Force-and-Motion Constrained Planning for Tool Use

by

Rachel Mara Holladay

B.S, Carnegie Mellon University (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

© Massachusetts Institute of Technology 2019. All rights reserved.

Author . . . .
Department of Electrical Engineering and Computer Science
August 9, 2019

Certified by . . . .
Alberto Rodriguez
Associate Professor of Mechanical Engineering
Thesis Supervisor

Certified by . . . .
Tomás Lozano-Pérez
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Force-and-Motion Constrained Planning for Tool Use

by

Rachel Mara Holladay

Submitted to the Department of Electrical Engineering and Computer Science
on August 9, 2019, in partial fulfillment of the
requirements for the degree of
Masters of Science

Abstract

The use of hand tools presents a challenge for robot manipulation in part because it calls for motions requiring continuous force application over a whole trajectory, usually involving large joint-angle excursions. The feasible application of a tool, such as pulling a nail with a hammer claw, requires careful coordination of the choice of grasp and joint trajectories to ensure kinematic and force limits are not exceeded - in the grasp as well as the robot mechanism. In this thesis, we formulate this type of problem as choosing the values of decision variables in the presence of various constraints. We evaluate the impact of the various constraints in some representative instances of tool use. To aid others in further investigating this class of problems, we have released materials such as printable tool models and experimental data. We hope that these can serve as the basis of a benchmark problem for investigating tasks that involve many kinematic, actuation, friction, and environment constraints.

Thesis Supervisor: Alberto Rodriguez
Title: Associate Professor of Mechanical Engineering

Thesis Supervisor: Tomás Lozano-Pérez
Title: Professor of Computer Science and Engineering

Acknowledgements

I am incredibly lucky to have Alberto Rodriguez and Tomás Lozano-Pérez as my advisors. Our meetings together are often one of my favorite parts of the week. They are filled with exciting technical discussions, invaluable critique, pearls of research (and life!) wisdom and a unique brand of humor. I am so grateful to have learned so much from both of you already and very excited to continue our journey - "full steam ahead!"

It is a true joy to be a member of both the MCube (Manipulation and Mechanisms at MIT) Lab and the LIS (Learning and Intelligent Systems) Group. It is my privilege to be part of, and draw wisdom from, two research families. From the MCube Lab, thank you to Daolin, Elizabeth, Elliott, Francois, Jose, Maria, Melody, Miquel, Neel, Nikhil, Nima, Oleguer, Orion, Peter, Selam and Siyuan. From the LIS Group, thank you to Alex, Anurag, Ariel, Beomjoon, Caelan, Caris, Clement, Ferran, Gustavo, Rohan, Skye, Tom, Zelda and Zi. Special thanks to Elliott, Francois and Nikhil for the fabrication assistance throughout this project. An additional special thanks to Nikhil and Caelan for the many, many discussions on mechanics and planning, respectively. Thank you to Teresa Cataldo, Lisa Mayer and Marcia Munger for everything you do for each lab, to everyone from the graduate office for answering a multitude of questions and to Leslie Pack Kaebling for her inspiring positivity.

A massive thank you is owed to my friends. Thank you to Mike, Marlyse, Liane and Lily for making classes bearable and always being sources of joy and laughter. I am grateful that through GW6, I befriended Vibhaa and Nili, who have both been critical to my sanity. As a roboticist, I am indebted to FIRST Robotics for sparking my love of robotics as a teen and to the Personal Robotics Lab at Carnegie Mellon for developing me as a robotics researcher.

Finally, the largest thank you goes to my family. My older siblings Ben and Annie and my womb buddy Sam have always been there for me with love, family in-jokes and encouragement. I am delighted to join the rest of my siblings in becoming "Master Holladay". Any of my success is fundamentally a product my my parents, Wendy and Ken Holladay. Their unconditional love, endless patience and unwavering confidence are the greatest gifts anyone could recieve. My parents have always both supported each of my choices and done everything in their power to enable my happiness. Any robotic capability enabled by this thesis is dedicated to my parents, in part because no one enjoys my robot demos or videos more than them. I love my family dearly and am eternally grateful to them.

Funding

This work was supported by the National Robotics Initiative IIS-1637753, MIT Merrill Lynch Fellowship and the NSF Graduate Research Fellowship.

Contents

1 *Introduction* 13

1.1 *Summary of Contributions* 14

2 *Background* 15

2.1 *Task-Constrained Grasping* 15

2.2 *Task-Constrained Motion* 16

2.3 *Tool Use* 17

2.4 *Terminology* 18

3 *Problem Definition* 21

3.1 *Example Tasks* 21

3.2 *Decision Variables* 22

3.3 *Constraints* 23

4 *Constraint Evalulation* 25

4.1 *Stable Grasping* 25

4.2 *Kinematically-Feasible Grasping* 26

4.3 *Force-Feasible Grasping* 27

4.4 *Path Following with Force* 27

5 *Experimental Setup* 29

6 Experiments 31

6.1 Grasping 31

6.2 Path Following 32

6.3 Collision-Free Planning 33

6.4 Domain Change 33

6.5 Computation Time 34

7 Conclusion 37

7.1 Future Directions 37

7.2 Concluding Remarks 38

Bibliography 41

Appendix: Collision-Free Planner 49

List of Figures and Tables

1.1 Constraint visualization on Example Task 13

3.1 3D Printed Tools 21
3.2 Task Variables 22
3.3 Grasp Set 22
3.4 `hammer_pulling` task 23
3.5 `screw_driving` task 23
3.6 `wrench_turning` task 23
3.7 `knife_cutting` task 23

4.1 Limit Surface 26
4.2 Torque Limits 27
4.3 Layered Graph 27

5.1 Reference force-torque profile 30

6.1 Grasp Constraint Evaulation 32
6.2 Impact of Grasp Constraints 32
6.3 Torque Constraint Evaluation 33
6.4 Impact of Torque Constraints 33
6.5 Force Torque execution traces 34
6.6 Grasp Constraint Timing 34
6.7 Algorithm Computation Time 35

1

Introduction

Physical interaction that produces physical changes in the world requires work exchange, i.e., the application of purposeful *forces* along intentional *motions*. We can think of a robot as a programmable force/motion generation machine, and that the aim of robotic manipulation is to master that force/motion generation process. For example, how do we get a robot to pick up a screw driver from a table, grasp its handle, latch gently onto the head of a screw, and turn it forcefully while maintaining a normal load?

In this thesis, we are particularly interested in the long-term decision making involved in choosing the grasp, the arm motions, and tool path to satisfy the many kinematic, actuation, friction, and environment constraints. Our overall goal is to enable robots to reason through that long-term combination of force and motion constraints to facilitate *forceful manipulation*.

Researchers in robotic manipulation have studied extensively both the problems of planning motion and of planning forces. The literature that studies their combined planning is, however, much more sparse and limited. To the best of our knowledge, there are no classical benchmark problems to study it. We study this problem in the context of robots using hand tools, as in Fig.1.1. Tool use is an illustrative task: Both the required forces and motions can be large. As humans, we use a tool in a particular grasp to overcome limited reachability, but also–and often specially–to use appropriately-sized muscles and sufficiently firm grasps for the tool to act on the environment.

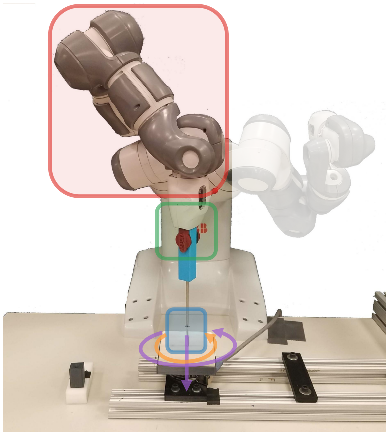

Figure 1.1: We develop a system that enables a robot to reason about force and motion constraints in order to complete complex tasks like wielding a screwdriver. We frame this type of manipulation as a constraint-satisfaction problem where there are constratints on the grasp (green), tool position (blue) and arm configurations (red).

This thesis is primarily an effort to define and investigate tool use problems to find integrated solutions such as that in Fig.1.1. This requires: 1) choosing motion variables: grasps, arm paths, and tool paths; while 2) satisfying force and kinematic constraints: joint travel limits, and joint torque limits, grasp stability, and environment collisions.

Summary of Contributions

This thesis proposes tool use as an illustrative example of forceful manipulation and formally defines the decisions and constraints that govern successful task execution. Specifically, we contribute:

- **Formulation** of a tool-use-problem as a constraint-satisfaction problem over continuous decision variables (Sec. 3).
- **Baseline** based on a rejection-sampling scheme that backtracks within the decision variables to produce a solution (Sec. 4).
- **Benchmark** for tool use. We provide a set of 3D printable tools with a common handle, a specification for their desired use, and experimental data to characterize them (Sec. 5).
- **Study** of the restrictiveness of each of the problem constraints, which gives an idea of the complexity of the planning problem (Sec. 6).

