Robust Manipulation with Active Compliance

Yifan Hou

CMU-RI-TR-21-03
March 2021

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Matthew T. Mason, Chair, CMU RI
Christopher G. Atkeson, CMU RI
Aaron M. Johnson, CMU MechE
Alberto Rodriguez, MIT MechE

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

Keywords: Manipulation, Force Control, Motion Planning, Robust Control

Abstract

Human manipulation skills are filled with creative use of physical contacts to move the object about the hand and in the environment. However, it is difficult for robot manipulators to enjoy this dexterity since contacts may cause the manipulation task to fail by introducing huge forces or unexpected change of constraints, especially when modeling uncertainties and disturbances are present. A properly designed robot compliance can provide the robot with the resilience and reliability in handling contacts.

This thesis proposes a framework for robust manipulation with contacts using active compliance. We provide quasi-static modeling that shows the necessity of compliance in rigid body manipulation. We further identify two causes of failure in manipulation: kinematic ill-conditioning and unexpected change of contact modes, and illustrate how the robot compliance can help avoid those failure types in manipulation tasks.

First, we propose robustness metrics for each type of the failures. The metrics measure the amount of modeling uncertainty and the magnitude of external disturbance forces the system can take before a failure happens.

Second, we provide methods to optimize the robustness metrics in a compliance control setting, as well as methods to improve the robustness of a contact-implicit motion planning.

Finally, we experimentally validate our proposed approaches in a variety of manipulation problems. Our method efficiently finds solutions with consistent high quality during testing. The result shows that our framework trades-off well between model complexity and accuracy, captures major factors in manipulation problems while keeping a low computation burden.

Acknowledgments

First of all, I would like to thank my advisor Matt Mason for taking me into the world of manipulation and showed me how to appreciate the beauty of it. I benefit from the numerous discussions with Matt, and learned to develop a problem-centric mindset. Matt is a great advisor who gave me the freedom to explore my interests, while influencing me to grow my own taste of research problems. I am fortunate to spend my PhD years with Matt, who is always a fun person to talk to and cooks amazing ribs. Special thanks to my thesis committee members: Chris Atkeson, Aaron Johnson, and Alberto Rodriguez. The quality of this thesis benefited from your questions, comments, and discussions.

I would like to thank my MCube collaborators and friends: Nikhil Chavan-Dafle, Maria Bauza, Ian Taylor, Daoling Ma, Siyuan Dong, Francois Hogan, Kuan-Ting Yu, Rachel Holladay, Orion Taylor, Nima Fazeli, Neel Doshi, Bernardo Aceituno, and Alberto Rodriguez. Thanks to Javier Peña for sharing your enormous knowledge on linear system conditioning and ill-posedness analysis. Thanks to Siyuan Feng, Naveen kuppuswamy, and Alex Alspach for the team work in TRI.

My life in CMU robotics institute would not be so fun and fulfilling without my friends here. I would like to thank my fellow MLabbers: Jiaji Zhou, Robbie Paolini, Zhengzhong Jia, Ankit Bhatia, Xianyi Cheng, Eric Huang, Jon King, Pragna Mannam, Alex Volkov, Reuben Aronson, Gilwoo Lee, and Amy Santoso. I will always remember the time we spent together building robots, debugging all night before deadlines, celebrating anything that worked, making the lab the messiest place on the planet then spending hours to clean it up. Thanks to Jean Harpley for taking care of the lab and us, and making sure we are never out of any supply.

Finally, I would like to thank my wife, Mo Li, who taught me that life has more purposes than work, and that happiness comes from not only achievements. Thank you for your endless care and encouragement through happy and difficult times, and for living through the PhD journey together with me. Thanks to my parents, Yurong Sha and Jianguo Hou, for trusting and supporting my choices and always be there when I need them.

Contents

List of Figures

Chapter 1

Introduction

Manipulation is the process of changing the state of the physical world by transmitting forces through contacts. Naturally, contacts play a central role in manipulation. Human manipulation gains dexterity by utilizing contacts in many ways: grasping, pushing, tumbling, pivoting, scooping, etc. When picking up a flat object lying at the bottom of a box, we touch the object by the side and press it against a corner to flip it up; when moving a heavy refrigerator, we hold it with two hands and pivot it about one of its feet; when positioning a wooden piece for band saw cutting, we press the piece against the workbench firmly to withstand disturbance forces. In a word, contacts provide human manipulation with more dexterity and reliability.

In robotic manipulation, however, contacts are usually considered to be dangerous and should be avoided. For example, most industrial applications of robot hands falls into the category of "pick-and-place": pick up an object, then place it at another location. In pick-and-place, making and breaking contacts only happens once during grasping and dropping; for the rest of the process the robot tries to avoid any collision with the environment. The result is the limited dexterity of robot's manipulation comparing with human's. An example is shown in Figure 1.1. In this thesis, we define *contact-rich manipulation* as a manipulation problem whose contact sequence is unknown, so as to distinguish from traditional pick-and-place manipulation.

There are several reasons why robots in real world applications don't make full use of contacts. First of all, it is difficult to reason about using contacts in motion planning because the contacts bring numerical difficulties. Computing a sequence of making and breaking contacts for a multi-DOF system could take tens of minutes. The problem is called *contact-implicit motion planning* [63] and has drawn attention of both the locomotion and the manipulation communities.

After computing a motion plan, it is still challenging to execute the planned trajectory in real world experiments because the constraints introduced by the contacts may cause the task to fail in many ways. The discrete nature of contacts makes it possible for a small disturbance to trigger a drastic change of contact constraints, e.g. unexpected slips. The original robot control is likely to be invalid under a different constraint which could eventually fail the task. To make things worse, the robot motion may incur huge internal forces and damage the system when it violates the actual contact constraints.

The robot hardware is another reason why the execution of contact-rich manipulation is difficult. Most robots, especially industrial robots, are not designed to make contacts. They have adequate accuracy and rigidity, however, they are not capable of high bandwidth and high accu-

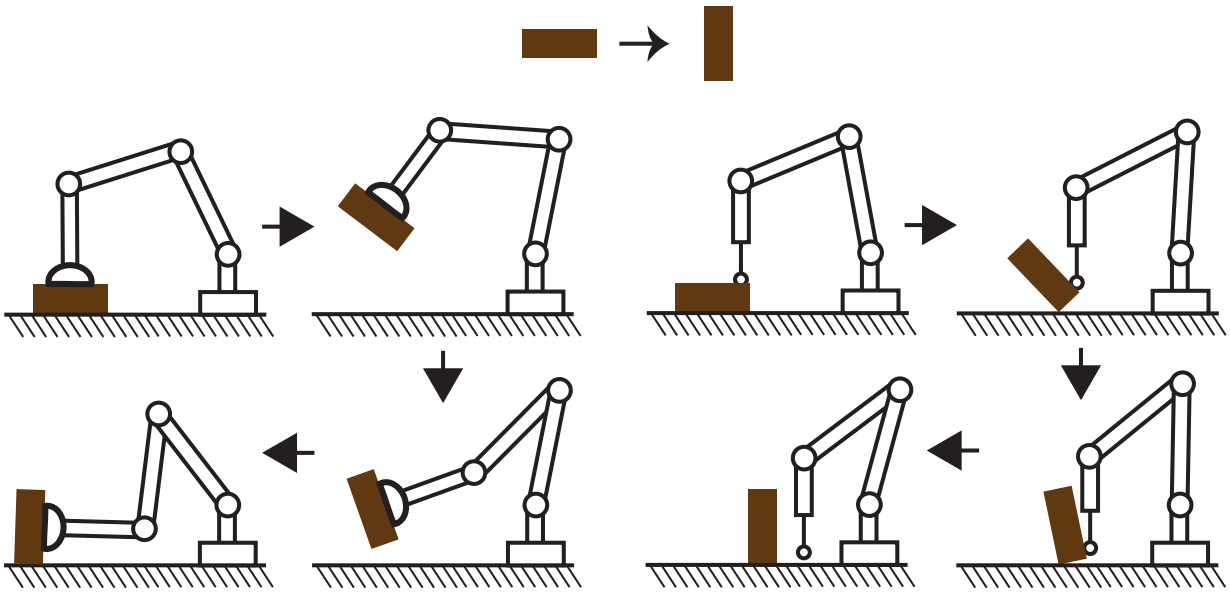

Figure 1.1: Two manipulation strategies to flip a box. Left: Pick-and-place. The robot must pick the object up and rotate the whole arm with the box. Right: A more human-like strategy that use the table contact to tumble the object. The required robot motion is minimal.

racy force control, which would make the handling of contacts easier and safer.

This thesis aims to provide a framework for reliable contact-rich manipulation by resolving the above difficulties. Unlike the traditional top-down approach which solves for a high level motion plan before worrying about its execution, this thesis presents a bottom-up approach as detailed below.

1.1 Outline: A Bottom-Up Approach to Robust Manipulation

A pipeline of manipulation usually starts with the perception of the system state, followed by computing a motion plan that describes the expected robot and object(s) motion, and ends with the execution of the motion plan under some low-level control law. However, if a motion plan is fragile and prune to disturbances, then its execution cannot be reliable no matter what controller the robot uses. Before solving the motion planning problem, we must understand what kind of motion can be executed reliably by the robot. With this philosophy, we present our solution in the following steps:

1. We start in Chapter 3 by introducing our quasi-static modeling for a manipulation system, including the object(s), the robot, the environment, and their contacts. We discuss the types of failures the contact may introduce to manipulation, including kinematic ill-conditioning and unexpected contact mode switching. The analysis demonstrates the benefits of modeling the robot compliance using *hybrid force-velocity control*.
2. In Chapter 4, we present two algorithms to find the best hybrid force-velocity control to make a manipulation robust to kinematic ill-conditioning. The algorithms are tracking

controllers that can be used to execute a pre-computed motion plan.

3. In Chapter 5 we provide a thorough geometrical analysis that shows how contact mode transitions happen. We propose controllers build upon Chapter 4 that can additionally maintain the desired contact mode. We design metrics to quantify the robustness of contact modes, and use them in high level planning to help generate motion plans that are robust to execute.
4. As an extension, in Chapter 6, we explain how to generalize our analysis and methods in Chapter 5 from planar problems to 3D manipulation problems. We reformulate the algorithms to handle polyhedral computation in high dimensional space and reduce unnecessary computations.

We review related literatures in Chapter 2, and conclude the thesis in Chapter 7.

Chapter 2

Related Work

2.1 Mechanics of Rigid Bodies in Contacts

2.1.1 Frictional Contact Modeling

Friction determines the interaction between robots and grasped objects in sliding [13, 16, 48]. The tribology community has extensive researches on precise friction modeling [5]. Static friction models treat friction as a memoryless function of contact normal force, contact sliding velocity and external force [61]. More detailed static friction phenomena and modeling can be found in [61].

The discontinuity and lack of expressiveness of static friction models motivates dynamic friction modeling [5, 23, 61], which provides smooth friction behavior even during friction direction transitions. Dynamic friction models use one or more hidden state variables to describe microscopic asperities in contact [5, 23]. Complicated friction models provide a more precise description of friction phenomena. The cost is more effort and more data required for parameter estimation.

2.1.2 Planar Pushing

Planar pushing studies the motion of a rigid object under face-to-face contact with a frictional support under gravity. Mason [51] was the first to analyze the mechanics of pushing and proposed the voting theorem to determine the sense of rotation given a push action and the center of pressure. Peshkin and Sanderson [64] give further bounds on the possible instantaneous motion. Goyal et al. [27] noted that all the possible static and sliding frictional wrenches, for any friction distribution, form a convex set whose boundary is termed as limit surface. Howe and Cutkosky [39] proposed to use ellipsoid approximation of the limit surface for a given pressure distribution. Zhou et al. [96] proposed a framework of representing limit surfaces using homogeneous even-degree convex polynomials with stochastic extension in [97]. It's also possible to do planar pushing in a prehensile manner. For example, Kao and Cutkosky [42] used a soft hand to rotate a paper card on a table by pressing on its top with two fingers.

Closely related to planar pushing, *prehensile pushing* studies the in-hand motion of a grasped object in the gravity plane driven by external contacts [16, 17]. For both pushing and prehensile

pushing, the feasible finger velocity directions are within a cone [18]. Similar to pushing, *pulling* analyzes the planar motion of an object driven by a single force whose direction is not confined within a friction cone [40]. Unlike pushing, pulling is a converging process.

Pushing, prehensile pushing and pulling have two nice properties. No matter in what direction the robot moves, 1) the object is always in static balance as long as the robot motion is slow; 2) the robot motion will not cause excessive contact forces (we discussion this phenomenon in 3.5.1). The two properties makes planar pushing and pulling stable and safe.

2.1.3 Extrinsic Dexterity

When the robot body can not provide the necessary contact forces for a manipulation task, the robot can try to utilize *extrinsic dexterity* [20], meaning external force resources. The force resource maybe contact-free, such as gravity and inertia forces: Brock [13] calculated possible slipping motion for an object in a multi-fingered hand under gravity by maximizing the virtual work. Rao et al. [69] studied how to grasp a polytope object so that after lifting up, the object rotates to a desired stable pose under gravity. Holladay et al. [33] extended [69] by planning an open loop trajectory for the gripper with the consideration of dynamics, and utilizing contact with the ground to discretize the final poses. It is possible to rotated a grasped object in-hand in the gravity plane by regulating gripping forces [87][85][77]. In-hand manipulation can also be done with inertia force by executing an acceleration profile [75] or feedback control [36].

External contacts are another type of extrinsic force resources. By pressing a grasped object against a support, the robot can do in-hand manipulation [82][16], reorient the object to a desired pose [33][37] or transport the object without bearing its full weight by rotating it about one of its corners [1][93] or edges [74]. These work computed the conditions of maintaining static balance throughout the motion, including minimal friction coefficient (all contacts were sticking) and range of finger locations. In a sense, pushing, prehensile pushing and pulling (2.1.2) also belong to this category.

Most of these work focus on one or two particular skills. Dafle et al. [20] demonstrated that a simple robot hand can do many tasks by sequencing multiple skills with external force resources, although each of the skills were designed manually. This thesis aims to provide the tools for analyzing and planning contact-rich skills.

2.2 Control of Manipulation under Contacts

Manipulation control under external contacts is difficult, because a controller designed for one contact mode can be catastrophic for another mode. Existing control methods usually focus on a particular type of manipulation system, which is marked by one or a few relevant contact modes.

2.2.1 Control of Pushing

Planar pushing model commonly has uncertainties due to the varying surface property on the support. There are in general two approaches to handle the uncertainty.

One approach is to use an edge pusher instead of point pusher in open loop control for uncertainty reduction. Lynch and Mason [47] gave results on controllability and stability for open loop edge pushing. The multiple constraints imposed by an edge can be utilized for uncertainty reduction include planning push-grasps [14], parts reorientation [2] and feeding [92]. The first closed-loop control for pushing was proposed in [48] where the orientation of object was controlled to achieve stable translation, using a round finger with only tactile sensing.

The other approach is to use feedback and close the loop. Hogan and Rodriguez [31] proposed using model predictive control for planar pushing with all but one prescribed mode sequence. Zhou et al. [98] also addressed model identification and adaptation for single point planar pushing, as well as providing a feedback controller based on fast re-planning.

2.2.2 Mode Scheduling and Trajectory Stabilization

To control a process that goes through multiple contact modes, it is popular to do control synthesis for each continuous mode, then switch to the right controller at mode switching. Work on both planning and control usually do mode scheduling (either planned or hand-coded). Examples include legged locomotion [81][66][65], planar pushing with both sticking and sliding pusher [31] and half-cylinder flipping [30]. If the detection of mode switching is fast enough, mode scheduling may not be necessary [36].

2.2.3 Compliance/Impedance Control

Surface finishing tasks like polishing and deburring require the robot end-effector to move along a surface. Peg-in-hole assembly involves the object moving along the edges of the hole. In both cases, the control of the robot compliance can help avoid penetration and excessive contact force. People have been using hybrid force-velocity control (HFVC) for these tasks since 1980s, when Mason [50] introduced a framework for identifying force and velocity controlled directions in a task frame given a task description. Raibert and Craig [68] completed the framework and demonstrated a working system. Yoshikawa [94] investigated hybrid force-velocity control in joint space under Cartesian space constraints, and proposed to use gradient of the constraints to find the normal of the constraint surface in the robot joint space. For the control of a manipulator subject to constraints, it was common to align the force and velocity control directions with the row and null space of the contact Jacobian [89].The approach has industrial applications including polishing [60] and peg-in-hole assembly [62]. There are also works on modeling the whole constrained robot system using Lagrange dynamics, such as analyzing the system stability under hybrid force-velocity control [53], or performing Cartesian space tracking for both positions and forces [54][89].

However, when the system contains one or more free objects with no attachment to any motor, most previous work was case by case study, such as control of pivoting and tumbling [24][1][93][84][74]. In some special cases, it is possible to design a HFVC from simple heuristics using local contact information, such as in multi-finger grasping or finger gaiting [13, 59] and locomotion [15][26]. Uchiyama and Dauchez [84] performed hybrid force-velocity control for a particular example: two point manipulators contacting one object. In this thesis, we consider

the general case when the object may have multiple contacts with both the robot hand and the environment.

There are lots of works on how to implement hybrid force-velocity controls on manipulators. Velocity control is essentially a high stiffness control; force control can be implemented by low stiffness control with force offset. Salisbury [71] described how to perform stiffness control on arbitrary Cartesian axes with a torque-controlled robot. Raibert and Craig [68] divided Cartesian space into force/velocity controlled parts, then controlled them with separated controllers. The impedance control [32] and operational space control [43] theory analyzed the force related behaviors of the end-effector for torque-controlled robots. Maples and Becker [49] described how to use a robot with position controlled inner loop and a wrist-mounted force-torque sensor to do stiffness control on Cartesian axes. Lopes and Almeida [46] enhanced the impedance control performance of industrial manipulators by mounting a high frequency 6DOF wrist. Whitney [90] and De Schutter et al. [21] provided overviews and comparisons for a variety of force control methods.

2.3 Motion Planning of Manipulation under Contacts

The contacts impose constraints to the object and hand motion. There have been methods for motion planning under constraints such as the constraint-based motion planning and state estimation framework [22], and motion planning on the reduced manifold Berenson et al. [10]. Most difficulty of planning through contacts is in the change of contacts, which is a discontinuous process. Gradients do not exist at the making or breaking of contacts, thereby making powerful continuous solvers unusable. A popular way around is to model the contacts by complementary constraints on continuous variables [65][78]. However, these constraints make the optimization problem close to ill-condition and sensitive to the choice of initial trajectory. Another direction is to approximate the contact dynamics with continuous models [55][56][83]. So far there have been limited successes [57][30] in transferring motions planned with simplified dynamics into experiments. Sampling based planning methods handle discrete states naturally. They find solutions by solving steering problems under contact constraints [88][17], or by projecting the sampled motion into the constrained manifold [10].

A key to successful experiments is the ability to handle uncertainties. Object pose uncertainties can be reduced by contacts [44][97][76]. However, there has been little work on making contact modes robust against modeling uncertainties and force disturbances. This thesis provides analysis tools for this purpose.

In recent years, learning based methods have also enabled robots to do manipulation tasks with external contacts, such as opening doors [29] and pushing in clutter for grasping [25][95]. However, the success of robot learning relies on the assumption that the policy can be improved by random sampling. This only holds for tasks that are intrinsically stable, meaning that temporarily falling into a bad contact mode will not fail the task.

Chapter 3

Modeling And Failure Mode Analysis

Modeling of mechanics phenomenon could be complex and delicate. For example, tribology is a field focusing on the frictional behavior between two contacting bodies. A friction model for a point contact in tribology may include eight parameters and five dimensional states [5]. Such modeling is necessary for tire or gear analysis, however, it is excessive for robotics because the system identification, state estimation, motion planning and control problems would become intractable.

Our modeling for manipulation draws lessons from George Box's famous aphorism: *"All models are wrong, but some are useful"* [12]. Instead of pursuing an ultra precise frictional contact model, we found it more practical to adopt a simple static friction model that enables robust planning and control methods, then rely on the robustness of those methods to handle modeling uncertainties including unexpected dynamic frictional behavior. Similarly, we only consider point contacts in the thesis. Other types of contact geometry, such as edge contacts and face contacts, can be approximated by multiple point contacts.

In the following sections, we introduce key concepts in our modeling of manipulation systems and the assumptions we made. We discuss the causes of failures in manipulation and motivate our choice of robot compliance control. We follow these conventions throughout the following chapters.

3.1 Robot, Object, and Environment

This thesis focuses on manipulation of rigid bodies. In other words, we assume the robot, the object and the environment will not deform significantly. This is common in applications where the object is made of metal, wood, plastic, or sometimes card boxes.

We assume the state of the system is known, which includes the pose of the objects and the configuration of the robot. The object could make contacts with both the robot and the environment, and we assume the locations of those contacts are also available.

Real world manipulation problems have many sources of uncertainties. The object pose may come from an inaccurate perception system; object shape may have error from poor measurement or deformation; robot end-effector location could be off due to deformation of the structure; all these factors could also affect the contact point position estimation. For all the modeling

information, we assume the nominal value has uncertainties.

For simplicity, we separate the manipulation problem from the control problem of the manipulator itself. We ignore the kinematics and dynamics of the robot and model it as one piece of rigid body that can move freely. The actual robot could be a serial robot arm, an XYZ station, or even a mobile robot, however, what matters to the object motion is only the list of contact points with the object and the motion of these contacts. In the case of multi-finger manipulation, we assume the fingertips contacting the object at the same time have no relative motion and treat those fingertips as one rigid body. Note that this assumption does not exclude making and breaking contacts, e.g. finger gaiting. We only need to redefine the hand rigid body after a contact change.

3.2 Contact Modeling

We consider point contact with clearly defined contact point location and contact normal. This is the case for point-to-face contacts. Edge-to-edge, edge-to-face, and face-to-face contacts can be approximated by one or more point contacts [16]. Point-to-point and point-to-edge contacts are not considered here due to their rare appearance.

We adopt Coulomb friction model and assume the magnitude of sliding friction is the same as stiction. The friction coefficients of all contacts are known. Maximum-dissipation principle holds, i.e. sliding friction force of a point contact is in the opposing direction of the sliding velocity.

We consider three types of contact modes: sticking, sliding, and separation. Both sticking and sliding contacts impose a linear constraint in the contact normal direction; a sticking contact also impose constraints in the contact tangential directions.

3.3 Quasi-static Mechanics

This thesis adopts quasi-static modeling, i.e. the system motion is slow so that all inertia forces could be ignored. Using quasi-static mechanics, we do not need to model the system velocity and acceleration, which reduces the dimensionality of the manipulation problem. Technically, the modeling only works for slow motion where the inertia forces are low. However, in many cases the system motion does not need to be slow because:

1. The objects are light-weighted. So its motion is dominated by contact forces.
2. The robot velocity/position controller has low tracking error, so that the robot dynamics can be safely ignored. This is the common case for most industrial robots.
3. The manipulation controller is robust and can tolerate inertia forces as disturbances.

Consider a robot and at least one object in a rigid environment. The robot, object(s), and the environment has n_a, n_u, and zero degree-of-freedoms (DOF), respectively. The total DOF of the system is $n = n_a + n_u$. The subscript 'a' and 'u' means 'actuated' and 'unactuated'. Denote $v = [v_u^T \; v_a^T]^T, f = [f_u^T \; f_a^T]^T \in \mathbb{R}^n$ as the generalized velocity and force vectors.

Contact points introduce linear constraints on the contact point velocity, which further translates to linear constraints on the system generalized velocity v. They are linear constraints on the

system velocity.

$$\mathbf{J}v = 0. \tag{3.1}$$

$$\mathbf{M}v \leq 0. \tag{3.2}$$

$\mathbf{J}$ and $\mathbf{M}$ are the contact Jacobians [59]. A sticking contact contributes equality constraints in the contact normal and tangential directions. A sliding contact contributes equality constraint in the contact normal direction. If the sliding direction is specified, it is an inequality constraint in the contact tangential direction. A separating contact point only contributes an inequality constraint in the contact normal direction.

Denote λ as the vector of contact forces. Using the principle of virtual work [86], we can write the contribution of λ to the generalized force space as $\tau = \mathbf{J'}^T\lambda$. Note the $\mathbf{J'}$ here is different from the $\mathbf{J}$ in (3.4), because $\mathbf{J'}$ may have more rows that correspond to sliding friction depending on the contact mode. The forces in the system is governed by the Newton's Second Law:

$$\mathbf{J'}^T\lambda + f + F = 0. \tag{3.3}$$

The three terms are contact forces λ, control actions (internal forces) $f \in \mathbb{R}^n$ and external forces $F \in \mathbb{R}^n$, respectively. The external force F may include gravity, disturbance forces, *etc*. In this formulation, the unactuated generalized force f_u is zero. However, there are places in the thesis where we represent gravity in f_u instead of F for convenience.

3.4 Control of the Robot Manipulator

In the Newton's Law (3.3), robot control is described by a vector of forces. This is a potential source of error because most robot manipulators are not capable of accurate force control. Two notable exceptions are direct-drive robots [6] and robots with Serial-Elastic Actuators (SEA) [67], which are not commonly seen in off-the-shelf robot manipulators.

In this section we first discuss the nature and limitation of several typical robot control schemes and establish a realistic expectation of their performance. Then we explain our choice of a compliance control method.

3.4.1 Velocity (Position) Control

Velocity control and position control are the most common control schemes provided by industrial robots. Both are high stiffness controls that can be implemented by closing a loop on acceleration or motor torque. They are equivalent from the modeling perspective: we can implement velocity control by sending incremental positional commands, or implement position control by closing a PID loop on a velocity-controlled robot.

To be concise, although all the controllers proposed in this thesis can be implemented on either position-controlled or velocity-controlled robot, we will only use the term "velocity control" in derivations.

