Exploiting Structure for Efficient Robotic Manipulation

by

Ben Michael Abbatematteo

B. S., University of Rochester, 2017

Sc. M., Brown University, 2019

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

October 2023

This dissertation by Ben Michael Abbatematteo is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

George Konidaris, Director

Recommended to the Graduate Council

Date _____ _____

Stefanie Tellex, Reader

Date _____ _____

Daniel Ritchie, Reader

Approved by the Graduate Council

Date _____ _____

Thomas A. Lewis
Dean of the Graduate School

In memory of Drew Michael Buxton

Abstract of "Exploiting Structure for Efficient Robotic Manipulation" by Ben Michael Abbatematteo, Sc. M, Brown University, October 2023.

With few exceptions, robots today are unable to quickly acquire new manipulation skills in the real world. The modern data-driven approach to skill learning is thwarted in practice by the sample complexity of learning algorithms and a lack of diverse, high-quality data. This thesis develops algorithms and frameworks that leverage structured models of perception and action to improve the efficiency of robot skill learning. I first describe our work on modeling articulated objects, posing a new problem formulation and introducing systems capable of autonomously manipulating novel objects. I subsequently show how these models can be used to bootstrap motor skill learning. The thesis then turns to study structure in behavior, illustrating a novel class of policies that are efficiently learnable and readily composable. Together, these methods present a path toward skill learning in real-time in the real world.

Acknowledgements

I was, admittedly, pretty green when I arrived at Brown CS. I had taken some robotics courses, and I knew my way around a terminal (largely thanks to Tom Howard), but boy, did I have a lot to learn. George Konidaris was patient, above all, as I learned the ropes of RL, and then vision, and then control theory, too. George's understanding of the field, foresight of where all of this is headed, and dedication to leave his mark on it are singular, and were invaluable time and time again. Thanks, George, for taking a chance, and for seeing me through.

George's schemes to solve AI were steadily balanced by Stefanie Tellex's consistent, pragmatic question: what can we make the robot do next? I am so fortunate to have had both perspectives while finding my footing and learning what makes a good problem. I would like to acknowledge Daniel Ritchie and James Tompkin for providing outstanding technical and professional advice at every turn: I'm a more capable researcher and presenter in no small part because of your time and effort over the years. Thank you. And to my undergraduate mentors — Laurel Carney and Shawn Newlands, for giving me a compelling introduction to research; and Tom Howard, for outstanding instruction, help in applying to PhD programs, and guidance in thinking about a dissertation topic in a new field.

Thinking back to the early days of George's group at Brown, I'm especially grateful for the guidance of more senior students who would become dear friends: Cam Allen, Ben Burchfiel, and Barrett Ames taught me much of RL, vision, and control, respectively; this thesis would not have been possible without each of you forging a path ahead of me. And to Matt Corsaro, for being the best pal through it all. Nakul Gopalan, Dave Abel, Steve James, and Kavosh Asadi also deserve recognition for their guidance in those early years.

This thesis would not be possible without the hard work of many brilliant collaborators. In particular, I would like to thank Eric Rosen, Skye Thompson, Tuluhan Akbulut, Hameed Abdul-Rashid, Miles Freeman, Seiji Shaw, Akhil Bagaria, Sreehari Rammohan, Matthew Corsaro, Omer Gottesman, Rachel Ma, and David Paulius for their efforts. It's been a pleasure to work with and learn from each of you. Nor would it have been possible without the support from A-staff and T-staff — thank you, especially to Suzanne Alden, for all the travel and grant support over the years.

To my many friends in the IRL and H2R not yet mentioned — Max, Thao, Ifrah, Kaiyu, Deemer, Jason, Saket, Sam, Rafa, David, to name only a few — thank you for many laughs and many happy memories.

Lastly, to my family.

Reyna Joyce can recite my job talk at this point — a testament to her patience and the depth of her love. I could not have done it with you. Thank you.

I have been fortunate enough to do a PhD in my hometown, which means I have had the distinct advantages of home-cooked meals, regular family gatherings, and summer days at the lake. Thank you to the Abbatematteo and Ashcroft clans for their boundless support.

And, above all, to my Mom and Dad. I'm so lucky to have been close to home all these years, to embark on this journey together. All of this is because of you — decades of hard work, sacrifices, and endless love. Thank you for making this wonderful life possible for us.

This research was supported by NSF CAREER Award 1844960, an Amazon Faculty Research Award, AFOSR DURIP FA9550-21-1-0308, and the ONR under contracts N00014-22-1-2592, N00014-21-1-2584.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The hallmark of natural intelligence is the ability to manipulate the world to accomplish one's goals. Across the animal kingdom, creatures grasp and use objects for gathering food and water, grooming, protection, construction, or just plain recreation. Primates are nature's most famous dexterous manipulators, but even birds, fish, and insects use tools to survive. Manipulation serves as a litmus test of intelligence — hard manipulation problems require not only physical dexterity but also perception, planning, directed exploration, and reasoning about uncertainty, all in real-time with constrained computational resources.

If we hope to build robots that are as intelligent as people, or even just robots that are useful, we must solve the grand challenge of designing robots that can similarly skillfully interact with the world around them. Manipulation, then, is at the core of the artificial intelligence problem. And it is hard: tasks that are extraordinarily challenging for humans (e.g. playing chess well) are relatively simple for computers, whereas simple tasks for humans (grasping, using tools, *manipulation*) are excruciatingly difficult for robots. This phenomenon, known as Moravec's paradox (Moravec, 1988), suggests that we have our work cut out for us.

While some natural agents are born capable of interacting with the world, the most sophisticated object manipulation behaviors require years to develop. Human children are born relatively hapless, and in the first few years of life acquire the physical dexterity and cognitive skills required to stack blocks, race toy cars, and survive in the world. Human adolescents and adults further refine these abilities on specific tasks, like shifting gears, using tools, or playing sports, for utility, financial gain, or enjoyment. Eons of evolution have provided the scaffolding for our learning systems to undergo

Figure 1.1: Some examples of natural agents exhibiting dexterous manipulation. *Top row*: a human child stacks blocks (image licensed from stock.adobe.com); a macaque uses a stone tool (photo by Michael D. Gumert, CC BY 2.5). *Bottom row*: a bonobo fishes for termites with a stick (photo by Mike Richey, CC BY 2.5.); a crow solves a puzzle with a tool (photo by Sarah Jelbert, CC BY 4.0); a capuchin cracks a tough nut (photo by Tiago Falótico, CC BY 4.0). These contact-rich skills are beyond the current capabilities of artificial agents.

remarkably consistent development — it is our task to do the same for our robots. The central question in robotics today is how we approach this endeavor: how can we, as designers, equip robots with this ability to learn to interact with the world?

Robotic manipulation presents three key challenges. The first is the outright *complexity* of interacting with the world: the dynamics of contact are discontinuous, making traditional control methods all but obsolete, even assuming perfect knowledge of the geometry and mechanics of the world. Moreover, robots are cursed to interact with the world through their sensors and actuators, presenting large, continuous observation and action spaces that make reinforcement learning of such skills tremendously challenging. The second is the constant *novelty* of interacting with an open world. Robots today are successful in factory environments where they perform fixed, predefined tasks with perfect knowledge of the environment. In human environments, however, our agents will lack complete knowledge of the geometry and dynamics of the world they encounter, and must instead try to make sense of the world's myriad appearances in order to accomplish their goals.

They will encounter objects they have never seen before, for which they lack good models, and be asked to perform tasks their designers had not foreseen. As such, they must adapt to ever-changing environments and demands — it is here that learning for perception and control is necessary. The third challenge is *data scarcity*: it is expensive and time consuming to collect even a small amount of data on a real robot. Relying solely on data and scaling computing resources, a tactic that has proven tremendously successful in other AI disciplines like computer vision and natural language processing, is not an option in robotics — we simply cannot gather enough robotic data to train systems in the same tabula rasa way. We must be more clever about how we scaffold the learning systems of our agents — this is the central effort of this thesis.

Despite the novelty and complexity of the world, there does exist *structure* in the regularities across the tasks that a robot must face — e.g. space is three-dimensional, objects persist and consist of parts, physical laws are constant, and time always marches forward. If we hope to build efficient learning systems, we must carefully consider which of these elements the system should be forced to learn and which can be designed a priori. This decision — of what to learn, and what not to — is central to the scientific endeavor of creating intelligent robots. This thesis seeks to identify this structure across manipulation problems, and develops models of perception and action which exploit that structure in order to make learning for manipulation a more practical reality. The central claim of the thesis is:

> *By exploiting structure present across tasks faced by a robotic agent, we may imbue our learning systems with greater data efficiency and generalization.*

One such critical element of structure is the presence of *articulated objects* — objects consisting of rigid parts connected by joints (doors, drawers, cabinets, refrigerators, etc.). These objects are ubiquitous in human environments, and interactions with them occur in nearly every task we would like a robot to perform. To make dinner, a robot must obtain ingredients from the refrigerator, tools from a kitchen drawer, and spices from the cabinet. To clean up, it must place dirty dishes into the racks of the dishwasher, and obtain the broom from the closet. All of these interactions require that the robot understands the degrees of freedom of these objects, and that the robot possesses the dexterity to robustly manipulate them. We first set out to build such systems, then leverage them to bootstrap more dexterous motor skill learning.

Second, behavior itself is structured. Behavior is *modular* and *compositional*, and the design of

our motor skills should reflect this. Long tasks inherently break down into simpler, reusable routines, many of which the robot leaves the factory capable of, like moving its arm in free space, or grasping objects. The truly challenging phases of manipulation are the contact-rich interactions with the environment that are necessarily difficult to model. To this end, we developed a structured policy class that is able to much more efficiently and safely learn a class of contact-rich motor skills.

