on a body are given as $\mathbf{x}_{c_i} = \partial \mathbb{F}_i / \partial u_i$ and $\mathbf{y}_{c_i} = \partial \mathbb{F}_i / \partial v_i$. We also assume that coordinate charts are orthogonal ($\mathbf{x}_{c_i} \cdot \mathbf{y}_{c_i} = 0$), and note that $\mathbf{x}_{c_i}$ and $\mathbf{y}_{c_i}$ are not necessarily unit vectors. The unit normal is given as $\mathbf{n}_{c_i} = (\mathbf{x}_{c_i} \times \mathbf{y}_{c_i}) / ||\mathbf{x}_{c_i} \times \mathbf{y}_{c_i}||$.

The normalized Gauss frame at a point $\mathbf{u}_i$ on body $i$ is defined as the coordinate frame $\{c_i\}$ with origin at $\mathbb{F}_i(\mathbf{u}_i)$ and coordinate axes given by

$$\mathbf{R}_{ic_i} = \left[ \frac{\mathbf{x}_{c_i}}{||\mathbf{x}_{c_i}||}, \frac{\mathbf{y}_{c_i}}{||\mathbf{y}_{c_i}||}, \mathbf{n}_{c_i} \right], \tag{5.24}$$

where $\mathbf{R}_{ic_i}$ expresses the Gauss frame in the object or hand frame $\{i\}$. The metric tensor $\mathbf{G}_i$ is a $2 \times 2$ positive-definite matrix defined as

$$\mathbf{G}_i = \begin{bmatrix} \mathbf{x}_{c_i} \cdot \mathbf{x}_{c_i} & \mathbf{x}_{c_i} \cdot \mathbf{y}_{c_i} \\ \mathbf{y}_{c_i} \cdot \mathbf{x}_{c_i} & \mathbf{y}_{c_i} \cdot \mathbf{y}_{c_i} \end{bmatrix}. \tag{5.25}$$

The coefficients $g_{jk,i}$ reference the indices of matrix $\mathbf{G}_i$, and $\mathbf{G}_i$ is diagonal ($g_{12,i} = g_{21,i} = 0$) when the coordinate chart $\mathbb{F}_i$ is orthogonal. The $2 \times 2$ matrix $\mathbf{L}_i$ is the second fundamental form given by the expression

$$\mathbf{L}_i = \begin{bmatrix} \frac{\partial^2 \mathbb{F}_i}{\partial u_i^2} \cdot \mathbf{n}_{c_i} & \frac{\partial^2 \mathbb{F}_i}{\partial u_i \partial v_i} \cdot \mathbf{n}_{c_i} \\ \frac{\partial^2 \mathbb{F}_i}{\partial v_i \partial u_i} \cdot \mathbf{n}_{c_i} & \frac{\partial^2 \mathbb{F}_i}{\partial v_i^2} \cdot \mathbf{n}_{c_i} \end{bmatrix}. \tag{5.26}$$

$\mathbf{H}_i$ combines the metric tensor $\mathbf{G}_i$ with the second fundamental form $\mathbf{L}_i$ and is given by,

$$\mathbf{H}_i = (\sqrt{\mathbf{G}_i})^{-1} \mathbf{L}_i (\sqrt{\mathbf{G}_i})^{-1}. \tag{5.27}$$

The $1 \times 2$ array $\Gamma_i$ is given by the expression

$$\Gamma_i = \left[\Gamma^2_{11,i}\ \Gamma^2_{12,i}\right], \tag{5.28}$$

where $\Gamma^l_{jk,i}$ are christoffel symbols of the second kind given by:

$$\Gamma^l_{jk,i} = \sum_{n=1}^{2} \left(\frac{\partial(\chi_i)_j}{\partial(\mathbf{u}_i)_k}\right)^{\mathsf{T}} (\chi_i)_n g_i^{nl} \tag{5.29}$$

where $(\chi_i)_j$ is the $j^{\text{th}}$ vector in the list $\chi_i = (\mathbf{x}_{c_i}, \mathbf{y}_{c_i})$, $(\mathbf{u}_i)_k$ is the $k^{\text{th}}$ variable in the list $\mathbf{u}_i = (u_i, v_i)$, and $g_i^{nl}$ are the entries $(n, l)$ of the metric tensor inverse $(\mathbf{G}_i)^{-1}$. This gives $\Gamma^2_{11,i}$ and $\Gamma^2_{12,i}$ as

$$\begin{aligned} \Gamma^2_{11,i} &= \left(\frac{\partial \mathbf{x}_{c_i}}{\partial u_i}\right)^{\mathsf{T}} \mathbf{x}_{c_i} g_i^{12} + \left(\frac{\partial \mathbf{x}_{c_i}}{\partial u_i}\right)^{\mathsf{T}} \mathbf{y}_{c_i} g_i^{22}, \\ \Gamma^2_{12,i} &= \left(\frac{\partial \mathbf{x}_{c_i}}{\partial v_i}\right)^{\mathsf{T}} \mathbf{x}_{c_i} g_i^{12} + \left(\frac{\partial \mathbf{x}_{c_i}}{\partial v_i}\right)^{\mathsf{T}} \mathbf{y}_{c_i} g_i^{22}. \end{aligned} \tag{5.30}$$

### 5.12.2. First-Order Kinematics

This form of the first-order kinematics was initially derived in [**61**]. We reproduce it in matrix form with rolling constraints ($v_x = v_y = v_z = 0$) in Eq. (5.1) as

$$\dot{\mathbf{q}} = \mathbf{K}_1(\mathbf{q})\omega_{\text{rel}},$$

with $\omega_{\text{rel}} = {}^{c_h}\omega_{p_h p_o} = [\omega_x \ \omega_y \ \omega_z]^{\mathsf{T}}$ and $\mathbf{K}_1(\mathbf{q})$ defined as:

$$\begin{aligned}
\mathbf{K}_1(\mathbf{q}) &= \begin{bmatrix} \mathbf{K}_{1o}(\mathbf{q}) & \mathbf{0}_{2\times 1} \\ \mathbf{K}_{1h}(\mathbf{q}) & \mathbf{0}_{2\times 1} \\ \sigma_o \Gamma_o \mathbf{K}_{1o}(\mathbf{q}) + \sigma_h \Gamma_h \mathbf{K}_{1h}(\mathbf{q}) & -1 \end{bmatrix}, \\
\mathbf{K}_{1o}(\mathbf{q}) &= (\sqrt{\mathbf{G}_o})^{-1} \mathbf{R}_\psi (\widetilde{\mathbf{H}}_o + \mathbf{H}_h)^{-1} \mathbf{E}_1, \\
\mathbf{K}_{1h}(\mathbf{q}) &= (\sqrt{\mathbf{G}_h})^{-1} (\widetilde{\mathbf{H}}_o + \mathbf{H}_h)^{-1} \mathbf{E}_1,
\end{aligned} \tag{5.31}$$

where $\mathbf{G}_i$ is the metric tensor of body $i \in [o, h]$ from Eq. (5.25), the $2 \times 2$ rotation matrix $\mathbf{R}_\psi$ and $\mathbf{E}_1$ are defined as

$$\mathbf{R}_\psi = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ -\sin(\psi) & -\cos(\psi) \end{bmatrix}, \ \mathbf{E}_1 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix},$$

$\mathbf{H}_i$ is a $2 \times 2$ matrix that gives the curvature of the surface from Eq. (5.27), $\widetilde{\mathbf{H}}_o$ is defined as $\widetilde{\mathbf{H}}_o = \mathbf{R}_\psi \mathbf{H}_o \mathbf{R}_\psi$, the scalar $\sigma_i$ is defined as $\sigma_i = \sqrt{g_{22,i}/g_{11,i}}$ where $g_{11,i}$ and $g_{22,i}$ are the diagonal entries of the metric tensor $\mathbf{G}_i$, and $\Gamma_i$ is a $1 \times 2$ matrix of the Christoffel symbols of the second kind from Eq. (5.28).