Velocity control is popular because of its high precision. Industrial robots under velocity control typically use high stiffness so as to overcome resistance forces from the load, joint friction,

and any other unmodeled disturbance forces while maintaining the commanded speed. Typical industrial robots have a repeatability on the order of 0.01mm, which is more than enough for our manipulation tasks.

As a result, in this thesis we treat a robot as a perfect source of velocity, meaning the robot can achieve any velocity command; the velocity control loop can generate arbitrarily large forces when necessary.

Considering its high stiffness, the velocity control needs to be modeled as a constraint on the robot velocity:

$$\mathbf{C}v = b_C, \tag{3.4}$$

where $\mathbf{C}$ is a matrix describing the directions of velocity control, b_C is a vector describing the magnitude of velocity control in those directions. Remember $v = [v_u^T \ v_a^T]^T$, so the first n_u columns of $\mathbf{C}$ must be zeros since the robot control cannot directly affect the free objects. In traditional velocity control, $\mathbf{C}$ has n_a rows to control the velocity of the robot in every possible direction. This is the case in applications like welding.

3.4.2 Force Control

Traditional mechanical analysis in Newton and Lagrange style assumes force or torque as robot action. It is easier to derive the dynamic model of the whole robot system using force or torque as robot control.

Although conceptually simple, force control is difficult to implement on real hardware. Off-the-shelf robot manipulators usually do not come with force control ability. Almost all industrial robots use high reduction transmission which brings large, nonlinear friction in the joints, making it impossible to estimate force precisely from motor current. People implemented force control on industrial robots by installing force/torque sensors in the joints or on the wrist, and close the force control loop outside. For a detailed list of references see Section 2.2.3. However, the "fake" force control has a bandwidth limited by the position control bandwidth. For those reasons, we must consider the inaccuracy when using force control.

Force control can be used to maintain contacts or avoid excessive contact forces in problems with geometrical uncertainties. Unlike velocity control, force control is not good at moving the robot or objects precisely. Additional outer loop needs to be closed on position/velocity feedback, which increases system complexity; its performance is also worse than the off-the-shelf velocity control since the force control bandwidth is much lower than the voltage/current control bandwidth used by velocity control.

Low-stiffness force control does not need to be modeled as a velocity constraint. Matrix $\mathbf{C}$ in Equation (3.4) for a force control has zero rows.

3.5 Causes of Failures in Manipulation

With all the modeling choices made, now we are ready to discuss the reasons why a manipulation could fail during execution. The uncertainties in modeling brings two potential problems: kinematic ill-conditioning and unexpected contact mode switching.

3.5.1 Kinematic Ill-Conditioning (Crashing)

The system generalized velocity is subject to two equality constraints: the contact constraints (3.1), and the velocity control constraints (3.4). The two constraints are not necessarily compatible with each other. For example, consider a manipulator whose finger is touching a rigid wall. If the velocity command is to move the finger towards the wall, this command is conflicting with the contact constraint from the wall. The situation breaks our modeling assumptions: the wall may deform and break the rigid body assumption, or the robot may stop moving under force limit and break the velocity constraint. Due to the high stiffness of the robot and the environment, a tiny robot motion can cause huge internal force to the system. We call this situation *crashing*.

During crashing, the two affine constraints (3.1) and (3.4) are infeasible. If we treat the robot velocity command as a kinematic constraint, then crashing means the system is kinematically ill-conditioned. This implies using the condition number of the combined affine system to evaluate the seriousness of ill-conditioning, which will be discussed in detail in the next chapter.

Jamming and wedging are two examples of crashing in mechanical assembly, which was analyzed in detail by Whitney [91]. Roughly speaking, jamming means a sliding contact turns into sticking unexpectedly. Wedging means two point contacts between an object and the environment form a force closure and fully immobilize the object. In both cases, the robot motion is interrupted and stopped by the contact constraints.

From the force balance perspective, a necessary condition of crashing is the existence of force balance at infinite force magnitude. When analyzing the force condition of crashing, finite forces (such inertia force, gravity, etc) should be ignored.

3.5.2 Unexpected Contact Mode Switching

A change of contact mode means a sudden change of system constraints. If the change is unexpected, the original robot control is unlikely to work under the new constraint. The task may fail immediately, for example, if a grasped object slips away between fingers. Or the contact mode change could cause a crashing, such as jamming in peg-in-hole assembly. To successfully execute a task, it is important to maintain desired contact modes even under real world uncertainties and disturbances. We provide methods to avoid unexpected contact mode switching in Chapter 5.

3.6 Hybrid Force-Velocity Control

Hybrid force-velocity control (HFVC) is a control technique that performs velocity control and force control in different directions simultaneously. With a properly designed HFVC, we could enjoy the benefits of both force and velocity control, which could help avoid the two failure modes. Specifically, using HFVC has the potential of the following benefits:

- **Robustly avoid kinematic ill-conditioning** using the force control component. By replacing some control DOFs to force control, we reduce the number of rows in the velocity control constraint (3.4), it is less likely to conflict with the contact constraint. In fact, we are able to maximize the robustness of a HFVC against crashing by picking proper force

controlled directions. In other words, the force control portion makes our robot resilient to geometrical uncertainties in the manipulation problem.

- **Robustly maintain the desired object motion** or system velocity under force disturbances. We can design the HFVC such that the velocity control portion together with the contact constraints fully ensure the desired velocity. This way the velocity control would generate whatever force that is necessary to overcome the resistance. In other words, the velocity control portion makes the manipulation robust to disturbance forces.
- **Construct a quasi-static problem**. Since the object motion is determined by the robot velocity control commands, the manipulation system becomes a kinematic system as long as the contact modes are maintained. The system is easier to analyze.
- **Requirement on force control accuracy is low**. For the same reason as above, small variations in robot force control or contact forces will not bring any change of velocity to the system. There is no need for an additional control loop to balance different force sources, which is the case in some multi-finger force controlled dexterous hands.
- **Robustly avoid unexpected contact mode switching**. The velocity control portion of HFVC can be designed to eliminate some undesired contact modes, while the force control portion can be used to select the desired mode among the rest. This is detailed in Chapter 5.

3.6.1 Quasi-Static Modeling of HFVC

We describe a hybrid force-velocity control as follows. Consider a HFVC with n_{av} dimensions of velocity control and n_{af} dimension of force control, $n_{av}+n_{af}=n_a$. We use matrix $\mathbf{T}\in\mathbb{R}^{n\times n}$ to describe the directions of force/velocity control. $\mathbf{T}=diag(\mathbf{I}_u,\mathbf{R}_a)$, where $\mathbf{I}_u\in\mathbb{R}^{n_u\times n_u}$ is an identity matrix, $\mathbf{R}_a\in\mathbb{R}^{n_a\times n_a}$ is an unitary matrix describing the control axes. Here we assume $\mathbf{R}_a$ is orthonormal, so that the force and velocity controls are reciprocal. Without loss of generality, we assume the last n_{av} rows of $\mathbf{T}$ are velocity-controlled directions, preceded by n_{af} rows of force-controlled directions.

Denote $w=\mathbf{T}v, \eta=\mathbf{T}f\in\mathbb{R}^n$ as the *transformed generalized velocity* and the *transformed generalized force*. We know $w=[w_u^T\ w_{af}^T\ w_{av}^T]^T$, where $w_u=v_u$ is the unactuated velocity, $w_{af}\in\mathbb{R}^{n_{af}}$ is the velocity in the force-controlled directions, $w_{av}\in\mathbb{R}^{n_{av}}$ is the velocity control magnitude. Similarly, $\eta=[\eta_u^T\ \eta_{af}^T\ \eta_{av}^T]^T$, where $\eta_u=f_u=0$ is the unactuated force, $\eta_{af}\in\mathbb{R}^{n_{af}}$ is the force control magnitude, $\eta_{av}\in\mathbb{R}^{n_{av}}$ is the force in the velocity-controlled directions. With our HFVC representation, $\mathbf{C}$ in Equation (3.4) is the last n_{av} rows of $\mathbf{T}$, and b_C is simply w_{av}.

To fully describe a HFVC, we need to solve for $n_{av}, n_{af}, \mathbf{R}_a, w_{av}$ and η_{af}.

3.6.2 Relation with Other Force Control Schemes

Our quasi-static modeling of HFVC hides the details of the low-level implementation. In fact, we do not require a specific type of implementation, we only require the robot to be able to apply different stiffness in different directions, and:

1. In the high stiffness directions, the control can be modeled as a velocity constraint.
2. In the low stiffness directions, the robot could apply force control.

Beside the original HFVC implementation [68], the robot can also do admittance control with force offset, or impedance control. Our quasi-static modeling could apply as long as the above conditions are met.

Chapter 4

HFVC with the Optimal Conditioning[1]

In this chapter, we provide methods to compute the HFVC with the best conditioning for a manipulation problem. To make a complete robust control algorithm, the algorithms also include methods to avoid unexpected contact modes. We call the control problem *hybrid servoing*, which is introduced formally in Section 4.2.

The algorithms for hybrid servoing in this chapter are limited to executing sticking contacts. In Chapter 5 we will present a more elaborate framework to maintain desired contact modes that is built upon algorithms presented in this Chapter.

We use the notation $\mathbb{NULL}(\cdot)$ and $\mathbb{ROW}(\cdot)$ to denote the null space and row space of the argument, respectively; use $\mathrm{Null}(\cdot)$ and $\mathrm{Row}(\cdot)$ to denote a matrix whose *rows* form an orthonormal basis of $\mathbb{NULL}(\cdot)$ and $\mathbb{ROW}(\cdot)$, respectively. We use $\mathrm{rows}(\cdot)$ to denote the number of rows in the argument.

First, we discuss how to quantify the kinematic conditioning of a manipulation system under HFVC.

4.1 Evaluate Kinematic Conditioning of a Manipulation System

In manipulator kinematic analysis, it is well-known that the condition number of the manipulator Jacobian is an indicator of the kinematic performance of the system [3][4][73]. In a manipulation problem with free objects, we need to consider two kinematic constraints including (3.1) and the velocity control portion of HFVC (3.4). We need to evaluate the conditioning of the whole kinematic system:

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{C} \end{bmatrix} v = \begin{bmatrix} 0 \\ b_C \end{bmatrix}. \tag{4.1}$$

[1]This chapter uses materials previously published in *Robust Execution of Contact-Rich Motion Plans by Hybrid Force-Velocity Control*, Yifan Hou and Matthew T. Mason, ICRA 2019, and *An Efficient Closed-Form Method for Optimal Hybrid Force-Velocity Control*, Yifan Hou and Matthew T. Mason, ICRA 2021

The condition number of the coefficient matrix needs to be minimized:

$$\min_{\mathbf{J},\mathbf{C}} \mathrm{cond}(\begin{bmatrix} \mathbf{J} \\ \mathbf{C} \end{bmatrix}). \tag{4.2}$$

Throughout this section, we use the 2-norm condition number, defined as

$$\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^\dagger\|_2 = \frac{\sigma_{\max}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}, \tag{4.3}$$

which is the ratio between the maximum and minimum singular values.

However, for our system, directly computing the above condition number makes little sense for two reasons. First, we only want to evaluate the influence of control $\mathbf{C}$, the rest entries of our coefficient matrix are constants. In fact, $\mathbf{J}$ itself could already be ill-conditioned if the contact modeling is redundant. To singulate the influence of $\mathbf{C}$, we replace $\mathbf{J}$ with an orthogonal basis of its rows, so it represents the same constraint as (3.1) but has a condition number of one. Second, the row scaling of $\mathbf{C}$ should not affect our criteria, since scaling both sides of (3.4) does not make a difference to our control. However, the condition number could be made arbitrarily large by row/column scaling. This problem is called the *artificial ill-condition*[79], the typical solution is to pre-normalize each row of our coefficient matrix. Thus our final expression of kinematic conditioning is:

$$\min_{\mathbf{C}} \mathrm{cond}(\begin{bmatrix} \hat{\mathbf{J}} \\ \hat{\mathbf{C}} \end{bmatrix}), \tag{4.4}$$

where rows of $\hat{\mathbf{J}}$ form an orthonormal basis of rows in $\mathbf{J}$; $\hat{\mathbf{C}}$ is $\mathbf{C}$ with each row normalized to unit norm. Figure 4.1 shows the condition number value of several planar examples. When the control is collinear with constraints, the condition number grows to infinity and a tiny motion can cause huge internal force. We have been calling this situation *crashing* in our previous work, and introduced a "crashing-avoidance score" in [34] to evaluating it. However, equation (4.4) is a more precise description, we call the cost function the *crashing index*.

4.2 The Hybrid Servoing Problem

We define the *hybrid servoing* problem as computing the best HFVC for a manipulation problem [35]. A hybrid servoing algorithm works as a tracking controller for executing a pre-computed motion plan. In hybrid servoing, we optimize the crashing index (4.4) defined above, while satisfying several constraints as detailed below in Section 4.2.1 and 4.2.2. We conclude this section with the complete problem formulation in Section 4.2.3.

4.2.1 Goal Description

Users of hybrid servoing should describe the desired motion. At a time instant, the desired velocity can be written as an affine constraint on the system generalized velocity:

$$\mathbf{G}v = b_G. \tag{4.5}$$

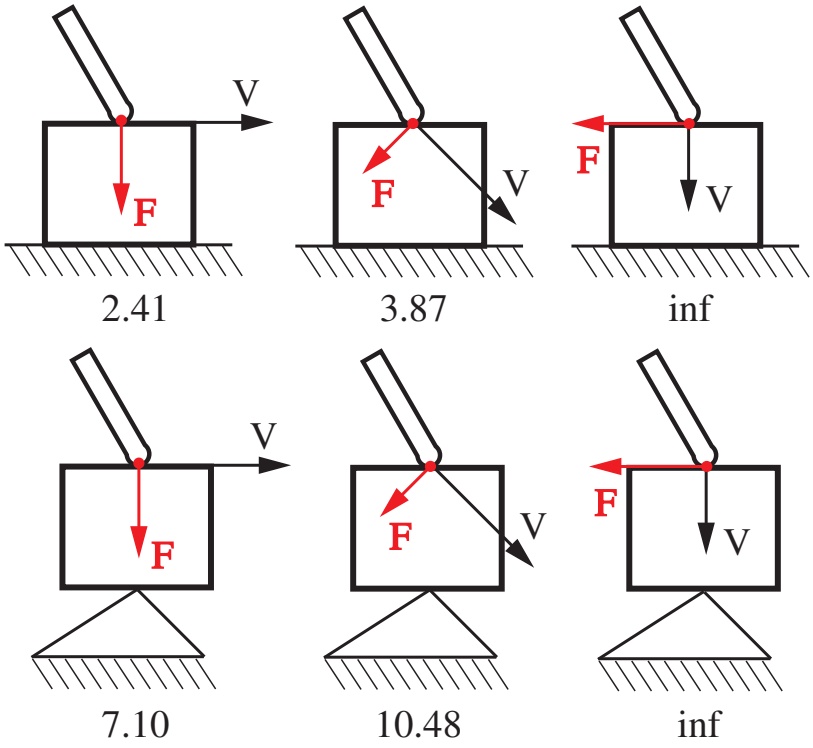

Figure 4.1: 2D examples of HFVC and their corresponding crashing indexes. The robot execute 2D HFVC, with 1D force control and 1D velocity control.

For example, the user may supply a coefficient matrix $\mathbf{G}$ and vector b_G with six rows to fully specify the desired velocity of a 3D rigid body, or only use three rows to specify its rotational velocity. The coefficients $\mathbf{G}, b_G$ can be derived from a given trajectory by taking first-order derivative. For example, denote $v \in \mathbb{R}^6$, $g \in \mathbb{R}^{4\times4}$ as the spatial velocity and pose of a 3D object, respectively. From the definition of spatial velocity [59] we know:

$$v^\wedge = \dot{g}g^{-1}, \tag{4.6}$$

where $v^\wedge$ denotes the wedge of v:

$$v^\wedge = \begin{bmatrix} 0 & -v_6 & v_5 & v_1 \\ v_6 & 0 & -v_4 & v_2 \\ -v_5 & v_4 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.7}$$

In other words,

$$v = (\dot{g}g^{-1})^\vee, \tag{4.8}$$

where $\vee$ denotes the inverse operation of wedge. At time step t, we can approximate $\dot{g}(t)$ with $(g(t+1) - g(t))/\delta t$, so the desired velocity of the object at time step t is:

$$v^*(t) = \big((g(t+1) - g(t))g^{-1}(t)\big)^\vee / \delta t. \tag{4.9}$$

Higher order approximations of $\dot{g}(t)$ could give higher precision, however, we find the forward Euler here to be accurate enough in our experiments.

The goal specification (4.5) must not be redundant with other constraints in the system. For example, to slide an object on a planar surface in 3D, the goal should have no more than three rows. It should not specify the object velocity in the contact normal direction, which is already limited to zero by the contact constraint. Given the active environmental contact Jacobian $\mathbf{J}_e$ at a time step t, we can populate $\mathbf{G}$ with the null space of $\mathbf{J}_e$ to avoid such redundancy:

$$\mathbf{G} = [0, ..., \mathrm{Null}(\mathbf{J}_e), ..., 0]\,. \tag{4.10}$$

Here $\mathrm{Null}(\mathbf{J}_e)$ only fills the entries that correspond to the object velocity, the rest entries in $\mathbf{G}$ are zeros. Then we can compute b_G by

$$b_G = \mathrm{Null}(\mathbf{J}_e)v^*(t). \tag{4.11}$$

4.2.2 Constraints on force

There are two kinds of force constraints. One is Equation (3.3), the Newton's Second Law under quasi-static approximation. The other force constraint is the condition for staying in the desired contact mode, we called them the *guard conditions* [35]. We borrow the term guard condition from the hybrid system theory, where it means the condition of discrete mode switching. It's usually a good practice to make the guard condition stricter than necessary to encourage conservative actions. We consider guard conditions that are affine constraints on force variables. Examples are friction cone constraints and lower/upper bounds on forces.

$$\mathbf{\Lambda} \begin{bmatrix} \lambda \\ f \end{bmatrix} \leq b_\Lambda. \tag{4.12}$$

Note that (4.12) has no equality constraints, so we don't consider sliding friction. This is because applying force on the friction cone is not a robust way to execute a sliding contact [34].

4.2.3 Problem Formulation

Now we can complete the definition of the hybrid servoing problem. The task of hybrid servoing is to solve for:

1. the dimensions of force-controlled actions and velocity-controlled actions, n_{af} and n_{av}, and
2. the directions to do force control and velocity control, described by the matrix $\mathbf{T}$, and
3. the magnitude of force/velocity actions: η_{af} and w_{av},

so as to minimize the crashing index (4.4) subject to the following constraints:

- Any v under the robot action shall satisfy the goal constraint (4.5);
- Any f under the robot action shall satisfy the guard conditions (4.12).

We use the word 'any' because a HFVC usually cannot uniquely determine v and f.

We do not consider avoiding unexpected new contacts in a hybrid servoing problem since it should be handled as collision avoidance in motion planning.

4.2.4 Algorithm Outline

Our approach to solve hybrid servoing has two steps. In the first step we solve for velocity controls, during which the dimensions and directions of both velocity and force controls are also determined. This step computes $n_{af}, n_{av}, \mathbf{T}$ and w_{av}. Then the only remaining unknown variable in a HFVC is the value of the force control magnitude η_{af}, which can be found by minimizing the magnitude of force variables:

$$\min_{\lambda,\eta} \lambda^T \lambda + \eta_a^T \eta_a \tag{4.13}$$

subject to the Newton's Second Law (3.3) and the guard conditions (4.12), which takes the form of a Quadratic Programming.

The challenge is how to compute the velocity control. In the following, we present two hybrid servoing algorithms that share the same force control steps as described above, but use different approaches to solve the velocity control. The method presented in Section 4.3 has an optimization formulation, which could incorporate more task specific cost terms. The method presented in Section 4.4 has a closed-form solution, which directly computes the HFVC with the best conditioning and is thus faster.

4.3 Algorithm One: Hybrid Servoing by Optimization

To solve for the velocity control, first let's look at the equations related to the system velocity:

- Contact constraint $\mathbf{J}v = 0$;
- Goal condition $\mathbf{G}v = b_G$;
- Velocity command $\mathbf{C}v = b_C$.

Denote the solution set of each equation above as $Sol(\mathbf{J}), Sol(\mathbf{G})$ and $Sol(\mathbf{C})$. We need to design the velocity command $\mathbf{C},\ b_C$ such that the resulted solution space (the solution set of natural constraints and velocity commands) becomes *a non-empty subset of the desired generalized velocities* (the solution set of natural constraints and goal condition):

$$Sol(\mathbf{J}\&\mathbf{C}) \ \in \ Sol(\mathbf{J}\&\mathbf{G}) \tag{4.14}$$

4.3.1 Determine dimensionality of velocity control

The first thing we can infer is the dimensionality of the velocity control. Denote $r_J = \mathrm{rank}(\mathbf{J}), r_{JG} = \mathrm{rank}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix})$. The minimum number of independent velocity control we must enforce is

$$n_{av}^{\min} = r_{JG} - r_J. \tag{4.15}$$

This condition makes sure the dimension of $Sol(\mathbf{J}\&\mathbf{C})$ is smaller or equal to the dimension of $Sol(\mathbf{J}\&\mathbf{G})$, so that their containing relationship becomes possible. The maximum number of independent velocity control we can enforce is

$$n_{av}^{\max} = n - r_J = \mathrm{Dim}(\mathbb{NULL}(\mathbf{J})), \tag{4.16}$$

This condition ensures the system will not be overly constrained.

In this section, we choose the minimal number of necessary velocity constraints:

$$n_{av} = n_{av}^{\min} = r_{JG} - r_J. \tag{4.17}$$

This choice gives the system more compliance. It also makes the force control problem easier to solve in Section 4.2.4.

4.3.2 Characterize the feasible velocity control directions

With our choice of n_{av}, we know $\mathrm{rank}([\mathbf{J};\mathbf{C}]) = \mathrm{rank}([\mathbf{J};\mathbf{G}])$. Then the condition $Sol(\mathbf{J}\&\mathbf{C}) \in Sol(\mathbf{J}\&\mathbf{G})$ implies

$$Sol(\mathbf{J}\&\mathbf{C}) \;=\; Sol(\mathbf{J}\&\mathbf{G}), \tag{4.18}$$

i.e. the two linear systems share the same solution space. (4.18) can be achieved by firstly choosing $\mathbf{C}$ such that the homogeneous linear systems $\begin{bmatrix}\mathbf{J}\\ \mathbf{C}\end{bmatrix} v = 0$ and $\begin{bmatrix}\mathbf{J}\\ \mathbf{G}\end{bmatrix} v = 0$ become equivalent (share the same solution space). We can ensure the equivalence in three steps. First, compute a basis for the solution space of $[\mathbf{J}^T\mathbf{G}^T]^T v = 0$: $[\sigma_1, ..., \sigma_{n-r_{JG}}]$. Second, ensure $\mathbf{C}$ satisfies:

$$\mathbf{C}\sigma_i = 0, \;\; i = 1, ..., n - r_{JG}. \tag{4.19}$$

And finally, compute b_C from any specific solution of $\{\mathbf{N}v = 0, \mathbf{G}v = b_G\}$. The original non-homogeneous systems now become equivalent.

Note that the condition in Equation (4.19) cannot uniquely determine the values in the matrix $\mathbf{C}$, and it leaves room for the crashing index to vary. We need to formulate an optimization problem to find the solution that minimize the crashing index while satisfying condition (4.19). However, the crashing index has the form of a condition number, whose gradient is not easy to obtain. In the following, we propose a polynomial approximation to the crashing index and optimize the approximation instead.

4.3.3 Polynomial Approximation to the Crashing Index

In the crashing index (4.4), the condition number is computed for a matrix whose rows have unit norms. Since the norms of the rows are fixed, what could influence the condition number is the linear dependency between different rows. Since the contact Jacobian $\hat{\mathbf{J}}$ is orthonormal and not part of the decision variables, we need to consider two kinds of linear independences:

- Each row of the velocity control $\mathbf{C}$ must not be a linear combination of rows of $\hat{\mathbf{J}}$. In other words, each row of $\mathbf{C}$ must contain some portion of the null space of $\hat{\mathbf{J}}$. They should be as close to $\mathbb{NULL}(\mathbf{J})$ as possible.
- Rows of the velocity control $\mathbf{C}$ must not be linearly dependent of each other. They should be as orthogonal to each other as possible.

When one of the linear dependency happens, the matrix loses rank and the condition number goes to infinity. The above preference is expressed in the following cost function:

$$\min_{\mathbf{C}=[c_1^T, c_2^T, ...]^T} \sum_{i \neq j} ||c_i^T c_j|| - \sum_i ||\mathrm{Null}(\mathbf{J})^T c_i||. \tag{4.20}$$

In this cost function we use $\mathbf{J}$ instead of $\hat{\mathbf{J}}$ for simplicity, since their null space is the same.

Previous work on hybrid force-velocity control dealt with fully-actuated systems, so they can choose the velocity commands from within the null space of the contact constraints $\hat{\mathbf{J}}$ [94] (Fig. 4.2, left). This is not possible for system with free objects.

We show the physical intuition behind the cost function in Figure 4.2 and Figure 4.3. The velocity controlled directions are preferred to overlap less with the natural constraints, more with its null space, so that the system of equations will be less likely to become infeasible under disturbances. An example is illustrated in Figure 4.2. Similar preference exists among different velocity commands. As shown in Fig. 4.3, different velocity controlled directions in the generalized velocity space are preferred to be more perpendicular to each other.

