Force-and-Motion Constrained Planning for Tool Use

Rachel Holladay[1], Tomás Lozano-Pérez[1], and Alberto Rodriguez[2]
https://mcube.mit.edu/tool-use/

***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 paper, 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.**

I. 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?

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. 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.

[1]Rachel Holladay and Tomás Lozano-Pérez are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, rhollada,tlp@mit.edu.

[2]Alberto Rodriguez is with the Mechanical Engineering Department, Massachusetts Institute of Technology, albertor@mit.edu.

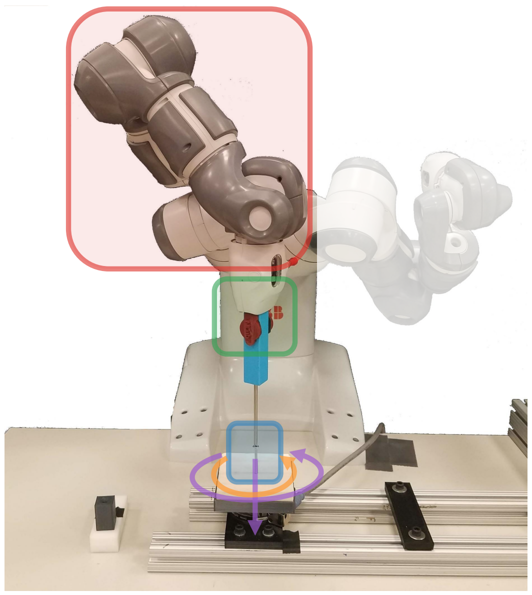


Fig. 1: We develop a system that enables a robot to reason about force and motion constraints in order to complete complex tasks like wielding a screwdriver. We frame this type of manipulation as a constraint-satisfaction problem where there are constraints on the grasp (green), tool position (blue) and arm configurations (red).

This paper is primarily an effort to define and investigate tool use problems to find integrated solutions such as that in Fig.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.

Our work makes four primary contributions:

- **Formulation** of a tool-use-problem as a constraint-satisfaction problem over continuous decision variables.
- **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.
- **Baseline** based on a rejection-sampling scheme that backtracks within the decision variables to produce a solution.
- **Study** of the restrictiveness of each of the problem constraints, which gives an idea of the complexity of the planning problem.

The experiments on this study are open loop, so a limitation of the current formulation is that it cannot adapt to dynamic changes in the environment. Sec. V discusses the prior knowledge our system assumes. While we currently provide these parameters and specifications, there are interesting opportunities for incoporating perception and learning.

II. Related Work

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. Finally, we briefly overview a subset of the literature on tool use. Tool use is a broad area with perspectives from psychology [1], animal behavior [2], [3], artificial intelligence [4], [5] and learning [6], [7], [8], to name a few. In this work we focus on those that are relevant to manipulation planning.

A. *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 [9].

Li and Sastry introduced the task wrench space, which involves modeling the task via task ellipsoids [10]. Pollard proposed the object wrench space, which incorporated the object's geometry into the metric [11]. Borst et al. presented an algorithm that combined evaluation of the task wrench and the object wrench spaces [12]. Other authors have presented various methods to model task-specific wrenches combined with algorithms to find grasps that best resist these wrenches [13], [14], [15].

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 [16], [17]. 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 [18], a probabilistic inference framework using human and robot examples [19] and deep learning approaches either for detecting object affordances and orientations, which is formulated into grasp constraints [20], or for joint grasping and manipulation policies trained via simulated self-supervision [21]. Each of these works apply their approaches to various hand-held tools.

B. *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. [22]. 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 [23]. 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 [24]. We draw inspiration from both of these approaches.

There are control strategies for exerting forces along a path of a tool that rely on classical notions of impedance or force control [25], [26]. However, these methods do not factor in the grasp on the object.

C. *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 [27], [28], [29], [30], [8]. Similarly, other have used the framework of affordances to describe the effect of simple tools in the world [31], [32]. Li et al. combined kinesthetic learning from demonstration with dynamic motor primitives to use power tools [33]. Rajeswaran et al. used deep reinforcement learning to grasp and wield a hammer from simulated demonstrations [34].