1.1 Contributions

This thesis makes three contributions.

1. First, we formulate the problem of category-level articulated object manipulation — perceiving and autonomously interacting with novel articulated objects from static observations. Solving this problem equips robots with the ability to enter human environments and immediately begin intelligently interacting with objects. The system we built leverages object recognition and exploits regularities in object categories; we then extended this system, relaxing its assumptions to recover object models by dynamically identifying parts and constructing graph representations of more diverse objects.

2. Second, we build upon these articulated object models to bootstrap motor skill learning using kinematic motion planning. Training robots to accomplish basic manipulation tasks is tremendously challenging — robots receive high-dimensional sensor observations from cameras, joint encoders, and tactile sensors, and need to coordinate long sequences of fine-grained motor control in order to accomplish their goals. Naively attempting to train this behavior from scratch for every new task simply is not an option when we want to train a robot in the real world. Humans use planning to develop approximate solutions when faced with a new task, then convert these planned, cognitively demanding behaviors to performant feedback policies executed with very little cognition. We formalize this process in a reinforcement learning setting, and build a system that leverages approximate models in order to initialize skill learning, showing dramatic improvements in sample efficiency.

3. Third, we develop a policy class for manipulation problems, Compositional Interaction Primitives, that is safely and efficiently learnable, and composable for generating temporally extended behavior. We achieve this design by exploiting structure present in behavior, and

leveraging well-studied robotics subproblems like motion planning and grasp detection. The result is a system capable of learning and sequencing skills in the real world in remarkably few trials.

Taken together, contributions constitute a step toward robots that are capable of learning new tasks in the real world, in real time.

Chapter 2

Background

In this section we present the concepts, formalisms, and notation relevant for the remainder of the dissertation.

2.1 Robotic Manipulation

Robotic manipulation is a term that has historically evaded a precise definition. For our purposes, we will use the broad definition suggested by Mason (2018):

Manipulation refers to an agent's control of its environment through selective contact.

Nearly everything a robot physically does falls under this umbrella, from pick-and-place interactions tool use, and even playing chess. In this sense, manipulation is the "business end" of the artificial intelligence problem, in which we must confront the complexity of interacting with the world to accomplish difficult, often abstract goals. Where "manipulation" research once referred to robotic grasping in particular, now it encompasses the effort to develop a wide range of skills from object rearrangement to tool use. As such, it requires that we engage with not only the control literature but also perception and planning, bringing techniques from across robotics and artificial intelligence disciplines to bear.

The central difficulty in manipulating the world is the inherent novelty and complexity involved. The physics of contact-rich manipulation inherently poses very challenging optimization problems: discontinuities introduced by changes in contact state make naive, model-based solutions intractable

for multi-step manipulation problems, even with full knowledge of the world's dynamics. Even with such perfect information, solving these problems is still an active research area — typically under the heading contact-implicit trajectory optimization (Posa et al., 2014; Manchester and Kuindersma, 2020; Huang et al., 2023). But, of course, robots (and humans alike) never have perfect information in real environments. It is impossible for our agents to discern latent physical quantities (inertia, friction coefficients, etc.) from visual inspection alone, and identifying them through physical interaction is prohibitively tedious. Thus, the novelty of objects and situations in the world also poses a tremendous difficulty to our robotic agents. Nonetheless, these are the problems we need robots to solve, both to be useful and generally intelligent. This section will cover the basic mathematical principles underlying the dynamics and control of robotic arms.

2.1.1 Manipulator Dynamics

The position of a robot can be described by its joint *configuration* $q \in \mathcal{C}$, where $\mathcal{C}$ is the *configuration space* (Lozano-Perez, 1981). In these generalized coordinates, we can prescribe the mechanics governing the robot's motion by the following equation (Tedrake, 2022):

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_g(q) + \tau + \sum_i J_i^T(q) f^{c_i} \tag{2.1}$$

with mass matrix M, Coriolis terms C, torques due to gravity τ_g, torque supplied by the robot's motors τ, and summation over contact forces from the environment. Each contact results in a force f^{c_i} that is propagated to the configuration space via a *contact Jacobian* J_i.

In the case in which the robot moves through free space, contact forces are zero and control is relatively straightforward. If we can find a trajectory through configuration space (e.g. via motion planning, see Section 2.2), we can command that trajectory directly by using the known manipulator dynamics. Suppose we have a desired trajectory $\{\ddot{q}_d, \dot{q}_d, q_d\}_{t=1}^T$, if we command

$$\tau = M(q)\ddot{q}_d + C(q,\dot{q})\dot{q} - \tau_g(q),$$

the robot should track the desired accelerations. Typically, we have small errors in the model of the robot, so we account for them with PD control with the following control law (perhaps also adding

an integral term):

$$\ddot{q}_d = \ddot{q}_d(t) + K_p(q_d(t) - q) + K_d(\dot{q}_d(t) - \dot{q}),$$

with gain matrices K_p and K_d. In this way, we can move the arm through free space, or in contact when the forces are negligible. Similar model-based strategies (e.g. force control, hybrid position/force control) exist when a precise model of the environment is available.

However, interesting manipulation problems are ones in which the contact forces are non-negligible and unknown a priori. In real settings, it is typically difficult or impossible to estimate expected contact forces ahead of interaction. The robot is not privy to a full dynamics model of the world complete with friction coefficients, inertia matrices, precise geometries, and contact Jacobians when it encounters new situations; moreover, these quantities may not be inferred from visual sensors alone. As a result, the model-based control methods described above fail to scale to diverse, real settings. Care must be taken such that the robot is naturally compliant and robust to modeling errors, and learning is typically required to compute the desired control.

2.1.2 Cartesian Impedance Control

Consider a robot turning a crank: it is clear that the robot needs to move its end-effector in a circular motion, but where precisely should this circle be placed? Small errors in this estimation can result in dramatically large forces, damaging the robot or the crank. Thus, robot motions must be *compliant* in the presence of imperfect models. One intermediate step is then to re-write the manipulator dynamics as a simple dynamical system in the world frame. This approach is typically referred to as *stiffness* or *impedance* control (Salisbury, 1980; Hogan, 1985).

We can write the robot's dynamics in the end-effector space as follows. Denote the pose of the end-effector as $p \in SE(3)$, computed via the robot's forward kinematics. The same dynamics equation above can be re-written in the frame of the end-effector as

$$\Lambda(q)\ddot{p} + C_X(q, \dot{q})\dot{q} = f_g(q) + f_u + f_{ext}, \tag{2.2}$$

where now we represent the mass matrix $\Lambda(q)$ and Coriolis terms $C_X(q, \dot{q})$ in the end-effector frame rather than the joint space (through some algebraic manipulations, see Tedrake (2022)). Gravity is now a Cartesian force f_g, the motor torques act as a virtual force on the end-effector f_u, and

contact force at the end-effector is given by f_{ext}. We can relate the motor torques to the force at the end-effector f_u via the Jacobian $\tau = J(q)^T f_u$. If we then command the following control,

$$f_u = -f_g(q) + K_p(p_d - p) + K_d(\dot{p}_d - \dot{p}),$$

the dynamics at the end-effector are akin to a spring-mass damper system with reference pose p_d and velocity $\dot{p}_d$. This enables us to naturally achieve compliant motions at the point of contact with the world, yielding robustness in the presence of approximate models of the world and/or learned control policies.

One further modification is required to deploy stiffness control on redundant manipulators: nullspace control. As the degrees of freedom of the robot are typically greater than the degrees of freedom of the end-effector in Cartesian space, the remainder of the joints must be stabilized, typically via a PD control term relative to a pre-specified reference pose (often the center of the joint range). The resulting torques commanded to the arm are then

$$\tau = J^T f_u + [I - J^+ J][K_{p,joint}(q_0 - q) - K_{d,joint}\dot{q}]$$

where q_0 denotes the nullspace reference pose. Taken together, this approach constitutes *operational space control* as proposed by Khatib (1987). This controller is deployed in a reinforcement learning context in the later chapters of this thesis, enabling the agent to learn to control the end-effector pose and regulate its compliance — it represents an important aspect of *structure* present in manipulation problems to be exploited, rather than forcing the robot to learn to command torques directly.

2.2 Motion Planning

Motion planning is the problem of finding a path (sequence of poses) through configuration space such that the robot is moved to a desired goal configuration, without encountering a collision. While there exist many different families of motion planning algorithms (LaValle, 2006), they all operate in a similar fashion: given a configuration space $\mathcal{C}$ and start and goal joint configurations $q_0, q^* \in \mathcal{C}$, return a collision-free path of joint configurations $\{q_t\}_{t=0}^T$ between the start and end configurations.

Sample-based motion planners provide a principled approach for quickly generating collision-free robot trajectories. However, motion planners are only as effective as their kinematic models

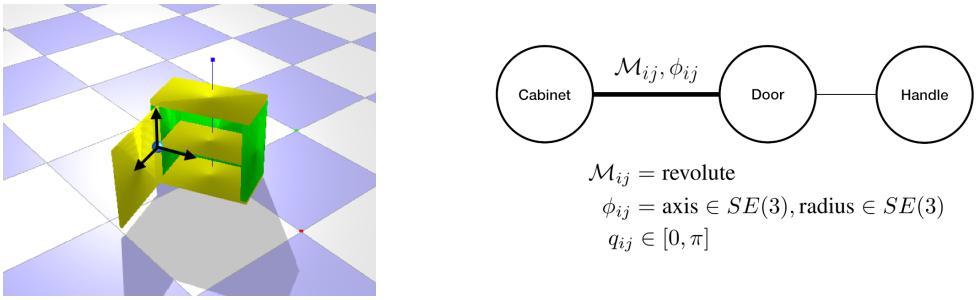


Figure 2.1: **Kinematic Graph.** A simulated object, annotated with the pose of its axis of rotation and corresponding kinematic graph.

are accurate: they generate trajectories directly, and thus cannot be improved through subsequent interaction and learning. Furthermore, kinematic planners typically produce trajectories that only account for kinematics, not dynamics: they explicitly do not account for forces involved in motion, such as friction, inertial forces, motor torques, etc, which are important for performing contact-rich, dexterous manipulation.