We use a five-dimensional representation $\dot{\mathbf{q}}$ for the relative rolling velocity at the contact, but these are subject to two constraints for rolling and pure rolling. The two rolling constraints are given by:

$$\mathbf{K}_{1o}(\mathbf{q})^{-1} \dot{\mathbf{u}}_o = \mathbf{K}_{1h}(\mathbf{q}))^{-1} \dot{\mathbf{u}}_h. \tag{5.32}$$

Pure rolling is subject to the constraint above as well as the no-spin constraint $\omega_z = 0$. From Eq. (5.31) this gives us:

$$\dot{\psi} = [\sigma_o \Gamma_o \mathbf{K}_{1o}(\mathbf{q}) + \sigma_h \Gamma_h \mathbf{K}_{1h}(\mathbf{q})] \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix}. \tag{5.33}$$

### 5.12.3. Second-Order Kinematics

Second-order contact equations were derived by Sarkar et al. in [**61**] and published again in later works [**62**,**63**]. Errors in the published equations for second-order contact kinematics in [**61**–**63**] were corrected in our recent work [**75**]. We first define some additional higher-order local contact geometry expressions used in the second-order kinematics first defined in [**61**], and then provide the corrected second-order-kinematics equations in a newly derived matrix form.

The first order kinematics includes expressions for $\Gamma_i$ $(1 \times 2)$ and $\mathbf{L}_i$ $(2 \times 2)$. We now give four additional expressions for $\bar{\Gamma}_i$ $(2 \times 3)$, $\bar{L}_i$ $(1 \times 3)$, $\bar{\bar{\Gamma}}_i$ $(1 \times 3)$, and $\bar{\bar{L}}_i$ $(2 \times 3)$:

$$\bar{\Gamma}_i = \begin{bmatrix} \Gamma^1_{11,i} & 2\Gamma^1_{12,i} & \Gamma^1_{22,i} \\ \Gamma^2_{11,i} & 2\Gamma^2_{12,i} & \Gamma^2_{22,i} \end{bmatrix}, \tag{5.34}$$

$$\bar{L}_i = \begin{bmatrix} \mathbf{L}_{11,i} & 2\mathbf{L}_{12,i} & \mathbf{L}_{22,i} \end{bmatrix}, \tag{5.35}$$

where $\Gamma^l_{jk,i}$ is the Christoffel symbol of the second kind defined in Eq. (5.29), and $\mathbf{L}_{jk,i}$ refers to the entry $(j,k)$ of matrix $\mathbf{L}_i$ in Eq. (5.26). The final two expressions for $\bar{\bar{\Gamma}}_i$ $(1 \times 3)$

and $\bar{\bar{L}}_i$ $(2 \times 3)$ are given as:

$$\bar{\bar{\Gamma}}_i =$$

$$\begin{bmatrix} (\Gamma^2_{21,i} - \Gamma^1_{11,i})\Gamma^2_{11,i} + \frac{\partial \Gamma^2_{11,i}}{\partial u_i} \\ (\Gamma^2_{21,i} - \Gamma^1_{11,i})\Gamma^2_{12,i} + (\Gamma^2_{22,i} - \Gamma^1_{12,i})\Gamma^2_{11,i} + \frac{\partial \Gamma^2_{12,i}}{\partial u_i} + \frac{\partial \Gamma^2_{11,i}}{\partial v_i} \\ (\Gamma^2_{22,i} - \Gamma^1_{12,i})\Gamma^2_{12,i} + \frac{\partial \Gamma^2_{12,i}}{\partial v_i} \end{bmatrix}^{\mathsf{T}}, \tag{5.36}$$

$$\bar{\bar{L}}_i =$$

$$\begin{bmatrix} \begin{pmatrix} \Gamma^1_{11,i}\mathbf{L}_{11,i} - \frac{\partial \mathbf{L}_{11,i}}{\partial u_i} \\ \Gamma^1_{11,i}\mathbf{L}_{12,i} + \Gamma^1_{12,i}\mathbf{L}_{11,i} - \frac{\partial \mathbf{L}_{12,i}}{\partial u_i} - \frac{\partial \mathbf{L}_{11,i}}{\partial v_i} \\ \Gamma^1_{12,i}\mathbf{L}_{12,i} - \frac{\partial \mathbf{L}_{12,i}}{\partial v_i} \end{pmatrix}^{\mathsf{T}} \\ \begin{pmatrix} \Gamma^2_{21,i}\mathbf{L}_{21,i} - \frac{\partial \mathbf{L}_{21,i}}{\partial u_i} \\ \Gamma^2_{21,i}\mathbf{L}_{22,i} + \Gamma^2_{22,i}\mathbf{L}_{21,i} - \frac{\partial \mathbf{L}_{22,i}}{\partial u_i} - \frac{\partial \mathbf{L}_{21,i}}{\partial v_i} \\ \Gamma^2_{22,i}\mathbf{L}_{22,i} - \frac{\partial \mathbf{L}_{22,i}}{\partial v_i} \end{pmatrix}^{\mathsf{T}} \end{bmatrix}. \tag{5.37}$$

The second-order kinematics expression can therefore be expressed as:

$$
\begin{aligned}
&\begin{bmatrix} \ddot{\mathbf{u}}_o \\ \ddot{\mathbf{u}}_h \end{bmatrix} = \begin{bmatrix} \mathbf{R}_\psi \sqrt{\mathbf{G}_o} & -\sqrt{\mathbf{G}_h} \\ \mathbf{R}_\psi \mathbf{E}_1 \mathbf{H}_o \sqrt{\mathbf{G}_o} & -\mathbf{E}_1 \mathbf{H}_h \sqrt{\mathbf{G}_h} \end{bmatrix}^{-1} \\
&\left\{ \begin{bmatrix} -\mathbf{R}_\psi \sqrt{\mathbf{G}_o} \bar{\Gamma}_o \\ \mathbf{R}_\psi \mathbf{E}_1 (\sqrt{\mathbf{G}_o})^{-1} \bar{\bar{L}}_o \end{bmatrix} \mathbf{w}_o + \begin{bmatrix} \sqrt{\mathbf{G}_h} \bar{\Gamma}_h \\ -\mathbf{E}_1 (\sqrt{\mathbf{G}_h})^{-1} \bar{\bar{L}}_h \end{bmatrix} \mathbf{w}_h \right. \\
&+ \begin{bmatrix} -2\omega_z \mathbf{E}_1 \mathbf{R}_\psi \sqrt{\mathbf{G}_o} & \mathbf{0}_{2\times 2} \\ -\omega_z \mathbf{R}_\psi \mathbf{H}_o \sqrt{\mathbf{G}_o} & -\dot{\psi} \mathbf{H}_h \sqrt{\mathbf{G}_h} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_o \\ \dot{\mathbf{u}}_h \end{bmatrix} \\
&\left. - \begin{bmatrix} \mathbf{0}_{2\times 1} \\ \sigma_o \Gamma_o \dot{\mathbf{u}}_o \begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{2\times 1} \\ \alpha_x \\ \alpha_y \end{bmatrix} - \begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{0}_{2\times 1} \end{bmatrix} \right\}, \\
&\ddot{\psi} = -\begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix}^\mathsf{T} \mathbf{R}_\psi \mathbf{E}_1 (\sqrt{\mathbf{G}_o})^{-1} \mathbf{L}_o \dot{\mathbf{u}}_o - \alpha_z \\
&+ \sigma_o (\Gamma_o \ddot{\mathbf{u}}_o + \bar{\bar{\Gamma}}_o \mathbf{w}_o) + \sigma_h (\Gamma_h \ddot{\mathbf{u}}_h + \bar{\bar{\Gamma}}_h \mathbf{w}_h),
\end{aligned}
\tag{5.38}
$$

where $\omega_{\text{rel}} = {}^{c_h}\omega_{p_h p_o} = [\omega_x\ \omega_y\ \omega_z]^\mathsf{T}$, $\dot{\mathcal{V}}_{\text{rel}} = {}^{c_h}\dot{\mathcal{V}}_{p_h p_o} = [\alpha_x\ \alpha_y\ \alpha_z\ \mathbf{a}_x\ \mathbf{a}_y\ \mathbf{a}_z]^\mathsf{T}$, and $\mathbf{w}_i$ comprises the velocity product terms $[\dot{u}_i^2\ \dot{u}_i \dot{v}_i\ \dot{v}_i^2]^\mathsf{T}$.