Toussaint et al. proposed logic-geometric programming to embed dynamic physical manipulations into the task and motion planning process [35]. They demonstrated this approach, which allowed them to describe sequences of modes relating to the kinematics and dynamics, on physical puzzles, including tool use.

III. 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. III-B). 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.

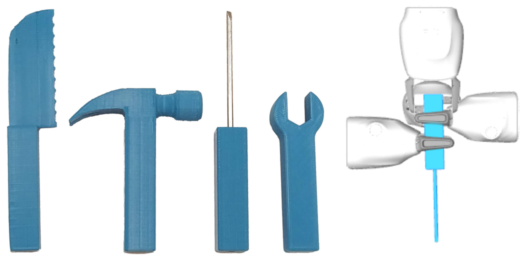
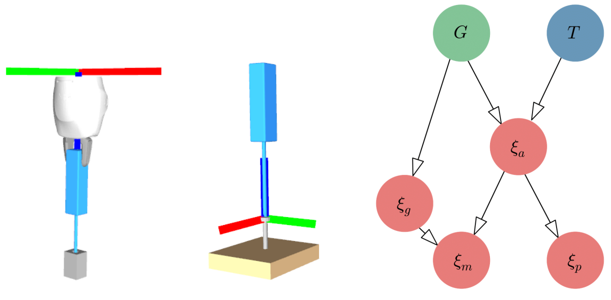

(a) 3D Printed Tools (b) Grasp Set

Fig. 2: a) Our 3D printed tools. b) A subset of some of our possible grasps. Each tool has the same set of available grasps.



(a) Grasp Frame G (b) Tool Frame T (c) Task Variables

Fig. 3: A particular choice of grasp G and tool starting pose T are shown in (a) and (b) respectively. c) 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.

A. Example Tasks

We demonstrate our approach on four tasks that use four different tools, illustrated in Fig.2a:

1) `hammer_pulling`: Remove a nail by pulling it out with the claw end of a hammer (Fig.4).
2) `screw_driving`: Drive a screw by turning a flathead screwdriver (Fig.5).
3) `wrench_turning`: Tighten a nut onto a bolt using an open-end wrench (Fig.6).
4) `knife_cutting`: Cut or score a sheet of material with a knife (Fig.7).

Walking through `hammer_pulling`, shown in Fig.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.

B. 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.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.3a 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 paper 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.3b 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.3c. Essentially our choice of grasp G and tool-path starting position T are two independent choices that define the desired manipulator path that wields the tool, ξ_a. These choices affect the remaining paths, ξ_g, ξ_m, ξ_p, which connect the other choices.

C. 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.2b shows three examples), the robot needs to pick one grasp that is forcefully and kinematically suited to the task (quantified in Sec. IV). 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 (force and torque). 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[1]. The algorithm for this is detailed in Sec. IV-D.
- 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.

IV. Constraint Evaluation

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. IV-A, Sec. IV-B, Sec. IV-C respectively. In Sec. IV-D we detail

[1]We follow the convention that ξ_x refers to a joint-space path and $\bar{\xi_x}$ refers to a task-space path in SE(3).