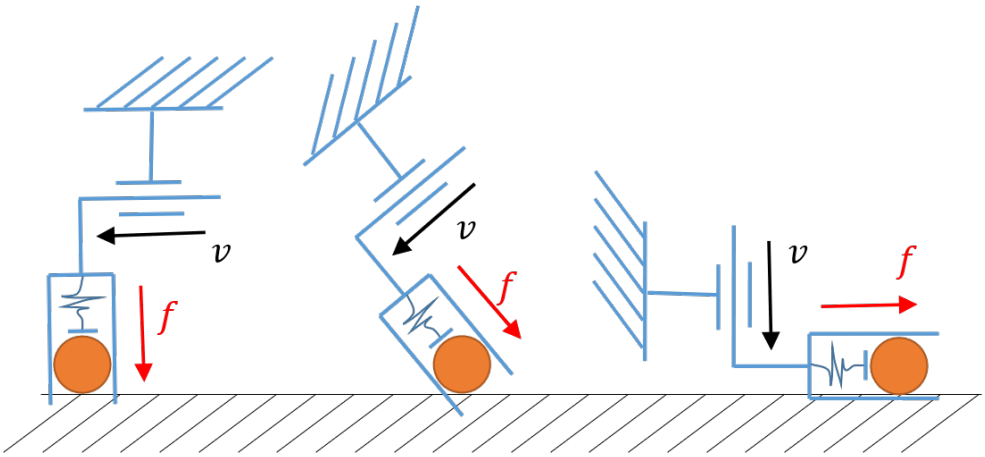

Figure 4.2: Relation between velocity commands and contact velocity constraints. The robot (blue) has two orthogonal translational joints, one force-controlled and another velocity controlled. The table provides a velocity constraint that stops the object from moving down. Assume no collision between the robot and the table. The systems in the left and middle figures are kinematically feasible. The system in the right figure is infeasible because the constraints on the object are ill-conditioned.

4.3.4 The Hybrid Servoing Algorithm

Now we formulate the complete optimization problem. Denote $c^T \in \mathbb{R}^{1\times n}$ as any row in $\mathbf{C}$. Rewrite equation (4.19) as a linear constraint on $\mathbf{c}$, also considering the first n_u columns in $\mathbf{C}$ are zeros:

$$\begin{bmatrix} \sigma_1^T \\ \vdots \\ \sigma_{n-n_{JG}}^T \\ \begin{bmatrix} \mathbf{I}_{n_u} & \mathbf{0}_{n_u\times n_a} \end{bmatrix} \end{bmatrix} c = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.21}$$

Its solution space has dimension of $n_c = n_a - (n - r_{JG}) = r_{JG} - n_u$. Since we need n_{av} independent constraints, we require $n_c = r_{JG} - n_u \geq n_{av} = r_{JG} - r_J$, which gives $r_J \geq n_u$, *i.e.*

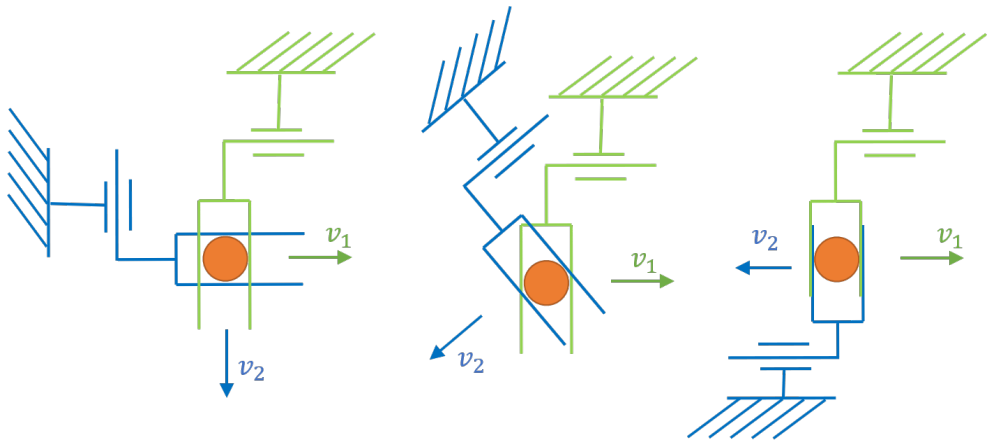

Figure 4.3: Relation between different velocity controls. The blue robot and the green robot are applying different velocity commands on the object. Assume no collision between the two robots. Systems in the left and middle figures are kinematically feasible. The on in the right figure is infeasible.

$$r_J + n_a \geq n. \tag{4.22}$$

For this algorithm to work, (4.22) says it must be possible for the actions and constraints to fully constrain the system. Denote matrix $\mathbb{B}_c = [c^{(1)} \ \cdots \ c^{(n_c)}]$ as a basis of the solution space of equation (4.21). Denote $\mathrm{Null}(\mathbf{J})$ as a basis of $\mathbb{NULL}(\mathbf{J})$. We can find a $\mathbf{C}$ that satisfies all the conditions by solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{k}_1,\cdots,\mathbf{k}_{n_{av}}} & \sum_{i\neq j} ||c_i^T c_j|| - \sum_i ||\mathrm{Null}(\mathbf{J})^{\mathbf{T}} \mathbf{c_i}|| \\ s.t. \quad & c_i^T c_i = 1, \quad \forall i \\ & c_i = \mathbb{B}_c \mathbf{k}_i, \quad \forall i \end{aligned} \tag{4.23}$$

Then $\mathbf{C}^* = (\mathbb{B}_c \ [\mathbf{k_1} \ \ ... \ \ \mathbf{k_{n_{av}}}])^T$. The optimization problem (4.23) is non-convex because of the norm constraint $c_i^T c_i = 1$. However, we can solve the problem numerically using projected gradient descent:

1. Start from a random $\mathbf{k} = [\mathbf{k_1} \ \ ... \ \ \mathbf{k_{n_{av}}}]$;
2. Perform a gradient descent step: $\mathbf{k} \leftarrow \mathbf{k} - t\nabla f$;
3. Projection: $\mathbf{k_i} \leftarrow \frac{\mathbf{k_i}}{||\mathbb{B}_c \mathbf{k_i}||}, \quad \forall i$;
4. Repeat from step two until convergence.

Here ∇f is the gradient of the cost reshaped to the same size as $\mathbf{k}$. t is a step length. In practice, we run the algorithm with N_s random initializations to avoid bad local minima.

After obtaining $\mathbf{C}^*$, we know the last n_{av} rows of $\mathbf{R}_a$. Denote the last n_a columns of $\mathbf{C}^*$ as $\mathbf{R}_{C^*}$, we can expand it into a full rank $\mathbf{R}_a$:

$$\mathbf{R}_a = \begin{bmatrix} \mathrm{Null}(\mathbf{R}_{C^*})^T \\ \mathbf{R}_{C^*} \end{bmatrix}, \tag{4.24}$$

it encodes the directions of force control. Then we have $\mathbf{T} = \mathrm{diag}(\mathbf{I}_{\mathrm{u}}, \mathbf{R}_{\mathrm{a}})$. The complete procedure is summarized in Algorithm 1.

Algorithm 1: Optimization-based Hybrid Servoing

```
Check condition (4.22) for feasibility;
Compute n_av from equation (4.17);
Compute a basis of [J^T G^T]^T v = 0, plug in equation (4.21) and compute a basis B_c;
Sample N_s sets of coefficients k ∈ R^{n_c × n_av};
foreach sample k do
    Solve the optimization problem (4.23);
    Compute C = (B_c k)^T from the solution;
    Compute the cost of C from equation (4.23);
end
Pick the C* with lowest cost;
Compute R_a from (4.24), then T = diag(I_u, R_a);
Compute one solution v* for Jv = 0, Gv = b_G;
Compute w_av = b_C = C* v*;
Compute the force control magnitude η_af by solving the QP defined in Section 4.2.4.;
```

4.4 Algorithm Two: Closed-Form Solution to Optimal Velocity Control

In this section, we present another method to compute the velocity control directions and magnitudes. The method is complete, in that it always produces an optimal/near optimal solution when a solution exists. It is efficient, since it is in closed form, avoiding the iterative search of our previous method. This closed-form method outperforms our previous method by being from 7 to 40 times faster, while consistently producing better solutions in the sense of robustness to kinematic singularity.

Before introducing our algorithm, we need to make some observations about the nature of the problem. Since the feasible velocity v under a HFVC may not be unique, the proper statement of the goal constraint is:

$$\mathbf{G}v = b_G, \ \ \forall v \in \{v \mid \mathbf{J}v = 0, \mathbf{C}v = b_C\}, \tag{4.25}$$

i.e. we need to ensure all possible solutions satisfy the goal. This is an inclusion relationship between the solution sets of two linear equations, which is equivalent to:

1. The null space of $\mathbf{G}$ contains the null space of $\begin{bmatrix} \mathbf{J} \\ \mathbf{C} \end{bmatrix}$;
2. There exists a common special solution: $\exists v^* : \mathbf{G}v^* = b_G, \mathbf{J}v^* = 0, \mathbf{C}v^* = b_C$.

We call them the *Goal-Inclusion conditions* and will refer to them repeatedly. Condition 1) is equivalent to

$$\mathbb{NULL}(\begin{bmatrix} \mathbf{J} \\ \mathbf{C} \end{bmatrix}) \subseteq \mathbb{NULL}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix}), \tag{4.26}$$

which further implies

$$\text{rank}(\begin{bmatrix} \mathbf{J} \\ \mathbf{C} \end{bmatrix}) \geq \text{rank}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix}). \tag{4.27}$$

Due to the orthogonal complement relation between the row and null space of a matrix, the null space inclusion (4.26) can be reformulated as a reverse row space inclusion:

$$\text{rank}\left(\begin{bmatrix} \mathbf{J} \\ \mathbf{C} \end{bmatrix}\right) = \text{rank}\left(\begin{bmatrix} \mathbf{J} \\ \mathbf{C} \\ \mathbf{G} \end{bmatrix}\right). \tag{4.28}$$

Our algorithm involves three steps. First, we derive the control axis directions to satisfy the Goal-Inclusion condition 1) while optimizing conditioning. Second, we compute the velocity control magnitudes to satisfy the Goal-Inclusion condition 2). Finally, we solve for the force control magnitudes to satisfy the guard conditions.

4.4.1 Pick Control Axes to Optimize Conditioning

The information of the control axes (n_{av}, n_{af}, and $\mathbf{T}$) is contained in the velocity control coefficient matrix $\mathbf{C}$ (3.4): $\mathbf{C}$ has n_{av} rows; $\mathbf{C}$ and its orthogonal complement forms $\mathbf{T}$.

First thing we need to know about the velocity control is its dimension. We can compute the instantaneous directions that the system can move without conflicting the contact constraints by computing the null space $\mathbf{U}$ of contact Jacobian:

$$\mathbf{U} = \text{Null}(\mathbf{J}). \tag{4.29}$$

The last n_a columns of $\mathbf{U}$, i.e. the actuated part, indicate the directions in which the robot can move freely. It is a linear space, a basis of which can be computed as:

$$\bar{\mathbf{U}} = \text{Row}(\mathbf{U}\mathbf{S}_a), \tag{4.30}$$

where $\mathbf{S}_a \in \mathbb{R}^{n \times n}$ is a selection matrix with only ones on the last n_a diagonal entries. Any linear combinations of rows of $\bar{\mathbf{U}}$ corresponds to a vector in $\mathbb{NULL}(\mathbf{J})$ and is thus a free robot motion direction. The dimensionality of $\bar{\mathbf{U}}$ indicates the maximum dimension of velocity control we can apply:

$$n_{av} \leq \text{rows}(\bar{\mathbf{U}}) \tag{4.31}$$

On the other hand, (4.27) suggests the minimum dimension of velocity control required to satisfy the goal (4.5):

$$n_{av} \geq \text{rank}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix}) - \text{rank}(\mathbf{J}). \tag{4.32}$$

Combining (4.31) and (4.32), we have a necessary condition for the feasibility of the problem:

$$\text{rows}(\bar{\mathbf{U}}) \geq \text{rank}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix}) - \text{rank}(\mathbf{J}). \tag{4.33}$$

If equation (4.33) is not satisfied, the problem has an infeasible goal. Otherwise, we can choose the dimension of velocity control within (4.31)-(4.32). Sometimes we want more velocity control so as to increase disturbance rejection ability [38]; sometimes we want less velocity control to have more compliance in the system [35]. We solve both situations and leave this choice to the user.

If the maximal velocity control is needed, we can simply do velocity control in all directions in $\bar{\mathbf{U}}$:

$$n_{av} = \text{rows}(\bar{\mathbf{U}}), \tag{4.34}$$

$$\mathbf{C} = \bar{\mathbf{U}}. \tag{4.35}$$

Then we check equation (4.28) to see if the problem is feasible. Otherwise, if the minimal velocity control is desired, we take

$$n_{av} = \text{rank}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix}) - \text{rank}(\mathbf{J}) \tag{4.36}$$

Then the n_{av} rows of velocity controls are linear combinations of rows of $\bar{\mathbf{U}}$:

$$\mathbf{C} = \mathbf{K}\bar{\mathbf{U}}, \tag{4.37}$$

where $\mathbf{K} \in \mathbb{R}^{n_{av} \times \text{rows}(\bar{\mathbf{U}})}$. We compute $\mathbf{K}$ using the null space form of Goal-Inclusion condition 1), which implies

$$\mathbf{C}\text{Null}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix}) = \mathbf{K}\bar{\mathbf{U}}\text{Null}(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix}) = 0. \tag{4.38}$$

Then $\mathbf{K}$ is an orthonormal basis of a null space:

$$\mathbf{K} = \text{Null}^T\left(\text{Null}^T(\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix})\bar{\mathbf{U}}^T\right). \tag{4.39}$$

The problem is feasible if $\mathbf{K}$ has enough rows:

$$\text{rows}(\mathbf{K}) \geq n_{av}. \tag{4.40}$$

If this is true, we keep the first n_{av} rows of $\mathbf{K}$ and recover $\mathbf{C}$ from equation (4.37). The $\mathbf{C}$ obtained this way has orthonormal rows, since it is the product of two orthonormal matrices.

Note that equation (4.35) and (4.37) compute the velocity control direction $\mathbf{C}$ in a closed form without explicitly optimize the crashing index (4.4), however, they do find optimal solutions. As shown in our numerical experiments, equation (4.37) always finds the solution with the minimal crashing index; equation (4.35) also always achieves the minimal crashing index among solutions with the same dimensionality.

After obtaining the velocity-controlled direction $\mathbf{C}$, we compute the force-controlled direction as its orthogonal complement to make the velocity and force controls reciprocal. Denote the last n_a columns of $\mathbf{C}$ as $\mathbf{R}_C$, we can expand it into a full rank $\mathbf{R}_a$:

$$n_{af} \quad = n_a - n_{av}. \tag{4.41}$$

$$\mathbf{R}_a = \begin{bmatrix} \mathrm{Null}(\mathbf{R}_C)^T \\ \mathbf{R}_C \end{bmatrix}. \tag{4.42}$$

Then we have $\mathbf{T} = \mathrm{diag}(\mathbf{I}_\mathrm{u}, \mathbf{R}_\mathrm{a})$.

We summarize the procedure in line 1 to line 13 in algorithm 2. Note that the method avoids the non-convex optimization in our previous method.

4.4.2 Solve for Velocity Control Magnitudes

Next, we use Goal-Inclusion condition 2) to compute b_C. Compute a special solution v^* from

$$\begin{bmatrix} \mathbf{J} \\ \mathbf{G} \end{bmatrix} v^* = \begin{bmatrix} 0 \\ b_G \end{bmatrix} \tag{4.43}$$

Such v^* must exist, otherwise the goal itself is infeasible. Use it to compute the velocity control magnitude:

$$w_{av} = b_C = \mathbf{C}v^* \tag{4.44}$$

This choice of b_C satisfies condition 2).

An important advantage of this algorithm over the first algorithm 1 is its ability to handle underactuated system. An example is a cube with one corner sticking on the ground and one

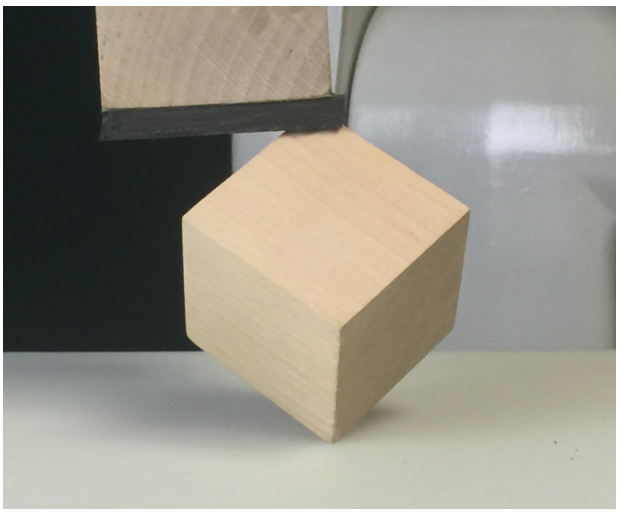

Figure 4.4: An underactuated example.

corner sticking on the robot finger (Figure 4.4): the robot has no control over the rotation of the cube about the line between the two contact points. Still, a control problem on this object may still be feasible, e.g. if the goal is to lift the center of mass of the object. Condition (4.28) tells us whether this is the case or not.

Algorithm 2: Optimally-Conditioned Hybrid Servoing (OCHS)

Input: Contact Jacobian J, Goal description G, b_G
Input: Guard condition,
Output: HFVC $(n_{af}, n_{av}, \mathbf{R}_a, w_{av}, \eta_{af})$
`// Solve for velocity control`
Compute free motion space $\mathbf{U}$ under constraint;
Compute free robot motion space $\bar{\mathbf{U}}$ (4.30) ;
Check necessary feasibility condition (4.33). ;
if *Maximal velocity control dimension* **then**
    Take $\bar{\mathbf{U}}$ as velocity-controlled directions (4.34)(4.35) ;
    Check goal feasibility using (4.28);
else
    Compute the minimal dimension of $\mathbf{C}$ from (4.36) ;
    Solve for the coefficient matrix $\mathbf{K}$ from (4.39) ;
    Check goal feasibility using (4.40) ;
    Compute the velocity control $\mathbf{C}$ from (4.37);
Complete control axes information using (4.41)-(4.42). ;
Compute the velocity control magnitude (4.44) using a special solution to (4.43). ;
`// Solve for Force control`
Compute the force control magnitude η_{af} by solving the QP defined in Section 4.2.4.;

4.5 An Example

We illustrate our method on the "block tilting" task shown in Fig. 4.5. The robot needs to flip a square block forward by pressing on its top. All contacts involved are sticking.

4.5.1 Variables

Denote W, H and O as the world frame, the hand frame and the object frame respectively. In the following, we use the form of ${}^A_B X$ to represent a symbol of frame B as viewed from frame A. The state of the system can be represented by the 3D pose of the object and the position of the hand: $q = [{}^W_O p^T,\ {}^W_O q^T,\ {}^W_H p^T]^T \in \mathbb{R}^{10}$. Define the generalized velocity for the system to be the object body twist ${}^O_O \xi \in \mathbb{R}^6$ and the hand linear velocity ${}^W_H v \in \mathbb{R}^3$:

$$v = [{}^O_O \xi^T,\ {}^W_H v^T]^T \in \mathbb{R}^9. \tag{4.45}$$

There is a linear relationship between the generalized velocity v and $\dot{q}$:

$$\dot{q} = \mathbf{\Omega}(q) v, \tag{4.46}$$

where

$$\mathbf{\Omega}(\mathbf{q}) = \begin{bmatrix} {}^W_O \mathbf{R} & & \\ & \mathbf{E}({}^W_O q) & \\ & & \mathbf{I}_H \end{bmatrix} \in \mathbb{R}^{10\times 9}. \tag{4.47}$$

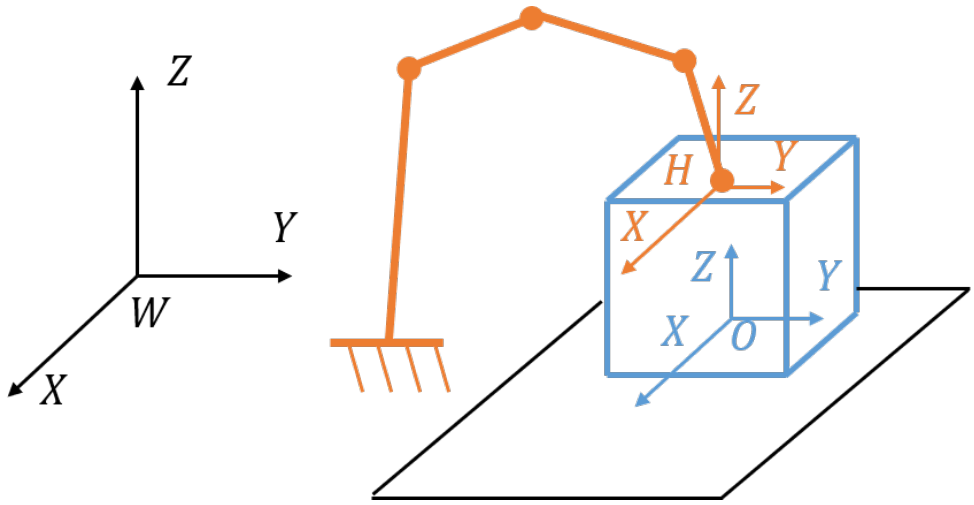

Figure 4.5: Illustration of the coordinate frames in the block tilting example.

In this definition, ${}^W_O\mathbf{R} \in SO(3)$ denotes the rotation matrix for ${}^W_O q$, $\mathbf{E}({}^W_O q)$ is the linear mapping from the body angular velocity to the quaternion time derivatives [28].

The generalized force corresponding to v is the object body wrench together with the hand pushing force:

$$f = [{}^O_O w^T,\ {}^W_H f^T]^T \in \mathbb{R}^9 \tag{4.48}$$

4.5.2 Goal Description

Denote ${}^W p_{tc}$ as a point on the line of contact, ${}^W \omega_g$ as the axis of rotation, $\dot{\theta}_g$ as the desired object rotation speed. The spatial twist for the object velocity is ${}^W \xi_g = (-{}^W \omega_g \times^W p_{tc},\ {}^W \omega_g)\dot{\theta}_g \in \mathbb{R}^6$, the corresponding body twist is:

$${}^O \xi_g = Ad_{{}^W_O g^{-1}}\, {}^W \xi_g \tag{4.49}$$

where $Ad_{{}^W_O g^{-1}}$ is the adjoint transformation associated with ${}^W_O g^{-1} = \begin{bmatrix} {}^W_O\mathbf{R} & {}^W_O p \\ 0 & 1 \end{bmatrix}^{-1}$. Then the goal can be specified as

$$\mathbf{G}v = b_G, \tag{4.50}$$

where $\mathbf{G} = \begin{bmatrix} \mathbf{I}_6 & 0_{6\times 3} \end{bmatrix}$, $b_G = {}^O \xi_g$.

4.5.3 Constraints

Contact Constraints

The contact between the object and the hand is a sticking point contact:

$${}^W_O Q({}^O p_{hc}) + {}^W_O p = {}^W p_{hc}, \tag{4.51}$$

where ${}^W p_{hc}, {}^O p_{hc}$ denote the location of the contact point, function ${}^W_O Q(p)$ rotates vector p by quaternion ${}^W_O q$.

The contact between the object and the table is a sticking line contact. We approximate it with two sticking point contacts at the corners:

$$ {}^W_O Q({}^O p_{tc,i}) + {}^W_O p = {}^W p_{tc,i}, \quad i = 1, 2. \tag{4.52} $$

Equation (4.51) and (4.52) together form a holonomic contact constraint for our system:

$$ \Phi(q) = \begin{bmatrix} {}^W_O Q({}^O p_{hc}) + {}^W_O p = {}^W p_{hc} \\ {}^W_O Q({}^O p_{tc,1}) + {}^W_O p = {}^W p_{tc,1} \\ {}^W_O Q({}^O p_{tc,2}) + {}^W_O p = {}^W p_{tc,2} \end{bmatrix} = 0. \tag{4.53} $$

Take the time derivative of both sides of Equation (4.53) and use Equation (4.46):

$$ \frac{\partial \Phi(q)}{\partial q} \mathbf{\Omega}(q) v = 0, \tag{4.54} $$

which gives us the Jacobian in (3.1):

$$ \mathbf{J} = \frac{\partial \Phi(q)}{\partial q} \mathbf{\Omega}(q). \tag{4.55} $$

Newton's second law

The reaction forces $\lambda = [{}^W\lambda_{hc}^T, {}^W\lambda_{tc,1}^T, {}^W\lambda_{tc,2}^T]^T \in \mathbb{R}^9$ associated with the constraints (4.53) are the three contact forces as viewed in world frame. In Newton's second law (3.3), $\mathbf{J}'$ is the same as the $\mathbf{J}$ computed above since all contacts are sticking; the external force F explains the gravity:

$$ F = \left[{}^O{G_O}^T, 0, {}^H{G_H}^T \right]^T \in \mathbb{R}^9. \tag{4.56} $$

${}^H G_H$ should be zero if the robot force controller already compensates for self weight.

Guard Conditions

For each contact we require:

1. the normal force to be greater than a threshold n_{min};
2. the contact force to be within the friction cone.

We approximate the 3D friction cone with eight-sided polyhedron [78] with

$$ d_i = [\sin(\pi i/4),\ \cos(\pi i/4),\ 0]^T \tag{4.57} $$

being the unit direction vectors for each ridge. Denote μ_{hc}, μ_{tc} as the estimated minimal possible friction coefficient, $z = [0\ 0\ 1]^T$ as the unit Z vector, the friction cone constraints becomes

$$ \begin{array}{ll} \mu_{hc} z^T ({}^O_W R^W \lambda_{hc}) \geq d_i^T ({}^O_W R^W \lambda_{hc}), & i = 1, ..., 8 \\ \mu_{tc} z^T {}^W\lambda_{tc,1} \geq d_i^T {}^W\lambda_{tc,1}, & i = 1, ..., 8 \\ \mu_{tc} z^T {}^W\lambda_{tc,2} \geq d_i^T {}^W\lambda_{tc,2}, & i = 1, ..., 8 \end{array} \tag{4.58} $$

The normal force lower bound can be written as

$$\begin{aligned} z^T({}^O_W R^W \lambda_{hc}) &\geq n_{\min} \\ z^T{}^W \lambda_{tc,1} &\geq n_{\min} \\ z^T{}^W \lambda_{tc,2} &\geq n_{\min} \end{aligned} \tag{4.59}$$

Equation (4.58) and (4.59) form the guard condition (4.12). This completes the problem formulation. Now we can use algorithm 1 or 2 to solve the system at each time step. The solution has one dimensional velocity control pointing in the tilting direction and two force control dimensions. The Y component of the force command is close to zero. The force in other component is roughly pressing against the rotation axis to maintain sticking.