The second order kinematics expression in Eq. (5.38) can be expressed in the form of Eq. (5.3)

$$
\ddot{\mathbf{q}} = \mathbf{K}_2(\mathbf{q}, \omega_{\text{rel}}) + \mathbf{K}_3(\mathbf{q}) \dot{\mathcal{V}}_{\text{rel}},
$$

which separates the velocity and acceleration components. The velocity terms are given by the matrix $\mathbf{K}_2(\mathbf{q}, \omega_{\text{rel}})$ defined as:

$$
\begin{aligned}
\mathbf{K}_2(\mathbf{q}, \omega_{\text{rel}}) &= \begin{bmatrix} \mathbf{K}_{2a} \\ \mathbf{K}_{2b} \end{bmatrix}, \\
\mathbf{K}_{2a} &= \begin{bmatrix} \mathbf{R}_\psi \sqrt{\mathbf{G}_o} & -\sqrt{\mathbf{G}_h} \\ \mathbf{R}_\psi \mathbf{E}_1 \mathbf{H}_o \sqrt{\mathbf{G}_o} & -\mathbf{E}_1 \mathbf{H}_h \sqrt{\mathbf{G}_h} \end{bmatrix}^{-1} \\
&\left\{ \begin{bmatrix} -\mathbf{R}_\psi \sqrt{\mathbf{G}_o} \bar{\Gamma}_o \\ \mathbf{R}_\psi \mathbf{E}_1 (\sqrt{\mathbf{G}_o})^{-1} \bar{\bar{L}}_o \end{bmatrix} \mathbf{w}_o + \begin{bmatrix} \sqrt{\mathbf{G}_h} \bar{\Gamma}_h \\ -\mathbf{E}_1 (\sqrt{\mathbf{G}_h})^{-1} \bar{\bar{L}}_h \end{bmatrix} \mathbf{w}_h \right. \\
&+ \begin{bmatrix} -2\omega_z \mathbf{E}_1 \mathbf{R}_\psi \sqrt{\mathbf{G}_o} & \mathbf{0}_{2\times 2} \\ -\omega_z \mathbf{R}_\psi \mathbf{H}_o \sqrt{\mathbf{G}_o} & -\dot{\psi} \mathbf{H}_h \sqrt{\mathbf{G}_h} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_o \\ \dot{\mathbf{u}}_h \end{bmatrix} \\
&\left. - \begin{bmatrix} \mathbf{0}_{2\times 1} \\ \sigma_o \Gamma_o \dot{\mathbf{u}}_o \begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix} \end{bmatrix} \right\}, \\
\mathbf{K}_{2b} &= -\begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix}^{\mathsf{T}} \mathbf{R}_\psi \mathbf{E}_1 (\sqrt{\mathbf{G}_o})^{-1} \mathbf{L}_o \dot{\mathbf{u}}_o \\
&+ \sigma_o \bar{\bar{\Gamma}}_o \mathbf{w}_o + \sigma_h \bar{\bar{\Gamma}}_h \mathbf{w}_h + [\sigma_o \Gamma_o \ \sigma_h \Gamma_h] \mathbf{K}_{2a}.
\end{aligned}
\tag{5.39}
$$

The matrix that operates on the acceleration terms of the second-order kinematics is given by $\mathbf{K}_3(\mathbf{q})$:

$$
\mathbf{K}_3(\mathbf{q}) = \begin{bmatrix} \mathbf{I}_{4\times 4} & \mathbf{0}_{4\times 1} \\ [\sigma_o \Gamma_o \ \sigma_h \Gamma_h] & -1 \end{bmatrix} \mathbf{K}_{3a}(\mathbf{q}),
\tag{5.40}
$$

where

$$\mathbf{K}_{3a}(\mathbf{q}) = \begin{bmatrix} \begin{bmatrix} \mathbf{R}_\psi\sqrt{\mathbf{G}_o} & -\sqrt{\mathbf{G}_h} \\ \mathbf{R}_\psi\mathbf{E}_1\mathbf{H}_o\sqrt{\mathbf{G}_o} & -\mathbf{E}_1\mathbf{H}_h\sqrt{\mathbf{G}_h} \end{bmatrix}^{-1} & \mathbf{0}_{4\times1} \\ \mathbf{0}_{1\times4} & 1 \end{bmatrix} \mathbf{E}_2,$$

$$\mathbf{E}_2 = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

**5.12.3.1. Second-Order Rolling and Pure-Rolling Constraints.** The relative linear accelerations $\mathbf{a}_{\text{rel}} = {}^{c_h}\mathbf{a}_{p_h p_o}$ are constrained by the second-order rolling constraints $\mathbf{a}_{\text{rel}} = \mathbf{a}_{\text{roll}}$. These were derived in Eq. (60) of [**61**]. The general version is given in Eq. (5.4) and the full form is reproduced below:

$$\mathbf{a}_{\text{roll}} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{\text{roll}} = \begin{bmatrix} -\omega_z\mathbf{E}_1 \\ \begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix}^{\mathsf{T}} \end{bmatrix} \mathbf{R}_\psi\sqrt{\mathbf{G}_o}\dot{\mathbf{u}}_o. \tag{5.41}$$

The relative rotational acceleration $\alpha_z$ is constrained by the second-order-pure-rolling constraints $\alpha_z = \alpha_{z,\text{pr}}$. This was given in [**61**] as $\alpha_{z,\text{pr}} = 0$. We found this to be valid for simple geometries such as sphere-on-sphere, sphere-on-plane, and ellipsoid-on-plane, but for more complex geometries such as ellipsoid-on-ellipsoid and sphere-on-ellipsoid, $\alpha_{z,\text{pr}} = 0$ did not enforce the no-spin constraint.

To derive the rolling constraint we set the derivative of the expression for $\omega_z$ from the first-order kinematics equal to zero:

$$\begin{aligned} \frac{d}{dt}\omega_z &= \frac{d}{dt}(\sigma_o \Gamma_o \dot{\mathbf{u}}_o + \sigma_h \Gamma_h \dot{\mathbf{u}}_h - \dot{\psi}), \\ &= 0. \end{aligned} \tag{5.42}$$

From Eq. (5.42) and the second-order kinematics in Eq. (5.39) we solved for an expresion of the form:

$$\alpha_{z,\mathrm{pr}} = d_1(\mathbf{q}, \omega_{\mathrm{rel}}) + d_2(\mathbf{q}) \begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix}. \tag{5.43}$$

For the ellipsoid-on-ellipsoid, and ellipsoid-on-sphere models we tested, Eq. (5.43) simplified to:

$$\alpha_{z,\mathrm{pr}} = (\omega_{\mathrm{rel}} \times {}^{c_h}\omega_{oc_o})^{\mathsf{T}} \mathbf{n}_h, \tag{5.44}$$

where $\mathbf{n}_h$ is the unit contact normal of $\{c_h\}$. This expression is equivalent to the following term from the $\ddot{\psi}$ expression in the second-order kinematics in Eq (5.38):

$$\alpha_{z,\mathrm{pr}} = - \begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix}^{\mathsf{T}} \mathbf{R}_\psi \mathbf{E}_1 (\sqrt{\mathbf{G}_o})^{-1} \mathbf{L}_o \dot{\mathbf{u}}_o. \tag{5.45}$$

**5.12.3.2. Relative Acceleration Expression.** Eq. (5.5) gives an expression for the body acceleration of the object given the body acceleration of the hand and the relative

acceleration at the contact, and is reproduced below:

$$
\begin{aligned}
{}^{o}\dot{\mathcal{V}}_{so} =& [\mathrm{Ad}_{\mathbf{T}_{oh}}]\,{}^{h}\dot{\mathcal{V}}_{sh} + [\mathrm{Ad}_{\mathbf{T}_{oc_h}}]\dot{\mathcal{V}}_{\mathrm{rel}} \\
&+ \mathbf{K}_4(\mathbf{q}, \omega_{\mathrm{rel}}, {}^{h}\omega_{sh}).
\end{aligned}
$$