We also include a discussion on related work with respect to task-constrained grasping and motion and manipulation planning for tool use (Sec. 2). Sec. 2.4 brief defines a few relevant terms used throughout the thesis. In addition to describing the benchmark, Sec. 5 discusses the prior knowledge our system assumes. We conclude the thesis with a discussion of future research questions and reflections (Sec. 7).

2

Background

We focus on manipulation tasks that have substantial kinematic and force constraints throughout the task, including choosing grasps and motions that can resist substantial interation forces. We first review work on task-constrained grasping and how previous literature has addressed some level of constraints. We next discuss task-constrained motion planning. Our approach brings together insights and algorithms from both areas in order to build a unified system.

We then briefly overview a subset of the literature on tool use. Tool use is a broad area with perspectives from psychology (Guerin et al., 2013), animal behavior (Shumaker et al., 2011; Beck, 1980), artificial intelligence (Sauer, 2015; Tee et al., 2018; Bicici and St Amant, 2003) and learning (Brown and Sammut, 2007; Wicaksono and Sammut, 2016), to name a few. In this work we focus on those that are relevant to manipulation planning.

We conclude with a brief overview of terminology used throughout the thesis.

Task-Constrained Grasping

The goal of task-constrained grasping is to grasp an object such that it can be used for a particular task. One popular way for encoding task suitability is to plan grasps that withstand the forces required by the task. In particular, there is a significant amount of work in developing analytical grasp quality metrics that score how well a grasp resists external wrenches (Prattichizzo and Trinkle, 2008).

Li and Sastry introduced the task wrench space, which involves modeling the task via task ellipsoids (Li and Sastry, 1988). Pollard proposed the object wrench space, which incorporated the object's geometry into the metric (Pollard, 1994). Borst et al. presented an algorithm that combined evaluation of the task wrench and the object wrench spaces (Borst et al., 2004). Other authors have presented various methods to model task-specific wrenches combined with algorithms to find grasps that best resist these wrenches (Haschke et al., 2005; Lin and Sun, 2015, 2016; Prattichizzo and Trinkle, 2008).

One major limitation of most of these approaches is that they are based on point contact models for the frictional interaction between fingers and objects. In this work we consider planar contacts, which naturally provide more firm grasps, with their associated limit surface friction models (Goyal et al., 1991; Chavan-Dafle et al., 2018). These models better reflect the force and torques that are required to pick up a tool and exert forces with it.

In addition to resisting wrenches, research has also focused on learning task-specific grasps that are kinematically suitable, for example: category-based generalization from training examples (Nikandrova and Kyrki, 2015), a probabilistic inference framework using human and robot examples (Song et al., 2015) and deep learning approaches either for detecting object affordances and orientations, which is formulated into grasp constraints (Kokic et al., 2017), or for joint grasping and manipulation policies trained via simulated self-supervision (Fang et al., 2018). Each of these works apply their approaches to various hand-held tools.

An alternative approach to task-constrained grasping is mechanically designing suitable hands. Specifically for the application of tool use, various works have presented or suggested building robot's with human-like dexterity (Bridgwater et al., 2012; Kemp et al., 2007), developing specialized grippers (Knepper et al., 2013; Stückler and Behnke, 2014) or using tool adaptors (Stückler et al., 2016). Aside from minor mechanical adjustments, we do not explore mechanical design in this thesis.

Task-Constrained Motion

Many of the constraints central to tool use relate to kinematic and force constraints over manipulator paths. For a comprehensive summary of sampling-based constrained motion planning, we refer the reader Kingston et al. (Kingston et al., 2018). They highlight that addressing forces while planning with constraints is an exciting and important area of future work that has seen little development.

There are, however, notable exceptions. Berenson incorporated torque constraints into a constrained sample-based motion planner by rejecting samples that fell outside of the robot's joint torque limits (Berenson et al., 2009). In the context of finding grasps that are stable under external forces, Chen et al. transform the applied forces into torques experienced by the robot and search for configurations that respect's the robot's torque limits (Chen et al., 2018). We draw inspiration from both of these approaches.

Alternatively, Jaquier et al. used manipulability ellipsoids as a metric for what robot configurations, from a force perspective, are best suited to a particular task (Jaquier et al., 2018). Chiu introduced the velocity and force manipulability ellipsoids as geometric representations the directions and magntitudes that a manipulator, in a given configuration, can exert velocity or force, respectively (Chiu, 1988). We further discuss manipulability ellipsoids in Sec. 7.1.

There are control strategies for exerting forces along a path of a tool that rely on classical notions of impedance control, force control or hybrid position/force control (Matsuzaki et al., 2013; Asada and Asari, 1988; Mu et al., 2019). However, these methods do not factor in the grasp on the object.

Tool Use

Much of the existing work on robotic manipulation for tool use focuses on learning from demonstrations. For example, previous work has demonstrated how robots can extend their kinematic reach by grasping various shaped sticks (or other objects) and using them as tools to pull or push objects in the environment (Sinapov and Stoytchev, 2008; Tikhanoff et al., 2013; Elliott et al., 2016; Jain and Inamura, 2014; Xie et al., 2019). Similarly, other have used the framework of affordances to describe the effect of simple tools in the world (Antunes et al., 2015; Stoytchev, 2005; Mar et al., 2015). Elliott et al. extracted contraints from human demonstrations to learn how to use surface cleaning tools (Elliott et al., 2017). Li et al. combined kinesthetic learning from demonstration with dynamic motor primitives to use power tools (Li and Fritz, 2015). Rajeswaran et al. used deep reinforcement learning to grasp and wield a hammer from simulated demonstrations (Rajeswaran et al., 2018).

Toussaint et al. proposed logic-geometric programming to embed dynamic physical manipulations into the task and motion planning process (Toussaint et al., 2018). They demonstrated this approach, which allowed them to describe sequences of modes relating to the kinematics and dynamics, on physical puzzles, including tool use.

Wilson reasoned over geometric constraints, i.e. the swept volume of using a wrench, in the context of tools for assembly applications (Wilson, 1996). Gajewski et al. presented a parameterized method for generating motion trajectories for tool use (Gajewski et al., 2018). However, neither of these methods considered the force or grasp constraints of using a tool.

While we focus on using tools for their conventional purpose, Levihn et al. and Xie et al. focused on using objects in the environment as tools, by performing constraint propagation with respect to the task(Levihn and Stilman, 2014; Levihn and Christensen, 2015) and combining imitation learning and visual MPC (Xie et al., 2019), respectively. Brown and Sammut presented an architecture for learning how to use objects as tools within a problem-solving context(Brown and Sammut, 2007) and Wicaksono and Sammut later extended this onto a real robot platform (Wicaksono and Sammut, 2016). Nair et al. demonstrated a system that constructed conventional tools from objects available in environment by reasoning over part shapes and potential ways to combine those shapes (Nair et al., 2019). Each of the example tool-use tasks in this thesis are quasistatic. While there has been work on dynamic tool-use tasks, such as swinging a hammer or flipping pancakes with a spatula, each focused on the trajectory generation of them arm and did not reason over the grasp of the object(Kunz and Stilman, 2014; Tsuji et al., 2015; Beetz et al., 2011).

Finally, some tool-use texts have focused on determining the transform between the robot's hand and the tip of the tool through visual or other sensory feedback (Kemp and Edsinger, 2006, 2005; Hoffmann et al., 2014). Within this work we assume this transformation is accessible via the grasp definition.

Terminology

Before defining our problem in Sec. 3, we briefly define a few relevant terms.

A configuration, q, of our robot completely describes the location of the robot. The configuration space, $\mathbb{C}$, is the set of all configurations (Lozano-Perez, 1990). In the case of a manipulator, the configuration space is equal to the joint space since we can completely describe the location of the arm by the joint angles. A path in configuration space is detonated by $\xi : [0, 1] \rightarrow \mathbb{C}$.

This section draws largely from (Holladay and Srinivasa, 2017).

Task space is the space defined by the pose of the robot's end effector, $SE(3)$. A path in task space is denoted as $\bar{\xi} : [0, 1] \rightarrow SE(3)$.

We operate with paths, which do not specify velocities or timing. This is in contrast to trajectories, which are parameterized by time. For a trajectory of length t we define trajectories in configuration space and task space, $\xi : [0, t] \rightarrow \mathbb{C}$ and $\xi : [0, t] \rightarrow SE(3)$ respectively. Before execution, we therefore need to time our generated paths into trajectories. In this work, we use a standard parabolic smoother and retimer (Hauser and Ng-Thow-Hing, 2010).

The robot induces a forward kinematics and an inverse kinematics mapping. Forward kinematics maps configurations to task space poses, $x = FK(q)$. Inverse kinematics maps a task space pose to a set of robot configurations, $Q = IK(x)$ such that $Q = \{q^1, q^2, ..q^k\}$.