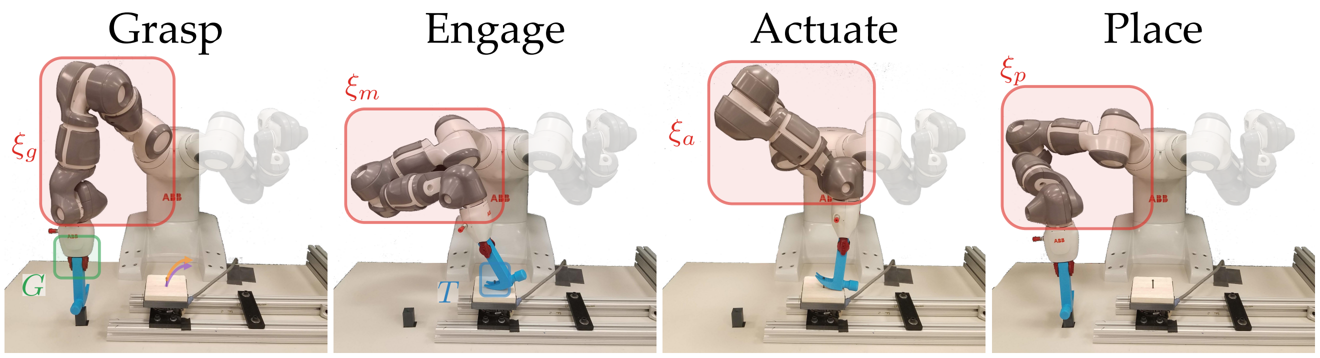


Fig. 4: `hammer_pulling`. As shown in the far left, the task motion (purple) and force (orange) are to pull the nail upward. With each stage, we annotate the relevant variables, categorized as: grasp-related (green), task-positioning related (blue) and kinematic (red).

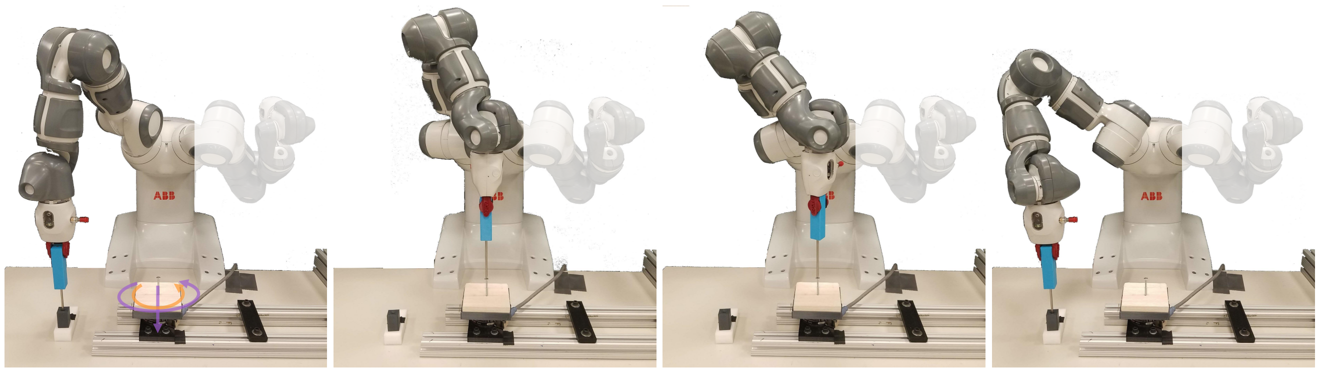


Fig. 5: `screw_driving`. The task motion (purple) is to turn the screw and the task forces (orange) are a downward twist (far left).

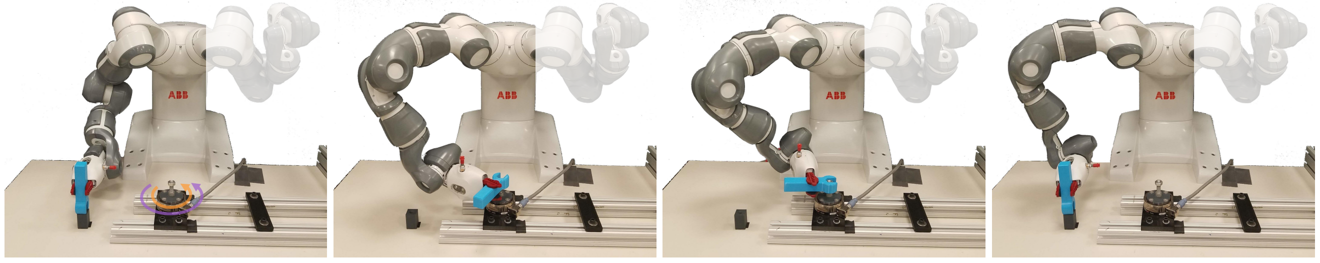


Fig. 6: `wrench_turning`. The task motion (purple) and force (orange) are both a twist about the nut (far left).