The following expression for $\mathbf{K}_4$ contains the velocity product terms:

$$
\begin{aligned}
\mathbf{K}_4(\mathbf{q}, \omega_{\mathrm{rel}}, {}^{h}\omega_{sh}) =& [\mathrm{Ad}_{\mathbf{T}_{oh}}] \begin{bmatrix} \mathbf{0}_{3\times 1} \\ [{}^{h}\omega_{sh}]([{}^{h}\omega_{sh}]\,{}^{h}\mathbf{r}_{hp_h}) \end{bmatrix} \\
&- [\mathrm{Ad}_{\mathbf{T}_{oc_h}}] \begin{bmatrix} [\omega_{\mathrm{rel}}](\mathbf{R}_{c_h o}\,{}^{o}\omega_{so}) \\ \mathbf{0}_{3\times 1} \end{bmatrix} \\
&- \begin{bmatrix} \mathbf{0}_{3\times 1} \\ [{}^{o}\omega_{so}]([{}^{o}\omega_{so}]\,{}^{o}\mathbf{r}_{op_o}) \end{bmatrix},
\end{aligned}
\tag{5.46}
$$

where ${}^{o}\omega_{so}$ comes from Eq (5.2), and the $[\mathrm{Ad}_{\mathbf{T}}]$, $\mathbf{r}$, and $\mathbf{R}$ expressions can be derived from the contact configuration $\mathbf{q}$.

## 5.13. Appendix C: Iterative Direct Collocation

We first describe the details of the direct collocation method, and then outline our iterative version.

Direct collocation is a method for trajectory optimization that optimizes an objective function $\mathcal{J}(\xi(t)) = \mathcal{J}(\mathbf{s}(t), {}^{h}\dot{\mathcal{V}}_{sh}(t))$ using polynomial spline approximations of the continuous states and controls. We chose to use trapezoidal collocation where the control trajectory ${}^{h}\dot{\mathcal{V}}_{sh}(t)$ is represented by piecewise-linear splines, the state trajectory $\mathbf{s}(t)$ is

represented by quadratic splines, and the trapezoidal rule is used for integration. Higher-order representations such as Hermite-Simpson collocation can also be used but with increased computational cost [**32**]. We define the objective function $\mathcal{J}(\mathbf{s}(t), {}^h\dot{\mathcal{V}}_{sh}(t))$ as the sum of the terminal cost and the running cost and omit the dependence on $t$ for clarity:

$$
\begin{aligned}
\mathcal{J}(\mathbf{s}, {}^h\dot{\mathcal{V}}_{sh}) &= \mathcal{M}(\mathbf{s}(t_f)) + \int_0^{t_f} \mathcal{L}(\mathbf{s}, {}^h\dot{\mathcal{V}}_{sh}) dt, \\
\mathcal{M}(\mathbf{s}(t_f)) &= \frac{1}{2}(\mathbf{s}(t_f) - \mathbf{s}_{\text{goal}})^{\mathsf{T}} \mathbf{P}_1 (\mathbf{s}(t_f) - \mathbf{s}_{\text{goal}}), \\
\mathcal{L}(\mathbf{s}, {}^h\dot{\mathcal{V}}_{sh}) &= \frac{1}{2}(\mathbf{s} - \mathbf{s}_{\text{des}})^{\mathsf{T}} \mathbf{Q} (\mathbf{s} - \mathbf{s}_{\text{des}}) + \frac{1}{2} {}^h\dot{\mathcal{V}}_{sh}{}^{\mathsf{T}} \mathbf{R}\, {}^h\dot{\mathcal{V}}_{sh},
\end{aligned}
\tag{5.47}
$$

where $\mathbf{P}_1$, $\mathbf{Q}$, and $\mathbf{R}$, penalize goal-state error, desired trajectory deviation, and control cost respectively, and $\mathbf{s}_{\text{des}}(t)$ is a desired trajectory. The path $\mathbf{s}_{\text{des}}(t)$ is chosen as the linear interpolation from $\mathbf{s}_{\text{start}}$ to $\mathbf{s}_{\text{goal}}$, which penalizes motions that do not move $\mathbf{s}$ towards the goal. Note that $\mathbf{s}_{\text{des}}(t)$ is not admissible in general (i.e. the states and controls do not satisfy the rolling dynamics equations).

The collocation method divides the trajectory $\xi(t)$ into $N$ segments, and the $N+1$ nodes at the ends of each segment are called collocation points. Each collocation point is expressed as $\xi_k(t) = (\mathbf{s}(t_k), {}^h\dot{\mathcal{V}}_{sh}(t_k))$ for $k \in [0, \ldots, N]$. For systems with $m$ state variables and $n$ control variables there are a total of $(N+1)(m+n)$ collocation points. The dynamics between each pair of sequential collocation points are enforced by the following condition:

$$
\begin{aligned}
\mathbf{s}_{k+1} - \mathbf{s}_k = \frac{1}{2}\Delta t_k (\mathcal{F}(\mathbf{s}_{k+1}, {}^h\dot{\mathcal{V}}_{shk+1}) + \mathcal{F}(\mathbf{s}_k, {}^h\dot{\mathcal{V}}_{shk})), \\
k \in [0, \ldots, N-1],
\end{aligned}
\tag{5.48}
$$

where $\Delta t_k = (t_{k+1} - t_k)$ indicates the interval duration and $\mathcal{F}(\mathbf{s}, {}^h\dot{\mathcal{V}}_{sh}) = \mathbf{K}_7(\mathbf{s}) + \mathbf{K}_8(\mathbf{s}){}^h\dot{\mathcal{V}}_{sh}$ is the rolling dynamics function from Eq. (5.12). Equation (5.48) is unique to the choice of trapezoidal collocation, and other integration methods require a different constraint [**32**].

The optimal control problem can be represented as the following nonlinear programming problem:

$$
\begin{aligned}
&\underset{\mathbf{s}(t_k),\ {}^h\dot{\mathcal{V}}_{sh}(t_k)}{\arg\min} && \mathcal{M}(\mathbf{s}(t_f)) + \sum_{i=0}^{N} \mathcal{L}(\mathbf{s}(t_k), {}^h\dot{\mathcal{V}}_{sh}(t_k))\Delta t_k \\
&\text{such that} && \mathcal{H}(\mathbf{s}(t_0) : \mathbf{s}(t_N); {}^h\dot{\mathcal{V}}_{sh}(t_0) : {}^h\dot{\mathcal{V}}_{sh}(t_{N-1})) = 0, \\
& && \mathcal{I}(\mathbf{s}(t_0) : \mathbf{s}(t_N); {}^h\dot{\mathcal{V}}_{sh}(t_0) : {}^h\dot{\mathcal{V}}_{sh}(t_{N-1})) \leq 0,
\end{aligned}
\tag{5.49}
$$

where $\mathcal{H}(\cdot)$ enforces the rolling dynamics in Eq. (5.48), and gives the equality constraints $\mathbf{s}(0) = \mathbf{s}_{\text{start}}$ (and optionally $\mathbf{s}(t_f) = \mathbf{s}_{\text{goal}}$ which can be relaxed by replacing with a high weighting matrix on the goal state $\mathbf{P}_1$). The expression $\mathcal{I}(\cdot)$ includes the contact wrench inequality constraints on Eq. (5.10), the controls limits (${}^h\dot{\mathcal{V}}_{sh\min} \leq {}^h\dot{\mathcal{V}}_{sh} \leq {}^h\dot{\mathcal{V}}_{sh\max}$), and enforces any constraints on the configurations (e.g., due to singularities in the coordinate chart). Equation (5.49) is a finite-dimensional nonlinear optimization problem, and a solution $\xi_{\text{iDC}}(t)$ can be found using nonlinear optimizers such as SNOPT, IPOPT, or MATLAB's `fmincon`.