We group together the forces and torques that act on a rigid body into a single term: wrench. In the context of this thesis, the wrenches are six-dimensional, owing to a three-dimensional force and three-dimensional torque, i.e:

$$w = \left[f_x \; f_y \; f_z \; t_x \; t_y \; t_z\right]^T.$$

3

Problem Definition

Our goal is to enable forceful manipulation by reasoning over force and motion constraints. While many manipulation tasks fit within this problem setup, we explore problems involving the use of hand tools. Specifically, we develop a robot system that picks up a tool and uses it to exert force on the environment. This requires planning a force-motion constrained path.

Tool use can be decomposed into multiple stages such as grasping, making and breaking contact, path following, etc. Each stage has a set of "decision" variables that need to be solved for, such as grasps or kinematic paths (Sec. 3.2). The dependencies between these variables, as well as task-specific constraints, limit the feasible choices. Tool use can therefore be viewed as an instance of a constraint satisfaction problem, but with variables whose domains are high-dimensional continuous values, in general, robot paths. Furthermore, the constraints, either motion or force related, must be maintained throughout the paths.

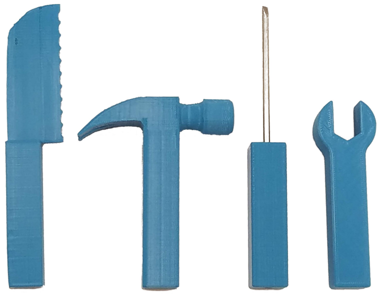

Figure 3.1: Our 3D printed tools

Example Tasks

We demonstrate our approach on four tasks that use four different tools, illustrated in Fig.3.1:

1. `hammer_pulling`: Remove a nail by pulling it out with the claw end of a hammer (Fig.3.4).
2. `screw_driving`: Drive a screw by turning a flathead screwdriver (Fig.3.5).
3. `wrench_turning`: Tighten a nut onto a bolt using an open-end wrench (Fig.3.6).
4. `knife_cutting`: Cut or score a sheet of material with a knife (Fig.3.7).

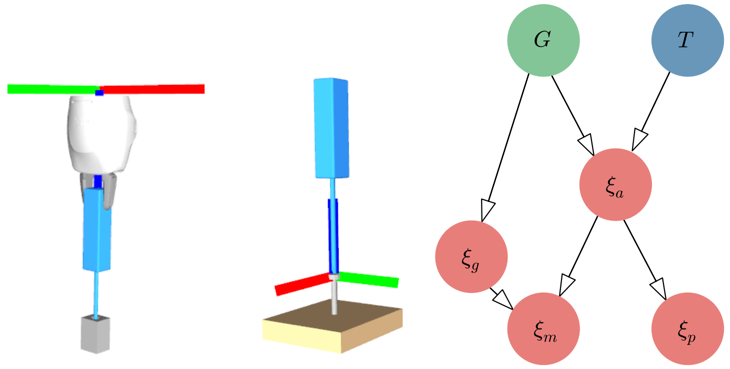

Figure 3.2: A particular choice of grasp G and tool starting pose T are shown in the left and center respectively. On the right, we define our task in terms of variables that are grasp-related (green), tool-positioning related (blue) or kinematic (red). The arrows encode the dependencies between variables.

Walking through `hammer_pulling`, shown in Fig.3.4: the robot grasps the tool, brings the tool in contact with the part it is acting on (i.e. the nail), forcefully acts on the part and places the tool back.

Each of four tasks follows a similar stage structure, differing in what part they act on, what motion is traced out and what forces are exerted through that motion. We next consider the choices that the robot needs to make at each of these stages.

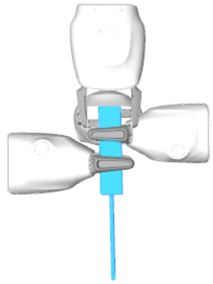

Figure 3.3: A subset of some of our possible grasps. Each tool has the same set of available grasps.

Decision Variables

We have broken each task into four stages, where each stage encompasses a series of variables. We sort these variables into three categories: grasp-related, tool-positioning related and kinematic. In Fig.3.4 these variables are highlighted in green, blue and red, respectively, and the arrows encode their dependencies.

In the first stage (far left), the robot must select a grasp G and plan a collision-free path, ξ_g from its starting position to the grasp. Fig.3.2(left) shows a choice of G for `screw_driving`.

In the second stage (middle left), the robot plans a collision-free path ξ_m to bring the tool in contact with the part it is operating on. We assume in this thesis that we know how to use the tool. We only need to determine the starting location of the tool, T. Therefore, the path ξ_m moves the tool from its starting location to some acceptable T. Fig.3.2(middle) shows a choice of T.

In the third stage (middle right), the robot executes a motion ξ_a that corresponds to using the tool. In the last stage (far right), the robot places the tool back in its initial location via collision-free path, ξ_p.

This tool use problem can therefore be framed as searching for several variables that include paths, configurations and poses. These variable are interdependent, as illustrated in Fig.3.2(right). Essentially our choice of grasp G and tool-path starting position T are two independent choices that define the desired manipulator path

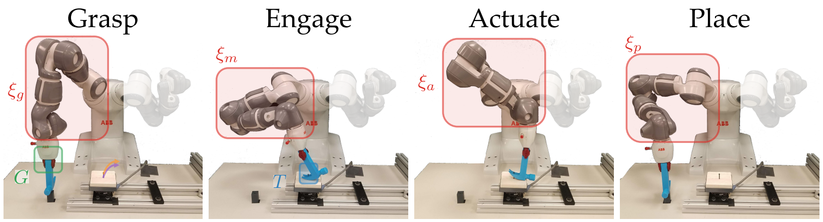

Figure 3.4: `hammer_pulling`. As shown in the far left, the task motion (purple) and force (orange) are to pull the nail upward. With each stage, we annotate the relevant variables, categorized as: grasp-related (green), task-positioning related (blue) and kinematic (red).

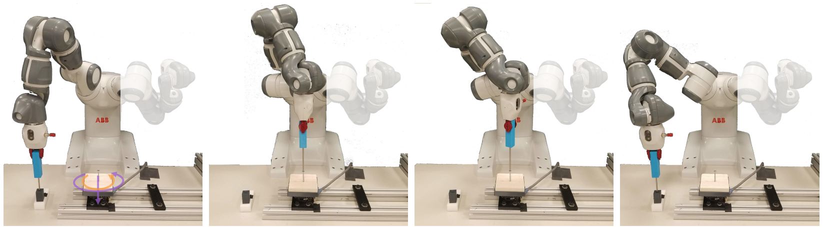

Figure 3.5: `screw_driving`. The task motion (purple) is to turn the screw and the task forces (orange) are a downward twist (far left).

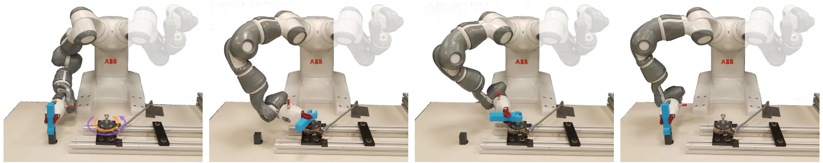

Figure 3.6: `wrench_turning`. The task motion (purple) and force (orange) are both a twist about the nut (far left).

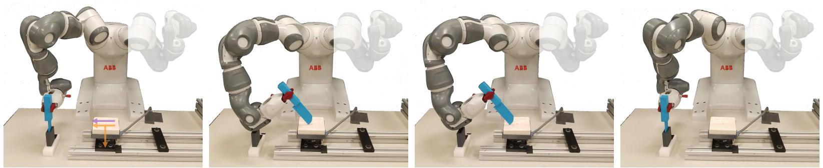

Figure 3.7: `knife_cutting`. The task motion (purple) is to cut across the block and the task force (orange) is to exert down and across (far left).

that wields the tool, ξ_a. These choices affect the remaining paths, ξ_g, ξ_m, ξ_p, which connect the other choices.

Constraints

Having outlined the variables and how they interact with each other, we focus on the constraints on each variable:

- G: From some set of possible grasps (i.e. Fig.3.3 shows three examples), the robot needs to pick one grasp that is forcefully and kinematically suited to the task (quantified in Sec. 4). The grasp must also be collision-free and reachable.
- T: This pose defines the start of the tool path and therefore should be collision-free and reachable.
- ξ_a: For each task, we define a reference path of the tool $\bar{\xi}_T$, that describes its expected operation relative to a starting pose T, through a series of waypoints, $\{t_0, t_1 ... t_m\}$ for $t_i \in SE(3)$. We also define a vector of expected task-specific wrenches. Our goal is to

generate a joint-space path ξ_a for the arm that enables the tool to follow $\bar{\xi}_T$ and sustain the required wrenches. The algorithm for this is detailed in Sec. 4.4.