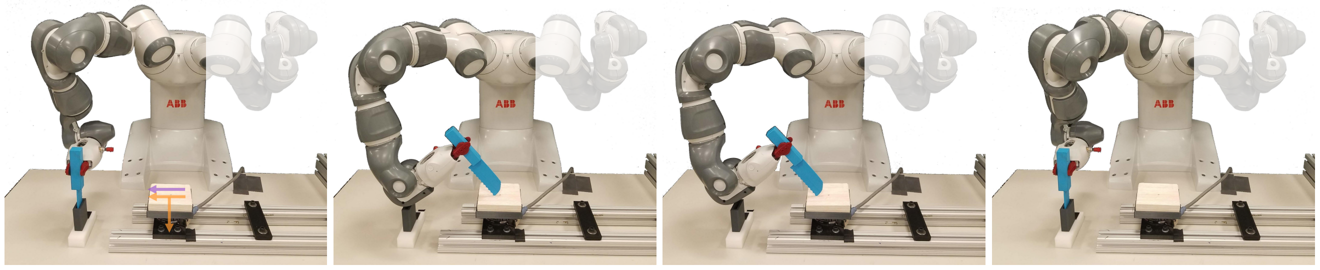


Fig. 7: `knife_cutting`. The task motion (purple) is to cut across the block and the task force (orange) is to exert down and across (far left).

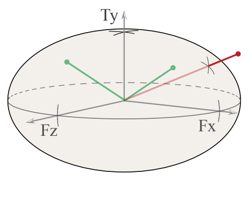
the planner used to generate a joint-space manipulator path ξ_a that follows the task-space tool path $\bar{\xi_T}$ while resisting external wrenches.

For all other collision-free planning, we use a bidirectional rapidly-exploring random tree (BiRRT) [36], [37].

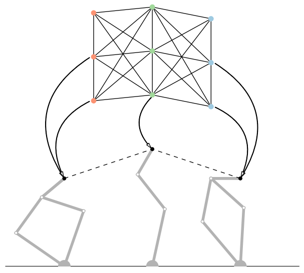
A. Stable Grasping

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.2a), finger contacts occur on parallel surfaces.

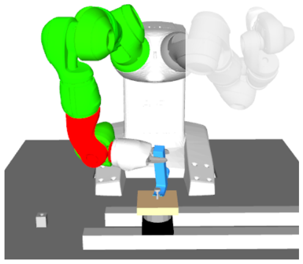


(a) Limit Surface

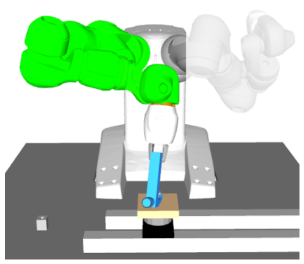


(b) Layered Graph

Fig. 8: a) 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. b) 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 [41].



(a) Torque Fault at Joints 5, 7



(b) Respecting Torque Limits

Fig. 9: 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 in (a), since joints five and seven both experience a torque fault. By contrast, enforcing the constraint leads to the successful grasp and path shown in (b).

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 [38].

The boundary of the set of frictional wrenches that a patch contact can provide is known as the limit surface [16]. 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 [39]. 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 [39], [40].

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:

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

Therefore a grasp is stable if (1) is true, since the contact can support the external wrenches. This check is illustrated in Fig.8a, 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).

B. *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. IV-D.

C. *Force-Feasible Grasping*

While in Sec. IV-B 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. IV-B to verify that, for each waypoint, there is an IK solution q such that:

$$J^T(q)w_{ext} < \tau_{lim}. \tag{2}$$

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.9a. By enforcing (2), the robot selects a different grasp, shown in Fig.9b, that leads to successful execution.

D. 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. IV-B, 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 [41], which measures closeness via the discrete Fréchet distance [42].

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.8b). 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. 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 (2).