The integration error can be determined by comparing the trajectory $\mathbf{s}_{\text{iDC}}(t)$ from the direct collocation method with the trajectory $\mathbf{s}_{\text{fine}}(t)$, where $\mathbf{s}_{\text{fine}}(t)$ is obtained by integrating the initial state over the interval $t = [0, t_f]$ using Eq. (5.12), the piecewise-linear output controls ${}^h\dot{\mathcal{V}}_{sh\text{iDC}}(t)$, and a higher-order integrator with small time steps (e.g. $dt \leq 0.001$). With fewer segments $N$, the integration error is larger, but there

are fewer constraints for the nonlinear solver. This means that the optimizer is more likely to find a solution, and with less computational cost. The choice of $N$ is therefore a trade-off between computational cost/optimizer convergence and integration error. We implemented the iterative direct collocation (iDC) method to address this.

We first run the nonlinear optimization method using MATLAB's `fmincon` for a small number of segments (e.g. $N = 25$) to find a trajectory $\xi_{\text{iDC}}(t)$. The recalculated path $\mathbf{s}_{\text{fine}}(t)$ is found using smaller integration timesteps and a higher-order integrator (`ode45`), and the planner is terminated if the goal-state tolerance of the fine trajectory is satisfied ($\mathbf{s}_{\text{error}}(t_f) < \eta$). If the goal-state error is too large, the previous output trajectory serves as the initial trajectory guess for the next iteration with twice as many segments ($N \to 2N$) ($N$ could also be increased by a fixed value $\Delta N$ between each iteration to add an additional tuning parameter for the planning method). This is repeated until a valid trajectory $\xi_{\text{sol}}(t)$ is found, the maximum number of iDC iterations is reached, or the optimization converges to an invalid point.

In our tests, an initial optimization with a fine control discretization often takes an unnecessarily long time to converge or even fails to converge to a feasible solution. The coarse initial guess followed by successive refinement yields higher-quality solutions faster and more consistently. The iterative refinement process acts as a form of regularization.

Additional methods can improve the planning success such as ignoring the friction inequality constraints for the initial iteration(s), and decreasing/zeroing the weight $\mathbf{Q}$ that penalizes trajectories that do not track a straight-line trajectory to the goal state.

CHAPTER 6

# Conclusion

This thesis presents methods of modeling, motion planning, and feedback control for hybrid, dynamic, and nonprehensile manipulation.

Chapter 2 outlined five subproblems to address such manipulation tasks: determining a set of manipulation primitives, choosing a sequence of tasks, picking transition states, motion planning for each individual primitive, and stabilizing each mode using feedback control. We apply the framework to plan a sequence of motions for manipulating a block with a planar 3R manipulator. We demonstrate preliminary experimental results for a block resting on the manipulator with a desired goal state on a ledge outside of the robot's workspace. The planned primitives reorient the block using a series of fixed, rolling, and sliding contact modes, and throw it to the goal state. This work provides a high-level framework to formulate these complex manipulation problems, and the specific subproblems can be built upon to further improve robotic manipulation capabilities.

Chapter 3 examines the problem of planning and stabilizing the trajectory of one smooth body rolling on the surface of another. The two control inputs are the angular velocity of the moving body about two orthogonal axes in the contact tangent plane; spinning about the contact normal is not allowed. To achieve robustness and computational efficiency, our approach to trajectory planning is based on solving a series of optimization problems of increasing complexity. To stabilize the trajectory in the face of perturbations,

we use a linear quadratic regulator. We apply the approach to examples of a sphere rolling on a sphere and an ellipsoid rolling on an ellipsoid.

Chapter 4 presents corrections to the second-order kinematics equations first derived by Sarkar et al. in [**61**] and sets up the work on dynamic manipulation. Chapter 5 presents methods to control the motion of objects that are in rolling contact with a robot manipulator or "hand" in three dimensions. We directly control the motion of the hand to indirectly control the motion of the rolling object. Our approach to dynamic rolling manipulation can be split into four subproblems: 1) calculating the first- and second-order rolling kinematic equations; 2) deriving the rolling dynamics; 3) planning rolling motions that satisfy the dynamics; and 4) feedback control of rolling trajectories. The results are validated against examples with analytical solutions in simulation, and tested experimentally.

## 6.1. Future Work

### 6.1.1. Hybrid Manipulation

Chapter 2 provides a high-level framework to address complex manipulation problems, and the specific subproblems can be built upon to further improve robotic manipulation capabilities. Future work will focus on automating the process of generating primitives, choosing mode sequences, picking transition states, planning within single modes, and stabilizing them.

**6.1.1.1. Determining Manipulation Primitives.** The primitives are defined by the location, types, and number of contacts between the object, manipulator, and the environment, as well as the control mode at each contact (such as position or force control).

These are the building blocks of the hybrid manipulation plans, and each additional primitive increases the manipulation capabilities of a robot. Future work will focus on automatically generating primitives and dynamic equations from information such as 3D models and vision data.

**6.1.1.2. Choosing a Sequence of Tasks.** Hybrid manipulation moves through multiple contact modes that connect the initial state to the goal state. Automating this selection is crucial for making hybrid manipulation tasks solvable in unstructured environments. Mode sequences can be weighted by desired properties such as minimum number of contact modes or penalizing uncertain transitions such as those that include impacts.

**6.1.1.3. Picking Transition States.** Understanding the boundaries between different contact modes is crucial for hybrid manipulation. There are often many choices of transitions (such as when to release an object before a throw), so defining desired properties of a transition state and choosing one is a complex problem. Deriving representations of the intermode constraints will allow us to analyze the local topology of the state-control space and plan mode transitions. Understanding belief propagation during mode transitions can also lead to more robust transitions.

**6.1.1.4. Motion Planning Within Individual Primitives.** Motion planning work in the literature often focuses on a single set of dynamic equations, so future work in intramode planning can borrow heavily from this research. The most interesting future work in this area for hybrid planning will involve the beginning and end of a single mode plan. The trajectory must avoid unwanted transitions and ensure a successful transition at the end.

**6.1.1.5. Feedback Control Within Modes.** The majority of manipulation tasks are done without state feedback on the object, and this is a necessary addition to making manipulation primitives reliable. Some potential methods for feedback include the application of linear controllers to linearized versions of the dynamics, receding horizon controllers, adaptive controllers, and hybrid force/motion controllers. We want to ensure smoothness when transitioning between control laws, and we will apply techniques such as sum of squares to estimate the local basins-of-attraction of each controller to know how robust they are to error. For contact modes where feedback control is impractical we will develop methods to explicitly estimate and manage uncertainty.

An important subproblem for this work will be the development of projection methods that prevent the controls from causing undesired mode transitions. Projection methods raise questions such as whether projected controls maintain their stability properties and whether extra degrees of freedoms at contacts can be used to increase the control authority of a contact. These will have to be addressed in future work to ensure the chosen control methods will still be applicable to the constraints of a given primitive.

### 6.1.2. Rolling Manipulation

There are many interesting areas for future work that extend the work on rolling manipulation outlined in Chapters 3, 4, and 5.

**6.1.2.1. Surface Parameterization.** The work in this thesis requires orthogonal parameterizations of surfaces, and while any smooth surface can be locally represented as such, the ability to use non-orthogonal parameterizations would improve the utility of

this method. We have used non-orthogonal parameterizations for simulating the second-order kinematics by defining local orthogonal frames at each point on the surface, but the dynamics derivation in Section 5.6 would need to be modified to relax the orthogonality assumption.

Moving beyond explicit parameterizations would also improve the scope of this work such as using smooth objects represented by point clouds. There is some relevant work in the computer graphics field that uses conformal (angle-preserving) mappings to map smooth objects to a sphere, and it could be interesting to explore how that could be applied to rolling manipulation.

Future research could process an object defined by a point-cloud, automatically generate an atlas of orthogonal coordinate charts that cover the surface, and plan motions with stabilizing feedback controllers through multiple coordinate charts.