- ξ_g, ξ_m, ξ_p: Each path needs to be collision-free.

We now have a set of variables, each with own constraints and dependencies. Our goal is to find an assignment of the variables that enables a solution.

4
Constraint Evalulation

Here we describe the key constraints in our tool use problem, namely: finding a task-suitable grasp G and planning ξ_a. We subject our grasp G to four constraints: G (1) is a stable grasp, (2) enables a kinematically-feasible path, (3) enables a force-feasible path and (4) is reachable and collision-free. The first three constraints are detailed in Sec. 4.1, Sec. 4.2, Sec. 4.3 respectively. In Sec. 4.4 we detail the planner used to generate a joint-space manipulator path ξ_a that follows the task-space tool path $\bar{\xi}_T$ while resisting external wrenches.

For all other collision-free planning, we use a bidirectional rapidly-exploring random tree (BiRRT) (LaValle and Kuffner Jr, 2000; Berenson, 2011). This algorithm is detailed in Sec. 7.2.

Stable Grasping

We define a grasp to be forcefully suitable if the grasp is stable under the force of gravity and under the required task-specific wrenches. This ensures that the tool does not slip from the hand while in use.

Given a parallel jaw gripper and the prismatic tool handles (seen in Fig.3.1), finger contacts occur on parallel surfaces. Therefore, we model each finger as a frictional hard patch contact, which means that the finger can exert normal and tangential forces, including torques in the contact plane (Chavan-Dafle and Rodriguez, 2015).

The boundary of the set of frictional wrenches that a patch contact can provide is known as the limit surface (Goyal et al., 1991). These are in charge of resisting external wrenches such as gravity and task-specific wrenches. The limit surface can be approximated as an ellipsoid in the contact frame, where the contact frame, $w = [f_x, f_z, m_y]$, is defined to be centered between where the two parallel fingers make contact with the object (Xydas and Kao, 1999). The ellipsoidal limit surface is then defined by $w^T A w = 1$ where, for isotropic friction:

$$A = \begin{bmatrix} \frac{1}{(N\mu)^2} & & 0 \\ & \frac{1}{(N\mu)^2} & \\ 0 & & \frac{1}{(Nk\mu)^2} \end{bmatrix}$$

such that μ is the coefficient of friction between the tool and the fingers and N is the normal force. With a parallel jaw gripper, where each finger exerts an equal amount of force, we directly control N by the commanded gripping force. For a circular patch contact with a uniform pressure distribution at the contact, $k \approx 0.6r$ where r is the radius of the contact (Xydas and Kao, 1999; Shi et al., 2017).

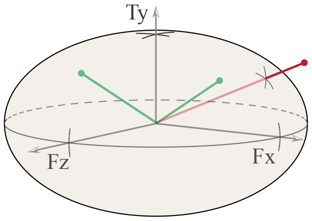

Figure 4.1: We visualize the ellipsoidal approximation of the limit surface. The green wrenches lie within the boundary of the limit surface, representing stable grasps. The red wrench lies outside the boundary, representing an unstable grasp.

To verify if the grasp is stable under external wrenches, we first map the external wrenches to the contact frame. We then check if this wrench falls within the ellipsoid, which from the definitions above becomes:

$$\frac{f_x^2}{(N\mu)^2} + \frac{f_z^2}{(N\mu)^2} + \frac{m_y^2}{(Nk\mu)^2} < 1 \tag{4.1}$$

Therefore a grasp is stable if (4.1) is true, since the contact can support the external wrenches. This check is illustrated in Fig.4.1, where the two green wrenches lie inside the ellipsoid (corresponding to stable grasps) and the red wrench breaches the boundary of the ellipsoid (corresponding to an unstable grasp).

Kinematically-Feasible Grasping

We want to select a grasp that makes it kinematically feasible to plan ξ_a, the manipulator motion that executes the tool action. One way to think about it is to see a stable grasp as augmenting the kinematic chain of the manipulator. The grasp relation serves as an additional joint, bounded by friction, between the hand and the tool and we want to select a value (i.e. a grasp G) such that the path ξ_a is in the reachable workspace of our augmented chain.

The tool path $\bar{\xi}_T$ is defined as a series of waypoints relative to the tool starting at pose T. A grasp G relates $\bar{\xi}_T$ to $\bar{\xi}_a$, i.e. we relate the path of the tool to the path of the end effector given the transformation between the tool and the end effector. Therefore $\bar{\xi}_a$ is also defined as a series of waypoints. A necessary condition to the existance of $\bar{\xi}_a$ is that there is an inverse kinematic (IK) solution, q, at each waypoint. If no IK solutions exist for some waypoint, we cannot generate a full path and can reject the grasp. For our redundant manipulator, this condition is not sufficient because it does not guarantee a smooth, continuous path. Still, it provides a useful heuristic. We address kinematic feasibility over the whole path in Sec. 4.4.

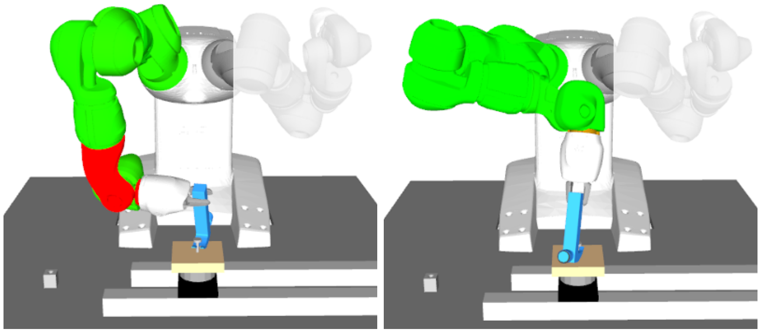

Figure 4.2: The color of each joint indicates the proximity of the expected torque load to the maximum torque limit. Therefore, green and even orange joints are within their torque limits while bright red joints are experiencing a torque fault. Disregarding the torque constraint leads to task failure, as shown in the left, since joints five and seven both experience a torque fault. By contrast, enforcing the constraint leads to the successful grasp and path shown on the right.

Force-Feasible Grasping

While in Sec. 4.2 we verify that a motion path could exist, we also need to ensure that a force-motion path exists. In our context, this means that each configuration in the path is stable under the external wrenches, namely gravity and the task-specific wrenches in the end effector frame. We relate the external wrenches at the end effector to robot joint torques through the manipulator Jacobian, J. Specifically, given a joint configuration q and external wrenches w_{ext}, the torque τ experienced at the joints is modeled by $\tau = J^T(q)w_{ext}$. Our goal is ensure that the expected vector of torques τ does not exceed the robot's torque limits τ_{lim}. Therefore, we augment our heuristic from Sec. 4.2 to verify that, for each waypoint, there is an IK solution q such that:

$$J^T(q)w_{ext} < \tau_{lim}. \tag{4.2}$$

Without this constraint, the robot can choose a grasp and then plan paths that are not strong enough to resist external wrenches, as illustrated in Fig.4.2(left). By enforcing (4.2), the robot selects a different grasp, shown in Fig.4.2(right), that leads to successful execution.

Path Following with Force

To completely generate ξ_a we need to plan a continuous collision-free manipulator path that wields the grasped tool, while sustaining external wrenches. We are given as input $\bar{\xi_T}$, the path of the tool in task space, as a function of the tool starting position T. As described in Sec. 4.2, using grasp G we transform $\bar{\xi_T}$ to $\bar{\xi_a}$, the path of the end effector in task space. We then want to plan a joint-space path ξ_a that closely follows the task-space path $\bar{\xi_a}$. We leverage Holladay et al.'s path following algorithm (Holladay et al., 2019), which measures closeness via the discrete Fréchet distance (Holladay and Srinivasa, 2016), a computational geometry measure of similarity between

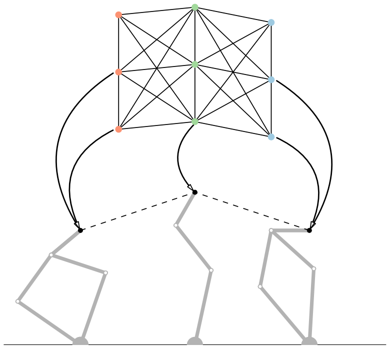

Figure 4.3: Our path-following algorithm constructs a layered graph (top) that maps to task-space poses of the arm (bottom) along our reference path, shown as the dotted line. For each pose, there are multiple IK solutions, which make up the elements of each layer. Reproduced from (Holladay et al., 2019).

curves that respects monotonicity (Fréchet, 1906).