While we can actuate robot joints by using its motors, when manipulating the degrees of freedom of an object, we must use the robot's body to indirectly actuate them. For example, a robot wishing to open a cupboard must first grasp the cupboard's door before it can manipulate its hinge joint. Nonetheless, we can form and exploit kinematic plans in the configuration space of *objects* as well for efficient exploration. In these cases, we often employ *constrained* motion planning (Berenson et al., 2009) in which we explicitly model the grasp constraint and plan motions jointly for the robot and the object. Chapters 3 and 4 employ a similar approach to interact with objects in the world; Chapter 5 leverages such motion planners to initialize motor skill learning.

2.3 Modeling Articulated Objects

One common set of tasks robots will need to be capable of in home and industrial environments is manipulating articulated objects: objects consisting of rigid parts connected by movable joints, like cabinets or drawers. Articulated objects are commonly represented by a graph that encodes kinematic relationships between parts (Sturm et al., 2011). With these models, robots can begin to derive appropriate plans and control sequences to interact with them.

A *kinematic model*, $\mathcal{M}$, encodes a motion constraint between two object parts; the most prevalent model types are revolute (like a door) and prismatic (like a drawer). A kinematic model's *parameter*

vector, ϕ, encodes the constraints on the pose of the object parts, represented as the 6-DoF pose of the axis of rotation/translation in the world frame, and the 6-DoF pose of the articulated sub-part's origin in the axis frame. This parameter vector varies with each object instance, depending upon the object's pose and geometry. For example, the exact trajectory of a microwave door will be different for different microwaves. The object's *configuration*, q, represents the articulated pose of the object, encoding angles in revolute models and displacements in prismatic models, and grows in dimensionality with the number of joints present in the object. An object's configuration may change with each observation as forces are applied to its parts.

A *kinematic graph* $G = (V_G, E_G)$ consists of a set of vertices $V_G = \{1, ..., p\}$ representing the object's p parts, and a set of undirected edges $E_G \subset V_G \times V_G$ specifying the motion constraints between the parts. Each joint (ij) is assigned a kinematic model $\mathcal{M}_{ij}$, parameter vector ϕ_{ij}, and configuration q_{ij}. The graph of a cabinet, for example, consists of three nodes (a body, a door, and a handle), with a revolute kinematic constraint encoded in the edge between the body and door, as illustrated in Figure 2.1. In this example, the parameter vector, ϕ, encodes the position and orientation of the axis of rotation between the body and the door as well as the radius of the door's rotation, and the configuration, q, encodes the angle made by the door and the body. The kinematic model $\mathcal{M}$, is revolute. Typically, it is assumed that these graphs are trees without cyclic dependencies in order to make inference tractable. In Chapter 4, these graph structures are referred to as *kinematic hierarchies*.

2.3.1 Related Work

The problem of estimating the kinematic model and articulated pose of an individual object has been studied extensively in the robotics and computer vision literature.

Many approaches strictly perform articulated pose estimation (e.g. Brookshire and Teller (2016) and Desingh et al. (2019)), estimating the configuration of an object with a known kinematic model. Several approaches estimate kinematic graphs from video demonstrations, tracking sub-parts either with fiducial markers or by clustering feature trajectories (Sturm et al., 2011; Pillai et al., 2015; Niekum et al., 2015; Daniele et al., 2017; Yan and Pollefeys, 2006). More recent learning-based approaches to category-level model and pose estimation from video (Jain et al., 2021, 2022) have shown promise. Some category-level models also recover object part size via canonicalization (Li et al., 2020; Weng et al., 2021). These approaches, however, are typically limited to a priori known

category-level graph templates. Exceptions include Heppert et al. (2022), which employs factor graphs to efficiently build articulation models and track parts given video data.

Other approaches explore object mechanisms and joint dependencies interactively, leveraging the embodiment of the robot, tracking sub-parts via fiducial markers or by clustering feature trajectories (Katz and Brock, 2008; Hausman et al., 2015; Barragän et al., 2014; Martín-Martin and Brock, 2014; Kulick et al., 2015; Baum et al., 2017; Sturm et al., 2010; Jain and Niekum, 2020; Gadre et al., 2021; Nie et al., 2022). These systems accurately estimate the kinematic structure of an object having first directly observed or estimated a sequence of sub-part poses. However, they require observing the very manipulation the robot aims to achieve. Moreover, these methods lack a notion of object type and typically assume uniform priors over model parameters.

Our work in Chapter 3 introduced the problem of estimating articulated object models from a *static* observation — essentially providing a prior distribution over the kinematic model with which the robot could begin interaction. Most subsequent approaches similarly assume category-level graph templates (Jain et al., 2021, 2022). Li et al. (2020) extend the idea of canonicalization to articulated objects to also recover part sizes from static observations.

Other approaches forgo kinematic graph estimation and attempt to recover per-part motion models without estimating the global graph structure. Zeng et al. (2021) estimate per-part motion models from static observations. Wang et al. (2019b) train a model to jointly predict movable parts and their motion parameters from an input point cloud. Yi et al. (2018) take a pair of articulated 3D shapes of the same category but in different kinematic poses, and jointly infer moving parts and their kinematic motion parameters. Yan et al. (2019) infer movable parts by hallucinating their motions over multiple time steps. These approaches train deep neural networks which generalize to geometrically and structurally different input objects. However, none infer the object's kinematic graph structure: they treat each articulated part as it if moves in isolation. RigNet (Xu et al., 2020b) is a system for inferring the skeletal joint structure of articulated humanoid and creature characters which uses a graph neural network for predicting graph topology information. Our work in Chapter 4 similarly employs graph neural networks to recover kinematic graphs for novel object instances, relaxing the assumption of an a priori known category-level template.

More recent methods eschew kinematic model estimation entirely, instead estimating *affordances* directly. For example, VAT-Mart (Wu et al., 2021), AdaAfford (Wang et al., 2022), and UMPNet (Xu et al., 2022) propose end-effector trajectories that accomplish a desired manipulation directly from

pointcloud input. Schiavi et al. (2022) extend these ideas to an agent-aware setting to handle joint limits and reachability, albeit under more restrictive assumptions. The FlowBot systems (Eisner et al., 2022; Zhang et al., 2023) predict a 3D scene flow to derive appropriate controls. Other approaches eschewing traditional kinematic model estimation include RoboBarista (Sung et al., 2018), which learns multi-modal embeddings of visual data, language commands, and trajectories for manipulating common object parts, and kinematic morphing networks (Englert and Toussaint, 2018) which transfer manipulation policies to novel doors. These affordance models are attractive in that they simultaneously solve the grasp detection problem, identifying candidate locations for interaction. However, they fail to recover articulation constraints or kinematic graphs, which are necessary for planning whole-body interactions (Mittal et al., 2022) or task and motion planning (Garrett et al., 2021).

Recent works have begun to leverage neural fields to create 3D representations of articulated objects, typically assuming known category-level templates and focusing on geometric reconstruction, e.g. Tseng et al. (2022); Mu et al. (2021); Lei et al. (2023); Kawana et al. (2022); Deng et al. (2020); Zhang et al. (2021), but sometimes inferring articulation properties from pairwise inputs (i.e. Ditto (Jiang et al., 2022; Hsu et al., 2023)). Human articulated pose estimation has also been studied extensively in the vision literature, e.g., Pavlakos et al. (2018); Wei et al. (2016); Zheng et al. (2020).

2.4 Reinforcement Learning

Consider a robot trying to perform a task involving contact, like opening a door or sweeping the floor. If we revisit the mechanics governing this interaction (Equation 2.1), the configuration q now not only includes the robot's joints but also the the state of the world: positions of objects and tools in the world, the states of object joints, etc. Now, the robot must confront the fact that it lacks complete knowledge of the dynamics of the world around it. While the robot has a good model of its own body, the precise mass and inertia of other objects are unknown a priori; their geometries, material properties, and the resulting contact forces are at best approximate; and even their positions and velocities must be estimated. The robot could attempt to recover all of this information through interaction (i.e. *system identification*), but it would be prohibitively tedious. Moreover, these contact-rich interactions are prohibitively complex: even with perfect knowledge

of the dynamics, solving for feedback controllers in such discontinuous, non-linear systems remains very challenging, even for state-of-the-art optimal control methods (Huang et al., 2023). As a result, a great deal of work instead seeks to leverage data to overcome these challenges, and employs *reinforcement learning* to train the robot's behavior via trial and error interaction.

Tasks of the robot can be formalized as a Markov Decision Process $M = (S, A, R, T, \gamma)$, where S is a set of states which describe the current configuration of the task; A is a set of actions available to the robot; $R(s, a, s')$ is a reward function describing the reward obtained for executing action a in state s and transitioning to state s'; $T(s'|s, a)$ is the transition function, describing the probability that executing a in s leaving the robot in state s', and $\gamma \in (0, 1]$ is a discount factor expressing a preference for immediate over delayed rewards. The robot's goal is to learn a policy π mapping a state to the action it should execute in that state, such that it maximizes the discounted sum of expected future rewards (or *return*) (Sutton and Barto, 1998).