**6.1.2.2. Feedback Control.** We showed one example of feedback control but there are many areas for extending feedback control for rolling objects. Our method utilizes a linearized LQR controller to stabilize planned trajectories and balance states which results in a simple feedback law given by Eq. (5.19) that can easily run at high speeds (1000 Hz in our implementation). This method requires knowledge of the contact coordinates which are difficult to estimate for general, 3D rolling motions. Hardware such as manipulators equipped with contact location sensing could help address this, especially if combined with an observability model such as the one developed in [**29**]. Other feedback methods such as energy-based feedback controllers could be used to stabilize trajectories [**14**] as well, and may avoid the need for estimating specific contact states.

**6.1.2.3. Hybrid Rolling, Sliding, and Free-Flight Dynamics.** This work derives dynamics for rolling and pure-rolling but does not consider roll-slide dynamics where there is relative linear velocity at the contact. This would allow more general modeling where the contact mode is determined based on the state and contact constraints, and the appropriate dynamics equations are applied. This is then a hybrid dynamics problem, where the state evolves within a single contact mode until a mode transition occurs. Chapter 2 outlines a framework for hybrid manipulation planning and there are many interesting applications of dynamic manipulation that combine contact modes such as rolling, pure-rolling, rolling + sliding, and free flight.

# References

[1] Agrachev, A. A., and Sachkov, Y. L. *Control theory from the geometric viewpoint*, vol. 87. Springer-Verlag Berlin Heidelberg, 2004.

[2] Alouges, F., Chitour, Y., and Long, R. A motion-planning algorithm for the rolling-body problem. *IEEE Transactions on Robotics 26*, 5 (Oct. 2010), 827–836.

[3] Anderson, B. D., and Moore, J. B. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[4] Anderson, B. D. O., and Moore, J. B. *Optimal Control: Linear Quadratic Methods*. Dover Books on Engineering. Dover Publications, 2007.

[5] Anitescu, M., Cremer, J. F., and Potra, F. A. Formulating three-dimensional contact dynamics problems. *Journal of Structural Mechanics 24*, 4 (1996), 405–437.

[6] Ansari, A. R., and Murphey, T. D. Sequential action control: closed-form optimal control for nonlinear and nonsmooth systems. *IEEE Transactions on Robotics 32*, 5 (2016), 1196–1214.

[7] Bätz, G., Yaqub, A., Wu, H., Kühnlenz, K., Wollherr, D., and Buss, M. Dynamic manipulation: Nonprehensile ball catching. In *18th Mediterranean Conference on Control and Automation, MED'10* (2010), IEEE, pp. 365–370.

[8] Becker, A., and Bretl, T. Approximate steering of a plate-ball system under bounded model perturbation using ensemble control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), IEEE, pp. 5353–5359.

[9] Berenson, D., Srinivasa, S. S., and Kuffner, J. Task Space Regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research 30*, 12 (2011), 1435–1460.

[10] Bicchi, A., and Sorrentino, R. Dexterous manipulation through rolling. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation* (May 1995), vol. 1, IEEE, pp. 452–457.

[11] Brescianini, D., Hehn, M., and D'Andrea, R. Quadrocopter pole acrobatics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013), IEEE, pp. 3472–3479.

[12] Brogan, W. L. *Modern control theory*, 3rd ed. Prentice Hall, 1991.

[13] Cai, C., and Roth, B. On the Spatial Motion of a Rigid Body with Point Contact. In *IEEE International Conference on Robotics and Automation* (1987), vol. 4, pp. 686–695.

[14] Cefalo, M., Lanari, L., and Oriolo, G. Energy-based control of the butterfly robot. *IFAC Proceedings Volumes 39*, 15 (2006), 1–6.

[15] Chitour, Y., Molina, M. G., and Kokkonen, P. The rolling problem: overview and challenges. In *Geometric Control Theory and Sub-Riemannian Geometry*. Springer, 2014, pp. 103–122.

[16] Choudhury, P., and Lynch, K. M. Rolling manipulation with a single control. *The International Journal of Robotics Research 21*, 5-6 (2002), 475–487.

[17] Çimen, T. State-Dependent Riccati Equation (SDRE) control: A survey. *IFAC Proceedings Volumes (IFAC-PapersOnline) 17*, 1 Part 1 (2008), 3761–3775.

[18] Date, H., Sampei, M., Ishikawa, M., and Koga, M. Simultaneous control of position and orientation for ball-plate manipulation problem based on time-state control form. *IEEE transactions on robotics and automation 20*, 3 (2004), 465–480.

[19] Duindam, V., and Stramigioli, S. Modeling the kinematics and dynamics of compliant contact. In *2003 IEEE International Conference on Robotics and Automation* (2003), vol. 3, IEEE, pp. 4029–4034.

[20] Erdmann, M. A. An Exploration of Nonprehensile Two-Palm Manipulation. *The International Journal of Robotics Research 17*, 5 (1998), 485–503.

[21] Erumalla, S. R., Pasupuleti, S., and Ryu, J.-C. Throwing, catching, and balancing of a disk with a disk-shaped end effector on a two-link manipulator. *Journal of Mechanisms and Robotics 10*, 5 (2018).

[22] Fliess, M., Lévine, J., Martin, P., and Rouchon, P. Flatness and defect of non-linear systems: introductory theory and examples. *International Journal of Control 61*, 6 (June 1995), 1327–1361.

[23] Furukawa, N., Namiki, A., Taku, S., and Ishikawa, M. Dynamic Regrasping Using a High-speed Multifingered Hand and a High-speed Vision System. *IEEE International Conference on Robotics and Automation* (2006), 181–187.

[24] Gahleitner, R. Ball on ball: Modeling and control of a novel experiment set-up. *IFAC-PapersOnLine 48*, 1 (2015), 796–801.

[25] Goebel, R., Sanfelice, R., and Teel, A. Hybrid dynamical systems. *IEEE Control Systems 29*, 2 (apr 2009), 28–93.

[26] Harada, K., Kawashima, T., and Kaneko, M. Rolling based manipulation under neighborhood equilibrium. *The International Journal of Robotics Research 21*, 5-6 (2002), 463–474.

[27] Hristu-Varsakelis, D. The dynamics of a forced sphere-plate mechanical system. *IEEE Transactions on Automatic Control 46*, 5 (2001), 678–686.

[28] Ivaldi, S., Peters, J., Padois, V., and Nori, F. Tools for simulating humanoid robot dynamics: a survey based on user feedback. In *2014 IEEE-RAS International Conference on Humanoid Robots* (2014), IEEE, pp. 842–849.

[29] Jia, Y.-B., and Erdmann, M. Local observability of rolling. In *Workshop on Algorithmic Foundation of Robotics* (1998), pp. 251–263.

[30] Johnson, A. M., Burden, S. A., and Koditschek, D. E. A hybrid systems model for simple manipulation and self-manipulation systems. *The International Journal of Robotics Research 35*, 11 (2016), 1354–1392.

[31] Jurdjevic, V. The geometry of the plate-ball problem. *Archive for Rational Mechanics and Analysis 124*, 4 (1993), 305–328.

[32] Kelly, M. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review 59*, 4 (2017), 849–904.

[33] Krakowski, K. A., Atima, F., and Leite, S. Why controllability of rolling may fail: a few illustrative examples. In *Pre-Publicacoes do Departamento de Matematica, no. 12-26.* (2012), Universidade de Coimbra, pp. 1–30.

[34] Lafferriere, G., and Sussmann, H. Motion planning for controllable systems without drift. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation* (April 1991), IEEE, pp. 1148–1153.

[35] Lee, K.-K., Batz, G., and Wollherr, D. Basketball robot: Ball-on-plate with pure haptic information. In *2008 IEEE International Conference on Robotics and Automation* (2008), IEEE, pp. 2410–2415.

[36] Li, Z., and Canny, J. Motion of two rigid bodies with rolling constraint. *IEEE Transactions on Robotics and Automation 6*, 1 (1990), 62–72.

[37] Lippiello, V., Ruggiero, F., and Siciliano, B. The effect of shapes in input-state linearization for stabilization of nonprehensile planar rolling dynamic manipulation. *IEEE Robotics and Automation Letters 1*, 1 (2016), 492–499.