Briefly, the algorithm begins by creating a layered graph that samples and organizes IK solutions by their task-space pose along the reference path (illustrated in Fig.4.3). In order to efficiently find the path within the layered graph closest to the reference path, we create a cross product graph of the layered graph and the reference path and search with a Bottleneck Shortest Path algorithm[1]. This algorithm outputs a joint-space path ξ_a whose forward kinematics maps to a task-space path that closely follows $\bar{\xi}_a$.

However, this formulation does not account our force constraints. Therefore, we only accept a configuration q as a waypoint in our layered graph if it satifies (4.2).

---

[1] We do not use the densification procedure described in the original algorithm. Additionally, instead of randomly sampling each IK solution, for half of the solutions we seed the IK solver with a solution from the previous layer.

5

Experimental Setup

Figs. 3-6 show our physical task setup. We used an ABB YuMi (Robotics, 2015) with custom printed fingers that allowed us to control the radius and coefficient of friction, two critical terms in the grasp stability evaluation. Each tool sits on a holder that maximizes the set of reachable grasps. We built a part-rig for each task that is mounted on top of an ATI Gamma Force/Torque sensor. The readings from the force-torque sensor are not used in-the-loop of the task and are instead only used for analysis.

Our system assumes, as input, several specifications and parameters:

- Plan Skeleton: the sequencing of the stages and choices described in Sec. 3.2.
- Force-Motion Tool Path: the use of each tool is specified as a series of waypoints, $\bar{\xi}_T$, that is a function of the tool starting position. The set of possible starting positions is defined via a chain of Task Space Regions (TSR) (Berenson et al., 2009), which allows for random sampling. For each task we experimentally approximate the upper bounds on the expected task-specific wrenches by having a human perform the task with the robot's tools (Fig.5.1). Once the robot has performed the tasks, we could use the force-torque readings to update the approximation of the expected wrenches.
- Grasp Set: the set of possible grasps (some of which are shown in Fig.3.3) is also defined via a TSR. Each tool is designed with the same size handle such that all tools share a common grasp set. Given a parallel-jaw gripper and rectangular prism handle, we define grasps along each of the faces.
- Tool Parameters: a geometric model of the tool, its mass, its center of mass and the coefficient of friction μ between the tool and the robot's finger.

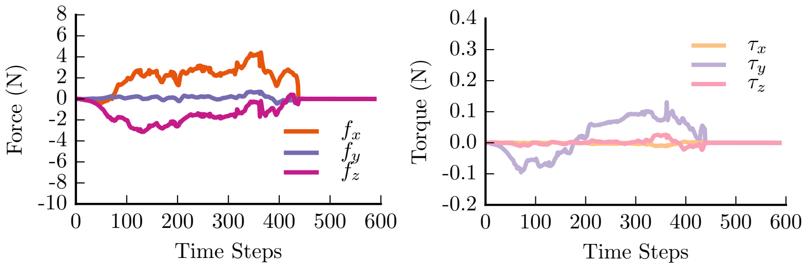

Figure 5.1: We track the force and torque output while a human completes the `knife_cutting` task. This as used as a reference for the force and torque needed to complete the task.

We make 3D printable models of our tools[1], tool holders and environment pieces, along with several other task descriptors available online [2]. We hope this enables and encourages others to explore this type of problem.

---

[1] For the screwdriver we 3D printed only the handle and inserted, as the shaft, a steel rod, which we filed into a flathead tip. We found 3D printing the shaft to be insufficient for practical application.

[2] MCube Lab. Tool use dataset. *URL: https://mcube.mit.edu/tool-use/*, 2019

6

Experiments

Using the physical setup described in Sec. 5, we explore how the task constraints impact the overall solution. The presentation in this section aligns with the variable dependency tree (Fig.3.2(right)), exploring each variable and their corresponding constraints. We begin with how task-specific grasping constraints impact the available grasps (Sec. 6.1) and how torque constraints impact the path-following algorithm (Sec. 6.2). We also briefly discuss feasibility of finding collision-free paths (Sec. 6.3). Next, we focus on real robot experiments that both demonstrate our approach and explore the impact of domain changes (Sec. 6.4). We conclude with a discussion about the computation time of our grasp constraints and overall algorith. (Sec. 6.5).

Grasping

We first explore how the constraints of the task impact the choice of grasp G and how this interacts with the choice of tool starting position, T. As stated in Sec. 3.3, we constrain the grasp to be stable, kinematically-feasible, force-feasible and reachable. For that, we sample 500 grasps and evaluate the four constraints. For each tool we sample up to 10 tool starting postions T and state that the kinematic and force constraints from Sec. 4.2 and Sec. 4.3 are satifisied if the heuristic succeeds for any T.

Table 6.1 shows the results. In addition to showing how many grasps satisfy each constraint, we show how many satisfy the two kinematic constraints (kinematically-feasible and reachable), the two force constraints (stable grasp and force-feasible) and how many grasps lie at the intersection of all constraints.

For most tasks we see that the force constraints are the most restrictive. This is possibly due to using a safe robot with a low payload. The number of grasps at the intersection of all of the constraints can be seen as a proxy for task difficulty.

Task	C_0	C_1	C_2	C_3	Kin.	Force	All
`screw_driving`	500	271	271	384	254	271	254
`wrench_turning`	288	397	349	389	332	190	186
`knife_cutting`	299	364	257	388	279	186	186
`hammer_pulling`	87	186	135	360	171	87	87

Table 6.1: For each task we state the number of grasps (out of 500) that satisfy each constraints. C_0 is grasp stablilty. C_1 is kinematic feasibility. C_2 force feasbility. C_3 is reachability. We also group our constraints by type such that Kinematics (Kin.) denotes $C_1 \cap C_3$ and force denotes $C_0 \cap C_2$

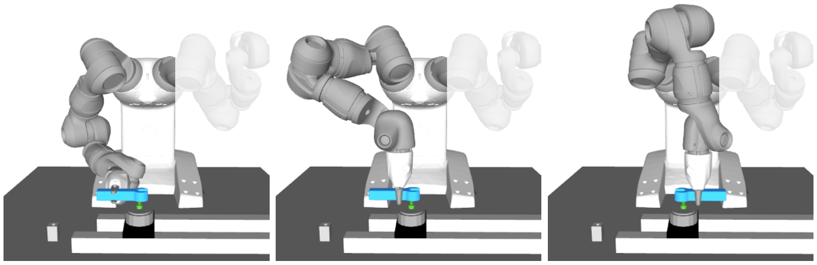

Figure 6.2: Here we show three grasps for `wrench_turning`. The leftmost is kinematically feasible but not stable. The middle is stable but kinematically infeasible. The right is both stable and kinematically feasible.

In the `screw_driving` task, the force requirements are less significant and all grasps are stable. In comparison, the remaining tasks require much more torque, leading to smaller intersection sets. The `hammer_pulling` task is the most difficult, demanding grasps that can resist a significant amount of torque and can create a constrained arc-motion.

Path Following

Given a suitable choice for G and T, we can now plan ξ_a, respecting kinematic and torque constraints. We sample ten suitable combinations of G and T, as determined by the previous section, and generate ten layered graphs as part of our path-following algorithm (Sec. 4.4). We generate ten inverse kinematic (IK) solutions per layer (waypoint), later eliminating those that fail to provide sufficient torque. Removing IK solutions shrinks the search space, decreasing the probability of finding a solution.

Table 6.3 shows the average number of IK solutions in each layer in the `knife_cutting` task. We explored two force requirements, corresponding to cutting through play-doh (easy) and balsa wood (hard). The torque requirements for balsa wood are considerably higher, leading to a smaller set of feasible IK solutions. Cutting play-doh, however, does not suffer from torque constraints. Fig.6.4 shows one particular time slice of the cutting trajectory for both play-doh (left) and basla wood (right), along with the the IK solutions that satisfy the constraints.

The waypoint IK tables for the `wrench_turning` task, `screw_driving` task and `hammer_pulling` task parallel the results from Table 6.1. At one extreme, the screwdriver task is easier and results in a nearly full

Force Requirements	w_0	w_1	w_2	w_3	w_4	w_5
None	8.69	9.06	8.96	8.99	8.85	9.08
Easy	8.69	9.06	8.96	8.99	8.85	9.08
Hard	6.23	6.78	6.45	6.68	6.57	6.76

Table 6.3: For `knife_cutting`, we list the number of IK solutions in each layer, averaged over 100 runs. As we increase the force requirement, some IK solutions drop out due to failing the torque constraint.