4.6 Evaluations

In this section, we evaluate the performance of the optimization-based hybrid servoing algorithm and the OCHS algorithm in randomly generated manipulation problems. We implement both algorithms in Matlab and test them on a desktop with an i7-9700k CPU clocked at 4.7GHz.

Test Problems

We consider a rigid object with one to three environmental contacts and one to three rigid body fingers. Each rigid body has three DOFs in planar problems, or six DOFs in 3D problems. The settings are listed in Table 4.1, where 'f' denotes a fixed (sticking) contact point, 's' denotes a sliding contact point. Each environment contact point can be sliding or sticking; finger contacts are all sticking.

Table 4.1: Types of Randomly Generated Test Problems

	Environment Contacts		Hand Contacts	
	# Contact Points	Contact Modes	# Contacts per finger	# Fingers
Planar	1	f,s	1,2	1
	2	ss	1,2	1
3D	1	f,s	1,2,3	1,2,3
	2	ff,fs,ss	1,2,3	1,2,3
	3	ffs,fss,sss	1,2,3	1,2,3

We randomly sample 1000 set of contact point locations and normals for each contact mode setting, making a total of 6000 planar and 72,000 3D test problems. The goal constraint is sampled randomly for each problem.

The results are summarized in Table 4.2 and 4.3, the difference is that in Table 4.3 we give the goal (4.5) the maximum possible dimension, so all algorithms must select the maximum velocity control dimension. In the tables, "OCHS(min)" and "OCHS(max)" denotes OCHS with the minimal and the maximal velocity control dimensions, respectively. "HS3" and "HS10" denotes

the optimization-based hybrid servoing algorithm. Since the projected gradient descent is the most time-consuming part of this algorithm, we test it with different number of initial guesses. HS3 uses three initial guesses and has a good computation speed; HS10 uses ten initial guesses to search the problem excessively. Both run each initial guess for 50 iterations.

Results

In all tests, OCHS(min) consistently finds solutions with better crashing indexes than HS3 and HS10, achieves lower average crashing index and fewer ill-conditioned solutions. OCHS(max) has a larger crashing index in Table 4.2 because it applies more dimensions of control. When all algorithms selects the same dimension of velocity control, OCHS(max) also always achieves better crashing indexes than HS3 and HS10 on every problem. HS3 and HS10 can find solutions with comparable conditioning as OCHS(min) most of the time, however, their solution are sometimes much worse due to the local minima in the non-convex optimization.

Both OCHS(min) and OCHS(max) are notably faster than HS3 and HS10. The velocity part of OCHS shows 7 to 13 times speedup comparing with HS3, 20 to 40 times speedup comparing with HS10.

4.6.1 Experiments

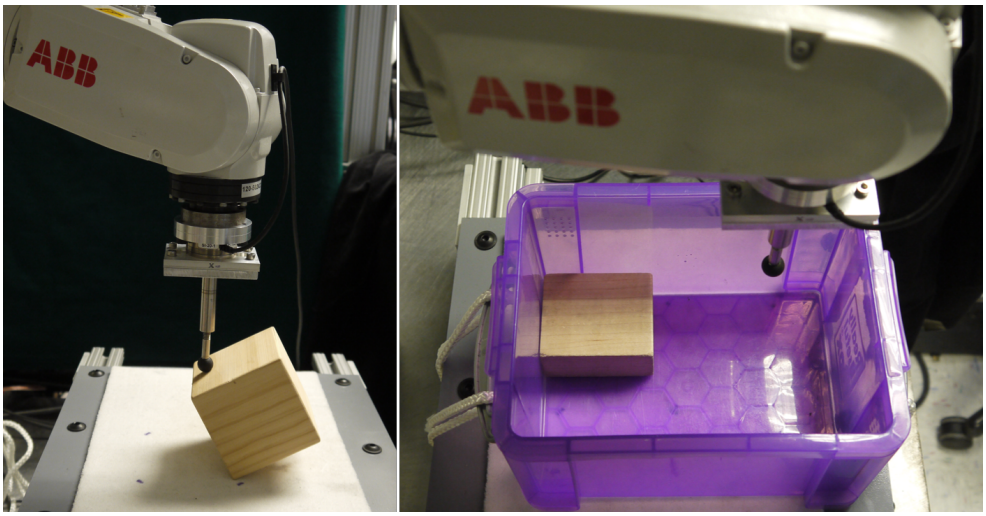

Figure 4.6: Our experiment setup. Left: block tilting. Right: tile levering-up.

We implemented block tilting (section 4.5) and another example: tile levering-up, in which the robot need to pivot the object up against a corner in a box. During the motion, the contacts between the object and the corner are sliding, while the contact on the robot finger is sticking.

In the block tilting task, the object is a 75mm-wide wooden block. We place a 2mm-thick piece of cloth on the table to introduce some passive compliance as well as increasing friction. In the tile levering-up example, the object is placed at a corner of an immobile plastic box. We experimented with a variety of objects. In both tasks, the robot hand is a rubber ball installed on a metal bar.

The control output computed by our algorithms can be implemented in many ways. We implemented hybrid force-velocity control with position-control inner loop according to [49], and added functionality for choosing axes in any orientation. We use a position-controlled ABB IRB 120 industrial robot with a wrist-mounted ATI Mini-40 force-torque sensor. In all experiments, we run OCHS off-line on a given motion plan to obtain a trajectory of HFVC, though the computation speed of OCHS supports feedback control at hundreds of Hz. In online execution, HFVC control loop is clocked at 200Hz. The lowest communication rate between the computer and the robot position controller is 250Hz with 25ms latency.

To test the optimization-based hybrid servoing algorithm (Section 4.3), we run the block tilting task 50 times in a row[2]. Each run contains 15 time steps. The robot successfully tilted the block 47 times. The three failures all experienced premature stops because the robot detected large force (about 25N on the FT sensor) at some time step. The reason could be a bad solution from our algorithm. We ran the tile levering-up task for about 20 times on different objects with two-thirds successful rate. The failures are caused by unexpected sticking between the object and the wall, or unexpected slipping between the robot hand and the object. One important reason for these failures is the slow response of the low level force control, which introduces large errors in the commanded contact normal force.

To test the OCHS algorithm and compare with the optimization-based hybrid servoing, we redo the block tilting experiment with the same setup. OCHS produces a hundred consecutive successes at with the robot velocity being 50% higher [3]. The result demonstrate the better consistency of OCHS comparing with the optimization-based method.

4.7 Summary and Discussion

In this section, we provide two algorithms to compute a hybrid force-velocity control for manipulation under contact constraints. The closed-form algorithm OCHS efficiently finds the solution with the best kinematic conditioning of the manipulation system. The optimization-based algorithm solves a non-convex optimization and takes more computation to converge, however, it is convenient to incorporate other cost function terms.

The hybrid servoing algorithms themselves can serve as a robust tracking controller to execute a pre-computed motion plan with sticking contacts only. It is also a building block for the more comprehensive contact stability analysis in the next chapter.

[2]You can find the 25min video at https://youtu.be/YIP8xIFATHE

[3]The full 40min video is available at `https://www.dropbox.com/s/ppimywwrgaelbw8/108_Block_tiltings.mp4?dl=0`

Table 4.2: Test results - Min Velocity Control Dimension

Planar (6 DOF)		OCHS(min)	OCHS(max)	HS3	HS10
# of Problems	Total	6000			
	Solved	5985	5981	5974	5977
Average Crashing Index		15.5	18.4	19.5	19.9
ill-conditioned solutions		15	19	26	23
Velocity Time(ms)	Average	0.14	0.14	1.74	5.32
	Worst	0.63	0.46	2.73	7.40
Force Time (ms)	Average	0.99	0.94	1.28	1.33
	Worst	2.21	2.44	2.09	3.34

3D, (12 to 24 DOF)		OCHS(min)	OCHS(max)	HS3	HS10
# of Problems	Total	72000			
	Solved	67506	67399	65950	65952
Average Crashing Index		3.98	13.8	5.20	4.68
ill-conditioned solutions		22	136	50	48
Velocity Time(ms)	Average	0.28	0.28	1.96	5.70
	Worst	0.85	0.77	3.82	10.7
Force Time (ms)	Average	1.09	1.04	1.57	1.61
	Worst	3.03	3.13	12.3	11.8

Table 4.3: Test results - Max Velocity Control Dimension

		OCHS(min)	OCHS(max)	HS3	HS10
# of Problems	Total	30000			
	Solved	27372	27372	22939	22951
Average Crashing Index		8.34	8.34	14.0	12.9
ill-conditioned solutions		55	55	37	31
Velocity Time(ms)	Average	0.17	0.17	2.35	7.25
	Worst	0.54	0.45	3.99	28.39
Force Time (ms)	Average	1.04	0.98	1.39	1.44
	Worst	2.39	2.18	3.76	3.62

Chapter 5

Contact Mode Control In Shared Grasping[1]

Among the two failure cases in manipulation, the kinematic ill-conditioning problem is resolved by the hybrid servoing algorithms in the last chapter. In this chapter, we provide a solution to resolve the second failure case, which is unexpected contact mode switching, in a class of single object manipulation problem called *shared grasping*.

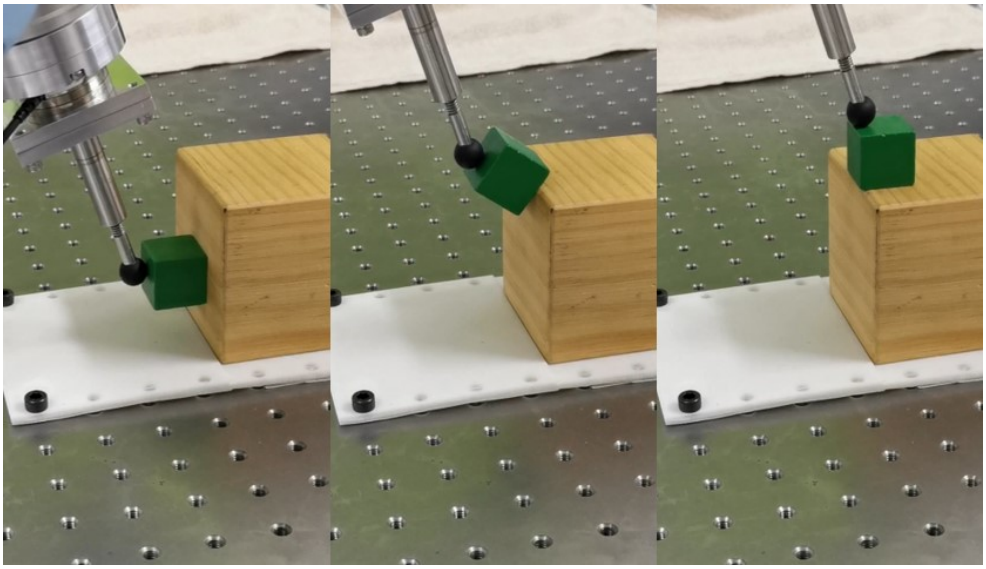

Figure 5.1: An example of shared grasping. The robot uses one point finger to lift a block up a stair.

A shared grasp is a grasp formed by contacts between the manipulated object and both the robot hand and the environment. Salisbury [72] introduced the idea of "whole arm manipulation", suggesting using all available surfaces on the robot for more ways to manipulate an object. Following this thought, if we treat the environment as another finger of the robot, manipulation

[1]This chapter is based on materials previously published in *Manipulation with Shared Grasping*, Yifan Hou, Zhenzhong Jia, and Matthew T. Mason, RSS 2020

under external contacts is like grasping from the object's point of view. This is the reason we call it shared grasping. Note that in-hand manipulation qualifies as shared grasping if we treat part of the fingers as the "environment".

Previous research has used shared grasps for non-prehensile manipulation such as pivoting and tumbling. This thesis treats the problem more generally, with methods to select the best shared grasp and robot actions for a desired object motion.

In shared grasping, the object can stay in a force closure even when no force closure grasps are available. Unlike traditional grasping where only sticking contacts are considered, shared grasping makes no assumption on contact mode. We need to consider how to avoid unexpected change of contact modes. In fact, even with perfect modeling parameters, rigid body modeling still gives shared grasping the problem of contact mode ambiguity, i.e. multiple contact modes could all be feasible under one robot control. Traditional force-based grasping stability analysis such as [80] cannot distinguish some contact modes from each other.

To overcome the mode ambiguity and control the exact motion of the shared-grasped object, our approach for shared grasping again uses HFVC instead of force control or velocity control alone. The velocity portion of HFVC filters out undesired contact modes and maintains the desired object velocity regardless of disturbance forces. The force portion of HFVC selects the desired contact mode while eliminating excessive internal forces. Note that the traditional force-based grasping analysis cannot model HFVC properly, because the forces in the velocity-controlled directions are not controllable.

We present our solution to shared grasping as follows. To begin with, we provide the definition of shared grasping and characterize the range of motion a shared-grasped object could have in Section 5.1.

Section 5.2 and Section 5.3 introduce our robust control and mode selection algorithm in detail. We propose to represent each contact mode by the set of feasible robot forces under this mode, making it possible to distinguish different contact modes by robot force command. We further make the process robust by adopting hybrid force-velocity control and analyze how the existence of velocity control affects the feasibility of every contact mode.

Section 5.4 analyzes the causes of contact mode failure and provides metrics to quantify the robustness of a contact mode. The construction of control in the algorithm reveals how a contact mode may fail under two influences: modeling uncertainty, and force disturbance. Modeling uncertainty could ruin the static force balance of a contact mode by affecting the contact geometries. Force disturbance could directly affect the robot force control in the mode selection step of our algorithm.

Section 5.5 shows how to apply our algorithm and the stability metrics to several common problems involving shared grasping, ranging from low-level control tasks to high-level trajectory planning. We demonstrate the efficacy of each of them in experiments in Section 5.6. In order to show the reliability of our approaches, we experiment with two different robots and multiple objects, and repeat every experiment multiple times to provide statistics.

5.1 Shared Grasping Definition and Properties

In this section, we provide a formal definition of shared grasping and discuss the properties of the motion of an object in a shared grasp. We focus on planar (2D) manipulation of rigid bodies. Beside the assumptions we made in Chapter 3, in this chapter we additionally assume the object weight is small enough comparing with contact forces, so we can ignore gravity and treat it as a disturbance force.

5.1.1 Shared Grasping

Consider a robot hand, a rigid object and a rigid environment in 2D. Denote $v_O \in \mathbb{R}^3$ as the velocity of the object. The contacts with the hand and the environment each imposes linear constraints on the object motion:

$$\begin{aligned} \mathbf{J}_{\mathrm{h}} v_O &= b_{\mathrm{h}}, \\ \mathbf{J}_{\mathrm{e}} v_O &= 0, \end{aligned} \tag{5.1}$$

where $\mathbf{J}_{\mathrm{h}}$ and $\mathbf{J}_{\mathrm{e}}$ are contact Jacobians derived from the contact point location and normal as in [59], b_{h} is determined by the robot hand velocity. Now we are ready to define shared grasping.

Definition 5.1.1. Shared Grasping. *Consider an object that is in contact with both a rigid, fixed environment and a robot. We say the object is in a shared grasp if:*

1. $\mathbf{J}_{\mathrm{h}}$ *and* $\mathbf{J}_{\mathrm{e}}$ *each does not have full row rank, i.e. the object velocity is not fully constrained by the environmental contacts or hand contacts alone.*
2. $\begin{bmatrix} \mathbf{J}_{\mathrm{e}} \\ \mathbf{J}_{\mathrm{h}} \end{bmatrix}$ *has full row rank, i.e. the object velocity is fully constrained by the environmental contacts and hand contacts combined.*

An example of shared grasping is shown in Figure 5.1. The conditions in definition 5.1.1 result in a nice property of the object motion in a shared grasp:

Theorem 1 (Motion Range of Shared Grasping)**.** *Consider an object in a shared grasp. Denote* $\mathbf{J}_{\mathrm{e}}$ *as the contact Jacobian of the environmental contacts on the object. With a suitable robot velocity, the object can have any velocity in the constrained subspace, i.e. the null space of* $\mathbf{J}_{\mathrm{e}}$*.*

Proof. We prove by constructing a robot action, i.e. the b_{h} in (5.1), then show that it results in the desired motion. Consider any $v^* \neq 0$ satisfying $\mathbf{J}_{\mathrm{e}} v^* = 0$. Such velocity must exist because J_{e} does not have full row rank. Next we show the choice $b_{\mathrm{h}}^* = \mathbf{J}_{\mathrm{h}} v^*$ satisfies our requirement. This choice leads to

$$\begin{bmatrix} \mathbf{J}_{\mathrm{h}} \\ \mathbf{J}_{\mathrm{e}} \end{bmatrix} v^* = \begin{bmatrix} b_{\mathrm{h}}^* \\ 0 \end{bmatrix}, \tag{5.2}$$

so v^* is a feasible object velocity under our robot action b_{h}^*. Moreover, since $\begin{bmatrix} \mathbf{J}_{\mathrm{h}} \\ \mathbf{J}_{\mathrm{e}} \end{bmatrix}$ has full row rank, its null space is empty. Then v^* must be the unique solution to the linear system (5.2). In other words, when the robot hand motion provides constraint $(\mathbf{J}_{\mathrm{h}}, b_{\mathrm{h}}^*)$ on the object, the resulted object velocity will be the given v^*. □

This theorem characterizes how dexterous shared grasping could be. Note the shared grasping definition and theorem 1 works for both planar problems and 3D problems, although this article focuses on planar problems.

5.2 A Distinguishable Contact Mode Representation

In this section we provide a representation of contact mode that can distinguish different modes from each other. This section lies the foundation for the mode selection algorithm and robustness analysis in the following sections. Denote H, O, W as the hand frame, object frame and world inertia frame, respectively. In the following we use the symbol ${}^{A}c_{B}$ to describe the quantity c of frame B measured in frame A, where c could be position p, normal n, force f, etc.

5.2.1 Polyhedral Convex Cone

In order to reveal the structure of forces in shared grasping, we re-derive the equations of force equilibrium. Define the generalized velocity vector $v \in \mathbb{R}^6$ and force vector $f \in \mathbb{R}^6$ of the system in the robot hand frame H:

$$v = \begin{bmatrix} {}^{H}v_O \\ {}^{H}v_H \end{bmatrix}, f = \begin{bmatrix} {}^{H}f_O \\ {}^{H}f_H \end{bmatrix}. \tag{5.3}$$

${}^{H}f_H \in \mathbb{R}^3$ is the robot control force, ${}^{H}f_O \in \mathbb{R}^3$ is zero since gravity is ignored. We choose the hand frame as the reference frame because it makes HFVC easier to describe. The Newton's Second Law for the hand-object system takes the following form:

$$\begin{bmatrix} \mathbf{J}'^{T}_{\mathrm{e}} & -\mathbf{J}'^{T}_{\mathrm{h}} \\ 0 & \mathbf{J}'^{T}_{\mathrm{h}} \end{bmatrix} \begin{bmatrix} \tau_{\mathrm{e}} \\ \tau_{\mathrm{h}} \end{bmatrix} + \begin{bmatrix} {}^{H}f_O \\ {}^{H}f_H \end{bmatrix} = 0. \tag{5.4}$$

Here the first term is the contact wrench written in terms of a positive linear combination of unitary wrenches as in [70]: each row of $\mathbf{J}'_{\mathrm{e}} \in \mathbb{R}^{n_{\mathrm{e}} \times 3}$ and $\mathbf{J}'_{\mathrm{h}} \in \mathbb{R}^{n_{\mathrm{h}} \times 3}$ represents the wrench of a friction cone edge of an environment contact and a hand contact, respectively; $\tau_{\mathrm{e}} \in \mathbb{R}^{n_{\mathrm{e}}}, \tau_{\mathrm{h}} \in \mathbb{R}^{n_{\mathrm{h}}}$ are non-negative vectors of contact wrench magnitudes. The number of unitary wrenches $n_{\mathrm{e}}, n_{\mathrm{h}}$ are determined by the number of contact points and their contact modes. Each contact point contributes up to two wrenches (i.e. two rows in $\mathbf{J}'_{\mathrm{e}}$ or $\mathbf{J}'_{\mathrm{h}}$), depending on its mode:

- **(s)** Separation: no wrench from the contact;
- **(f)** Fixed: wrenches for both the left and right edges;
- **(l)** Left sliding: only the wrench for the right edge;
- **(r)** Right sliding: only the wrench for the left edge.

Following the convention in [52], we use a string of the abbreviation letters to describe a contact mode, e.g. 'ffsl'. Note the $\mathbf{J}'_{\mathrm{e}}$ and $\mathbf{J}'_{\mathrm{h}}$ are different from the contact Jacobians used in velocity constraint. We distinguish them using the prime superscript.

In grasping analysis, it is common to exam the Minkowski sum of all contact wrenches on the object, which forms a polyhedral convex cone (PCC). Examples can be found in [59] and

[70]. The PCC represents the range of feasible forces from the given set of contact points. If the PCC spans the whole wrench space, the object is said to be in force closure. However, this definition of PCC is not able to distinguish different contact modes because every force closure contact mode looks the same (spans the whole wrench space). We resolve this ambiguity with a modified PCC in the following subsection.

5.2.2 The Cone of a Mode

Rewrite equation (5.4) using ${}^{H}f_{O} = 0$:

$$\mathbf{J'}_{\mathrm{e}}^{T}\tau_{\mathrm{e}} = \mathbf{J'}_{\mathrm{h}}^{T}\tau_{\mathrm{h}} = -{}^{H}f_{H}. \tag{5.5}$$

This equation maps the environmental contact force τ_{e} and the hand contact force τ_{h} into the generalized force space of the hand, which is also the space of robot action ${}^{H}f_{H}$. Since $\tau_{\mathrm{e}} \geq 0, \tau_{\mathrm{h}} \geq 0$, $\mathbf{J'}_{\mathrm{e}}^{T}\tau_{\mathrm{e}}$ and $\mathbf{J'}_{\mathrm{h}}^{T}\tau_{\mathrm{h}}$ each again forms a closed polyhedral convex cone in this wrench space, call them the *environment cone* $\mathbb{C}_{\mathrm{e}}$ and the *hand cone* $\mathbb{C}_{\mathrm{h}}$:

$$\begin{aligned} \mathbb{C}_{\mathrm{e}} &= \{\mathbf{J'}_{\mathrm{e}}^{T}\tau_{\mathrm{e}} \mid \tau_{\mathrm{e}} \geq 0\} \\ \mathbb{C}_{\mathrm{h}} &= \{\mathbf{J'}_{\mathrm{h}}^{T}\tau_{\mathrm{h}} \mid \tau_{\mathrm{h}} \geq 0\} \end{aligned} \tag{5.6}$$

Different contact modes have different hand cones and/or environmental cones. An example is shown in Figure 5.2. Rows of $\mathbf{J'}_{\mathrm{e}}$ and $\mathbf{J'}_{\mathrm{h}}$ are generators of the cones. The set of feasible

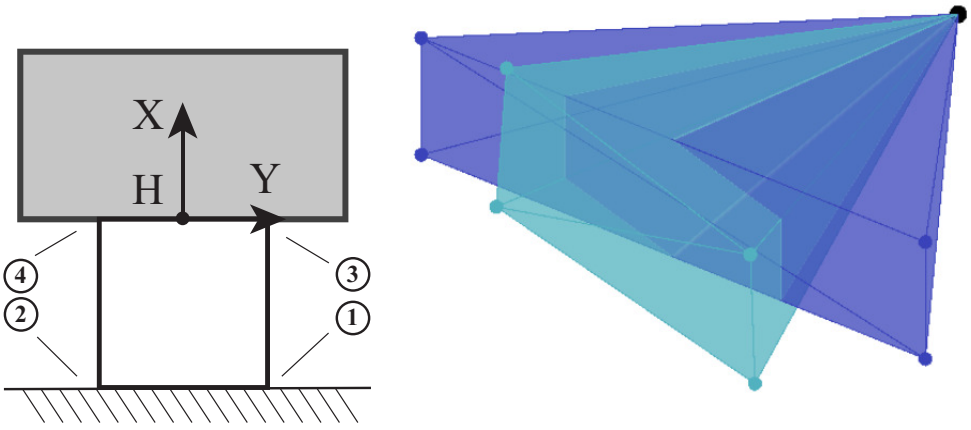

Figure 5.2: Left: a shared grasping system with a cube object (bottom) and a robot palm (top). The circled numbers show the ordering of contact points. Right: the hand cone (purple) and environment cone (blue) for the "ffff" mode.

robot force action ${}^{H}f_{H}$ under force balance (5.5) is the intersection between the hand cone and the environmental cone. The intersection is again a PCC, however, the intersection is more distinguishable for different modes comparing with the Minkowski sum, especially for force closure modes. We use this intersection to represent a contact mode m and call it the *cone of the mode* $\mathbb{C}_{\mathrm{m}}$:

$$\mathbb{C}_{m} = \mathbb{C}_{\mathrm{e}} \cap \mathbb{C}_{\mathrm{h}} \tag{5.7}$$

To demonstrate the benefit of this representation, in Figure 5.3 we draw all the feasible cones of the system shown in Figure 5.2. In this example, all 13 feasible modes have force closure, yet their $\mathbb{C}_m$ are different.