[38] Liu, T., and Wang, M. Y. Computation of three-dimensional rigid-body dynamics with multiple unilateral contacts using time-stepping and gauss-seidel methods. *IEEE Transactions on Automation Science and Engineering 2*, 1 (2005), 19–31.

[39] Lynch, K. M. Underactuated robots. In *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. Springer-Verlag, 2015, pp. 1503–1510.

[40] Lynch, K. M., Bloch, A. M., Drakunov, S. V., Reyhanoglu, M., and Zenkov, D. Control of nonholonomic and underactuated systems. In *The Control Handbook*, W. Levine, Ed. Taylor and Francis, 2011.

[41] Lynch, K. M., and Mason, M. T. Dynamic Nonprehensile Manipulation: Controllability, Planning, and Experiments. *The International Journal of Robotics Research 18*, 1 (jan 1999), 64–92.

[42] Lynch, K. M., and Park, F. C. *Modern robotics: mechanics, planning, and control*. Cambridge University Press, 2017.

[43] Lynch, K. M., Shiroma, N., Arai, H., and Tanie, K. The roles of shape and motion in dynamic manipulation: the butterfly example. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation* (1998), vol. 3, pp. 1958–1963 vol.3.

[44] Maeda, Y., and Arai, T. Planning of graspless manipulation by a multifingered robot hand. *Advanced Robotics 19*, 5 (jan 2005), 501–521.

[45] Marigo, A., and Bicchi, A. Rolling bodies with regular surface: controllability theory and applications. *IEEE Transactions on Automatic Control 45*, 9 (2000), 1586–1599.

[46] Marigo, A., and Bicchi, A. A local-local planning algorithm for rolling objects. In *Proceedings 2002 IEEE International Conference on Robotics and Automation* (2002), vol. 2, IEEE, pp. 1759–1764.

[47] Milne, E. A. *Vectorial mechanics*. Methuen & Company, 1957.

[48] Miyazawa, K., Maeda, Y., and Arai, T. Planning of graspless manipulation based on rapidly-exploring random trees. *Proceedings of the IEEE International Symposium on Assembly and Task Planning 2005* (2005), 7–12.

[49] Montana, D. J. The kinematics of contact and grasp. *The International Journal of Robotics Research 7*, 3 (1988), 17–32.

[50] Müller, P. C., and Weber, H. I. Analysis and optimization of certain qualities of controllability and observability for linear dynamical systems. *Automatica 8*, 3 (1972), 237–246.

[51] Murray, R., and Sastry, S. Steering nonholonomic systems in chained form. In *30th IEEE Conference on Decision and Control* (1991), IEEE, pp. 1121–1126.

[52] Murray, R. M. Nilpotent bases for a class of nonintegrable distributions with applications to trajectory generation for nonholonomic systems. *Mathematics of Control, Signals, and Systems 7*, 1 (1994), 58–75.

[53] Oriolo, G., and Vendittelli, M. A framework for the stabilization of general nonholonomic systems with an application to the plate-ball mechanism. *IEEE Transactions on Robotics 21*, 2 (2005), 162–175.

[54] Pasqualetti, F., Zampieri, S., and Bullo, F. Controllability metrics, limitations and algorithms for complex networks. *IEEE Transactions on Control of Network Systems 1*, 1 (2014), 40–52.

[55] Pekarovskiy, A., Nierhoff, T., Schenek, J., Nakamura, Y., Hirche, S., and Buss, M. Online deformation of optimal trajectories for constrained nonprehensile manipulation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (may 2015), IEEE, pp. 2481–2487.

[56] Posa, M., Cantu, C., and Tedrake, R. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research 33*, 1 (jan 2014), 69–81.

[57] Posa, M., Tobenkin, M., and Tedrake, R. Stability Analysis and Control of Rigid-Body Systems With Impacts and Friction. *IEEE Transactions on Automatic Control 61*, 6 (jun 2016), 1423–1437.

[58] Rehan, M., and Reyhanoglu, M. Control of Rolling Disk Motion on an Arbitrary Smooth Surface. *IEEE Control Systems Letters 2*, 3 (July 2018), 357–362.

[59] Ryu, J.-C., and Lynch, K. M. Contact juggling of a disk with a disk-shaped manipulator. *IEEE Access 6* (2018), 60286–60293.

[60] Sarkar, N. *Control of Mechanical Systems with Rolling Contacts: Applications to Robotics*. PhD thesis, University of Pennsylvania, 1993.

[61] Sarkar, N., Kumar, V., and Yun, X. Velocity and acceleration analysis of contact between three-dimensional rigid bodies. *Journal of Applied Mechanics 63*, 4 (1996), 974.

[62] Sarkar, N., Xiaoping Yun, and Kumar, V. Control of contact interactions with acatastatic nonholonomic constraints. *The International Journal of Robotics Research 16*, 3 (June 1997), 357–374.

[63] Sarkar, N., Yun, X., and Kumar, V. Dynamic control of 3-D rolling contacts in two-arm manipulation. *IEEE Transactions on Robotics and Automation 13*, 3 (June 1997), 364–376.

[64] Serra, D., Ferguson, J., Ruggiero, F., Siniscalco, A., Petit, A., Lippiello, V., and Siciliano, B. On the experiments about the nonprehensile reconfiguration of a rolling sphere on a plate. In *2018 26th Mediterranean Conference on Control and Automation (MED)* (2018), IEEE, pp. 13–20.

[65] Serra, D., Ruggiero, F., Donaire, A., Buonocore, L. R., Lippiello, V., and Siciliano, B. Control of nonprehensile planar rolling manipulation: A passivity-based approach. *IEEE Transactions on Robotics 35*, 2 (2019), 317–329.

[66] Shi, J., Woodruff, J. Z., and Lynch, K. M. Dynamic in-hand sliding manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (sep 2015), IEEE, pp. 870–877.

[67] Srinivasa, S., Erdmann, M., and Mason, M. Using projected dynamics to plan dynamic contact manipulation. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2005), IEEE, pp. 3618–3623.

[68] Surov, M., Shiriaev, A., Freidovich, L., Gusev, S., and Paramonov, L. Case study in non-prehensile manipulation: planning and orbital stabilization of one-directional rollings for the "butterfly" robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), IEEE, pp. 1484–1489.

[69] Tassa, Y., and Todorov, E. Stochastic Complementarity for Local Control of Discontinuous Dynamics. *Robotics: Science and Systems* (2010).

[70] Taylor, O., and Rodriguez, A. Optimal shape and motion planning for dynamic planar manipulation. *Autonomous Robots 43*, 2 (2019), 327–344.

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

[72] Walsh, G., Tilbury, D., Sastry, S., Murray, R., and Laumond, J. Stabilization of trajectories for systems with nonholonomic constraints. *IEEE Transactions on Automatic Control 39*, 1 (1994), 216–222.

[73] Weltner, K. Stable circular orbits of freely moving balls on rotating discs. *American Journal of Physics 47*, 11 (1979), 984–986.

[74] Woodruff, J. Z., and Lynch, K. M. Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), IEEE, pp. 4066–4073.

[75] Woodruff, J. Z., and Lynch, K. M. Second-order contact kinematics between three-dimensional rigid bodies. *Journal of Applied Mechanics 86*, 8 (May 2019).

[76] Woodruff, J. Z., Ren, S., and Lynch, K. M. Motion planning and feedback control of rolling bodies. *IEEE Access 8* (2020), 31780–31791.

[77] Xiao, M., and Ding, Y. Contact Kinematics between three-dimensional rigid bodies with general surface parameterization. *Journal of Mechanisms and Robotics* (2020), 1–28.

[78] Yashima, M., Shiina, Y., and Yamaguchi, H. Randomized manipulation planning for a multi-fingered hand by switching contact modes. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)* (2003), vol. 2, IEEE, pp. 2689–2694.

APPENDIX A

# Design and Control of 3-DOF Planar Robot

This chapter outlines details of the experimental setup used in this thesis

## A.1. Experimental setup description

A diagram of the experimental setup is shown in Figure A.1, and a picture of the experimental setup is shown in Figure A.2. Experiments are conducted at 40% full gravity by inclining the table at 24 degrees with respect to horizontal. Each link is actuated by a DC Harmonic Drive motor, current controlled using Junus motor amplifiers. The 1000 Hz motion controller runs on a PC104 embedded computer running the QNX real-time operating system. Vision feedback is given by a 250 Hz IR Optitrack camera. Desired trajectories and experimental results are transmitted between the PC104 and a PC running MATLAB using a TCP/IP connection.