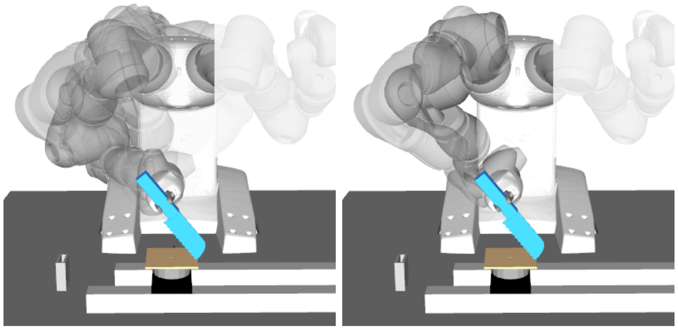

Figure 6.4: The left shows a set of IK solutions for a particular layer. The right shows that only two configurations satisfy the torque constraint.

set of solutions and, at the other extreme, the hammer task is quite difficult and has a very limited set of kinematic solutions.

Collision-Free Planning

Following the chain of dependencies in Fig.3.2(right), there are three collision-free paths, ξ_g, ξ_m, ξ_p that connect each part a task execution. Again, we sample ten feasible variable assignments for G, T, ξ_a, and execute the motion planner ten times for each of our three paths. The planner rarely, if ever, fails to find a solution. This is unsurprising due to the low clutter in the scene.

Domain Change

We next execute each of the tasks on the real robot (videos available online[1]). Using the `knife_cutting` task, we explore the impact of task changes on path execution. We plan a solution expecting the reference force-torque load, shown in Fig.5.1, We successfully cut into the block, experiencing a force profile shown in Fig.6.5(left). We then raise the block slightly and execute the same solution. As a result, the knife tries to cut deeper, incurring a much higher force-torque load, as shown in Fig.6.5(right). Since we did not plan accounting for this increased level of force-torque, the robot experiences a torque fault around time slice 600 and fails to complete the task. This underscores the need to generate plans that respect each of the constraints.

[1] MCube Lab. Tool use dataset. *URL: https://mcube.mit.edu/tool-use/*, 2019

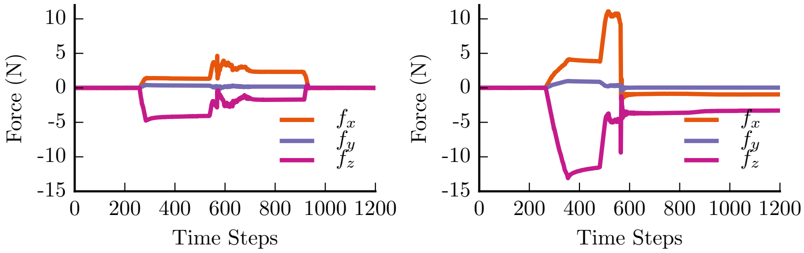

Figure 6.5: Force traces of the robot executing the `knife_cutting` task. The left shows results from the standard version of the experiment. The right shows the results if we raise the block, thus asking the robot to cut deeper. The resultant task forces were higher then what we planned for, leading to the torque-limit fail at the 600 mark.

Task	C_0		C_1 and C_2		C_3	
`screw_driving`	0.0002	-	2.14	2.60	4.30	10.00
`wrench_turning`	0.0003	0.0010	5.28	12.48	3.73	10.00
`knife_cutting`	0.0003	0.0003	4.46	13.94	4.00	10.00
`hammer_pulling`	0.0003	0.0100	3.58	9.57	4.13	10.00

Table 6.6: Time to evaluate each of our grasp constraints. C_0 is grasp stability. C_1 and C_2 is kinematic and force feasibility. C_3 is reachability. The results are split into success (left, green) and failure (right, red).

Computation Time

Returning to the grasp constraint experiments from Sec. 6.1, in Table 6.6 we list the time to evaluate each of the grasping constraints. Here we combine the computation time for heuristic that evaluates kinematic and force feasility. Across the 500 grasps, we split the evaluation time by success (left) versus failure (right). The stability constraint is quite fast to evaluate since it only requires to evaluation of Eqn. (4.1). The other constraints are much more expensive since their evaluation requires path planning or many inverse kinematic calls. For this experiment we set our path planning evaluation to timeout after 10 seconds, which explains the uniformity in the failure timing. Additionally, since all grasps for `screw_driving` satified the stability constraint, there is no failure timing information.

Fig.6.7 gives the time to find a complete solution for each task, averaged over twenty-five trials. Within each task the computation is split into three categories: rejection sampling for a suitable G and T, planning ξ_a and planning for ξ_g, ξ_m, ξ_p. This ordering and split is motivated by our dependency graph shown in Fig.3.2(right). Written on each bar is the average number of times the algorithm executed that category. For example, for the screwdriver task, the algorithm sampled an average of 2.44 G and T combinations, but usually only had to plan each of the paths once. This matches our results from Sec. 6.3, which showed that our collision-free planning rarely fails.

When rejection sampling for G, we only evaluate grasp stability, kinematic feasibility and force feasbility. The reachability check, shown to be expensive in Table 6.6, is delayed until the third stage when we plan all collision-free paths. As mentioned above, planner gives a maximum of ten seconds of planning time with up to a

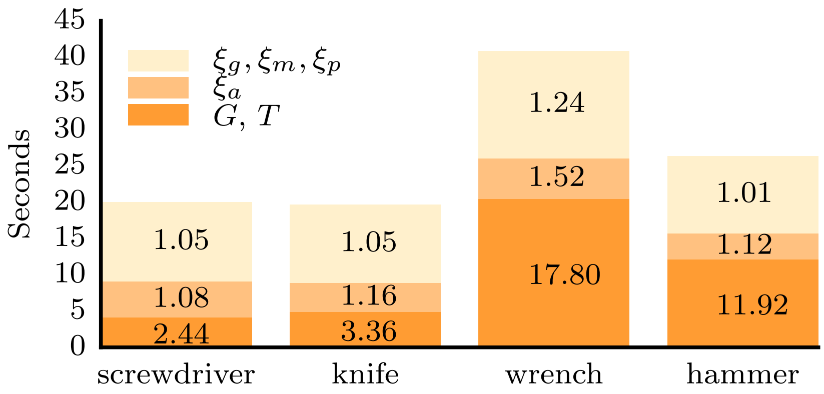

Figure 6.7: Computation time for each task averaged over twenty-five iterations. We note the average number of times the algorithm executed each category.

second of shortcutting[2].

Our timing results show that we may quickly sample through multiple combinations of G and T and therefore minimize the number of times we make expensive planning calls for $\xi_a, \xi_g, \xi_m, \xi_p$.

---

[2] The paths generated in the videos have up to four seconds to shortcutting.

7
Conclusion

The goal of this thesis is to enable a robot to complete tasks that require both planning motions and forces. We present tool use as an illustative example of this type of forceful manipulation and model it as an instance of constraint satification. We first decompose tool use into a series of decision variables that need to be solved for, such as grasp, poses and paths. We then connect the dependencies between these variables and identify the kinematic, actuation, friction, and environment constraints that impact task success. Two key constraints in our tool use problem are finding a task-suitable grasp and planning our action path. In Sec. 4 we detail algorithms to validate a grasp, which can be used in a rejection sampling framework, and a constrained path planning algorithm.

In Sec. 6, we evaluate how our constraints impact the overall solution. We can view the degree to which a task's constaints limit the feasible solution space a measure of the task's difficulty. We also brief discuss the computation time for our constraints and the overall algorithm. Our approach was validated on four tasks: `screw_driving`, `wrench_turning`, `hammer_pulling`, `knife_cutting`. We have publicly released benchmarking materials such as experimental data, 3D printable tool models, etc., as described in Sec. 5. We hope this encourages and enables other researchers to explore similar tool-use problems.

Below we briefly describe avenues of future research as well as a few concluding remarks.

Future Directions

While there are many possible future direction, we highlight two: searching over plan skeletons and incoporating manipulability metrics.

As detailed in Sec. 5, an input to our system is a fixed plan skeleton, i.e. we proscribe the stages and choices within tool use. Allowing flexibility within this sequence would enable a more general-purpose solver. For example, we currently fix a grasp throughout the duration of the task, which can be very kinematically limiting. Regrasping could enable the robot to complete longer-scale tasks and greater flexibility (Tournassoud et al., 1987). Likewise, we currently constrain each task to be completed in one continuous motion. This could be relaxed to allow the robot to exploit stop-readjust-continue strategies, much like humans do. While we consider a fixed robot and workcell position, a mobile robot or movable workcell would introduce an additional set of decision variables. More generally speaking, searching over possible plan skeletons, i.e. recasting tool use within the task and motion planning domain (Garrett et al., 2018, 2017; Kaelbling and Lozano-Pérez, 2013; Dantam et al., 2016), is an exciting future direction.