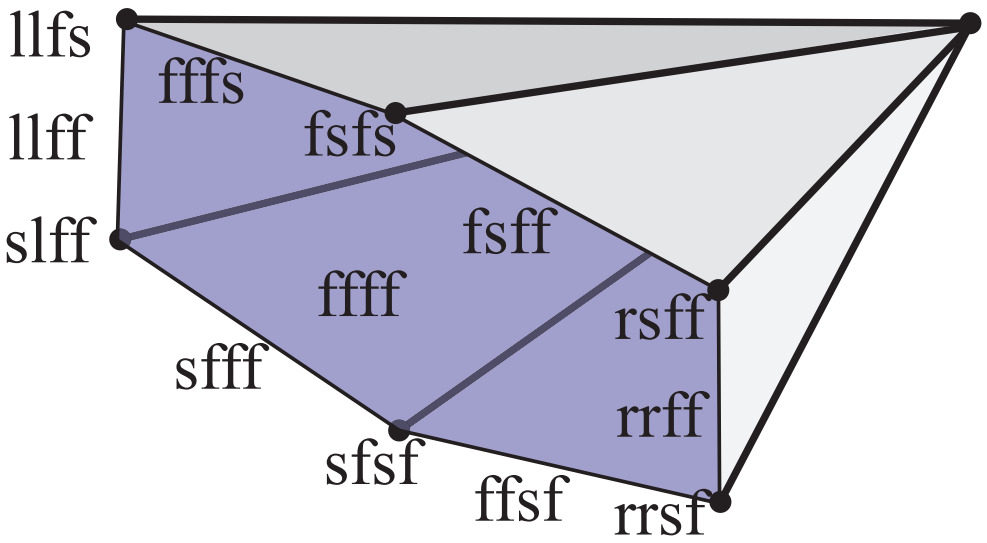

Figure 5.3: A closer look at the cones of the example in Figure 5.2 with annotation. In this example, each non-zero face of the 3D cone is the cone of a contact mode.

5.2.3 Properties of the Cone of the Mode

The cone of the mode $\mathbb{C}_{\mathrm{m}}$ illustrates the structure of the robot action space or the object wrench space. From the definition, we can easily see that the existence of the cone of the mode indicates force balance:

The existence of a nonzero intersection between $\mathbb{C}_{\mathrm{e}}$ and $\mathbb{C}_{\mathrm{h}}$ is equivalent to the existence of a non-trivial (all zero) solution $(\tau_{\mathrm{e}}, \tau_{\mathrm{h}}, {}^{H}f_{H})$ to the static balance equation (5.5)*.*

Depending on the existence of a non-empty $\mathbb{C}_m$, we call mode m *F-feasible* or *F-infeasible*.

The union of all the cones is a complete set of force equilibrium solutions at a given configuration. Since the cones are closed polyhedra, this set is exactly the cone of the all-fixed mode, denoted as $\mathbb{C}_{\mathrm{AF}}$. Denote $\mathbb{C}_{\mathrm{h,AF}}$ as the hand cone of the all-fix mode. Also denote $\mathbb{W}$ as the whole N_D dimensional wrench space. If the robot applies a wrench w, one of the following four situations must happen:

- $w \in \mathbb{C}_{\mathrm{AF}}$: the object can have static force balance;
- $w \in \mathbb{C}_{\mathrm{h,AF}} - \mathbb{C}_{\mathrm{AF}}$: the robot action is not balanced, the object is accelerating;
- $w \in \mathbb{W} - \mathbb{C}_{\mathrm{h,AF}}$: impossible because some contact forces are outside of their friction cones.

5.3 Robust Mode Selection with HFVC

5.3.1 A Naive Approach with Force Control

From (5.5) we know the robot can directly apply a wrench $w = -{}^H f_H$ in the wrench space. If

$$w \in \mathbb{C}_m, \tag{5.8}$$

then the mode m is feasible under robot action w. Moreover, if the robot action w is contained only in $\mathbb{C}_m$, then the system must be in mode m since it is the only feasible mode. Now it seems straightforward to enforce a contact mode m: pick a robot wrench w in $\mathbb{C}_m$ while staying away from the cones of other modes. The system will be in mode m if such w exists. However, this approach lacks robustness because the cones of different modes are not well separated from each other. For example, in Figure 5.3 the cone of the mode "fsff" is a facet of the cone of the "ffff" mode. If we choose "fsff" mode by applying a wrench in its cone, we might get "ffff" instead. What's worse is that a tiny disturbance force could shift the actual combined wrench strictly into the "ffff" mode. This is a demonstration of the mode ambiguity problem and shows a limitation of purely force-based method in multi-contact manipulation.

In this section, we show how to overcome this limitation by replacing pure force control w with HFVC. The key idea is that the velocity control part of HFVC won't be compatible with every contact mode; the incompatible modes are infeasible and can be ignored under certain conditions. Then we can select the desired mode with the force control part of HFVC in a set of well-separated cones.

5.3.2 Mode Filtering by Velocity Control

HFVC decompose the robot action space into a velocity control subspace and force control subspace. Denote n_{af} and n_{av} as the dimension of force and velocity control subspace in HFVC, $n_{\text{af}} + n_{\text{av}} = N_d$. Define transformation $\mathbf{R}_{\text{a}} \in \mathbb{R}^{N_d \times N_d}$, we can describe a HFVC by:

$$\begin{bmatrix} \omega_{\text{af}} \\ \omega_{\text{av}} \end{bmatrix} = \mathbf{R}_{\text{a}}{}^H v_H, \quad \begin{bmatrix} \eta_{\text{af}} \\ \eta_{\text{av}} \end{bmatrix} = \mathbf{R}_{\text{a}}{}^H f_H, \tag{5.9}$$

where $\eta_{\text{af}} \in \mathbb{R}^{n_{\text{af}}}$ and $\omega_{\text{av}} \in \mathbb{R}^{n_{\text{av}}}$ are the force and velocity control, respectively. η_{av}, ω_{af} are uncontrollable, since it is not possible to control both force and velocity in the same direction. The HFVC imposes a velocity constraint on the system generalized velocity v:

$$\begin{bmatrix} 0 & \mathbf{R}_{\text{a}}^{(n_{\text{av}})} \end{bmatrix} v = \omega_{\text{av}}. \tag{5.10}$$

$\mathbf{R}_{\text{a}}^{(n_{\text{av}})}$ is the last n_{av} rows of $\mathbf{R}_{\text{a}}$. Denote $\mathbf{C}_{\text{v}} = [0 \;\; \mathbf{R}_{\text{a}}^{(n_{\text{av}})}]$ and write the velocity control as

$$\mathbf{C}_{\text{v}} v = \omega_{\text{av}}. \tag{5.11}$$

Compute a velocity control.

We use the Optimally-Conditioned Hybrid Servoing (OCHS) algorithm in Section 4.4 to compute the force-velocity decomposition $\mathbf{R}_{\mathrm{a}}$ and velocity control magnitude ω_{av}. In shared grasping we only need the velocity related steps of OCHS and always pick the maximum possible dimension of velocity controls, so the complete OCHS is not necessary. To use OCHS, first we obtain the contact Jacobian at the desired contact mode m, which imposes a linear constraint on the system velocity:

$$\mathbf{J}_m v = 0, \tag{5.12}$$

where $\mathbf{J}_m$ is the contact Jacobian for mode m. Denote $\mathbf{U}$ as a matrix whose rows form an unitary basis of the null space: $\mathbf{U} = \mathrm{Null}(\mathbf{J}_m)$. We can express the free motion space of the robot by projecting U onto the controllable subspace, and use it as the velocity-controlled direction:

$$\mathbf{C}_{\mathrm{v}} = \mathrm{Row}(\mathbf{U}\mathbf{S}_a), \tag{5.13}$$

$\mathrm{Row}(\cdot)$ returns a matrix whose rows form an unitary basis of the rows of the argument, $\mathbf{S}_a \in \mathbb{R}^{6\times 6}$ is a selection matrix with only ones on the last three diagonal elements. The last three columns of $\mathbf{C}_{\mathrm{v}}$ are the velocity-controlled directions $\mathbf{R}_{\mathrm{a}}^{n_{\mathrm{av}}}$, which can be used to compute $\mathbf{R}_{\mathrm{a}}$:

$$\mathbf{R}_{\mathrm{a}} = \begin{bmatrix} \mathrm{Null}(\mathbf{R}_{\mathrm{a}}^{n_{\mathrm{av}}}) \\ \mathbf{R}_{\mathrm{a}}^{n_{\mathrm{av}}} \end{bmatrix} \tag{5.14}$$

To compute a velocity control magnitude ω_{av} that satisfies the goal (4.5), OCHS first solves for a special solution v^* to the goal and contact constraint:

$$\begin{bmatrix} \mathbf{J}_m \\ \mathbf{G} \end{bmatrix} v^* = \begin{bmatrix} 0 \\ b_G \end{bmatrix}. \tag{5.15}$$

Then use it to compute ω_{av}:

$$\omega_{\mathrm{av}} = \mathbf{C}_{\mathrm{v}} v^*. \tag{5.16}$$

(5.13) and (5.16) together describe the velocity control.

Check mode feasibilities

Each contact mode m is associated with a set of contact constraints, including non-penetration constraint, sticking constraint and sliding direction constraint. Beside the equality constraint (5.12), there could also be inequalities:

$$\mathbf{M}_m v \leq 0, \tag{5.17}$$

We call mode m **V-feasible** if the velocity constraints (5.12),(5.17), and (5.11) have a solution, **V-infeasible** otherwise.

Being V-infeasible means the velocity constraints have conflicts, so the first failure mode, crashing, could happen. An example is shown in Figure 5.4. Note that although a V-feasible mode exists (object is sliding to the left, finger is sticking), the all-sticking mode is V-infeasible

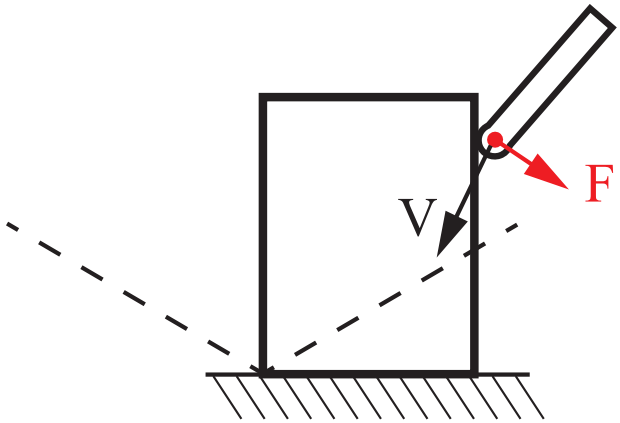

Figure 5.4: An example where the all-sticking mode is V-infeasible and causes crashing. A block is sitting on a rigid table. A finger touches the block on the right with a point contact and executes a HFVC as shown in the figure. Model the table -object edge contact with two contact points on the corner. The dashed lines show the friction cone of the left contact point.

and causing crashing since the table contact force can balance infinite force from the velocity-controlled direction.

As explained in Section 3.5.1, V-infeasible modes will not cause crashing and can be safely ignored if they do not have force balance at infinite magnitude. Since the force/wrench in the force-controlled directions cannot go to infinity, for crashing to happen the contact forces must be able to balance the force in the velocity-controlled directions. In other words, $\mathbb{C}_m$ must intersect the set of positive forces in the velocity-controlled directions, denoted as $\mathbb{C}_{\text{v}}$:

$$\mathbb{C}_m \cap \mathbb{C}_{\text{v}} \neq \{0\}. \tag{5.18}$$

This is a necessary condition for crashing. $\mathbb{C}_{\text{v}}$ here can be derived from (5.6) and (5.9):

$$\mathbb{C}_{\text{v}} = \{-\mathbf{R}_{\text{av}}^{-1}\tau_v | \tau_v \geq 0\}, \tag{5.19}$$

where $\mathbf{R}_{\text{av}}^{-1}$ represents the last n_{av} columns of $\mathbf{R}_a^{-1}$. $\mathbb{C}_{\text{v}}$ is again a PCC, we call it the *velocity cone*.

Remember $\mathbb{C}_{\text{AF}}$ contains all $\mathbb{C}_m$. Using (5.18) we can derive the sufficient condition for crashing-avoidance:

$$\mathbb{C}_{\text{AF}} \cap \mathbb{C}_{\text{v}} = \{0\}, \tag{5.20}$$

This condition ensures no crashing in all possible modes. If it is satisfied, we can safely ignore all the V-infeasible modes, which makes it easier to separate the desired mode from the rest. This is illustrated in the middle figure of Figure 5.5.

5.3.3 Mode Selection by Force Control

There is usually more than one contact mode remaining after the velocity filtering. The robot needs to secure the desired one by applying suitable force control. Note that given a force

control command η_{af}, the force in the velocity-controlled directions will automatically move the total robot wrench w into one of the remaining feasible cones since these cones are the only places for w to maintain static force equilibrium. As a result, although it is not possible to directly place the robot wrench w into the cone of the desired mode, we can still use the force control to do mode selection. From Equation (5.9) and (5.5) we know:

$$\mathbf{J}_{\text{e}}'^{T}\tau_{\text{e}} = \mathbf{J}_{\text{h}}'^{T}\tau_{\text{h}} = w = -\mathbf{R}_a^{-1}\begin{bmatrix}\eta_{\text{af}}\\ \eta_{\text{av}}\end{bmatrix}, \tag{5.21}$$

so the top n_{af} rows of $\mathbf{R}_a^{-1}$ form a basis of the force-controlled subspace in the robot action space. To select a mode, we project all V-feasible cones $\mathbb{C}_m$ into the force-controlled subspace. Those projections $\bar{\mathbb{C}}_m$ are PCCs in a lower dimensional space. Then we pick the force control η_{af} to be within the projection of the desired mode, while staying away from the projections of other modes:

$$\eta_{\text{af}} \in \bar{\mathbb{C}}_{\text{goal}} - \bar{\mathbb{C}}_{\text{others}}. \tag{5.22}$$

The minus sign denotes set difference. For planar problems, the projections $\bar{\mathbb{C}}_m$ are at most two dimensional; we use simple heuristics to select the best force control η_{af}. If such η_{af} exists, the actual w has to be in the desired mode because projections are surjective. An example of is shown in Figure 5.5, bottom. Because of the projection into a lower-dimensional space in the wrench space, we call this approach *Wrench Stamping*.

If the set $\bar{\mathbb{C}}_{\text{goal}} - \bar{\mathbb{C}}_{\text{others}}$ is empty, we call this mode *F-indistinguishable* since there is no force action the robot can take to avoid other feasible modes.

5.4 Metrics to Evaluate Contact Mode Stability

In this section, we show how to evaluate the robustness of a contact mode by looking at two causes of mode transitions:

5.4.1 Perturbations on Contact Geometry

Contact geometries plus friction coefficients determine the shape of hand and environment cones $\mathbb{C}_{\text{e}}, \mathbb{C}_{\text{h}}$. Modeling uncertainties change the shape of these cones, which could cause their intersection to disappear and the corresponding mode to be F-infeasible. For planar systems, we define a stability margin $\Phi_{\text{g}}(\mathbb{C}_{\text{e}}, \mathbb{C}_{\text{h}})$ that describes the "depth" of the intersection, i.e. the minimal angular rotation $\mathbb{C}_{\text{e}}$ needs to take to eliminate non-trivial intersections with $\mathbb{C}_{\text{h}}$. Note that the volume (solid angle) of the intersection $\mathbb{C}_m$ is NOT the right metric. Cones of most modes have zero volume. To introduce our metric, we first define a distance function $\Delta(F, E)$ which computes the angular distance between ray E and hyperplane F:

$$\Delta(F, E) = \arcsin(n_F \cdot E), \tag{5.23}$$

where n_F denotes the normal of the hyperplane F. Both n_F and E are unit vectors. Next, define a function that evaluates the "depth" of a cone inside another cone. Consider two PCCs $\mathbb{C}_a, \mathbb{C}_b$

that satisfies $\mathbb{C}_b \subseteq \mathbb{C}_a$. Define

$$\sigma(\mathbb{C}_a, \mathbb{C}_b) = \min_j \max_i \Delta(F_j, E_i), \tag{5.24}$$

where F_j is the jth facet of $\mathbb{C}_a$, E_i is the ith edge of $\mathbb{C}_b$. A facet F_j defines a hyperplane with the positive normal pointing inside of the cone. We reduce dimension whenever possible, for example, if $\mathbb{C}_a$ has two generators, a facet only has one ray and defines a one-dimensional hyperplane. If $\mathbb{C}_a$ only has one generator, we define $\sigma(\mathbb{C}_a, \mathbb{C}_b) = 0$. Finally, define the *geometrical stability margin* $\Phi_{\mathrm{g}}(\mathbb{C}_{\mathrm{e}}, \mathbb{C}_{\mathrm{h}})$ as:

$$\Phi_{\mathrm{g}}(\mathbb{C}_{\mathrm{e}}, \mathbb{C}_{\mathrm{h}}) = \min(\sigma(\mathbb{C}_{\mathrm{e}}, \mathbb{C}_{\mathrm{e}} \cap \mathbb{C}_{\mathrm{h}}),\ \sigma(\mathbb{C}_{\mathrm{h}}, \mathbb{C}_{\mathrm{e}} \cap \mathbb{C}_{\mathrm{h}})). \tag{5.25}$$

5.4.2 Perturbations on Force Control

Disturbance forces can move the force control η_{af} out of the region defined by equation (5.22). We define the *control stability margin* Φ_{c} as the angular distance from η_{af} to the closest cone projections of other modes $\bar{\mathbb{C}}_{\mathrm{others}}$.

$$\Phi_{\mathrm{c}}(\eta_{af}, \bar{\mathbb{C}}_{\mathrm{others}}) = \mathrm{dist}(\eta_{af}, \bar{\mathbb{C}}_{\mathrm{others}}). \tag{5.26}$$

Figure 5.6 shows how the control in Figure 5.5 can fail under disturbance force.

Φ_{c} and Φ_{g} each describes the minimal disturbance force required to trigger mode transition in terms of angular distance in object wrench space. Define the *stability margin* Ψ:

$$\Psi = \min(\Phi_{\mathrm{c}},\ \Phi_{\mathrm{g}}). \tag{5.27}$$

With proper scaling between force and torque, as well as a nominal force magnitude K_{F}, the product $K_{\mathrm{F}}\Psi$ represents the minimal magnitude of the disturbance force that can change contact mode m. Just like grasping, we can make the system more robust simply by applying larger force.

We summarize wrench stamping in Procedure WrenchStamping. $\mathbf{J}_{\mathrm{AF}}, \mathbf{M}_{\mathrm{AF}}, \mathbb{C}_{\mathrm{e,AF}}$, and $\mathbb{C}_{\mathrm{h,AF}}$ represents the variables defined in (5.12)(5.17)(5.6) for the all-fixed mode. $\mathbb{C}_{\mathrm{e},m}$,$\mathbb{C}_{\mathrm{h},m}$ denote the environmental cone and hand cone for mode m, respectively. Stability margins $\Phi_{\mathrm{g},m}, \Phi_{\mathrm{c},m}, \Psi_m$ for mode m are defined similarly. We call the 2D contact mode enumeration algorithm implemented in [41] to obtain a list of contact modes in the input.

5.5 Applications of Shared Grasping

In this section, we demonstrate how to use our wrench stamping algorithm and the stability margins to solve several common problems in manipulation with shared grasping.

5.5.1 Hybrid Servoing with Sliding Modes

Procedure WrenchStamping can be used directly to solve the hybrid servoing problem defined in Section 4.2, even if the problem involves sliding contacts. Wrench stamping adopts some procedures of Algorithm 2, and has two advantages over a plain OCHS:

Procedure WrenchStamping(Mode m^*)

Input: $\mathbf{J}_{\mathrm{AF}}, \mathbf{M}_{\mathrm{AF}}, \mathbb{C}_{\mathrm{e,AF}}, \mathbb{C}_{\mathrm{h,AF}}$
Input: G, b_{G}, set of modes $\mathbb{M}$
Output: Ψ, HFVC $(n_{\mathrm{af}}, n_{\mathrm{av}}, \mathbf{R}_{\mathrm{a}}, w_{\mathrm{av}}, \eta_{\mathrm{af}})$

Compute $n_{\mathrm{af}}, n_{\mathrm{av}}, \mathbf{R}_{\mathrm{a}}, w_{\mathrm{av}}$ from (5.13)(5.14)(5.16);
if *Crashing happens* (5.19)(5.20) **then return** $\emptyset$;
Initialize empty set $\hat{\mathbb{M}} \leftarrow \emptyset$;
`// Mode filtering by F-feasibility`
foreach $m \in \mathbb{M}$ **do**
    Assemble $\mathbb{C}_{\mathrm{e},m}$,$\mathbb{C}_{\mathrm{h},m}$ from columns of $\mathbb{C}_{\mathrm{e,AF}}, \mathbb{C}_{\mathrm{h,AF}}$;
    Compute $\mathbb{C}_m = \mathbb{C}_{\mathrm{e},m} \cap \mathbb{C}_{\mathrm{h},m}$;
    if $\mathbb{C}_m = \emptyset$ **then** Continue;
    Compute $\Phi_{\mathrm{g},m}(\mathbb{C}_{\mathrm{e},m}, \mathbb{C}_{\mathrm{h},m})$ (5.25);
    if $\Phi_{\mathrm{g},m} = 0$ **then** Continue;
    Assemble $\mathbf{J}_m, \mathbf{M}_m$ from rows of $\mathbf{J}_{\mathrm{AF}}, \mathbf{M}_{\mathrm{AF}}$;
    $\hat{\mathbb{M}} \leftarrow \{\hat{\mathbb{M}}, m\}$;
end
$\mathbb{M} \leftarrow \hat{\mathbb{M}}, \hat{\mathbb{M}} \leftarrow \emptyset$;
`// Mode filtering by V-feasibility`
foreach $m \in \mathbb{M}, m \neq m^*$ **do**
    if (5.11)(5.12)(5.17) *are feasible* **then** $\hat{\mathbb{M}} \leftarrow \{\hat{\mathbb{M}}, m\}$;
end
`// Projections`
$\bar{\mathbb{C}}_{\mathrm{goal}} \leftarrow \mathbf{R}_{\mathrm{af}}^{-T}\mathbb{C}_{\mathrm{m}^*}$, $\mathbf{R}_{\mathrm{af}}^{-1}$ is the first n_{af} cols of $\mathbf{R}_{\mathrm{a}}^{-1}$;
$\bar{\mathbb{C}}_{\mathrm{others}} \leftarrow \emptyset$;
foreach $m \in \hat{\mathbb{M}}$ **do**
    $\bar{\mathbb{C}}_{\mathrm{others}} \leftarrow \{\bar{\mathbb{C}}_{\mathrm{others}}, \mathbf{R}_{\mathrm{af}}^{-T}\mathbb{C}_m\}$
end
$\bar{\mathbb{C}}_{\mathrm{remain}} \leftarrow \bar{\mathbb{C}}_{\mathrm{goal}} - \bar{\mathbb{C}}_{\mathrm{others}}$;
if $\bar{\mathbb{C}}_{\mathrm{remain}} = \emptyset$ **then return** $\emptyset$;
Pick $\eta_{\mathrm{af}} \in \bar{\mathbb{C}}_{\mathrm{remain}}$, compute $\Phi_{\mathrm{c,m}^*}$;
$\Psi_{\mathrm{m}^*} \leftarrow \min(\Phi_{\mathrm{c,m}}, \ \Phi_{\mathrm{g},m})$;

- Wrench stamping does not require the user to provide the guard conditions, i.e. force constraints for the desired contact mode. On the contrary, wrench stamping reasons about contact mode switching conditions automatically based on contact geometry information.
- As a result, wrench stamping can handle contact modes with sliding contacts, while the hybrid servoing problem formulation cannot because the guard condition on contact force is not sufficient to specify a sliding contact.

We validate wrench stamping in experiments in Section 5.6.1.

5.5.2 Robust Control with Mode Selection

Sometimes when multiple contact modes are feasible, it is not obvious for the user to decide which mode is the best. In this case we can use the stability margin to select the most robust one for the desired motion. We wrap Procedure WrenchStamping into algorithm 3 to solve this problem. Given a desired velocity, the algorithm finds the most robust contact mode and computes a hybrid force-velocity control that achieves the desired motion and maintains the most robust contact mode. We demonstrate the use of this algorithm in Section 5.6.2.

Algorithm 3: Robust Control with Mode Selection

Input: $\mathbf{J}_{\mathrm{AF}}, \mathbf{M}_{\mathrm{AF}}, \mathbb{C}_{\mathrm{e,AF}}, \mathbb{C}_{\mathrm{h,AF}}, G, b_G$

Output: The chosen contact mode m^*, stability margin Ψ_{m^*}, and HFVC $(n_{\mathrm{af}}, n_{\mathrm{av}}, R_{\mathrm{a}}, w_{\mathrm{av}}, \eta_{\mathrm{af}})$

1 $\mathbb{C}_{\mathrm{AF}} \leftarrow \mathbb{C}_{\mathrm{e,AF}} \cap \mathbb{C}_{\mathrm{h,AF}}$;
2 $\mathbb{M}_{\mathrm{all}} \leftarrow$ all contact modes, $\mathbb{M} \leftarrow \emptyset$;
3 Line 4 - 12 of Procedure WrenchStamping for $m \in \mathbb{M}_{\mathrm{all}}$, save results in $\mathbb{M}$;
4 Initialize the solution set: $\mathbf{S} \leftarrow \emptyset$;
5 **foreach** $m \in \mathbb{M}$ **do**
6 $\quad \mathbf{S} \leftarrow \{\mathbf{S}, \text{WrenchStamping}(m)\}$
7 **end**
8 **return** $s \in \mathbf{S}$ *with the highest* Ψ_m

5.5.3 Motion Planning with Robustness Guarantee

With the stability margins, we can quantify how robust the shared grasping system is under a certain control. This information can be incorporated into a motion planning framework to plan robust contact-rich shared grasping motions. One idea is to use the stability margin to filter out unstable motions. In a Rapidly-exploring Random Tree (RRT) pipeline, we compute HFVC and check the stability margin of each new branch before adding it to the existing tree. The new branch is not added to the tree if its stability margin is smaller than a pre-defined threshold, so we maintain a tree with only robust motions. We refer the reader to [19] for a complete algorithm design. Example of executions of planned motions are shown in Section 5.6.3.