V. Experimental Setup

Figs. 3-6 show our physical task setup. We used an ABB YuMi [43] 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. III-B.
- 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) [23], 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.10). 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.2b) 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.

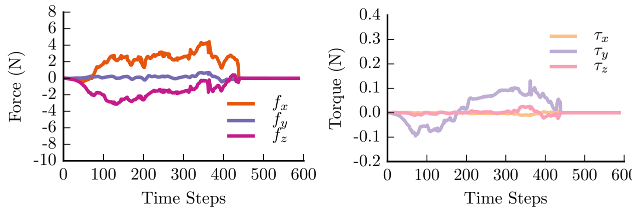


Fig. 10: We track the force and torque output while a human completes the `knife_cutting` task. This as used as a reference for the force and torque needed to complete the task.

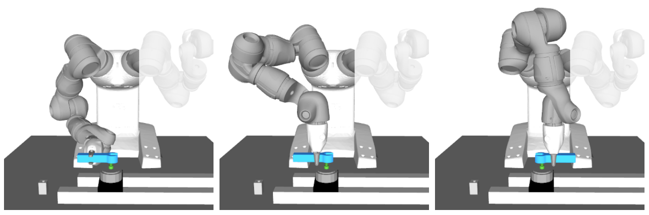


Fig. 11: Here we show three grasps for `wrench_turning`. The left-most is kinematically feasible but not stable. The middle is stable but kinematically infeasible. The right is both stable and kinematically feasible.

- 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.

We make 3D printable models of our tools[2], tool holders and environment pieces, along with several other task descriptors available online [44]. We hope this enables and encourages others to explore this type of problem.

VI. Experiments

Using the physical setup described in Sec. V, we explore how the task constraints impact the overall solution. The presentation in this section aligns with the variable dependency tree (Fig.3c), exploring each variable and their corresponding constraints. We begin with how task-specific grasping constraints impact the available grasps (Sec. VI-A) and how torque constraints impact the path-following algorithm (Sec. VI-B). We also briefly discuss feasibility of finding collision-free paths (Sec. VI-C). We conclude with real robot experiments that both demonstrate our approach and explore the impact of domain changes (Sec. VI-D).

A. 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. III-C, 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. IV-B and Sec. IV-C are satifisied if the heuristic succeeds for any T.

Table I shows the results. In addition to showing how many grasps satisfy each constraint, we show how many

[2]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.

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 I: 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 $C_0 \cap C_2$

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.

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.

B. *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. IV-D). 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 II 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.12 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 I. At one extreme, the screwdriver task is easier and results in a nearly full set of solutions and, at the other extreme, the hammer task is quite difficult and has a very limited set of kinematic solutions.

C. *Collision-Free Planning*

Following the chain of dependencies in Fig.3c, 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

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 II: 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.

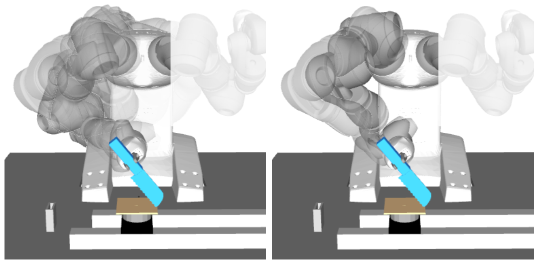


Fig. 12: The left shows a set of IK solutions for a particular layer. The right shows that only two configurations satisfy the torque constraint.

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.

D. *Domain Change*

We next execute each of the tasks on the real robot, as shown in [44]. 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.10, We successfully cut into the block, experiencing a force profile shown in Fig.13(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.13(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.

VII. Discussion

The goal of this work is to enable a robot to complete tasks that require both planning motions and forces. We present tool use as an illustrative example of this type of forceful manipulation, which involves grasp, force and motion constraints. We model tasks as instances of constraint satisfaction over continuous variables such as grasps, poses and paths.

In the future, this initial benchmark can be expanded to encompass other areas of manipulation and planning. 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 [45]. 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 focus on tool use, the principles for forceful manipulation apply across a wide range of applications.