In Sec. 4 we quantified force and motion constraints via kinematic heuristics and torque-limit aware planning. An alternative way to quantify task suitability is via velocity and force manipulability ellipsoids, which indicate how easily a manipulator can generate velocity or exert force in certain directions (Yoshikawa, 1984, 1985). For a given joint configuration q, where J is the manipulator Jacobian, the principal axes and corresponding lengths of the semi-axes of the velocity manipulability ellipsoid are defined by the eigenvectors of $J(q)J^T(q)$ and the square roots of the eigenvalues, respectively. Likewise the force manipulability ellipsoids are defined by the eigenvectors of $(J(q)J^T(q))^{-1}$ and the square roots of the eigenvalues. Chiu used these manipulability ellipsoids to define the task compatibility index, which scores configuration by how well they can control velocity or exert forces in directions that are useful for task execution (Chiu, 1988). More recently, Jaquier et al. track a desired profile of manipulability ellipsoidsas a primary or secondary task objective (Jaquier et al., 2018).

In Sec. 4.4, we filtered IK solution by whether they satified torque limits while experiencing a task-specific wrench. An alterative and perhaps complementary approach would be to rank configuration according to task compatibility indices, which requires no assumption with respect to the task-specific wrench. Our search would then be a combination of optimizing manipulability while finding the bottleneck shortest path.

Concluding Remarks

While this thesis focuses on tool use as an illustrative example, the principles of forceful manipulation apply across a wide range of applications. For example, opening a child-proof lid requires grasping the bottle and exerting a downward force on the lid while twisting it. This force-motion description, of a downward twist, is similar to how a screwdriver is used. However, the differences in the required force needed and the contact configurations, among other differences, lead us to execute the force-motion via different means. This leads to a critical, larger question: how do we specify forceful manipulation tasks?

In this thesis, we encoded the tool action as a series of waypoints in task space and an upper bound on the task-specific wrench we expected the robot to encounter. How would that specification generalize to the child-proof lid example, where a tool might not be needed to complete the task? More generally speaking, we could imagine the robot reasoning over whether it needs to use a tool or any object in the environment to accomplish the desired task. Moving foward, our goal is to develop this unifying force-and-motion constrained manipulation framework that can enable a wide variety of forceful manipulation tasks.

Bibliography

Alexandre Antunes, Giovanni Saponaro, Atabak Dehban, Lorenzo Jamone, Rodrigo Ventura, Alexandre Bernardino, and José Santos-Victor. Robotic tool use and problem solving based on probabilistic planning and learned affordances. In *IROS Workshop*, 2015.

Haruhiko Asada and Yukio Asari. The direct teaching of tool manipulation skills via the impedance identification of human motions. In *ICRA*, pages 1269–1274. IEEE, 1988.

Benjamin B Beck. *Animal tool behavior*. Garland STPM Pub., 1980.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic roommates making pancakes. In *Humanoids*, pages 529–536. IEEE, 2011.

Dmitry Berenson. *Constrained manipulation planning*. Carnegie Mellon University, 2011.

Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner. Manipulation planning on constraint manifolds. In *ICRA*, pages 625–632. IEEE, 2009.

Ergun Bicici and Robert St Amant. Reasoning about the functionality of tools and physical artifacts. *Department of Computer Science, North Carolina State University, Tech. Rep*, 22:1–34, 2003.

Ch Borst, Max Fischer, and Gerd Hirzinger. Grasp planning: How to choose a suitable task wrench space. In *ICRA*, volume 1, pages 319–325. IEEE, 2004.

Lyndon B Bridgwater, CA Ihrke, Myron A Diftler, Muhammad E Abdallah, Nicolaus A Radford, JM Rogers, S Yayathi, R Scott Askew, and D Marty Linn. The robonaut 2 hand-designed to do work with tools. In *ICRA*, pages 3425–3430. IEEE, 2012.

Solly Brown and Claude Sammut. An architecture for tool use and learning in robots. In *Australian Conference on Robotics and Automation*. Citeseer, 2007.

Nikhil Chavan-Dafle and Alberto Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *IROS*, pages 6215–6222. IEEE, 2015.

Nikhil Chavan-Dafle, Rachel Holladay, and Alberto Rodriguez. In-Hand Manipulation via Motion Cones. In *RSS*, 2018.

Lipeng Chen, Luis FC Figueredo, and MR Dogar. Manipulation Planning under Changing External Forces. In *IROS*, pages 3503–3510. IEEE, 2018.

Stephen L Chiu. Task compatibility of manipulator postures. *IJRR*, 7 (5):13–21, 1988.

Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. Incremental Task and Motion Planning: A Constraint-Based Approach. In *RSS*, volume 12, page 00052. Ann Arbor, MI, USA, 2016.

Sarah Elliott, Michelle Valente, and Maya Cakmak. Making objects graspable in confined environments through push and pull manipulation with a tool. In *ICRA*, pages 4851–4858. IEEE, 2016.

Sarah Elliott, Zhe Xu, and Maya Cakmak. Learning generalizable surface cleaning actions from demonstration. In *RO-MAN*, pages 993–999. IEEE, 2017.

Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision. In *RSS*, 2018.

M Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22(1):1–72, 1906.

Pawel Gajewski, Paulo Ferreira, Georg Bartels, Chaozheng Wang, Frank Guerin, Bipin Indurkhya, Michael Beetz, and Bartlomiej Sniezynski. Adapting everyday manipulation skills to varied scenarios. *arXiv preprint arXiv:1803.02743*, 2018.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Sample-Based Methods for Factored Task and Motion Planning. In *RSS*, 2017.

Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. FFRob: Leveraging symbolic planning for efficient task and motion planning. *IJRR*, 37(1):104–136, 2018.

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.

Frank Guerin, Norbert Kruger, and Dirk Kraft. A survey of the ontogeny of tool use: from sensorimotor experience to planning. *TAMD*, 5(1):18–45, 2013.

Robert Haschke, Jochen J Steil, Ingo Steuwer, and Helge Ritter. Task-oriented quality measures for dextrous grasping. In *CIRA*, pages 689–694. IEEE, 2005.

Kris Hauser and Victor Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *ICRA*, pages 2493–2498. IEEE, 2010.

Heiko Hoffmann, Zhichao Chen, Darren Earl, Derek Mitchell, Behnam Salemi, and Jivko Sinapov. Adaptive robotic tool use under variable grasps. *Robotics and Autonomous Systems*, 62(6):833–846, 2014.

Rachel Holladay and Siddhartha Srinivasa. Distance metrics and algorithms for task space path optimization. In *IROS*, pages 5533–5540. IEEE, 2016.

Rachel Holladay and Siddhartha Srinivasa. Following Paths in Task Space: Distance Metrics and Planning Algorithms. Master's thesis, Carnegie Mellon University Pittsburgh, PA, 2017.

Rachel Holladay, Oren Salzman, and Siddhartha Srinivasa. Minimizing task space Frechet error via efficient incremental graph search. *Robotics and Automation Letters*, 4(2):1999–2006, 2019.

Raghvendra Jain and Tetsunari Inamura. Learning of usage of tools based on interaction between humans and robots. In *CYBER*, pages 597–602. IEEE, 2014.

Noémie Jaquier, Leonel Dario Rozo, Darwin G Caldwell, and Sylvain Calinon. Geometry-aware Tracking of Manipulability Ellipsoids. In *RSS*, 2018.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *IJRR*, 32(9-10):1194–1227, 2013.

Charles C Kemp and Aaron Edsinger. Visual tool tip detection and position estimation for robotic manipulation of unknown human tools. *CSAIL Technical Report*, 83, 2005.

Charles C Kemp and Aaron Edsinger. Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *ICDL*, volume 42, 2006.

Charles C Kemp, Aaron Edsinger, and Eduardo Torres-Jara. Challenges for robot manipulation in human environments [grand challenges of robotics]. *Robotics & Automation Magazine*, 14(1):20–29, 2007.

Zachary Kingston, Mark Moll, and Lydia E Kavraki. Sampling-based methods for motion planning with constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:159–185, 2018.

Ross A Knepper, Todd Layton, John Romanishin, and Daniela Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *ICRA*, pages 855–862. IEEE, 2013.

Mia Kokic, Johannes A Stork, Joshua A Haustein, and Danica Kragic. Affordance detection for task-specific grasping using deep learning. In *Humanoids*, pages 91–98. IEEE, 2017.

Tobias Kunz and Mike Stilman. Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. In *IROS*, pages 3713–3719. IEEE, 2014.

MCube Lab. Tool use dataset. *URL: https://mcube.mit.edu/tool-use/*, 2019.

Steven M LaValle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. *WAFR*, 2000.

Martin Levihn and Henrik Christensen. Using environment objects as tools in unknown environments. In *Humanoids*, pages 160–165. IEEE, 2015.

Martin Levihn and Mike Stilman. Using environment objects as tools: Unconventional door opening. In *IROS*, pages 2502–2508. IEEE, 2014.

Wenbin Li and Mario Fritz. Teaching robots the use of human tools from demonstration with non-dexterous end-effectors. In *Humanoids*, pages 547–553. IEEE, 2015.

Zexiang Li and S Shankar Sastry. Task-oriented optimal grasping by multifingered robot hands. *IEEE Journal on Robotics and Automation*, 4(1):32–44, 1988.