5.5.4 Geometry Optimization

We can optimize contact geometry parameters to maximize the geometrical stability margin Φ_g for a given mode. Note that the function $\Phi_g(\mathbb{C}_e, \mathbb{C}_h)$ boils down to the Δ distance between a facet F and an edge E. The facet has either two generators C_i, C_j or just C_i, we can backtrack its location in $\mathbb{C}_e$ and $\mathbb{C}_h$. The edge is a generator of $\mathbb{C}_e \cap \mathbb{C}_h$, which could be either an edge C_k of $\mathbb{C}_e$ or $\mathbb{C}_h$, or an intersection of a facet (C_1, C_2) in $\mathbb{C}_e$ and a facet (C_3, C_4) in $\mathbb{C}_h$. In the latter case, the expression of the edge is $(C_1 \times C_2) \times (C_3 \times C_4)$. So we can explicitly write down Φ_g as a function of contact screws:

$$\begin{aligned} \Phi_g(\mathbb{C}_e, \mathbb{C}_h) &= \arccos\left(\frac{C_i}{||C_i||} \cdot \frac{C_k}{||C_k||}\right) \\ &\text{or} \\ &= \arccos\left(\frac{C_i}{||C_i||} \cdot \frac{(C_1 \times C_2) \times (C_3 \times C_4)}{||(C_1 \times C_2) \times (C_3 \times C_4)||}\right). \end{aligned} \tag{5.28}$$

for one-dimensional facet F, or

$$\begin{aligned} \Phi_g(\mathbb{C}_e, \mathbb{C}_h) &= \arcsin\left(\frac{C_i \times C_j}{||C_i \times C_j||} \cdot \frac{C_k}{||C_k||}\right) \\ &\text{or} \\ &= \arcsin\left(\frac{C_i \times C_j}{||C_i \times C_j||} \cdot \frac{(C_1 \times C_2) \times (C_3 \times C_4)}{||(C_1 \times C_2) \times (C_3 \times C_4)||}\right) \end{aligned} \tag{5.29}$$

for two-dimensional F. We can further expand the contact screws into contact geometry parameters and compute the gradient of Φ_g in analytical form. We outline a gradient descent algorithm in Algorithm 4. The parameter $\mathbf{p}$ may contain contact locations, normals and friction coefficients. Note that computation in (5.28) and (5.29) only involves contact screws that are the current bottleneck of stability margin. This set of contact screws will change between iterations. We observed this phenomenon in the experiments section.

In order to get analytical expression of the gradient of Φ_g, we ignore the normalizations in (5.28) and (5.29) when computing their derivatives. To obtain the correct gradient about a contact screw, we just project the gradient computed this way onto the tangent plane of the contact screw, as shown in line 7.

We can apply geometrical parameter optimization on hand contacts to optimize finger placement, as demonstrated in Section 5.6.4. We can also optimize environment contact geometry, which is useful for tooling or fixture design.

5.6 Experimental Validations

We implement our algorithms in MATLAB and test them on several tasks in experiments. We use an ABB IRB120 robot in section 5.6.1, 5.6.2, 5.6.3, and a UR5e robot in section 5.6.4. Both are 6-axis robot and position-controlled; The hybrid force-velocity controller is implemented in C++ with feedback from a wrist-mounted ATI Mini-40 FT sensor. The communication rate is 250Hz

Algorithm 4: Geometrical Parameter Optimization

Input: Contact geometry parameter $\mathbf{p}$

Output: Optimized contact geometry parameter $\mathbf{p}^*$

repeat

`// 1. Evaluate` Φ_g

Compute $\mathbb{C}_e$, $\mathbb{C}_h$ (5.6) using $\mathbf{p}$;

Compute $\mathbb{C}_m = \mathbb{C}_e \cap \mathbb{C}_h$, record the mapping M_1 between columns of $\mathbb{C}_m$ and columns of $\mathbb{C}_e, \mathbb{C}_h$.;

Compute Φ_g (5.25), record the mapping M_2 between $C_{i,j,k,1,2,3,4}$ and columns of $\mathbb{C}_e, \mathbb{C}_h, \mathbb{C}_m$;

Use M_1 and M_2 to compute the mapping M_3 between $C_{i,j,k,1,2,3,4}$ and columns of $\mathbb{C}_e, \mathbb{C}_h$;

`// 2. Compute gradient`

Compute $\Phi_t = \partial\Phi_g/\partial C_t, t \in \{i, j, k, 1, 2, 3, 4\}$ by differentiating (5.28)(5.29) without denominators;

Remove the components of Φ_t that scales C_t:
$\Phi_t = \Phi_t - \Phi_t \cdot C_t, t \in \{i, j, k, 1, 2, 3, 4\}$;

Use M_4 to construct $\Phi_C = \partial\Phi_g/\partial C$ from Φ_t, $C \in$ columns of $\mathbb{C}_e, \mathbb{C}_h$;

Compute the Jacobian $C_{\mathbf{p}} = \partial C/\partial\mathbf{p}$ by differentiating the expression of $\mathbb{C}_e, \mathbb{C}_h$ (5.6).;

$\Phi_{\mathbf{p}} \leftarrow \sum_C (\Phi_C \cdot C_{\mathbf{p}})$;

`// 3. Update`

$\mathbf{p} \leftarrow \mathbf{p} + d\Phi_{\mathbf{p}}$, d is a step length;

until *converge*;

for the ABB robot and 500Hz for the UR5e robot. In both cases, the force control bandwidth is roughly 1Hz.

5.6.1 Robust control of desired contact modes

First, we validate the ability of our method in computing robust actions under different contact modes. Consider the shared grasping system in Figure 5.2 as an example. The friction coefficients are estimated to be 1.2 and 0.25 for hand-object and table-object contacts, respectively.

There are 81 contact modes for the four contact points. Algorithm 3 computes 13 modes with positive geometrical stability margin Φ_g. We hand-pick six modes with object motion and solve them with Procedure WrenchStamping. The results are shown in Figure 5.7.

5.6.2 Contact Mode Selection and Control

Using the same object and robot, we demonstrate Algorithm 3 in the task of cube rotation with a palm, as shown in Figure 5.8. The task is challenging because the object could fall at any time, the control must always be robust. The task has four stages. The goal velocity for the first two stages are:

$$\mathbf{G}_1 = [0\ 0\ 1\ 0\ 0\ 0],\ b_{G1} = -0.1. \tag{5.30}$$

$$\mathbf{G}_2 = \begin{bmatrix} Adj_{WH} & \mathbf{I}_3 \\ 0\ 0\ 0 & 0\ 0\ 1 \end{bmatrix}, b_{G2} = [0\ 0\ 0\ -0.1]^T. \tag{5.31}$$

First three rows of $\mathbf{G}_2, b_{G2}$ constrain the object to be static. Stage three and four are the same but in a different plane. Each stage is divided into eight time steps, at which Algorithm 3 decides which mode and what action to take. There is no vision feedback, the object pose is estimated from robot hand pose and has an accumulating error. We execute the four-stage task eight times and have four complete successes. The failures all happen in the last two stages.

5.6.3 Execution of Robust Motion Plans

We obtain trajectories of robot motion for several planar shared grasping problems using the sampling-based planning method in [19]. The planning algorithm use our stability margin to filter out unstable motions, so the generated motion plan is guaranteed to be robust. Figure 5.9 shows the experiment results of some of the trajectories found by the planner. To demonstrate the reliability of the HFVC, we execute each trajectory at least three times consecutively without object state feedback.

5.6.4 Finger Placement Optimization and Evaluation

We test Algorithm 4 on a point finger manipulation task, as shown in Figure 5.10, left. The width and height of the block are both 0.1m. We optimize finger location for two tasks: 1. pivoting about the left corner with mode "sff"; 2. Sliding to the left with mode "llf". We show the result of 200 iterations of gradient descent in Figure 5.11, right, which takes 0.45s in MATLAB. The

stability margin increases and reaches a plateau. Note that on the plateau the solution is still zig-zagging by a small magnitude, because the set of contact screws being optimized is varying between iterations, as explained in section 5.5.4.

We can graphically illustrate how robust the optimized control is. Using the optimized finger location and a HFVC computed from Algorithm 3, we compute the range of table contact locations that produce a positive stability margin for the desired mode "llf". We randomly sample 1000 pairs of object right and left corner locations within the range $(-0.2m, 0.2m)$. The result is shown in Figure 5.12. A point is marked green if the stability margin is positive. We can see there is roughly a safety range of $0.05m$ around the nominal point, i.e. the sliding motion can still be successful if the actual environmental contact location changes by 0.05m. We can do the same computation for varying object height, friction coefficient, etc.

We implement this sliding action with a wide range of objects. To add more difficulty, we perform the pushing in the vertical direction to add gravity as a disturbance force. The same control law works for a variety of different objects reliably, as shown in Figure 5.13.

5.7 Conclusion and Discussion

To conclude, in this chapter we provide a solution to planar rigid body shared grasping problems. The solution includes metrics to quantify the robustness of contact modes and methods to maximize the robustness in controllers. We demonstrate the use of our method in several common problems including robust planning and control and verify the solution to these problems in a variety of experiments.

Although shared grasping is designed for making environmental contacts, the stability analysis and algorithms also work for real in-hand manipulation problems if there are two (groups of) fingers.

Shared grasping complements grasping as another category of manipulations with good robustness. We shows its potential to make contact-rich manipulations reliable enough in real-world applications.

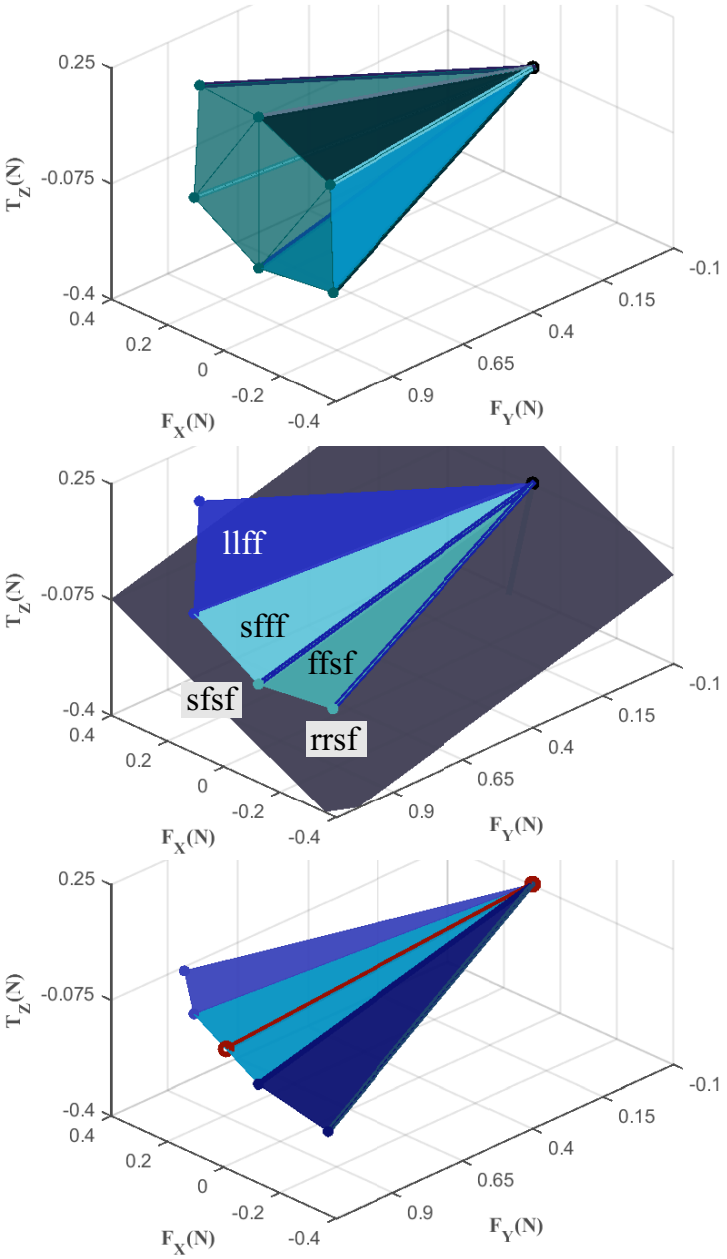

Figure 5.5: Procedures of wrench stamping for the problem in Figure 5.2. The torque is scaled into Newton by the object length. Top: All nonempty cones. Middle: the cones of V-feasible modes (one or two generators). The gray plane is the force-controlled subspace. Bottom: Projection of feasible cones onto the force-controlled subspace. The red ray is the chosen wrench for mode "sfff".

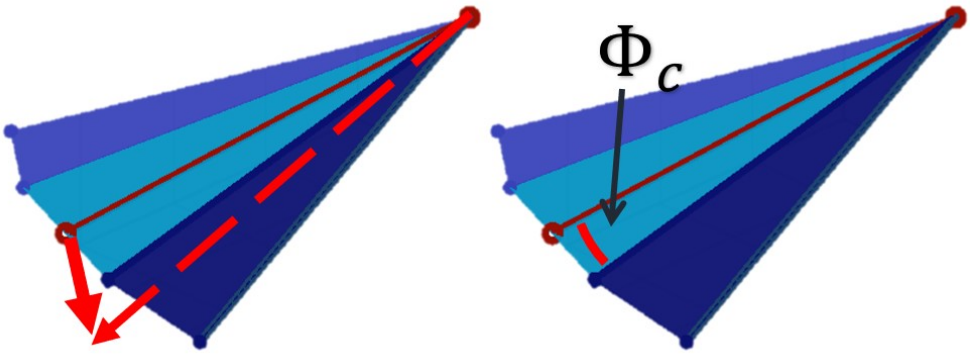

Figure 5.6: Left: A disturbance force (bold red arrow) changes the contact mode. Right: illustration of the control stability margin.

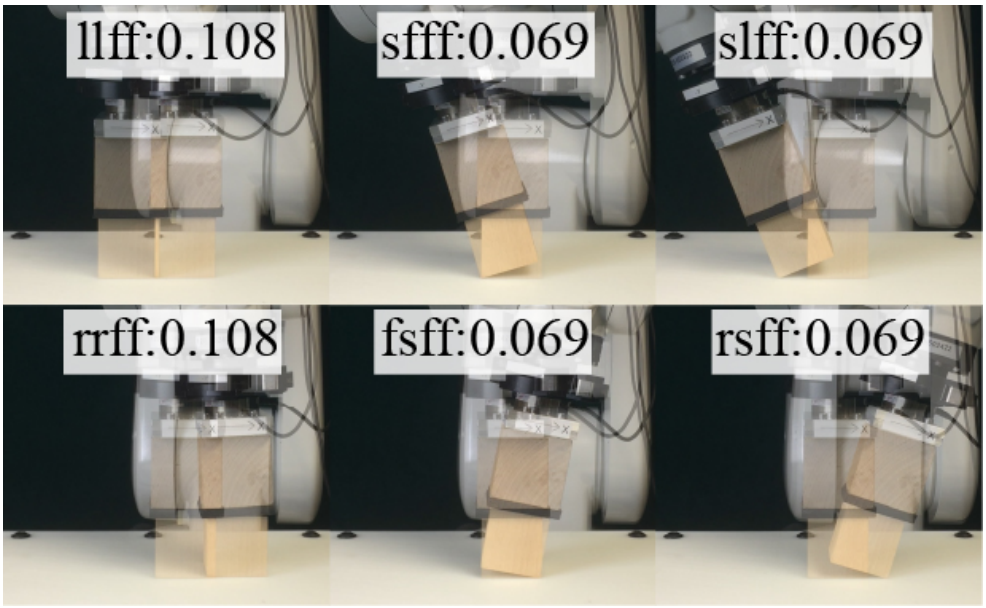

Figure 5.7: Different contact modes and their stability margins. The order of contacts follows Figure 5.2.

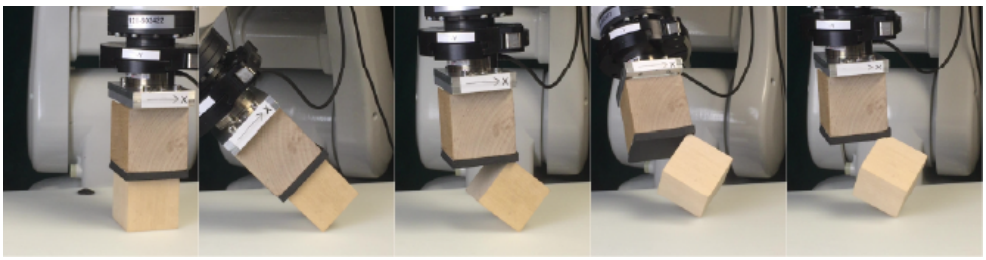

Figure 5.8: Flipping a cube on its corners.

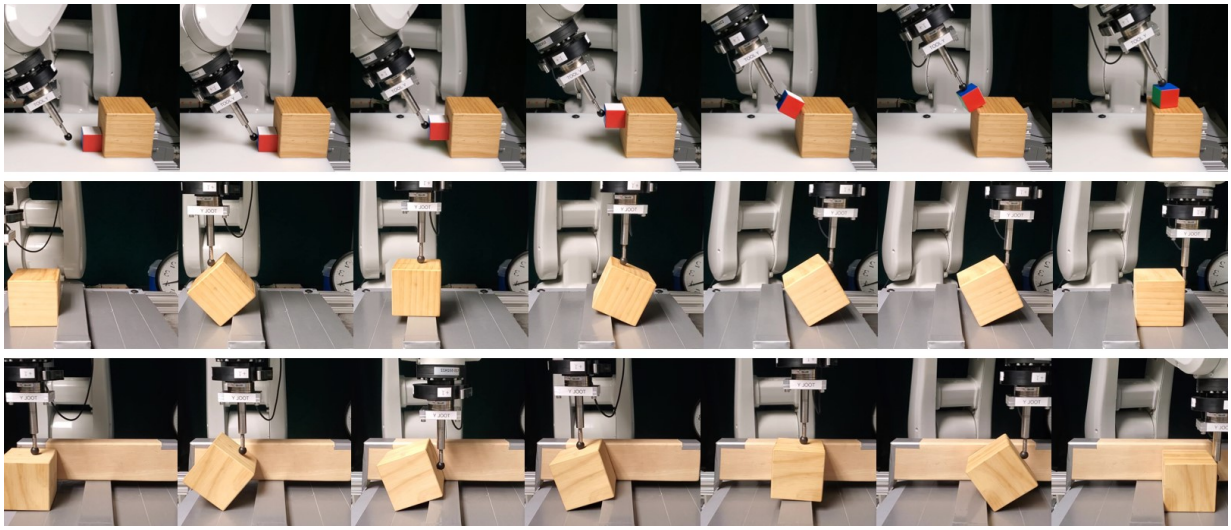

Figure 5.9: Snapshots of executing motion plans. Top row: lifting an object over a stair. Middle row: transport an object over an obstacle. Bottom row: another solution to the obstacle traverse problem. The wood board in the back is used to reduce out of plane motions.

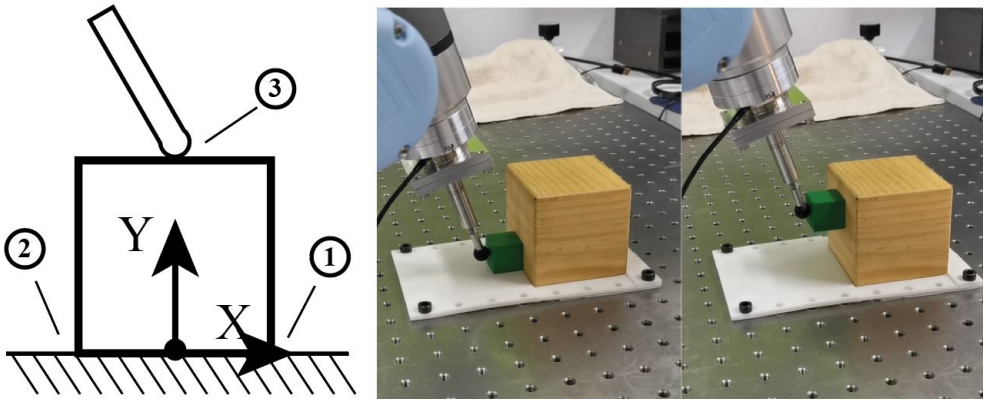

Figure 5.10: Left: the Finger-block example and the ordering of contacts. Right: Experiment setup. The yellow block is fixed.

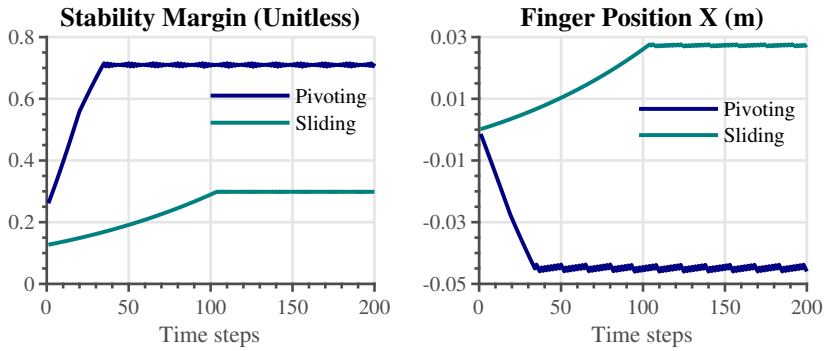

Figure 5.11: The evolution of finger position and stability margin during the optimization.

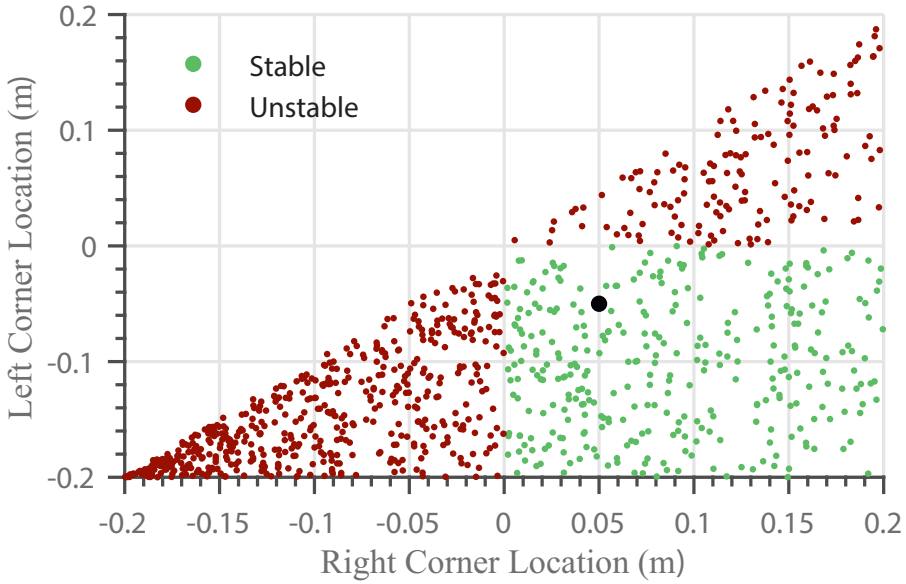

Figure 5.12: Feasibility of different model parameters. The control is computed for the black dot.

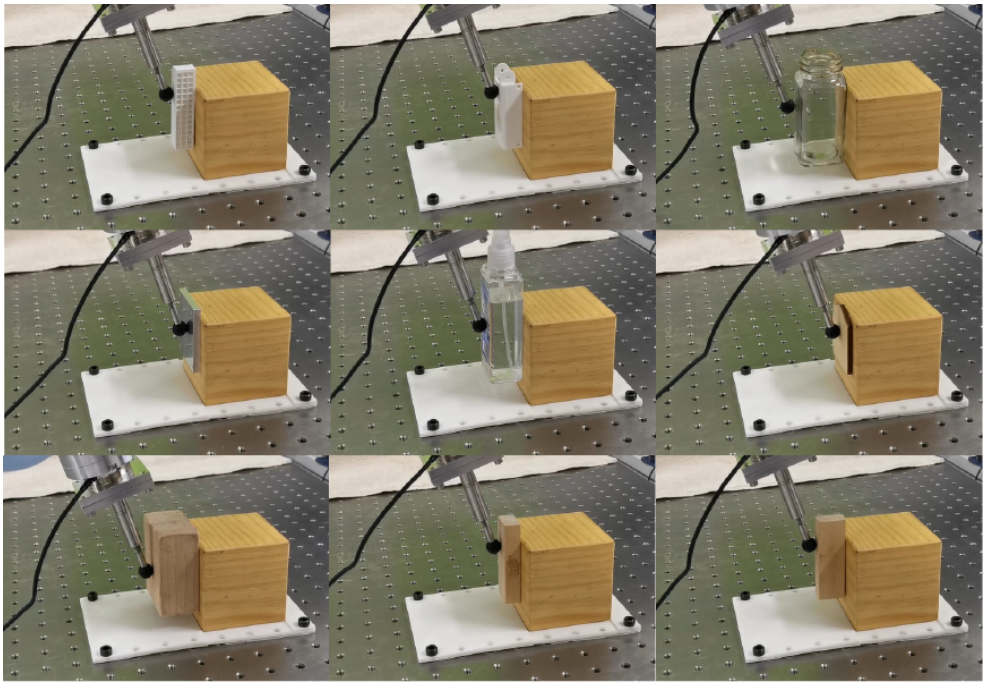

Figure 5.13: The same control law works for different objects.