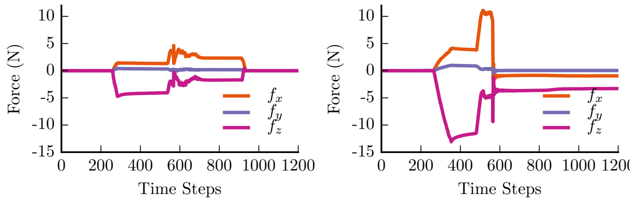


Fig. 13: Force traces of the robot executing the `knife_cutting` task. The left shows results from the standard version of the experiment. The right shows the results if we raise the block, thus asking the robot to cut deeper. The resultant task forces were higher then what we planned for, leading to the torque-limit fail at the 600 mark.

ACKNOWLEDGMENTS

This work was supported by the National Robotics Initiative IIS-1637753, MIT Merrill Lynch Fellowship and the NSF Graduate Research Fellowship. We thank the members of the MCube Lab and LIS Group for helpful discussion and advice. We would especially like to thank Elliott Donlon and Francois Hogan for their fabrication assistance and Nikhil Chavan-Dafle for the many mechanics discussions.

REFERENCES

[1] F. Guerin, N. Kruger, and D. Kraft, "A survey of the ontogeny of tool use: from sensorimotor experience to planning," *TAMD*, vol. 5, no. 1, pp. 18–45, 2013.

[2] R. W. Shumaker, K. R. Walkup, and B. B. Beck, *Animal tool behavior: the use and manufacture of tools by animals*. JHU Press, 2011.

[3] B. B. Beck, *Animal tool behavior*. Garland STPM Pub., 1980.

[4] L. Sauer, "An AI formalization of Betty the Crow's sequential geometric tool use," Master's thesis, University of Stuttgart, 2015.

[5] K. P. Tee, J. Li, L. T. P. Chen, K. W. Wan, and G. Ganesh, "Towards Emergence of Tool Use in Robots: Automatic Tool Recognition and Use without Prior Tool Learning," in *ICRA*, IEEE, 2018.

[6] S. Brown and C. Sammut, "An architecture for tool use and learning in robots," in *Australian Conference on Robotics and Automation*, Citeseer, 2007.

[7] H. Wicaksono and C. Sammut, "Relational Tool Use Learning by a Robot in a Real and Simulated World," in *ACRA*, 2016.

[8] A. Xie, F. Ebert, S. Levine, and C. Finn, "Improvisation through Physical Understanding: Using Novel Objects as Tools with Visual Foresight," *arXiv preprint arXiv:1904.05538*, 2019.

[9] D. Prattichizzo and J. C. Trinkle, "Grasping," in *Springer handbook of robotics*, pp. 671–700, Springer, 2008.

[10] Z. Li and S. S. Sastry, "Task-oriented optimal grasping by multifingered robot hands," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 32–44, 1988.

[11] N. S. Pollard, "Parallel methods for synthesizing whole-hand grasps from generalized prototypes," tech. rep., MIT AI Lab, 1994.

[12] C. Borst, M. Fischer, and G. Hirzinger, "Grasp planning: How to choose a suitable task wrench space," in *ICRA*, vol. 1, pp. 319–325, IEEE, 2004.

[13] R. Haschke, J. J. Steil, I. Steuwer, and H. Ritter, "Task-oriented quality measures for dextrous grasping," in *CIRA*, pp. 689–694, IEEE, 2005.

[14] Y. Lin and Y. Sun, "Grasp planning to maximize task coverage," *IJRR*, vol. 34, no. 9, pp. 1195–1210, 2015.

[15] Y. Lin and Y. Sun, "Task-oriented grasp planning based on disturbance distribution," in *Robotics Research*, pp. 577–592, Springer, 2016.

[16] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction part 1. limit surface and moment function," *Wear*, vol. 143, no. 2, pp. 307–330, 1991.

[17] N. Chavan-Dafle, R. Holladay, and A. Rodriguez, "In-Hand Manipulation via Motion Cones," in *RSS*, 2018.