Yun Lin and Yu Sun. Grasp planning to maximize task coverage. *IJRR*, 34(9):1195–1210, 2015.

Yun Lin and Yu Sun. Task-oriented grasp planning based on disturbance distribution. In *Robotics Research*, pages 577–592. Springer, 2016.

Tomas Lozano-Perez. Spatial planning: A configuration space approach. In *Autonomous robot vehicles*, pages 259–271. Springer, 1990.

Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of grasp dependent tool affordances on the iCub Humanoid robot. In *ICRA*, pages 3200–3206. IEEE, 2015.

Ryohei Matsuzaki, Makoto Kamibayashi, Sho Sakaino, and Toshiaki Tsuji. Classification of a hybrid control system for robotic tool use. In *ICM*, pages 712–717. IEEE, 2013.

Xiaoqian Mu, Yuechuan Xue, and Yan-Bin Jia. Robotic Cutting: Mechanics and Control of Knife Motion. In *ICRA*, 2019.

Lakshmi Nair, Jonathan Balloch, and Sonia Chernova. Tool macgyvering: Tool construction using geometric reasoning. In *RSS*, 2019.

Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of dynamic systems, measurement, and control*, 108(3):163–171, 1986.

Ekaterina Nikandrova and Ville Kyrki. Category-based task specific grasping. *RAS*, 70:25–35, 2015.

Nancy S Pollard. Parallel methods for synthesizing whole-hand grasps from generalized prototypes. Technical report, MIT AI Lab, 1994.

Domenico Prattichizzo and Jeffrey C Trinkle. Grasping. In *Springer handbook of robotics*, pages 671–700. Springer, 2008.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.

ABB Robotics. Abb yumi. *URL: http://new. abb. com/products/robotics/yumi/. Last visited*, 9, 2015.

Lukas Sauer. An AI formalization of Betty the Crow's sequential geometric tool use. Master's thesis, University of Stuttgart, 2015.

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

Robert W Shumaker, Kristina R Walkup, and Benjamin B Beck. *Animal tool behavior: the use and manufacture of tools by animals*. JHU Press, 2011.

Jivko Sinapov and Alexadner Stoytchev. Detecting the functional similarities between tools using a hierarchical representation of outcomes. In *ICDL*, pages 91–96. IEEE, 2008.

Dan Song, Carl Henrik Ek, Kai Huebner, and Danica Kragic. Task-based robot grasp planning using probabilistic inference. *IEEE Transactions on Robotics*, 31(3):546–561, 2015.

Alexander Stoytchev. Behavior-grounded representation of tool affordances. In *ICRA*, pages 3060–3065. IEEE, 2005.

Jörg Stückler and Sven Behnke. Adaptive tool-use strategies for anthropomorphic service robots. In *Humanoids*, pages 755–760. IEEE, 2014.

Jörg Stückler, Max Schwarz, and Sven Behnke. Mobile manipulation, tool use, and intuitive interaction for cognitive service robot cosero. *Frontiers in Robotics and AI*, 3:58, 2016.

Keng Peng Tee, Jun Li, Lawrence Tai Pang Chen, Kong Wah Wan, and Gowrishankar Ganesh. Towards Emergence of Tool Use in Robots: Automatic Tool Recognition and Use without Prior Tool Learning. In *ICRA*. IEEE, 2018.

V Tikhanoff, U Pattacini, L Natale, and G Metta. Exploring affordances and tool use on the iCub. In *Humanoids*, pages 130–137. IEEE, 2013.

Pierre Tournassoud, Tomás Lozano-Pérez, and Emmanuel Mazer. Regrasping. In *ICRA*, volume 4, pages 1924–1928. IEEE, 1987.

Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *RSS*, 2018.

Toshiaki Tsuji, Jun Ohkuma, and Sho Sakaino. Dynamic object manipulation considering contact condition of robot with tool. *Transactions on Industrial Electronics*, 63(3):1972–1980, 2015.

Handy Wicaksono and Claude Sammut. Relational Tool Use Learning by a Robot in a Real and Simulated World. In *ACRA*, 2016.

Randall H Wilson. A framework for geometric reasoning about tools in assembly. In *ICRA*, volume 2, pages 1837–1844. IEEE, 1996.

Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through Physical Understanding: Using Novel Objects as Tools with Visual Foresight. *arXiv preprint arXiv:1904.05538*, 2019.

Nicholas Xydas and Imin Kao. Modeling of contact mechanics and friction limit surfaces for soft fingers in robotics, with experimental results. *IJRR*, 18(9):941–950, 1999.

Tsuneo Yoshikawa. Analysis and control of robot manipulators with redundancy. In *Robotics research: the first international symposium*, pages 735–747. MIT press Cambridge, MA, USA, 1984.

Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *IJRR*, 4(2): 3–9, 1985.

Appendix: Collision-Free Planner

For collision-free planning, we utilize a bidirectional rapidly-exploring random tree (BiRRT) motion planner, detailed below (LaValle and Kuffner Jr, 2000). The details of the implementation were inspired by CBiRRT, a constrained BiRRT, and we borrow their algorithmic description (Berenson, 2011).

We plan from a starting joint configuration $q_s \in \mathcal{C}$ to either a goal configuration $q_g \in \mathcal{C}$ or goal pose $x_g \in SE(3)$. The BiRRT algorithm, given in Algorithm 1, at each step, can either sample new goal configurations or explore configuration space. This decision is, in part, controlled by a variable P_{sample}, which we set to 0.1 in order to bias computation toward building the path (Berenson, 2011).

If the planner samples a new goal, it either adds the goal configuration to the tree or samples for collision-free inverse kinematic solutions of the goal pose. If the planner explores configuration space, it samples a configuration q_{rand} and tries to grow one tree as far as possible towards q_{rand} using the Extend function (Algorithm 2). The Extend function will grows as far as it can, (respecting joint limits and collisions[1]) returning where it stops at some q^a_{reach}. It then tries to grow the other tree towards q^a_{reach} using same Extend operation. If the two trees meet, there is a path from start to goal.

Before returning the path, we use a shortcutting operation (Algorithm 3). While we have time left in our budget, we randomly sample sections of our path and attempt to create shorter paths to replace it with. RRTs are known to provide very erratic paths, due to their randomness, so this shortcutting serves as a smoothing function.

[1] Within the original constrained version of this planner (Berenson, 2011), the sample q_s is projected to the constraint manifold. Our planner, however, takes a simpler approach of clipping to respect joint limits

Algorithm 1 BiRRT(q_s, Q_g

```
$T_a$.Init($q_s$); $T_b$.Init($Q_g$)
while TimeRemaining() do
    $T_{goal}$ = GetBackwardTree($T_a, T_b$)
    if size($T_{goal}$) = 0 or rand(0, 1) < $P_{sample}$ then
        AddRoot($T_{goal}$)
    else
        $q_{rand}$ = RandomConfig()
        $q^a_{near}$ = NearestNeighbor($T_a, q_{rand}$)
        $q^a_{reach}$ = Extend($T_a, q^a_{near}, q_{rand}$)
        $q^b_{near}$ = NearestNeighbor($T_b, q^a_{reach}$)
        $q^b_{reach}$ = Extend($T_b, q^b_{near}, q^a_{reach}$)
        if $q^a_{reach} = q^b_{reach}$ then
            $P$ = ExtractPath($T_a, T_b, q^a_{reach}, q^b_{reach}$)
            return ShortenPath($P$)
        else
            Swap($T_a, T_b$)
```

Algorithm 2 Extend(T, q_{near}, q_{target})

```
$q_s = q_{near}; q_s^{old} = q_{near}$
while True do
    if $q_{target} = q_s$ then
        return $q_s$
    else if $\| q_{target} - q_s \| > \| q_s^{old} - q_{target} \|$ then
        return $q_s^{old}$
    $q^s_{old} = q_s$
    $q^s_p = q_s + \min(\Delta q_{step}, \| q_{target} - q_s \|) \frac{(q_{target} - q_s)}{\|q_{target} - q_s\|}$
    $q_s$ = ClampJointLimits($q^s_p$)
    if CollisionFree($q^s$) then
        $T$.AddVertex($q_s$)
        $T$.AddEdge($q^s_{old}, q_s$)
    else
        return $q_s^{old}$
```

Algorithm 3 ShortenPath(P)

```
while TimeRemaining() do
    $T_{short} = \{\}$
    $i$ = RandomInt(0, size($P$)-1)
    $j$ = RandomInt($i$, size($P$))
    $q_{reach}$ = Extend($T_{short}, P_i, P_j$)
    if $q_{reach} = P_j$ and Len($T_{short}$) < Len($P_i, P_j$) then
        $P = [P_1, \ldots P_i, T_{short}, P_{j+1}, \ldots P.size]$
    return $P$
```