Chapter 6

Generalizing Shared Grasping to 3D Problems

The derivations in Chapter 5 assumes planar manipulation. Now that efficient contact mode enumeration in 3D is possible with Huang et al. [41], we also propose modifications to generalize shared grasping analysis to 3D problems. We need to revisit the derivation and design of stability margins ((5.25) and (5.26)), which relies on 3D geometry.

Also, in order to model object gravity, the cones will have offsets which means they are no longer homogeneous.

There are two challenges in generalizing the shared grasping analysis to 3D. First, we need to use the right mathematical language to describe geometric objects in 6D and non-holonomic cones because of gravity. Second, we need to handle to exponential grows of computational complexity with dimensionality.

6.1 Modeling of 3D Shared Grasping

Definitions and properties in Section 5.1 translate directly to 3D problems after simply replacing $\mathbb{R}^3$ by $\mathbb{R}^6$. Reuse the definition of generalized velocity vector v and force vector f:

$$v = \begin{bmatrix} {}^H v_O \\ {}^H v_H \end{bmatrix}, f = \begin{bmatrix} {}^H f_O \\ {}^H f_H \end{bmatrix}. \tag{6.1}$$

The only difference is that the quantities ${}^H v_O, {}^H v_H, {}^H f_O, {}^H f_H$ are all six-dimensional vectors instead of three dimensional.

For 3D friction modeling, we approximate the quadratic Coulomb friction cone with m-side polyhedral cone. Since we are considering the gravity of the object, ${}^H f_O$ is not zero. The Newton's Law 5.4 can be re-written as:

$$\mathbf{J'}_{\mathrm{e}}^T \tau_{\mathrm{e}} + {}^H f_O = \mathbf{J'}_{\mathrm{h}}^T \tau_{\mathrm{h}}. \tag{6.2}$$

Again $\mathbf{J'}_{\mathrm{e}}$ and $\mathbf{J'}_{\mathrm{h}}$ are dependent on the contact mode. The meaning of $\mathbf{J'}_{\mathrm{e}}$ and $\mathbf{J'}_{\mathrm{h}}$ has a slight change. Each row of $\mathbf{J'}_{\mathrm{e}} \in \mathbb{R}^{n_{\mathrm{e}} \times 6}$ and $\mathbf{J'}_{\mathrm{h}} \in \mathbb{R}^{n_{\mathrm{h}} \times 6}$ represents a unit generator of the approximated polyhedral friction cone of an environment contact and a hand contact, respectively;

$\tau_{\mathrm{e}} \in \mathbb{R}^{n_{\mathrm{e}}}, \tau_{\mathrm{h}} \in \mathbb{R}^{n_{\mathrm{h}}}$ are non-negative vectors of contact wrench magnitudes. The dimensionality $n_{\mathrm{e}}, n_{\mathrm{h}}$ are determined by the number of contact points and their contact modes. Each contact point contributes up to m wrenches depending on its mode:

- **(Sticking)**: wrenches for all the m generators;
- **(Sliding)**: only the wrench for the sliding direction;
- **(Separation)**: no wrench from the contact point.

From Equation 6.2, we can define the *environment cone* $\mathbb{C}_{\mathrm{e}}$ and the *hand cone* $\mathbb{C}_{\mathrm{h}}$ in the same way as Equation 5.5:

$$\begin{aligned} \mathbb{C}_{\mathrm{e}} &= \{ {\mathbf{J}'}_{\mathrm{e}}^{T} \tau_{\mathrm{e}} + {}^{H}f_{O} \mid \tau_{\mathrm{e}} \geq 0 \} \\ \mathbb{C}_{\mathrm{h}} &= \{ {\mathbf{J}'}_{\mathrm{h}}^{T} \tau_{\mathrm{h}} \mid \tau_{\mathrm{h}} \geq 0 \}. \end{aligned} \tag{6.3}$$

Now $\mathbb{C}_{\mathrm{e}}$ and $\mathbb{C}_{\mathrm{h}}$ are cones in the six dimensional object wrench space, and they are offset from each other by the object gravity ${}^{H}f_{O}$. Their intersection still represent the set of robot wrenches under force equilibrium at this mode, however, the intersection may not be a cone because of the offset. We define this intersection for contact mode m as the *polyhedron of the mode* $\mathbb{P}_{\mathrm{m}}$ instead of the cone of a mode:

$$\mathbb{P}_{m} = \mathbb{C}_{\mathrm{e}} \cap \mathbb{C}_{\mathrm{h}} \tag{6.4}$$

6.2 Wrench Stamping in 6D Space

The wrench stamping algorithm (Procedure WrenchStamping) is the main analysis tool for shared grasping problems. We need to make a few changes to accommodate 3D problems. The main challenge is the significant increase of the number of contact modes. In planar problem, one contact point may have one of four modes: sticking, left sliding, right sliding, and separation. A system with four contact points has at most $4^4 = 256$ possible modes. For 3D problems, there could be infinite sliding directions. If we simplify sliding into eight discrete directions, a system with four contact points may have at most $10^4 = 10,000$ possible modes. The 2D wrench stamping requires computing the cone of every contact mode, the computation time of which is too much for 3D problems.

In this section, we re-design wrench stamping to avoid the need of computing every sliding contact mode. Our algorithm only needs enumeration of sticking-sliding-separation modes, instead of individual sliding directions. The key idea is to use the velocity control for the desired mode to filter out as many sliding directions as possible, then only do the expensive 6D wrench space polyhedral computation on the remaining ones.

The key steps in wrench stamping is summarized as follows:

1. Compute a velocity control for the desired mode, check if crashing could happen;
2. Filter out modes that are not compatible with the velocity control;
3. Project each cone of the remaining modes to the force-controlled subspace;
4. Select a force control in the projection of the desired cone, while staying away from the projection of the other cones.

Comparing with the planar algorithm WrenchStamping, the 3D algorithm does not compute the cones of each mode before the velocity filtering. Before we explain the implementation of those

steps, first we introduce some concepts and algorithms in polyhedral computation. A polyhedron $P \subseteq \mathbb{R}^d$ in d dimensional space can be described as a sum of a convex hull of a finite set of points plus a conical combination of vectors:

$$P = \text{conv}(V) + \text{cone}(Y) \tag{6.5}$$

for some $V \in \mathbb{R}^{d\times n}, Y \in \mathbb{R}^{d\times n'}$. This is called the vertex representation. The polyhedron P can also be described by an intersection of closed halfspaces:

$$P = \{x \in \mathbb{R}^d | Ax \leq z\} \tag{6.6}$$

for some $A \in \mathbb{R}^{m\times d}, z \in \mathbb{R}^m$. This is called the halfspace representation. The *Main Theorem for polyhedra* says the two representations above are equivalent [99], i.e. given a polyhedron in vertex representation, its halfspace representation must exist, and vice versa. Computing the vertex representation of a polyhedron given its halfspace representation is called *vertex enumeration*, the other direction is called *facet enumeration*. Both are basic operations in computational geometry and has efficient algorithms available such as double description [58] and reverse search [7]. Our 3D wrench stamping algorithm calls facet and vertex enumeration multiple times.

Now we are ready to introduce our 3D wrench stamping algorithm.

6.2.1 Step I: Velocity Control and Crashing Check

Same as in Section 5.3.2, the first step is to call OCHS on the desired mode to compute a velocity control $(\mathbf{C}_\text{v}, \omega_\text{av})$:

$$\mathbf{C}_\text{v} V = \omega_\text{av}. \tag{6.7}$$

OCHS also computes the velocity control dimension n_av and the control axes $\mathbf{R}_\text{a}$, which can be used to compute the velocity cone $\mathbb{C}_\text{v} = \{-\mathbf{R}_\text{av}^{-1}\tau_v | \tau_v \geq 0\}$. The next step is to check the crashing condition (5.20), which should be written as

$$\mathbb{P}_\text{AF} \cap \mathbb{C}_\text{v} = \{0\}, \tag{6.8}$$

where $\mathbb{P}_\text{AF} = \mathbb{P}_\text{e,AF} \cap \mathbb{P}_\text{h,AF}$ is the polyhedron of the all-fixed mode. What we have includes the vertex representation of $\mathbb{P}_\text{e,AF}, \mathbb{P}_\text{h,AF}$ and the velocity cone $\mathbb{C}_\text{v}$. To check the condition, we perform facet enumeration on each of them, and perform a linear programming with no cost function to check if the combination of the resulting linear inequalities have a solution.

6.2.2 Step II: Mode Filtering by Velocity Feasibility

The goal of step two is to decide which modes should be considered in the force control. We check the compatibility between the velocity control (6.7) and contact constraint of every contact mode, and discard the modes that are not compatible. The challenge here is how to efficiently handle the many sliding contact modes.

We postpone the decision of sliding direction. When calling the contact mode enumeration algorithm [41], we re-group the contact modes into three state modes, i.e. each contact point

could be in one of sticking, sliding, or separation mode, the sliding direction (which was discretized into m directions) is ignored. We call this contact mode the SSS mode, the total number of which is significantly smaller than the total number of contact modes. If a mode with sliding contact(s) is feasible, we sample several feasible sliding directions and record them.

The contact velocity constraints for a SSS mode includes:

$$\begin{aligned} \mathbf{J}_{\mathrm{m}} v &= 0 \\ \mathbf{M}_{\mathrm{m}} v &\leq 0. \end{aligned} \tag{6.9}$$

The equality constraint includes contact normal and tangential velocity constraint for a sticking contact, normal velocity constraint for a sliding contact, no constraint for a separation contact. The inequality constraint only includes normal velocity constraint for a separating contact.

We examine the feasibility of (6.7) and (6.9) and sample feasible sliding directions as follows. First we solve the equality part of the system $\mathbf{C}_{\mathrm{v}} V = \omega_{\mathrm{av}}, \mathbf{J}_{\mathrm{m}} v = 0$. If the equality part has no solution, the whole system has no solution and we can ignore this contact mode. If the equality part has solution(s), denote the solution space as

$$\mathbf{v} = \{v^* + M\sigma\}, \tag{6.10}$$

the matrix M denotes the null space. If the null space is not empty, we sample several velocities $\mathbf{v}'$ from the solution set, otherwise $\mathbf{v}'$ only contains v^*. Using the rows of the contact Jacobian that corresponds to contact i, $\mathbf{J}_i \in \mathbb{R}^{3\times 12}$, we can project a generalized velocity to the 3D velocity space of each contact:

$$v_i = \mathbf{J}_i v. \tag{6.11}$$

We project each sample in $\mathbf{v}'$ onto each separation contact to check its feasibility, then project it onto each sliding contact to compute its corresponding sliding direction. We record all the feasible SSS contact modes and their feasible velocity samples.

If the system velocity is unique under a contact mode, this approach can obtain its exact sliding direction and avoid any discretization approximation.

6.2.3 Step III: Force Feasibility Check and Projection

For every remaining mode after the filtering in step II, we need to check if the mode has force equilibrium, i.e. check if the polyhedron of the mode $\mathbb{P}_{\mathrm{m}}$ exists. In fact, we can evaluate not only the existence of $\mathbb{P}_{\mathrm{m}}$ but also its robustness against modeling error by computing its geometrical stability margin $\Phi_{\mathrm{g,m}}(\mathbb{P}_{\mathrm{e,m}}, \mathbb{P}_{\mathrm{h,m}})$. We explain the definition of $\Phi_{\mathrm{g,m}}$ in Section 6.3. If the margin is zero, we know the contact mode is F-infeasible.

Next, we project the remaining F-feasible polyhedra onto the force-controlled subspace. Each polyhedron $\mathbb{P}_{\mathrm{m}}$ is the intersection of two cones given by their vertex representation (6.3). Note the coordinates of force-controlled directions are orthogonal to those of the velocity-controlled directions $-\mathbf{R}_{\mathrm{av}}^{-1}$, so we can compute the projection as follows.

First, compute the halfspace representation of the hand cone and the environment cone:

$$\begin{aligned} \mathbb{C}_{\mathrm{e}} &= \{f \in \mathbb{R}^6 | \mathbf{A}_{\mathrm{e}} f \leq \mathbf{b}_{\mathrm{e}}\} \\ \mathbb{C}_{\mathrm{h}} &= \{f \in \mathbb{R}^6 | \mathbf{A}_{\mathrm{h}} f \leq \mathbf{b}_{\mathrm{h}}\}. \end{aligned} \tag{6.12}$$

The coefficients $\mathbf{A}_\mathrm{e}, \mathbf{b}_\mathrm{e}, \mathbf{A}_\mathrm{h}, \mathbf{b}_\mathrm{h}$ are obtained by facet enumeration. Combine the constraints together, we have a (redundant) halfspace representation of the polyhedron of the mode:

$$\mathbb{P}_\mathrm{m} = \mathbb{C}_\mathrm{e} \cap \mathbb{C}_\mathrm{h} = \{f | \mathbf{A}_\mathrm{e} f \leq \mathbf{b}_\mathrm{e}, \mathbf{A}_\mathrm{h} f \leq \mathbf{b}_\mathrm{h}\}. \tag{6.13}$$

Perform vertex enumeration on $\mathbb{P}_m$ to get its generators:

$$\mathbb{P}_\mathrm{m} = \{\mathbf{J}_{0,\mathrm{m}} \tau_0 + \mathbf{J}_{1,\mathrm{m}} \tau_1 | \tau_0 \geq \mathbf{0}, \mathbf{0} \leq \tau_1 \leq 1\}, \tag{6.14}$$

where $\mathbf{J}_{0,\mathrm{m}}$ represents rays, $\mathbf{J}_{1,\mathrm{m}}$ represent vertices. The 0 and 1 are vectors of suitable dimensions. To project a polyhedron $\mathbb{P}_\mathrm{m}$ onto the force-controlled directions $-\mathbf{R}_\mathrm{af}^{-1}$, we first cylindrificate it in the orthogonal directions $-\mathbf{R}_\mathrm{av}^{-1}$ by adding these directions as lines to its generators:

$$\mathbb{P}_{\mathrm{m,v}} = \{\mathbf{J}_{0,\mathrm{m}} \tau_0 + \mathbf{J}_{1,\mathrm{m}} \tau_1 + \mathbf{R}_\mathrm{av}^{-1} \tau_v | \tau_0 \geq 0, 0 \leq \tau_1 \leq 1, \tau_v \in \mathbb{R}^{n_{av}}\}, \tag{6.15}$$

then compute the facet enumeration of the cylindrificated polyhedron:

$$\mathbb{P}_{\mathrm{m,v}} = \{f \in \mathbb{R}^6 | \mathbf{A}_{\mathrm{m,v}} f \leq \mathbf{b}_{\mathrm{m,v}}\}. \tag{6.16}$$

In the cylindrificated polyhedron $\mathbb{P}_{\mathrm{m,v}}$, every facet normal is in the force-controlled subspace. We can extract a halfspace representation of the projection by looking at $\mathbb{P}_{\mathrm{m,v}}$ in the transformed wrench space (Section 3.6.1):

$$\{f \in \mathbb{R}^6 | \mathbf{A}_{\mathrm{m,v}} \mathbf{R}_\mathrm{a}^{-1} f \leq \mathbf{b}_{\mathrm{m,v}}\}. \tag{6.17}$$

The last n_{av} columns of $\mathbf{A}_{\mathrm{m,v}} \mathbf{R}_\mathrm{a}^{-1}$ must be zeros. Denote $\bar{\mathbf{A}}_\mathrm{m}$ as the first n_{af} columns of $\mathbf{A}_{\mathrm{m,v}} \mathbf{R}_\mathrm{a}^{-1}$, denote $\bar{\mathbf{b}}_\mathrm{m} = \mathbf{b}_{\mathrm{m,v}}$, we obtain a halfspace representation of the projection of $\mathbb{P}_\mathrm{m}$ onto the force-controlled subspace, expressed in the force-controlled subspace:

$$\bar{\mathbb{P}}_\mathrm{m} = \{f \in \mathbb{R}^{n_{af}} | \bar{\mathbf{A}}_\mathrm{m} f \leq \bar{\mathbf{b}}_\mathrm{m}\}. \tag{6.18}$$

By computing $\hat{\bar{\mathbb{P}}}_\mathrm{m}$, we have "stamped" the wrench space polyhedra.

6.2.4 Step IV: Force Control

Now we have the polyhedron projections $\bar{\mathbb{P}}_\mathrm{m}$. The final step of 3D wrench stamping is to select a wrench in the projection of the desired polyhedron, $\bar{\mathbb{P}}_\mathrm{goal}$, while staying away from the projection of other polyhedra to maximize stability. To limit the force magnitude, denote F_L as the maximum allowed wrench magnitude. Maximizing the distance with polyhedra is a non-convex optimization problem. Unlike the planar case (Section 5.3.3) where the force control can be selected based on heuristics, for 3D problem we design a non-convex optimization algorithm for finding the force control.

The algorithm iteratively performs hyperplane approximation to all the undesired polyhedra, then compute the largest inscribed sphere inside of those hyperplanes, as shown in Algorithm 5. In the algorithm, "InscribeSphere(A, b, A_1, b_1, L)" is a linear programming formulation

Algorithm 5: Iterative Inscribed Sphere

Input: $\bar{\mathbb{P}}_{\mathrm{m}} = \{f \in \mathbb{R}^{n_{af}} | \bar{\mathbf{A}}_{\mathrm{m}} f \leq \bar{\mathbf{b}}_{\mathrm{m}}\}, m \in \{1, 2, ...\}$
Input: $\bar{\mathbb{P}}_{\mathrm{goal}} = \{f \in \mathbb{R}^{n_{af}} | \bar{\mathbf{A}}_{\mathrm{goal}} f \leq \bar{\mathbf{b}}_{\mathrm{goal}}\}$
Input: Initial point w_0, Maximum magnitude F_L
Output: Solution w, distance d

$w \leftarrow w_0, d_{\mathrm{improvement}} \leftarrow \infty, d \leftarrow 0$;
while $d_{\mathrm{improvement}} > \Delta$ **do**
  Solve quadratic programming $x_m \leftarrow \arg\min_x \|x - w\|, \text{ s.t.} \mathbf{A}_{\mathrm{m}} x \leq \mathbf{b}_{\mathrm{m}}$ to find the point x_m in each $\bar{\mathbb{P}}_{\mathrm{m}}$ that is closest to w.;
  $d_{\mathrm{improvement}} \leftarrow \min_m \|x_m - w\| - d$;
  $d \leftarrow \min_m \|x_m - w\|$;
  if $d = 0$ **then**
    return *No solution, since w is infeasible.*
  Construct hyperplanes $(x - x_m)^T (w - x_m)/\|w - x_m\| \geq 0, \forall m$, combine all of them into $\mathbf{A}'x \leq \mathbf{b}'$;
  $w \leftarrow \mathrm{InscribeSphere}(\mathbf{A}', \mathbf{b}', \mathbf{A}_{\mathrm{goal}}, \mathbf{b}_{\mathrm{goal}}, F_L)$;
end
return (w, d)

that solves for the largest inscribed sphere in a convex polyhedron $Ax \leq b$ subject to additional linear constraints $A_1 x \leq b_1, -L \leq x \leq L$ on the sphere center location:

$$\begin{aligned} x = \max_{r,x} \quad & r \\ s.t. \quad & A_1 x \leq b_1 \\ & r \geq 0 \\ & -L \leq x \leq L \end{aligned} \tag{6.19}$$

where $-L \leq x \leq L$ should be interpreted as an element-wise constraint.

Since the force control problem is non-convex, we sample multiple start points then run Algorithm 5 from each of them. When attacking non-convexity by multiple initial start points, it is critical to have those initial points well-distributed. To perform uniform sampling in high dimensional polyhedron, we modify the hit-and-run sampling algorithm [45][9] for polytope to accommodate unbounded polyhedron. The complete algorithm for force control is summarized in Algorithm 6.

The wrench w computed by Algorithm 6 is our force control expressed in the force-controlled subspace, which is exactly the force control magnitude vector w_{af}. This completes the HFVC and concludes our 3D wrench stamping algorithm.

6.3 Stability Margins in 3D

Similar to Section 5.4, we can define the two stability margins in 3D. The wrench space distance d returned by the force control algorithm 6 is exactly the *control stability margin* Φ_C, which

Algorithm 6: Force Control

Input: Desired polyhedron $\bar{\mathbb{P}}_{\text{goal}}$, other polyhedra $\{\bar{\mathbb{P}}_{\text{m}}\}$
Input: Maximum magnitude F_L, number of initial points K_{init}
Output: Solution w, distance d

Get an internal point :$w_0 \leftarrow \text{InscribeSphere}(\bar{\mathbf{A}}_{\text{goal}}, \bar{\mathbf{b}}_{\text{goal}}, -, -, F_L)$;
Use w_0 to seed hit-and-run sampling in $\bar{\mathbb{P}}_{\text{goal}}$ and sample $100K_{\text{init}}$ uniformly distributed points;
Sample $100K_{\text{init}}$ groups of K_{init} points, pick the group with the maximum minimal distances between points, denote as $\{w_1, w_2, ..., w_{K_{\text{init}}}\}$.;
foreach w_i **do**
    (w_i, d_i) =IterativeInscribedSphere($\{\bar{\mathbb{P}}_{\text{m}}\}, \bar{\mathbb{P}}_{\text{goal}}, w_i, F_L$);
end
return (w_i, d_i) *with the largest* d_i

describes the magnitude of the disturbance force on our robot action w that could potentially drive the robot action into another mode.

The *geometrical stability margin* describes the robustness of the system against modeling uncertainties, i.e. uncertainties in the contact Jacobians $\mathbf{J'}_{\text{e}}$ and $\mathbf{J'}_{\text{h}}$. These uncertainties affects the shapes of the hand cone and the environment cone (6.3). When they don't have an intersection, the polyhedron of the mode $\mathbb{P}_m$ (6.4) does not exist and the system has no force equilibrium. We can express the feasibility as the following statement:

Statement 6.3.1. *Given* $\mathbf{J'}_{\text{e}}$, $\mathbf{J'}_{\text{h}}$, *and* ${}^H f_O$, *there exists a solution* $(\tau_1, \tau_2) \geq 0$ *such that:*

$$\mathbf{J'}_{\text{h}}\tau_1 = \mathbf{J'}_{\text{e}}\tau_2 + {}^H f_O. \tag{6.20}$$

We need to evaluate how much uncertainties in $\mathbf{J'}_{\text{h}}, \mathbf{J'}_{\text{e}}$ can exist while the above statement holds. We approach the problem in three steps. First, we reformulate the above statement to make it suitable for perturbation analysis. Second, we propose a quantification of uncertainty in $\mathbf{J'}_{\text{h}}, \mathbf{J'}_{\text{e}}$ under which the statement still holds. Finally, we provide implementations to compute the quantification. $\mathbf{J'}_{\text{h}} \in \mathbb{R}^{n\times n_h}, \mathbf{J'}_{\text{e}} \in \mathbb{R}^{n\times n_e}$, $n = 6$ for 3D shared grasping. However, all the derivations below in this section do not assume a certain dimensionality.

6.3.1 The Reformulation

Denote

$$\begin{aligned} A &= [\mathbf{J'}_{\text{h}}, -\mathbf{J'}_{\text{e}}] \in \mathbb{R}^{n\times(n_e+n_h)} \\ b &= {}^H f_O \in \mathbb{R}^n \\ x &= \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \in \mathbb{R}^{n_e+n_h}, \end{aligned} \tag{6.21}$$

We can rewrite statement 6.3.1 in a compact form:

Statement 6.3.2. *There exists* x *such that*

$$Ax = b, x \geq 0. \tag{6.22}$$

In this formulation, the uncertainties exist in A. To take trivial solutions (e.g. when $\mathbb{P}_{\mathrm{h}}$ and $\mathbb{P}_{\mathrm{e}}$ intersects at one single point only) out of our discussion, we introduce the following lemma:

Lemma 2. *Statement 6.3.2 holds for a small perturbation $\Delta_A \in \mathbb{R}^{n\times(n_e+n_h)}$, i.e. there exists x such that*

$$(A + \Delta_A)x = b, x \geq 0, \tag{6.23}$$

if and only if the following condition holds: for any $y \in \mathbb{R}^n$, there exists $x \in \mathbb{R}^{n_e+n_h}, t \in \mathbb{R}$ such that

$$Ax - tb = y, x, t \geq 0 \tag{6.24}$$

Geometrically, the condition says that the rays $G_1, -G_2$, and $-b$ span the whole n dimensional space.

Proof. **Sufficiency:**
First, we show that $Ax = b, x \geq 0$ is feasible. Taking $y = b$, (6.24) becomes

$$Ax - tb = b, \tag{6.25}$$

$$A\frac{x}{1+t} = b. \tag{6.26}$$

$x, t \geq 0$, so $\frac{x}{1+t} \geq 0$ is a solution to $Ax = b, x \geq 0$. Next, we show the feasibility considering Δ_A. Suppose (6.23) is infeasible, i.e. vector b is not in the cone spanned by columns of $A + \Delta_A$. Then from Farkas' lemma, there must exist a hyperplane that separates vector b from rays $A+\Delta_A$:

$$c(A + \Delta_A) > 0, \tag{6.27}$$

$$cb \leq 0, \tag{6.28}$$

where c^T is the normal of the hyperplane. Taking the limit $\Delta_A \to 0$, Equation (6.27) results in

$$cA \geq 0. \tag{6.29}$$

Now, take $y = -c^T$, (6.24) says there must exist $x \geq 0, t \geq 0$ such that

$$Ax - tb = -c^T. \tag{6.30}$$

Multiply both sides with c:

$$cAx - cbt = -cc^T. \tag{6.31}$$

However, (6.31) is not possible because $cAx \geq 0, -cbt \geq 0, -cc^T \leq 0$. As a result, (6.23) must be feasible.