The interface between different components of the setup is shown in Figure A.3.

## A.2. Workflow description

There are two main software components for the setup. The first is the multi-threaded control loop running on the PC104, and the second is the trajectory generation/data analysis software written in MATLAB and running on the PC.

The controller for the arm is written in c, and runs on the PC104 QNX RTOS. There are three threads running, the highest priority 1000 Hz control thread, the medium

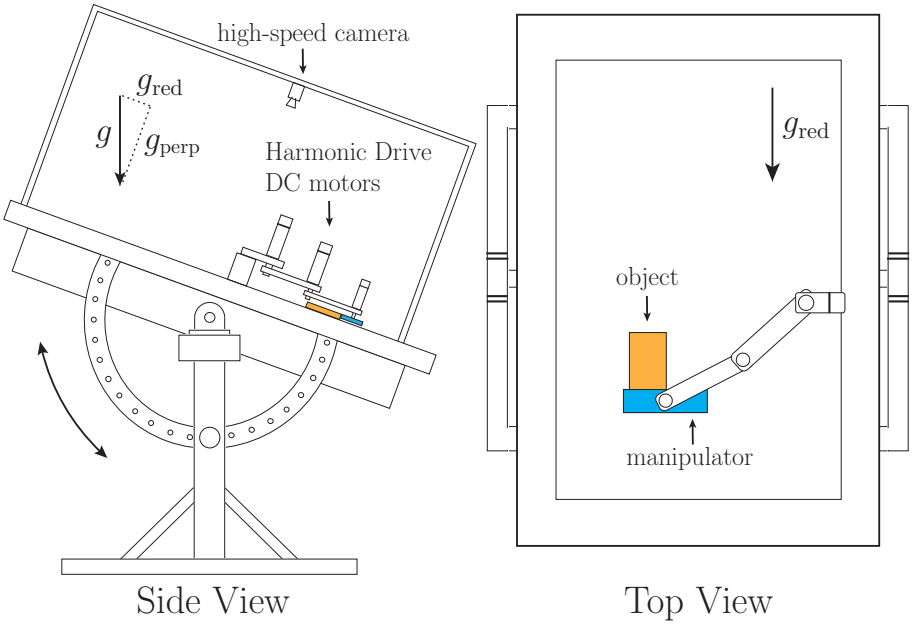



Figure A.1. Our experimental setup consists of an inclined air hockey table with a planar 3R robot driven by Harmonic Drive DC motors, and an OptiTrack s250e 250 Hz camera. The angle of the table allows 2D dynamic manipulation experiments in reduced gravity ($0.4g$ shown here), and the camera system gives feedback on object positions.

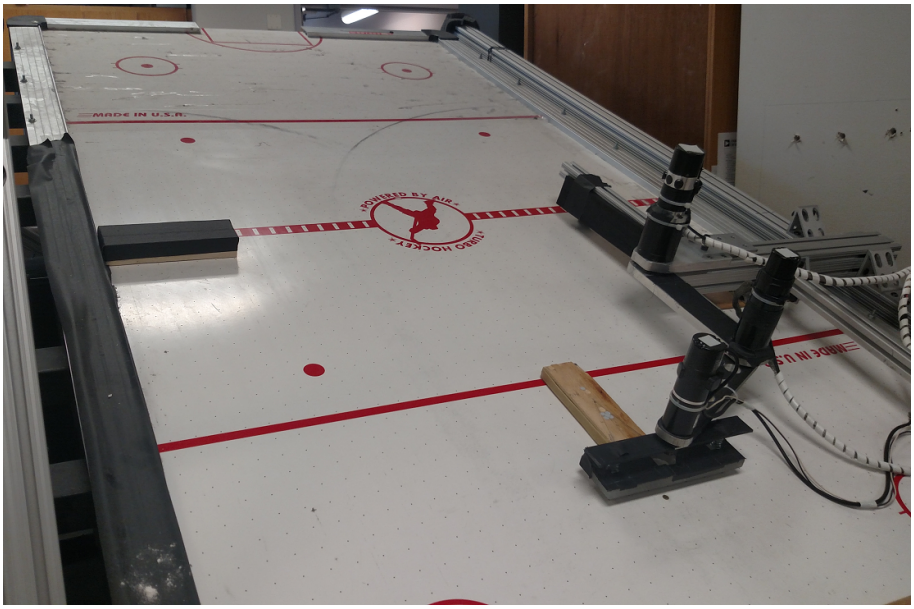


Figure A.2. A picture of the experimental setup.

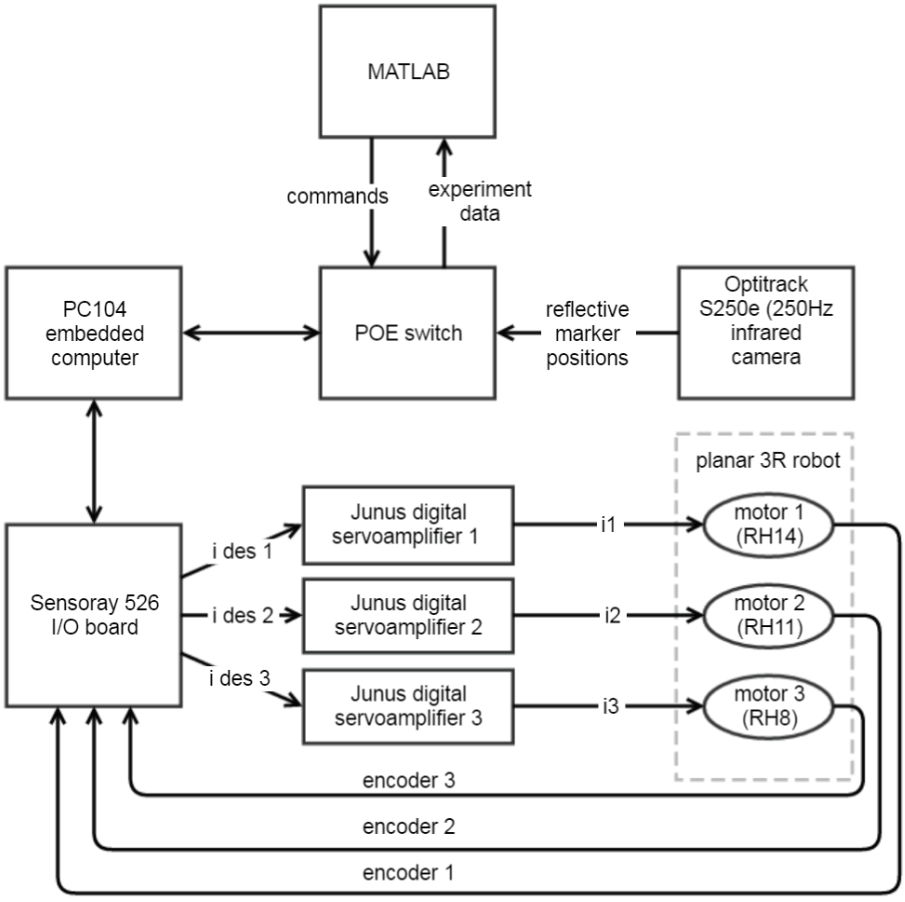



Figure A.3. Interface between different components of the experimental setup.

priority vision thread, and the lowest priority communication thread. To run a trajectory, the PC104 code initializes motors, the vision system, and parameters, and then receives desired trajectories from the PC along with the desired control mode. Depending on the control mode, the trajectories can be task/joint space trajectories for the arm, or desired positions of the object for manipulation tasks that include feedback control such as balancing. Once the trajectory is ready, it is run from MATLAB and the data is sent back to MATLAB, and stored in a time-stamped trajectory data-log. Desired data can then be plotted and simulated from this log to see how well the system performed.