In many cases, a target motor skill is not the entirety of the robot's task, but should instead be used as an executable subroutine used as part of the solution. Such skills are typically modeled using the *options framework* (Sutton et al., 1999), where an option o is defined by a tuple (I_o, π_o, β_o), where $I_o \subseteq S$ is the *initiation set*, the set of states from which the robot may choose to execute the option; $\beta_o : S \to [0, 1]$ is the *termination condition*, giving the probability that option execution ceases in state s; and π_o is the *option policy*. The robot can choose to execute o if the current state is inside I_o, whereupon execution proceeds according to π_o and halts at each encountered state according to β_o. Modeling motor skills as options naturally supports reasoning about sequential compositionality —option o_2 can be executed after option o_1 if the state that o_1 leaves the robot in lies within o_2's initiation set (Lozano-Perez et al., 1984; Burridge et al., 1999; Tedrake, 2009; Konidaris and Barto, 2009). Hierarchical RL studies the discovery and composition of such options (Nachum et al., 2018; Gupta et al., 2018; Huo et al., 2020).

2.4.1 Policy Search

Policy search methods (Deisenroth et al., 2013) are a family of model-free reinforcement learning algorithms that search within a parametric class of policies to maximize reward. Formally, given a Markov Decision Process $\mathcal{M}$, the objective of policy search is to maximize the expected return of

the policy π_θ:

$$\max_\theta \mathop{\mathbb{E}}_{\mathcal{M},\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t r_t\right]. \tag{2.3}$$

These approaches can learn motor skills through interaction and therefore do not require an explicit environment model, and are typically agnostic to the choice of policy class (though their success often depends on the policy class having the right balance of expressiveness and compactness). However, their model-free nature leads to high sample complexity, often making them infeasible to apply directly to robot learning.

2.4.2 Policy Representations

The form of the policy function $\pi : S \to A$ is a design decision critical to efficient reinforcement learning. In the natural world, we have discovered that *motor primitives* underlie many behaviors, like gaits or reaching motions (Thoroughman and Shadmehr, 2000; Flash and Hochner, 2005) — as such, there is an effort in robotics to produce similar compact, expressive representations of behavior.

The most historically significant motor primitive in robotics is the Dynamic Movement Primitive (Schaal, 2006; Ijspeert et al., 2002; Saveriano et al., 2021) which yields a dynamical system defined on the robot's state space:

$$\tau\dot{z} = \alpha_z(\beta_z(g-y)-z) + f(x), \tag{2.4}$$

$$\tau\dot{y} = z, \tag{2.5}$$

$$\tau\dot{x} = \alpha_x x, \tag{2.6}$$

where τ here represents a time constant, y represents position, x is a phase variable, z is an auxiliary variable, g defines the goal position, and α_z and β_z are gain parameters. The primitive is parameterized through weights w and basis functions Ψ in the term $f(x)$ which serve to modulate the shape of solution trajectories to this dynamical system, typically given as

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \tag{2.7}$$

where the basis functions Ψ are usually chosen as Radial Basis Functions. This simple parameterization enables fitting from a single demonstration (Schaal et al., 2002). Each DMP governs one degree

of freedom, typically either in the joint space or the Cartesian space of the robot's end-effector, where special care must be taken to address the dynamical system for orientation on $SO(3)$ (Ude et al., 2014; Koutras and Doulgeri, 2020). This policy class yields robot behavior that is smooth and asymptotically stable while retaining a low-dimensional parameterization, and as a result has been wildly successful — accomplishing tasks ranging from ball-in-cup (Peters and Schaal, 2008) and table tennis (Muelling et al., 2010, 2013) to contact-rich manipulation (Kalakrishnan et al., 2011). We refer the reader to the recent review paper from Saveriano et al. (2021) for a comprehensive survey, and discuss limitations and extensions below in Section 2.4.4. More recently, neural networks are commonly employed as the policy representation, sacrificing data efficiency for generality (Levine et al., 2016). The central effort of the latter half of this thesis is to develop a novel class of policies which retains a compact parameterization while increasing the expressivity of possible behaviors.

Related design considerations are the state and action spaces. The robot's sensors do not always constitute a Markov state space, so we instead deal in *observation* spaces, sidestepping a complicated and unknown a priori underlying state space. The robot receives high-dimensional observations in the form of data from cameras, tactile sensors, motor encoders, and force/torque sensors, and is ultimately forced to command torques to its motors. Often, though, reinforcement learning in these native observation and action spaces is prohibitively challenging, and neglects the structure present in each. In the tasks studied in this thesis, we typically assume the robot is able to estimate the pose of objects and their articulated state rather than learning in image observation spaces directly. This represents a tradeoff — lower-dimensional observation spaces should result in faster reinforcement learning (Sutton and Barto, 1998), potentially at the cost of less generalizable policies (Driess et al., 2022). The action spaces employed in this dissertation vary on the task; the work in Chapter 5 employs joint position control through dynamic movement primitives, whereas Chapters 6 and 7 permit the agent to control its end-effector pose and compliance as described in Section 2.1.2 and recently reviewed comprehensively by Abu-Dakka and Saveriano (2020). Formally, in the Variable Impedance Control in End-Effector Space (VICES) action space (Martín-Martín et al., 2019), the agent policy π maps observations to a desired delta end-effector position $\Delta^{pos} = (p_d - p)$ and rotation $\Delta^{ori} = R_d \ominus R$, as well as commanded stiffness terms $k_p^{pos} \in \mathbb{R}^{3\times3}$ and $k_p^{ori} \in \mathbb{R}^{3\times3}$ for position and rotation respectively. These terms are then used to map to joint torques τ via the following

equation, modified slightly from Equation 2.2:

$$\tau = J_{pos}^T[\Lambda^{pos}[k_p^{pos}(p_d - p) - k_d^{pos}v]] + J_{ori}^T[\Lambda^{ori}[k_p^{ori}(R_d \ominus R) - k_d^{ori}\omega]], \tag{2.8}$$

where Λ^{pos} and Λ^{ori} are the position and orientation components of the inertia matrix $\Lambda \in \mathbb{R}^{6\times6}$ in the end-effector frame, J_{pos} and J_{ori} are the position and orientation components of the end-effector Jacobian J, and $R_d \ominus R$ corresponds to subtraction in $SO(3)$. Linear and angular velocity are denoted by v and ω, respectively; $k_d^{pos} \in \mathbb{R}^{3\times3}$ and $k_d^{ori} \in \mathbb{R}^{3\times3}$ are the damping matrices for position and rotation, usually set such that the system is critically damped. This enables the agent to be compliant enough to be robust to perturbations and control end-effector pose directly, rather than being forced to confront control in the joint space. This leads to more performant, generalizable policies in most practical manipulation settings (Martín-Martín et al., 2019).

As the observation and action spaces are continuous in manipulation problems, policies are learned using policy search methods like CMAES (Stulp and Sigaud, 2012) or policy gradient actor-critic methods like TD3 (Fujimoto et al., 2018).

2.4.3 Learning from Demonstration

Due to the complexity of learning manipulation behaviors, in practice, researchers resort to human demonstrations of desired behavior. Typically this is done via teleoperation or "kinesthetic teaching", though recently learning from observations alone has been receiving attention (Brown et al., 2019; Torabi et al., 2019). Learning from Demonstration methods (Argall et al., 2009) broadly consist of two families of approaches that either mimic (Behavioral Cloning) or generalize (Inverse Reinforcement Learning) demonstrated behavior. Inverse reinforcement learning methods estimate a latent reward signal from a set of demonstrations; throughout this thesis, we assume a given reward function, and omit a discussion of inverse reinforcement learning methods here.

Behavioral cloning methods (Atkeson and Schaal, 1997; Pastor et al., 2009; Ho and Ermon, 2016) attempt to directly learn a policy that reproduces the demonstrated data. Given a dataset of expert demonstration trajectories D, the objective of behavioral cloning is:

$$\max_{\theta} \sum_{(s,a)\in D} \pi_\theta(a|s).$$

These methods often result in policies with undesirable behavior in states not observed during demonstrations, though this can addressed with interactive learning (Ross et al., 2011; Nair et al., 2017; Sun et al., 2017).

Many approaches investigate incorporating human-provided demonstrations into policy search to drastically reduce sample complexity via a reasonable initial policy and/or the integration of demonstrations in the learning objective (Cheng et al., 2018; Rajeswaran et al., 2017; Levine and Koltun, 2013). Others seek to generate demonstrations autonomously, e.g. with Model-Predictive Control (MPC) (Pan et al., 2017) or heuristically derived controllers (Kurenkov et al., 2019). Guided Policy Search (GPS) (Levine and Koltun, 2013) uses LQR to similarly aid in exploration. Residual learning has also been recently proposed, in which learn a policy is superimposed on hand-designed or model-predictive controllers (Silver et al., 2018; Johannink et al., 2019; Zeng et al., 2020). In Chapter 5, we study the problem of generating demonstrations autonomously, bootstrapping skill learning with useful exploration strategies derived from motion planning with kinematic models. Our method enables motion planning to bootstrap policies that are more expressive than the original planner.