[18] E. Nikandrova and V. Kyrki, "Category-based task specific grasping," *RAS*, vol. 70, pp. 25–35, 2015.

[19] D. Song, C. H. Ek, K. Huebner, and D. Kragic, "Task-based robot grasp planning using probabilistic inference," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 546–561, 2015.

[20] M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic, "Affordance detection for task-specific grasping using deep learning," in *Humanoids*, pp. 91–98, IEEE, 2017.

[21] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision," in *RSS*, 2018.

[22] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 159–185, 2018.

[23] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *ICRA*, pp. 625–632, IEEE, 2009.

[24] L. Chen, L. F. Figueredo, and M. Dogar, "Manipulation Planning under Changing External Forces," in *IROS*, pp. 3503–3510, IEEE, 2018.

[25] R. Matsuzaki, M. Kamibayashi, S. Sakaino, and T. Tsuji, "Classification of a hybrid control system for robotic tool use," in *ICM*, pp. 712–717, IEEE, 2013.

[26] H. Asada and Y. Asari, "The direct teaching of tool manipulation skills via the impedance identification of human motions," in *ICRA*, pp. 1269–1274, IEEE, 1988.

[27] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *ICDL*, pp. 91–96, IEEE, 2008.

[28] V. Tikhanoff, U. Pattacini, L. Natale, and G. Metta, "Exploring affordances and tool use on the iCub," in *Humanoids*, pp. 130–137, IEEE, 2013.

[29] S. Elliott, M. Valente, and M. Cakmak, "Making objects graspable in confined environments through push and pull manipulation with a tool," in *ICRA*, pp. 4851–4858, IEEE, 2016.

[30] R. Jain and T. Inamura, "Learning of usage of tools based on interaction between humans and robots," in *CYBER*, pp. 597–602, IEEE, 2014.

[31] A. Antunes, G. Saponaro, A. Dehban, L. Jamone, R. Ventura, A. Bernardino, and J. Santos-Victor, "Robotic tool use and problem solving based on probabilistic planning and learned affordances," in *IROS Workshop*, 2015.

[32] A. Stoytchev, "Behavior-grounded representation of tool affordances," in *ICRA*, pp. 3060–3065, IEEE, 2005.

[33] W. Li and M. Fritz, "Teaching robots the use of human tools from demonstration with non-dexterous end-effectors," in *Humanoids*, pp. 547–553, IEEE, 2015.

[34] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *RSS*, 2018.

[35] M. Toussaint, K. Allen, K. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *RSS*, 2018.

[36] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," *WAFR*, 2000.

[37] D. Berenson, *Constrained manipulation planning*. Carnegie Mellon University, 2011.

[38] N. Chavan-Dafle and A. Rodriguez, "Prehensile pushing: In-hand manipulation with push-primitives," in *IROS*, pp. 6215–6222, IEEE, 2015.

[39] N. Xydas and I. Kao, "Modeling of contact mechanics and friction limit surfaces for soft fingers in robotics, with experimental results," *IJRR*, vol. 18, no. 9, pp. 941–950, 1999.

[40] J. Shi, J. Z. Woodruff, P. B. Umbanhowar, and K. M. Lynch, "Dynamic in-hand sliding manipulation," *Transactions on Robotics*, vol. 33, no. 4, pp. 778–795, 2017.

[41] R. Holladay, O. Salzman, and S. Srinivasa, "Minimizing task space Frechet error via efficient incremental graph search," *Robotics and Automation Letters*, vol. 4, no. 2, pp. 1999–2006, 2019.

[42] R. Holladay and S. Srinivasa, "Distance metrics and algorithms for task space path optimization," in *IROS*, pp. 5533–5540, IEEE, 2016.

[43] A. Robotics, "Abb yumi," *URL: http://new. abb. com/products/robotics/yumi/. Last visited*, 2015.

[44] M. Lab, "Tool use dataset," *URL: https://mcube.mit.edu/tool-use/*, 2019.

[45] P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," in *ICRA*, vol. 4, pp. 1924–1928, IEEE, 1987.