Necessity:
First, we prove the proposition for $b \neq 0$. Given any y, from (6.23) we know the following holds for a small enough ϵ:

$$(A - \epsilon y\mathbf{1}^T)x = b, x \geq 0 \tag{6.32}$$

where $\mathbf{1}^T$ denotes a row vector of ones. Since $b \neq 0$, we know $x \neq 0$. Expand it into

$$Ax - b = \epsilon y \mathbf{1}^T x. \tag{6.33}$$

Since $x \geq 0, x \neq 0$, we know $\epsilon \mathbf{1}^T x > 0$, so

$$A\frac{x}{\epsilon \mathbf{1}^T x} - \frac{b}{\epsilon \mathbf{1}^T x} = y, \tag{6.34}$$

which proves (6.24).

If $b = 0$, make the same choice on Δ_A and (6.32) becomes:

$$(A - \epsilon y \mathbf{1}^T)x = 0, x \geq 0 \tag{6.35}$$

so

$$Ax = \epsilon y \mathbf{1}^T x = \epsilon \mathbf{1}^T x y \tag{6.36}$$

$$A\frac{x}{\epsilon \mathbf{1}^T x} = y, \tag{6.37}$$

this proves (6.24) for $b = 0$. □

6.3.2 The Quantification

We quantify the robustness of (6.24) in the following theorem.

Theorem 3. Geometrical Stability Margin

Define r by

$$r = \max\{r : B(0, r) \subseteq \{Ax - tb : x, t \geq 0, ||x|| \leq 1\}\} \tag{6.38}$$

Then if

$$||\Delta_A|| < r \tag{6.39}$$

The following system must be feasible for any y:

$$(A + \Delta_A)x - tb = y, x, t \geq 0 \tag{6.40}$$

Proof. Suppose $(A + \Delta_A)x - tb = y, x, t \geq 0$ is infeasible. We need to show that $||\Delta_A|| \geq r$. Again by Farkas' Lemma, there exists $\|v\| = 1$ such that

$$v^T y < 0 \leq v^T[(A + \Delta_A)x - tb], \quad \forall x, t \geq 0 \tag{6.41}$$

The right hand side is greater than zero for any $x, t \geq 0$, so we must have

$$(A + \Delta_A)^T v \geq 0 \tag{6.42}$$

$$-b^T v \geq 0 \tag{6.43}$$

Now pick $(\bar{x}, \bar{t}) \geq 0$ such that

$$A\bar{x} - \bar{t}b = -rv, ||\bar{x}|| \leq 1 \tag{6.44}$$

The existence of such choice comes directly from the definition of r. Then

$$\begin{aligned} 0 &\leq v^T \left[(A+\Delta_A)\bar{x} - t\bar{b}\right] \\ &= v^T(A\bar{x} - \bar{t}b) + v^T\Delta_A\bar{x} \\ &= -rv^Tv + v^T\Delta_A\bar{x} \\ &= -r + v^T\Delta_A\bar{x} \end{aligned} \tag{6.45}$$

So

$$v^T\Delta_A\bar{x} \geq r. \tag{6.46}$$

Finally, using $\|\bar{x}\| \leq 1, \|v\| = 1$, and Cathy-Schwarz Inequality, we finally have:

$$||\Delta_A|| \geq ||v||\,||\Delta_A||\,||\bar{x}|| \geq v^T\Delta_A\bar{x} \geq r. \tag{6.47}$$

□

6.3.3 The Implementation

Now, evaluating geometrical stability margin is to compute the r defined in Equation (6.38)

$$r = \max\{r : B(0,r) \subseteq \{Ax - tb : x, t \geq 0, \|x\| \leq 1\}\} \tag{6.48}$$

given A and b defined in (6.21). This is equivalent to

$$r = \max\{r : B(0,r) \subseteq \{Ax - tb : x, t \geq 0, \|x\| = 1\}\} \tag{6.49}$$

Further, we can make our lives easier by picking the 1-norm, then $\|x\| = \mathbf{1}^T x$ for $x \geq 0$. Then we are looking for the distance from origin to the boundary of a polyhedron:

$$\mathbb{P}_{\text{geo}} = \{Ax - tb : x, t \geq 0, \mathbf{1}^T x = 1\} = conv(A) + ray(-b) \tag{6.50}$$

This polyhedron is constructed from vertices that corresponds to contact wrenches, and a ray corresponds to the gravity vector. This is closely related to, but different from the grasping metric that measures the maximum resistance wrench [70]. In such grasping metric, the maximum resistance wrench is computed as the radius of the ball inscribed to a different polyhedron that represents the set of possible wrenches applied to the object. This polyhedron has one vertex corresponds to the finite gravity wrench, and rays corresponds to contact forces. For any force closure grasps, this polyhedron span the whole 6D wrench space so it provides no method to evaluate and compare the stability against modeling uncertainties. The polyhedron defined in Equation (6.50), however, never spans the whole space since it only has one infinite ray. We call it the *Geometrical Stability Polyhedron* $\mathbb{P}_{\text{geo}}$. Run facet enumeration on $\mathbb{P}_{\text{geo}}$, obtain its halfspace representation:

$$\mathbb{P}_{\text{geo}} = \{f \in \mathbb{R}^6 | \mathbf{A}_{\text{geo}} f \leq \mathbf{b}_{\text{geo}}\} \tag{6.51}$$

Then the distance from the origin to the boundary of $\mathbb{P}_{\text{geo}}$ can be computed as:

$$r = \min_i \frac{\mathbf{b}_{\text{i,geo}}}{\|\mathbf{A}_{\text{i,geo}}\|} \tag{6.52}$$

where $\mathbf{b}_{\text{i,geo}}$ and $\mathbf{A}_{\text{i,geo}}$ denotes the ith row of $\mathbf{b}_{\text{geo}}$ and $\mathbf{A}_{\text{geo}}$, respectively. If any element of $\mathbf{b}_{\text{geo}}$ is negative, the origin is outside of the polyhedron and the system has no force equilibrium.

6.4 Implementation and Computational Efficiency

We implemented the 3D wrench stamping algorithm in C++. We use the Parma Polyhedra Library (PPL) [8] for common polyhedral computations including facet and vertex enumeration, adding generators, and redundancy removal. PPL implements rational arithmetic that performs exact computation. We tested our implementation on a desktop with i7-9700k CPU clocked at 4.7GHz.

Considering the complexity of polyhedral computations, we make two compromises to reduce the time complexity of our algorithm. First, we use four-sided pyramid to approximate a friction cone, i.e. each contact point contributes at most four contact wrenches. Second, we assume the finger contact is always sticking in our testing. We test our algorithm on four different problem settings and show the results in Table 6.1. In the results, step I and II are relatively fast because they do not involve polyhedral computation. The computation time variations in step I and II mainly come from the number of environment contact points, which determines the number of modes to consider in the velocity filtering. The computation of geometrical stability margin and the projection both are dominated by polyhedral computations including facet/vertex enumeration. Their computation time grows quickly with the complexity of the polyhedrons, which is determined by the total number of contact wrenches. The time complexity of the force control step depends on the dimensionality of the force-controlled subspace and the complexity of the polyhedrons in it. We use three to eight initial solutions, depending on the dimensionality. The iterative inscribed sphere is limited to at most three iterations.

Table 6.1: Time Consumption Statistics of the 3D Wrench Stamping Algorithm

Problem Settings	Hand Contacts	2 sticking	2 sticking	2 sticking	4 sticking
	Env Contacts (Goal)	2 sliding	4 sliding	2 sticking	4 sliding
	# of contact wrenches	10	12	16	20
Average Time Statistics (ms)	Step I & II	3.1	9.2	3.9	11.03
	Step III Compute r	1.3	3.0	8.0	30.0
	Step III Projection	1.9	8.2	14.6	43.0
	Step IV Force Control	3.5	5.1	60.9	23.5
	Total	9.8	25.5	87.4	107.3

Chapter 7

Summary

This thesis makes the following contributions:

- **Modeling** of rigid body manipulation that explains the causes of failures related to contacts. Our analysis demonstrates the advantages and necessity of using hybrid force-velocity control to handle manipulation under contact constraints. Our modeling uses a contact mode representation that can distinguish the wrenches in different modes from each other.
- **Algorithms** that compute a HFVC to control the object motion while enforcing a desired contact mode. The two hybrid servoing algorithms can efficiently computes the HFVC with the best kinematic conditioning, while the wrench stamping algorithm can additionally avoid unexpected contact modes.
- **Metrics** to quantify the quasi-static stability of a manipulation system under any contact mode. The crashing index quantifies the robustness of a system against the dangerous kinematic ill-conditioning. The geometric stability margin evaluates the resistance of a contact mode against modeling uncertainties, including variations on contact point locations, normals, and their friction coefficients. The control stability margin evaluates the minimal force disturbance that is required to break the contact mode.
- **Applications** of the proposed algorithms and metrics in solving several common manipulation problems, including contact mode selection, motion planning of robust trajectory, finger placement optimization, and computing the range of feasible parameters.
- **Experimental validations** of the applications in several representative scenarios. We implemented HFVC on industrial robots and solved tasks that are infeasible without utilizing environmental contact. Several of the tasks involve a sequence of making and breaking contacts. We run each experiment multiple times to demonstrate the reliability of the method.

There are two future work directions that can potentially improve the performance of the proposed methods. The first is to incorporate feedback control. All experiments demonstrated in this thesis are open loop in the sense that the robot has no object state feedback. Apart from time limitation, we choose open loop experiments to demonstrate the robustness of our method is enough to handle the uncertainties. However, the disturbance rejection ability and tracking performance could be even better if feedback is indeed available.

The second is to use high force-control bandwidth robots. As we explained in the introduction, most industrial robots, including the ones used in this thesis, are not designed to make contacts. This is why all the experiments are slow comparing with human motion for the same task. Our quasi-static modeling is not the limitation of speed, as explained in Section 3.3. There is a potential of speed improvement using robot designs such as the Direct-Drive Hand [11] to execute the hybrid force-velocity controls computed by our methods.

To conclude, in this thesis we show that it is possible to plan robust contact-rich manipulation efficiently and execute contact-rich manipulation reliably. It is our hope that our modeling framework, controller designs, and stability metrics could motivate research work on more dexterous manipulation skills with contacts, and ultimately make it easier to deploy contact-rich manipulation in the real world.

Bibliography

[1] Yasumichi Aiyama, Masayuki Inaba, and Hirochika Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, volume 1, pages 136–143. IEEE, 1993. 2.1.3, 2.2.3

[2] Srinivas Akella and Matthew T. Mason. Posing polygonal objects in the plane by pushing. *IJRR*, 17(1):70–88, January 1998. 2.2.1

[3] Jorge Angeles and Carlos S López-Cajún. Kinematic isotropy and the conditioning index of serial robotic manipulators. *The International Journal of Robotics Research*, 11(6):560–571, 1992. 4.1

[4] Jorge Angeles, Farzam Ranjbaran, and Rajnikant V Patel. On the design of the kinematic structure of seven-axes redundant manipulators for maximum conditioning. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 494–495. IEEE Computer Society, 1992. 4.1

[5] Brian Armstrong-Hélouvry, Pierre Dupont, and Carlos Canudas De Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30(7):1083–1138, 1994. 2.1.1, 3

[6] Haruhiko Asada and Takeo Kanade. Design of direct-drive mechanical arms. 1983. 3.4

[7] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete applied mathematics*, 65(1-3):21–46, 1996. 6.2

[8] Bagnara, Roberto and Hill, Patricia M. and Zaffanella, Enea and Bagnara, Abramo. Parma polyhedra library. URL `https://www.bugseng.com/parma-polyhedra-library`. 6.4

[9] Tim Benham. Uniform distribution over a convex polytope. URL `https://www.mathworks.com/matlabcentral/fileexchange/34208-uniform-distribution-over-a-convex-polytope`. 11

[10] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner. Manipulation planning on constraint manifolds. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 625–632. IEEE, 2009. 2.3

[11] Ankit Bhatia, Aaron M. Johnson, and Matthew T. Mason. Direct drive hands: Force-motion transparency in gripper design. In *Proceedings of Robotics: Science and Systems*, June 2019. 7

[12] George EP Box. Science and statistics. *Journal of the American Statistical Association*, 71 (356):791–799, 1976. 3

[13] David L Brock. Enhancing the dexterity of a robot hand using controlled slip. In *1988 IEEE International Conference on Robotics and Automation*, pages 249–251, 1988. 2.1.1, 2.1.3, 2.2.3

[14] Randy C Brost. Automatic grasp planning in the presence of uncertainty. *The International Journal of Robotics Research*, 7(1):3–17, 1988. 2.2.1

[15] Thomas Buschmann, Sebastian Lohmeier, and Heinz Ulbrich. Biped walking control based on hybrid position/force control. pages 3019–3024, 2009. 2.2.3

[16] Nikhil Chavan-Dafle and Alberto Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6215–6222, 2015. 2.1.1, 2.1.2, 2.1.3, 3.2

[17] Nikhil Chavan-Dafle and Alberto Rodriguez. Sampling-based planning of in-hand manipulation with external pushes. *arXiv preprint arXiv:1707.00318*, 2017. 2.1.2, 2.3

[18] Nikhil Chavan-Dafle, Rachel Holladay, and Alberto Rodriguez. In-hand manipulation via motion cones. *Robotics: Science and Systems*, 2018. 2.1.2

[19] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. 2021. 5.5.3, 5.6.3

[20] Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S Srinivasa, Michael Erdmann, Matthew T Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1578–1585. IEEE, 2014. 2.1.3

[21] Joris De Schutter, Herman Bruyninckx, Wen-Hong Zhu, and Mark W Spong. Force control: a bird's eye view. In *Control Problems in Robotics and Automation*, pages 1–17. Springer, 1998. 2.2.3

[22] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007. 2.3

[23] C Canudas De Wit, Henrik Olsson, Karl Johan Astrom, and Pablo Lischinsky. A new model for control of systems with friction. *IEEE Transactions on Automatic Control*, 40 (3):419–425, 1995. 2.1.1

[24] Niels Dehio, Joshua Smith, Dennis Leroy Wigand, Guiyang Xin, Hsiu-Chin Lin, Jochen J Steil, and Michael Mistry. Modeling and control of multi-arm and multi-leg robots: Compensating for object dynamics during grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 294–301. IEEE, 2018. 2.2.3

[25] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017. 2.3

[26] Yasutaka Fujimoto and Atsuo Kawamura. Proposal of biped walking control based on robust hybrid position/force control. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2724–2730. IEEE, 1996. 2.2.3

[27] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991. 2.1.2

[28] Basile Graf. Quaternions and dynamics. *arXiv preprint arXiv:0811.2889*, 2008. 4.5.1

[29] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017. 2.3

[30] Weiqiao Han and Russ Tedrake. Local trajectory stabilization for dexterous manipulation via piecewise affine approximations. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8884–8891. IEEE, 2020. 2.2.2, 2.3

[31] François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. *arXiv preprint arXiv:1611.08268*, 2016. 2.2.1, 2.2.2

[32] Neville Hogan. Impedance control: An approach to manipulation: Part ii—implementation. *Journal of dynamic systems, measurement, and control*, 107(1):8–16, 1985. 2.2.3

[33] Anne Holladay, Robert Paolini, and Matthew T Mason. A general framework for open-loop pivoting. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3675–3681, 2015. 2.1.3

[34] Yifan Hou and Matthew T Mason. Criteria for maintaining desired contacts for quasi-static systems. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, November 2019. 4.1, 4.2.2

[35] Yifan Hou and Matthew T Mason. Robust execution of contact-rich motion plans by hybrid force-velocity control. In *International Conference on Robotics and Automation (ICRA) 2019*. IEEE Robotics and Automation Society (RAS), May 2019. 4.2, 4.2.2, 4.4.1

[36] Yifan Hou, Zhenzhong Jia, Aaron M Johnson, and Matthew T Mason. Robust planar dynamic pivoting by regulating inertial and grip forces. In *The 12th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Springer, 2016. 2.1.3, 2.2.2

[37] Yifan Hou, Zhenzhong Jia, and Matthew T Mason. Fast planning for 3d any-pose-reorienting using pivoting. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1631–1638. IEEE, 2018. 2.1.3

[38] Yifan Hou, Zhenzhong Jia, and Matthew T Mason. Manipulation with shared grasping. *Robotics: Science and Systems*, 2020. 4.4.1

[39] Robert D Howe and Mark R Cutkosky. Practical force-motion models for sliding manipulation. *IJRR*, 15(6):557–572, 1996. 2.1.2

[40] Eric Huang, Ankit Bhatia, Byron Boots, and Matthew Mason. Exact bounds on the contact driven motion of a sliding object, with applications to robotic pulling. In *Proceedings of Robotics: Science and Systems*, July 2017. 2.1.2

[41] Eric Huang, Xianyi Cheng, and Matthew T Mason. Efficient contact mode enumeration in 3d. In *Workshop on the Algorithmic Foundations of Robotics*, 2020. 5.4.2, 6, 6.2.2

[42] Imin Kao and Mark R Cutkosky. Quasistatic manipulation with compliance and sliding. *The International journal of robotics research*, 11(1):20–40, 1992. 2.1.2

[43] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. 2.2.3

[44] Michael Koval. *Robust Manipulation via Contact Sensing*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, September 2016. 2.3

[45] Dirk P Kroese, Thomas Taimre, and Zdravko I Botev. *Handbook of monte carlo methods*, volume 706. John Wiley & Sons, 2013. 11

[46] António Lopes and Fernando Almeida. A force–impedance controlled industrial robot using an active robotic auxiliary device. *Robotics and Computer-Integrated Manufacturing*, 24(3):299–309, 2008. 2.2.3

[47] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996. 2.2.1

[48] Kevin M Lynch, Hitoshi Maekawa, and Kazuo Tanie. Manipulation and active sensing by pushing using tactile feedback. In *IROS*, pages 416–421, 1992. 2.1.1, 2.2.1

[49] J. Maples and J. Becker. Experiments in force control of robotic manipulators. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 695–702, Apr 1986. doi: 10.1109/ROBOT.1986.1087590. 2.2.3, 4.6.1

[50] Matthew T Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, 1981. 2.2.3

[51] Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986. 2.1.2

[52] Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001. 5.2.1

[53] N Harris McClamroch and Danwei Wang. Feedback stabilization and tracking of constrained robots. *IEEE Transactions on Automatic Control*, 33(5):419–426, 1988. 2.2.3

[54] James K Mills and Andrew A Goldenberg. Force and position control of manipulators during constrained motion tasks. *IEEE Transactions on Robotics and Automation*, 5(1): 30–46, 1989. 2.2.3

[55] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144. Eurographics Association, 2012. 2.3

[56] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43, 2012. 2.3

[57] Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids. In *Intelligent Robots and Systems*

(IROS), 2015 IEEE/RSJ International Conference on, pages 5307–5314. IEEE, 2015. 2.3

[58] Theodore S Motzkin, Howard Raiffa, Gerald L Thompson, and Robert M Thrall. The double description method. *Contributions to the Theory of Games*, 2(28):51–73, 1953. 6.2

[59] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994. 2.2.3, 3.3, 4.2.1, 5.1.1, 5.2.1

[60] Fusaomi Nagata, Tetsuo Hase, Zenku Haga, Masaaki Omoto, and Keigo Watanabe. Cad/cam-based position/force controller for a mold polishing robot. *Mechatronics*, 17(4-5): 207–216, 2007. 2.2.3

[61] Henrik Olsson, Karl J Åström, C Canudas De Wit, Magnus Gäfvert, and Pablo Lischinsky. Friction models and friction compensation. *European journal of control*, 4(3):176–195, 1998. 2.1.1

[62] Hyeonjun Park, Jaeheung Park, Dong-Hyuk Lee, Jae-Han Park, Moon-Hong Baeg, and Ji-Hun Bae. Compliance-based robotic peg-in-hole assembly strategy without force feedback. *IEEE Transactions on Industrial Electronics*, 64(8):6299–6309, 2017. 2.2.3

[63] Amir Patel, Stacey Leigh Shield, Saif Kazi, Aaron M Johnson, and Lorenz T Biegler. Contact-implicit trajectory optimization using orthogonal collocation. *IEEE Robotics and Automation Letters*, 4(2):2242–2249, 2019. 1

[64] Michael A Peshkin and Arthur C Sanderson. The motion of a pushed, sliding workpiece. *IEEE Journal on Robotics and Automation*, 4(6):569–598, 1988. 2.1.2

[65] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014. 2.2.2, 2.3

[66] Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373. IEEE, 2016. 2.2.2

[67] Gill A Pratt and Matthew M Williamson. Series elastic actuators. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 399–406. IEEE, 1995. 3.4

[68] Marc H Raibert and John J Craig. Hybrid position/force control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 103(2):126–133, 1981. 2.2.3, 3.6.2

[69] Anil Rao, David J Kriegman, and Kenneth Y Goldberg. Complete algorithms for feeding polyhedral parts using pivot grasps. *IEEE Transactions on Robotics and Automation*, 12 (2):331–342, 1996. 2.1.3

[70] Máximo A Roa and Raúl Suárez. Grasp quality measures: review and performance. *Autonomous robots*, 38(1):65–88, 2015. 5.2.1, 6.3.3

[71] J Kenneth Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, volume 19, pages 95–100. IEEE, 1980. 2.2.3

[72] J Kenneth Salisbury. Whole-arm manipulation. In *Proc. of the 4th Int. Symp. of Robotics*

Research. Published by the MIT Press, 1987. 5

[73] J Kenneth Salisbury and John J Craig. Articulated hands: Force control and kinematic issues. *The International journal of Robotics research*, 1(1):4–17, 1982. 4.1

[74] Naoyuki Sawasaki, Masayuki Inaba, and Hirochika Inoue. Tumbling objects using a multi-fingered robot. In *Proceedings of the 20th International Symposium on Industrial Robots and Robot Exhibition*, pages 609–616, 1989. 2.1.3, 2.2.3

[75] Jian Shi, J Zachary Woodruff, Paul B Umbanhowar, and Kevin M Lynch. Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795, 2017. 2.1.3

[76] Arne Sieverling, Clemens Eppner, Felix Wolff, and Oliver Brock. Interleaving motion in contact and in free space for planning under uncertainty. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4011–4073. IEEE, 2017. 2.3

[77] Avishai Sintov, Or Tslil, and Amir Shapiro. Robotic Swing-Up regrasping manipulation based on the Impulse–Momentum approach and cLQR control. *Ieee T Robot*, 32(5):1079–1090, 2016. ISSN 1552-3098. doi: 10.1109/TRO.2016.2593053. 2.1.3

[78] David E Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996. 2.3, 4.5.3

[79] G. W. Stewart. Collinearity and least squares regression. *Statistical Science*, 2(1):68–84, 1987. ISSN 0883-4237. 4.1

[80] Raúl Suárez, Jordi Cornella, and Máximo Roa Garzón. *Grasp quality measures*. Citeseer, 2006. 5

[81] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012. 2.2.2

[82] Hajime Terasaki and Tsutomu Hasegawa. Motion planning of intelligent manipulation by a parallel two-fingered gripper equipped with a simple rotating mechanism. *IEEE Transactions on Robotics and Automation*, 14(2):207–219, 1998. 2.1.3

[83] Marc Toussaint, Kelsey Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems*, 2018. 2.3

[84] Masaru Uchiyama and Pierre Dauchez. A symmetric hybrid position/force control scheme for the coordination of two robots. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 350–356. IEEE, 1988. 2.2.3

[85] Francisco E Vi, Yiannis Karayiannidis, Christian Smith, Danica Kragic, et al. Adaptive control for pivoting with visual and tactile feedback. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 399–406. 2.1.3

[86] Luigi Villani and Joris De Schutter. Force control. In *Springer handbook of robotics*, pages 161–185. Springer, 2008. 3.3

[87] B Vina, E Francisco, Yiannis Karayiannidis, Karl Pauwels, Christian Smith, and Danica

Kragic. In-hand manipulation using gravity and controlled slip. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5636–5641. 2.1.3

[88] Dustin J Webb and Jur Van Den Berg. Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE International Conference on Robotics and Automation*, pages 5054–5061. IEEE, 2013. 2.3

[89] Harry West and Haruhiko Asada. A method for the design of hybrid position/force controllers for manipulators constrained by contact with the environment. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 251–259. IEEE, 1985. 2.2.3

[90] Daniel E Whitney. Historical perspective and state of the art in robot force control. *The International Journal of Robotics Research*, 6(1):3–14, 1987. 2.2.3

[91] Daniel E Whitney. *Mechanical assemblies: their design, manufacture, and role in product development*, volume 1. Oxford university press New York, 2004. 3.5.1

[92] J Wiegley, K Goldberg, M Peshkin, and M Brokowski. A complete algorithm for designing passive fences to orient parts. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1133–1139. IEEE, 1996. 2.2.1

[93] Eiichi Yoshida, Mathieu Poirier, Jean-Paul Laumond, Oussama Kanoun, Florent Lamiraux, Rachid Alami, and Kazuhito Yokoi. Pivoting based manipulation by a humanoid robot. *Autonomous Robots*, 28(1):77–88, 2010. 2.1.3, 2.2.3

[94] Tsuneo Yoshikawa. Dynamic hybrid position/force control of robot manipulators–description of hand constraints and calculation of joint driving force. *IEEE Journal on Robotics and Automation*, 3(5):386–392, 1987. 2.2.3, 4.3.3

[95] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. 2018. 2.3

[96] Jiaji Zhou, Robert Paolini, J Andrew Bagnell, and Matthew T Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377. IEEE, 2016. 2.1.2

[97] Jiaji Zhou, Robert Paolini, Aaron M Johnson, J Andrew Bagnell, and Matthew T Mason. A probabilistic planning framework for planar grasping under uncertainty. *IEEE Robotics and Automation Letters*, 2(4):2111–2118, 2017. 2.1.2, 2.3

[98] Jiaji Zhou, Yifan Hou, and Matthew T Mason. Pushing revisited: Differential flatness, trajectory planning, and stabilization. *The International Journal of Robotics Research*, 38 (12-13):1477–1489, 2019. 2.2.1

[99] Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 2012. 6.2