2.4.4 Related Work

This thesis builds upon a tremendous amount of work that has applied reinforcement learning to problems in robotic manipulation. A great deal of recent work has examined the setting where a robot learns to map its sensor input directly to motor torques via deep reinforcement learning (Levine et al., 2016; Khazatsky et al., 2021). These methods offer flexibility, generality, and autonomy by exploiting recent advances in deep learning. However, that generality has a cost: such methods rely on access to massive amounts of compute and data and therefore typically require additional methods that implicitly encode design insight into the data set (Finn et al., 2017; Sadeghi and Levine, 2016; Andrychowicz et al., 2020), collect experience from multiple robots in parallel (Gu et al., 2017), or include human demonstration (Schaal, 1999; Argall et al., 2009; Nair et al., 2018). Additionally, these approaches make it difficult to incorporate the structural knowledge that robotics as a field has developed around techniques like forward and inverse kinematics, motion planning, wrench closure, safety, and feedback control.

The alternative approach, taken by earlier works, is to carefully design and structure a policy class to guarantee desirable properties (e.g., stability, joint and torque limits, and safety constraints)

while exploiting the properties of a broad class of target tasks to support sample-efficient learning — like parameterizing gaits (Saggar et al., 2007) or trajectories via Dynamic Movement Primitives (DMPs) (Kober and Peters, 2010; Muelling et al., 2013). The key assumption underlying DMPs is that dynamic motions can be represented largely as a trajectory shape—represented separately for each degree of freedom, as a linear combination of learned weights with basis functions over time—coupled with a second-order dynamical system that stably controls the robot towards the shape trajectory. Although DMPs have seen wide usage (Kroemer et al., 2021), they have largely not been integrated with high-dimensional, multi-modal data for contact-rich tasks (Saveriano et al., 2021), and typically must be bootstrapped by an expert demonstration trajectory (Argall et al., 2009). Other important policy classes overcome the standard shortcomings of DMPs, such as Probabilistic Movement Primitives (Paraschos et al., 2013), Conditional Movement Primitives (Seker et al., 2019; Akbulut et al., 2021b,a), and Riemannian Motion Policies (RMPs) (Cheng et al., 2021; Ratliff et al., 2018; Shaw et al., 2022; Xie et al., 2023). These approaches help account for variability across demonstrations, high-dimensional task parameters, and different task spaces but still fail to robustly handle the contact dynamics and high-dimensional observation spaces present in dexterous manipulation (Saveriano et al., 2021). We investigate combining RMPs with variable impedance control for safe manipulation skill learning in Chapter 6.

A burgeoning area of research is incorporating learned motor skills into task-and-motion planning (TAMP) solvers, e.g. Wang et al. (2021); see Garrett et al. (2021) for a recent review. Briefly, TAMP systems alternate search between abstract, symbolic spaces and low-level geometric spaces to solve long-horizon manipulation problems. TAMP methods have generally assumed composable skills, focusing on bilevel planning rather than how such skills might be constructed. Recent work by Silver et al. (2023) learns skills in a TAMP framework by segmenting demonstrations consisting of sequences of skills (ala Niekum et al. (2012)). Other related approaches include the works of Sharma et al. (2021); Sharma and Kroemer (2022) which seek to identify and select among object-centric, single-axis controllers for pushing or using tools, and the work of Liang et al. (2022) which seeks to identify skill preconditions for hybrid force-velocity controllers. Our work in Chapter 7 seeks to design a policy class that is compatible with TAMP systems by construction and efficiently learnable. We do so by exploiting structure — building upon the variable impedance control action space (Martín-Martín et al., 2019), using motion planning to control the arm in free space, and exploiting grasp detection methods (ten Pas and Platt, 2015; ten Pas et al., 2017; Mahler et al.,

2017; Corsaro et al., 2021).

2.5 Safe End-Effector Variable Impedance Control

A central effort in Chapters 6 and 7 is develop safer policy classes in order to enable learning with real robots in the real world. Safety has been studied extensively in the reinforcement learning literature (Garcıa and Fernández, 2015) as well as in the optimal control literature (Chen and Sankaranarayanan, 2022). Here we briefly overview related approaches to safe impedance control in Cartesian space.

The traditional approach in the control literature is to compute safety sets (using known dynamics models), and ensure that the policy stays within those bounds. Wabersich and Zeilinger (2018) propose a general formulation of a learning problem where these bounds can be computed. Once the safety set is known, a method such as control barrier functions (Wei and Liu, 2019; Cheng et al., 2019) can be used to ensure that the policy stays within that set (termed *forward set-invariance*). However, finding these safety sets is computationally expensive. While these methods are compelling due to their formal safety guarantees, we have yet to see them applied for safety of an impedance-control scheme in learning to solve contact-rich manipulation tasks, at least in part because of their assumption of a priori known dynamics.

Many have discussed formulations of impedance control to ensure Lyapunov stability. Khader et al. (2021) constrain the stiffness parameters of the action space to ensure that their controller is Lyapunov-stable, and thus resilient to perturbations and unexpected behavior. Others prove Lyapunov stability for more general formulations of learning problems (Chow et al., 2018; Berkenkamp et al., 2017). Lyapunov stability guarantees a convergence property of the system showing that the system state must be contained in smaller regions of the space as dynamics evolve in accordance to the Lyapunov function. However, since Lyapunov stability discusses *convergence* and not *safety-set forward-invariance*, it is difficult to guarantee the robot will remain free from unsafe behavior in the presence of unmodeled contact dynamics (e.g. violations of joint position limits, joint torque limits, or collisions with objects).

Chapter 3

Learning to Generalize Kinematic Models to Novel Objects

This work was presented at the Conference on Robot Learning in November of 2019, in Osaka, Japan.

3.1 Introduction

As robots move out of controlled laboratory environments and into homes and work environments, it is critical that they are capable of readily manipulating common objects with functional parts. This implies that our robots must be able to robustly perceive and manipulate articulated objects as soon as they are encountered. For example, a robot should be able to recognize an object like a microwave and infer whether the door is open or closed, as well as the information necessary to change the kinematic state of the door.

Existing approaches to learning to manipulate articulated objects are instance-based (Katz and Brock, 2008; Sturm et al., 2011; Pillai et al., 2015; Hausman et al., 2015). They estimate the kinematic model of an object having observed the object's motion, either demonstrated by a human or generated by the robot through slow, deliberate interaction. This enables manipulation but requires the robot to learn about each object in its environment from scratch, regardless of how similar the object is to those it has previously experienced. This chapter introduces a method that achieves generalization via object types: objects may be categorized according to their shared

kinematic structure, such that all objects in a particular class have the same set of kinematic constraints between parts. This allows a robot to be taught the prototypical kinematic structure for many types of objects commonly experienced in human environments, then exploit that knowledge to readily manipulate novel instances of those objects after having recognized them.

Grouping objects in this way helps to relate similar manipulation problems, aiding the generalization of learned policies to new tasks, a long-standing challenge in robotics and reinforcement learning (Taylor and Stone, 2009). Additionally, a robust notion of object type enables efficient high-level planning through abstraction: objects of the same type can be treated identically in a plan (Diuk et al., 2008).

This chapter introduces a framework capable of jointly estimating the kinematic model parameters, kinematic state, and geometry of a novel object from RGB-D observations, given its categorization and prior experience with objects of the same type. We use object recognition to select a kinematic model, and use deep neural networks to learn object-specific mappings from depth sensor observations to kinematic model parameters, kinematic state variables, and geometric parameters. Inspired by recent approaches to object recognition and pose estimation (Wei et al., 2016; Burchfiel and Konidaris, 2018; Wang et al., 2019a), we cast kinematic model parameter inference as a regression task given a known kinematic model type specified by the object's categorization. To train these networks, we developed a dataset of simulated articulated objects, recorded ground-truth geometry and kinematic models, then simulated object articulation and rendered depth images. We evaluated the performance of our system with withheld simulated objects, real objects observed with a Kinect sensor, and manipulation scenarios. Our results demonstrate that our networks learn to localize kinematic mechanisms, infer kinematic state, and predict object geometry sufficiently well to enable the manipulation of novel articulated objects on a real robot.

3.2 Learning to Generalize Object Kinematics

Our goal is to map a segmented depth image of an object to its kinematic graph parameters, configuration, and geometry. For example, the robot will observe a depth image of a microwave and estimate the location of the microwave door's axis of rotation (shown in Figure 3.1), the transform from that axis to the center of the door, as well as the angle between the door and the body, and a simple parameterization of the object's geometry. We begin by making the assumption that

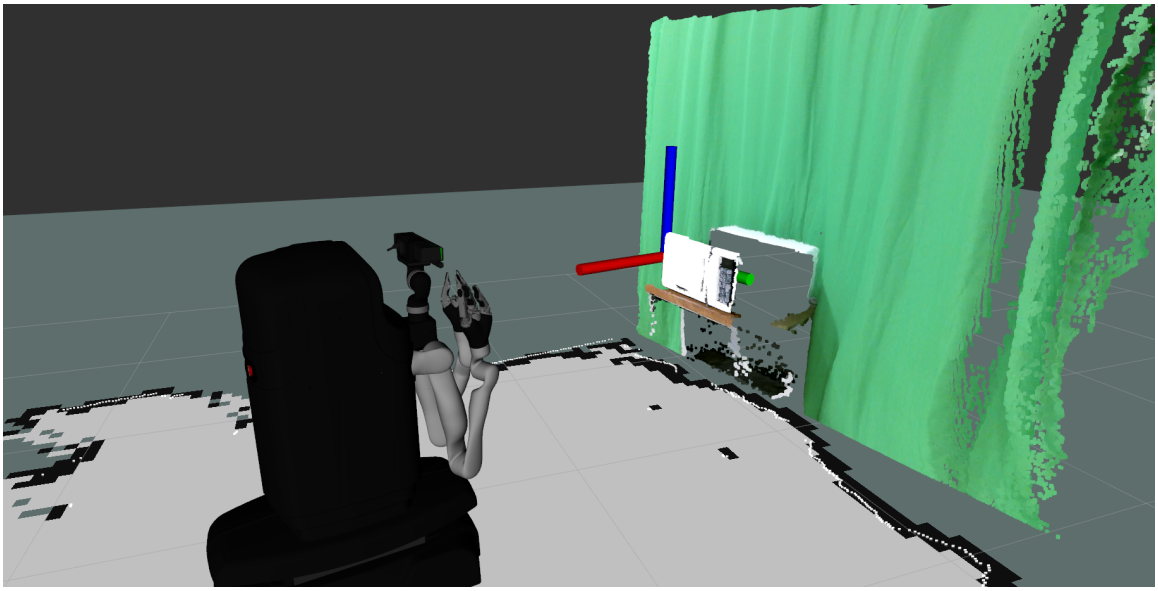


Figure 3.1: **Kinematic model prediction for a novel object.** Our model predicts the pose and state of a microwave's hinge from a point cloud captured with a Kinect depth sensor.

an object's kinematic structure is directly specified by its category. Thus, for an object to be a member of a class, it must possess the particular kinematic graph connectivity shared among all instances of that class. Additionally, we assume that the kinematic graph structure of each object class is known a priori, leaving the estimation of prototypical graphs for each class to future work. This assumption enables us to forgo fitting a kinematic model type to a sequence of observations and proceed to estimating model parameters immediately following object recognition. As such, our system is able to generalize within object categories without first observing a trajectory. We cast this as a regression problem, and develop a learning framework that is trained on a dataset of simulated articulated objects before being tested in reality.

3.2.1 Model Description

The robot takes as input an RGB-D image. We use off-the-shelf object detection algorithms to classify and segment the object from the rest of the scene. We manually specify a kinematic model for each object category. We aim to estimate $p(\phi, q_t, \theta | x_t, c)$, the probability of the object's kinematic model parameters, ϕ, its present configuration, q_t, and a simple parameterization (length, width, height, chirality) of the object's geometry, θ, conditioned upon a single depth image, x_t, and the object's class label, c. In practice, we train a unique mixture density network (Bishop, 1994) for

each object class c, which consumes a depth image and regresses to the parameters of a mixture of Gaussians:

$$p(\phi, q_t, \theta | x_t, c) = \sum_i^m \pi_i^c(x_t)\ \mathcal{N}(\phi, q_t, \theta | \mu_i^c(x_t), \sigma_i^c(x_t)), \tag{3.1}$$

where x_t is a depth image sampled from the object's articulation at time t, the object class c is known to the learner, and $\pi_i^c(x_t), \mu_i^c(x_t), \sigma_i^c(x_t)$ are neural networks which consume depth images and output mixing parameters, means, and diagonal covariances, respectively. We use a ResNet-18 (He et al., 2016) backbone with a fully connected network head for each parameter (π, σ, μ) of the mixture of Gaussians. We found $m = 20$ mixture components to be sufficient.

3.2.2 Training

Training the model requires point clouds annotated with the underlying pose, geometry, and kinematics of the articulated object. To generate this data, we created a dataset of synthetic articulated objects (see Section 3.2.3). Ground truth kinematic models allowed us to train inference networks by maximizing the probability of the labels (ϕ, q_t, θ) under the model $p(\phi, q_t, \theta | x_t, c)$:

$$\mathcal{L} = -\mathbb{E}\left[\log p(\phi, q_t, \theta | x_t, c)\right]. \tag{3.2}$$

We sampled batches of individual image-label pairs rather than training on full object articulation demonstrations to enable one-shot model inference with new objects.

3.2.3 Synthetic Dataset

Our synthetic dataset consists 6 object categories (cabinet, drawer, microwave, toaster oven, two-door cabinet, refrigerator). The kinematic graph for each of these classes, known a priori to the learner, was specified in Universal Robot Description Format (URDF) and rendered via Mujoco (Todorov et al., 2012). Example objects are shown in Figure 3.2. Two-door cabinets and refrigerators are modeled as having two degrees of freedom, while the rest of the objects are modeled with one degree of freedom.

A generator specific to each object class sampled object pose and geometry, and computed the location(s) of kinematic mechanism(s). The simulator rendered observations from each object's

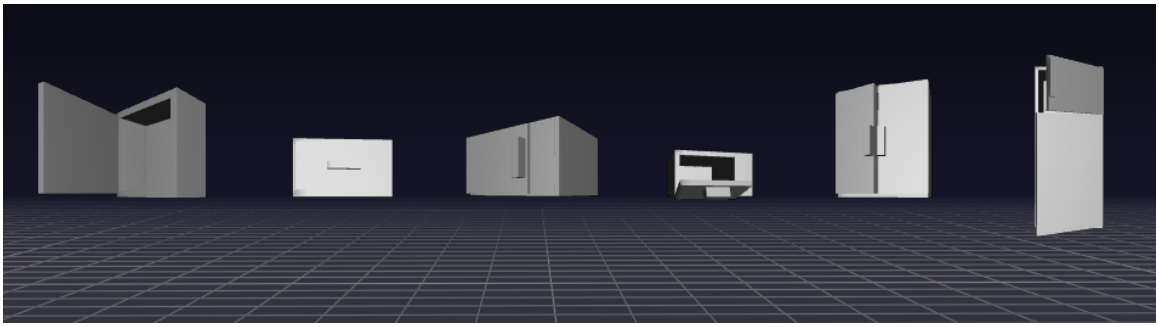


Figure 3.2: **Example synthetic objects:** cabinet, drawer, microwave, toaster oven, cabinet2, refrigerator, respectively.

articulation. We sampled object position uniformly from the view frustum of the camera up to a maximum depth dependent upon object size. Object orientations were sampled uniformly from the range $[-\frac{\pi}{4}, \frac{\pi}{4}]$ about the z-axis. A unique neural network was trained for each object class, using $160,000$ depth-image/label pairs ($10,000$ objects) for training and $16,000$ pairs ($1,000$ withheld objects) for testing per class. We make the dataset publicly available here for broader use by the community. More details are provided in Appendix A.

3.3 Experimental Evaluation

We evaluated our approach's ability to jointly estimate kinematic model parameters and kinematic configuration of articulated objects from depth images using withheld simulated objects and real objects observed with a Kinect depth sensor. Additionally, to demonstrate that our approach indeed enables the manipulation of real articulated objects, we use our model to open a microwave door and a drawer with a real mobile manipulator.

3.3.1 Predicting Kinematics From Depth Images

Parameter estimation accuracy was evaluated by measuring the Euclidean distance between the estimated and the true pose of the axis of rotation/translation, and the orientation error of the axis of rotation/translation about the z-axis. Kinematic state estimation accuracy was evaluated by measuring the error in kinematic state, reported in degrees for revolute mechanisms and centimeters for prismatic mechanisms. These estimates are produced from a single depth image, and are not accumulated over time. The error is computed using the maximum likelihood sample from the

mixture density network for each class. As a baseline, we compared against the mean of each quantity for each object class (Mean), nearest neighbor in depth-image space (NN), and a variant of ICP in which we registered each test observation to the mean model for each object class in 16 different poses and found the best match (ICP). The ICP baseline finds a 6-DoF transformation, but we report error only about the z-axis for comparison. Table 3.1 displays these results for $1,000$ withheld objects for each of the six object classes studied. Our framework learns to infer the position, orientation, and state of simulated kinematic mechanisms to within a few centimeters/degrees.

	Axis Pos. Error (cm)	**Axis Rot. Error** (deg)	**Config. Error**
Cabinet - Mean	53.83 ± 18.36	10.81 ± 14.29	40.62 ± 26.06 °
Cabinet - NN	18.08 ± 11.31	10.39 ± 12.02	19.66 ± 17.30 °
Cabinet - ICP	35.68 ± 17.36	34.75 ± 26.76	40.58 ± 26.13 °
Cabinet - Ours	**1.84 ± 4.43**	**0.36 ± 1.03**	**1.61 ± 1.70** °
Drawer - Mean	55.59 ± 17.98	11.22 ± 14.42	9.75 ± 5.43 cm
Drawer - NN	16.51 ± 9.99	8.72 ± 11.84	9.89 ± 8.03 cm
Drawer - ICP	22.8 ± 12.36	19.33 ± 25.02	16.47 ± 11.16 cm
Drawer - Ours	**5.23 ± 3.91**	**1.14 ± 3.38**	**4.64 ± 4.05** cm
Microwave - Mean	50.78 ± 16.73	11.33 ± 14.28	22.74 ± 12.88 °
Microwave - NN	18.08 ± 11.31	10.39 ± 12.02	19.66 ± 17.30 °
Microwave - ICP	21.29 ± 10.76	25.3 ± 20.0	40.64 ± 26.14 °
Microwave - Ours	**1.28 ± 1.31**	**0.34 ± 1.58**	**1.73 ± 2.11** °
Toaster - Mean	61.65 ± 18.58	11.48 ± 14.08	22.75 ± 12.89 °
Toaster - NN	14.59 ± 8.37	10.58 ± 12.40	18.79 ± 20.70 °
Toaster - ICP	16.22 ± 5.9	21.26 ± 20.22	40.73 ± 26.15 °
Toaster - Ours	**2.33 ± 1.46**	**0.80 ± 1.60**	**3.22 ± 4.06** °
Cabinet2 - Mean	62.43 ± 20.04	11.16 ± 14.41	42.91 ± 21.52 °
Cabinet2 - NN	17.16 ± 10.20	10.81 ± 11.95	39.95 ± 35.99 °
Cabinet2 - ICP	19.56 ± 9.9	26.22 ± 22.92	66.08 ± 33.77 °
Cabinet2 - Ours	**3.81 ± 2.71**	**2.14 ± 4.59**	**11.97 ± 9.56** °
Refrigerator - Mean	108.96 ± 56.02	11.17 ± 14.23	35.23 ± 23.09 °
Refrigerator - NN	24.69 ± 16.00	8.92 ± 11.39	24.07 ± 32.20 °
Refrigerator - ICP	58.25 ± 35.16	27.15 ± 19.89	42.3 ± 29.6 °
Refrigerator - Ours	**4.67 ± 9.17**	**0.75 ± 3.93**	**6.59 ± 11.17** °

Table 3.1: **Simulation Results.** Mechanism localization and configuration results are presented as mean and standard deviation of the error, evaluated over 1000 withheld objects in simulation, measured using the model's maximum likelihood sample.

Next, we evaluated the system's accuracy on objects in the real world. We evaluated our trained models with one real object per class, show in Figure 3.3. Recognition and segmentation were

performed using Mask R-CNN (He et al., 2017; Girshick et al., 2018) pretrained on the MS COCO dataset (Lin et al., 2014) for the microwave and refrigerator object classes, and manually for the others. Kinematic mechanism poses were acquired using AR tags, and configurations were held constant. Between 47 and 170 samples were acquired for each class. For real objects, we found the model's accuracy improved when only considering points in the point cloud as candidates for the mechanism position. To compute the estimates for Table 3.2, points in the point cloud were scored by computing their likelihood under the model, and the maximally likely point was selected as the system's estimate. Our framework learns to accurately generalize kinematic models to real, novel objects after being trained exclusively in simulation. Performance on the simulated refrigerator

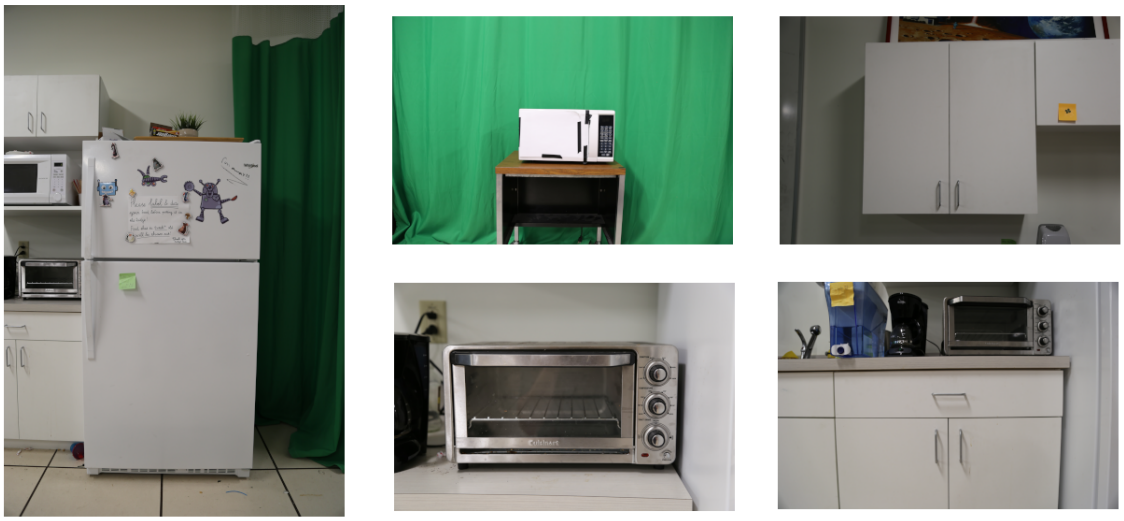


Figure 3.3: **Real objects studied**. Clockwise: refrigerator, microwave, 2-DoF cabinet, drawer, toaster oven. To obtain 1-DoF cabinet data, we segmented the 2-DoF cabinet into its respective halves.

object class is relatively low due to more pose variation in the dataset in order to accommodate for the refrigerator's size, and more samples with only partial views of the object. Interestingly, networks trained on this object class generalized relatively well to the real object. Additionally, the drawer object class is challenging in both simulation and reality due to occlusion of the mechanism's pose during actuation; this affects the accuracy of the model on the real data of the open microwave, as well.

The error rate of the model on the 1-DoF cabinet class is higher due to ambiguity in which side of the object the mechanism resides on. In simulation, there are sufficient artifacts to predict this

	Axis Pos. Error (cm)	**Axis Rot. Error** (deg)	**Config. Error**
Cabinet	14.96 ± 13.37	7.82 ± 8.68	15.27 ± 10.75 °
Drawer	9.12 ± 5.97	1.32 ± 1.83	11.45 ± 3.77 cm
Microwave - Closed	3.96 ± 3.48	2.65 ± 3.60	12.96 ± 14.34 °
Microwave - Open	11.87 ± 6.97	5.45 ± 4.81	19.39 ± 9.72 °
Toaster	6.30 ± 3.56	2.79 ± 3.31	10.32 ± 10.47 °
Cabinet2	6.55 ± 3.10	3.02 ± 3.39	16.49 ± 8.87 °
Refrigerator	3.56 ± 2.32	2.59 ± 3.62	1.30 ± 2.85 °

Table 3.2: **Real Object Results.** Mechanism localization and configuration results are presented for real sensor data with real objects. Objects were imaged in a closed configuration unless denoted otherwise. Errors were measured using the maximally likely point in the point cloud under the model. Ground truth measurements were obtained using AR tags.

accurately, but in reality, the model makes some erroneous predictions. Resulting samples from the model are visualized in Figure 3.4, wherein the agent has assigned probability to both sides of a 1-DoF cabinet after several estimates of the cabinet's hinge pose.

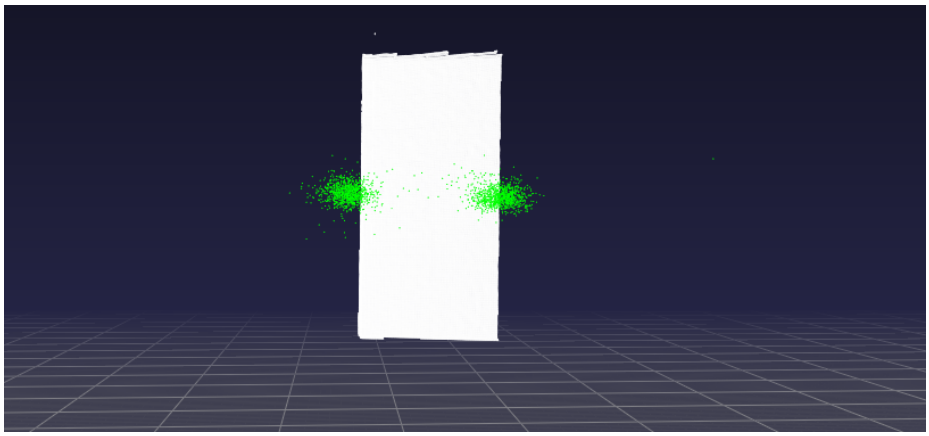


Figure 3.4: **Model estimates under uncertainty.** Samples drawn from the model of a cabinet's hinge pose, accumulated over time and overlaid on a point cloud. With the door closed, it is challenging to discern whether the cabinet opens left or right. Accordingly, our model has assigned probability to both sides of the object.

3.3.2 Using Estimated Kinematic Models for Manipulation

Finally, we evaluated our system's accuracy by demonstrating successful manipulation of novel articulated objects in the real world. The robot, a Kinova MOVO mobile manipulator, was driven around the area in front of each object, and recorded observations of the object and its own pose

in a pre-computed map of the room. Using these observations, the agent estimated the position and orientation of the object's axis of rotation/translation, articulated pose, and geometry. We provided the agent with a fixed grasp on the object's handle, and computed a series of waypoints as a function of the estimated object kinematics, end-effector pose, and desired object configuration. Using our system's estimates, we were able to generate kinematic motion plans that achieved the desired manipulations, as shown in Figures 3.5 and 3.6.

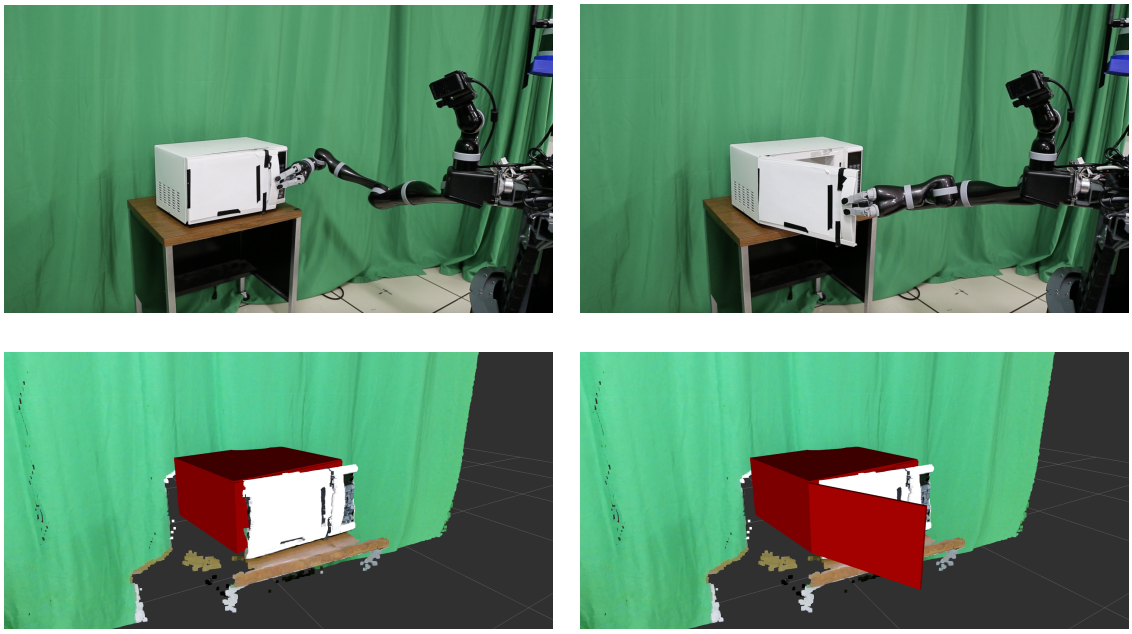


Figure 3.5: **Novel object manipulation.** *Top:* A MOVO robot manipulating a novel microwave using our system's estimates of the object's kinematics and geometry. *Bottom:* Rendered estimates of the object's pose, geometry, and kinematic state before and after the manipulation.

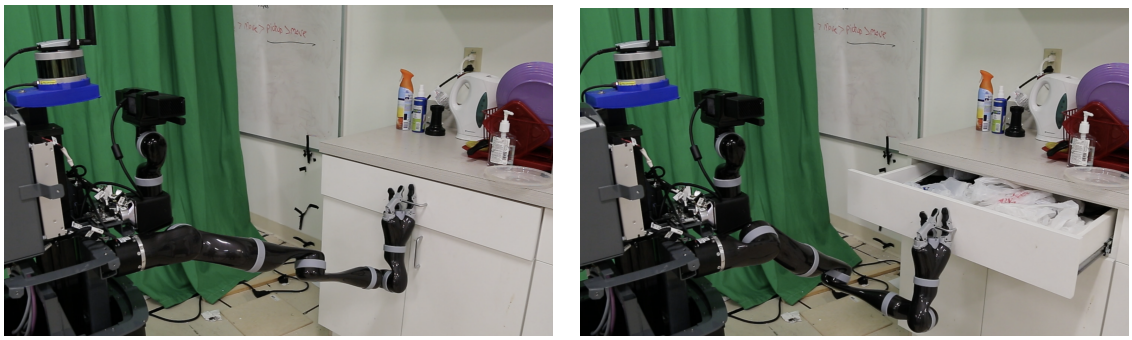


Figure 3.6: **Novel object manipulation.** A MOVO robot manipulating a novel drawer using a kinematic model estimated from a single depth image.

3.4 Discussion and Conclusions

We presented a framework for the perception and manipulation of articulated objects, without the need for first observing the object's articulation at runtime, by leveraging object recognition and a novel dataset of synthetic articulated objects.

Our work assumes that models are directly specified by object class, and that each object class has exactly one appropriate kinematic model. This assumption is relaxed in the next chapter to accommodate the variation present in many real object classes. Conversely, while previous approaches to learning object classes have relied exclusively on appearance, our work suggests that functionality should also play a key role.

Our work introduces and relies upon a simulated dataset which has several shortcomings. The simulated objects present artifacts and are not photo-realistic, forcing us to use only depth images and to regularize the models heavily during training. Additionally, the length, width, and height of objects was sampled independently, which does not accurately reflect the true joint distribution of object shapes in many categories. Furthermore, our dataset only includes objects in isolation; objects are often occluded in real scenes, and this presents a challenge for our models. This avenue of research could benefit immensely from a large dataset of real articulated objects; the Sapien dataset (Xiang et al., 2020) has since been developed and seen wide adoption. Nonetheless, our real-world experiments provide evidence that our system does generalize well to real objects.

Robots will have to be able to generalize their manipulation skills in order to be useful in real environments. Our system accurately produces probabilistic estimates of kinematic model parameters, articulated pose, and object geometry from individual sensor observations; our experiments show that our approach enables the perception of household articulated objects immediately following object detection—thus enabling a robot to manipulate novel instances of familiar object classes.

We now turn our attention to the aforementioned limitation of this system – assumption of a category template – and describe a system which is capable of recovering kinematic *hierarchies* from observation.

Chapter 4

Learning to Infer Kinematic Hierarchies for Novel Object Instances

This work was done in collaboration with Hameed Abdul-Rashid and Miles Freeman, under the supervision of Daniel Ritchie and George Konidaris. The work appeared in the 2022 IEEE International Conference on Robotics and Automation in Philadelphia, PA.

4.1 Introduction

In the previous chapter, we introduced the challenge of inferring the kinematic models of novel articulated objects from sensor data. To a human, interactions with such objects are effortless, but the underlying process is complex: one must perceive the *kinematic hierarchy* of the object (i.e. which parts can move, how they move, and how those motions are coupled). Then, one must plan a sequence of actions to transform the object from its current kinematic pose into another, and finally, manipulate the object to execute that plan. The perception part of this process is particularly challenging, as objects with similar function can vary considerably in both their structure and their geometry. Wardrobes, for example, can have differing numbers of parts in different arrangements, and many different doorknob shapes. To succeed at manipulating such objects in the wild, an

agent must infer the *kinematic hierarchies* of never-before-seen object instances with these kinds of variations. In the previous chapter, we assumed that we could determine this hierarchy based on object classification, and focused on inferring fine-grained properties like the motion models between parts. But, in practice, this assumption does not always hold: real object classes like *storage furniture* or *chairs* posses wide variety in their structure.

In this chapter, we describe a perception system that infers kinematic hierarchies, generalizing to novel object geometries and structures. Our system infers the moving parts of an object and the kinematic couplings between them. To infer parts, it uses a point cloud instance segmentation neural network. To infer kinematic hierarchies, it uses a graph neural network which learns to infer the existence, direction, and type of edges (i.e. joints) that relate the inferred parts. This system is trained on simulated scans of a large set of synthetic 3D models, enabling it to learn to generalize to new, never-before-seen object instances which may differ geometrically and structurally from the training data. We evaluate our system's perception abilities on synthetic point clouds, and find that our system reliably detects moving parts and reconstructs kinematic hierarchies on synthetic data. We also demonstrate a proof-of-concept application of using the output of our perception system to drive manipulation executed by a real-world robot platform.

In summary, the contributions in this chapter are twofold: the first method for inferring kinematic hierarchies from never-before-seen instances of 3D objects, without reliance on a schema or template; and a novel graph neural network approach for predicting hierarchies from part segments.

4.2 Approach

In this section, we overview our method for kinematic mobility perception. Fig. 4.2 shows a schematic of our pipeline.

4.2.1 Part Segmentation

The input to our system is an unlabeled point cloud depicting an articulated object; a robot can obtain such input by unprojecting and consolidating frames from an onboard depth sensor. Our system first segments this point cloud into parts, some of which are movable (Fig. 4.2a). This stage relies upon an existing neural network architecture for point cloud instance segmentation (i.e. segmenting a point cloud without consistent part labels) (Mo et al., 2019).

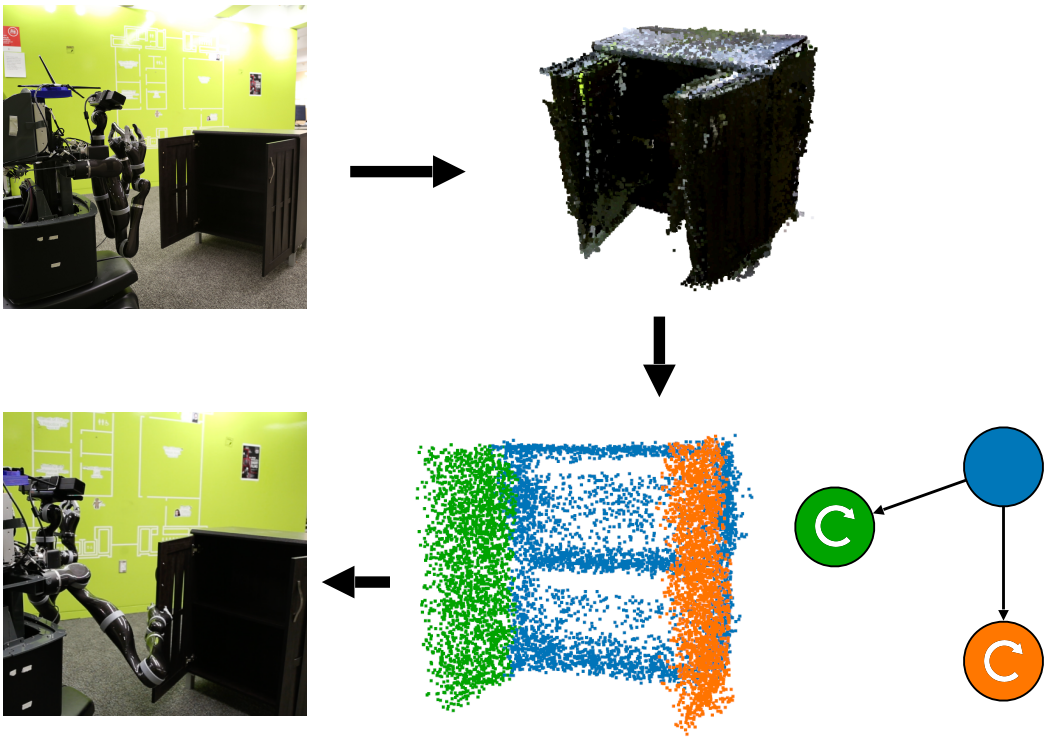


Figure 4.1: We propose a system for robot perception of the kinematic structure of never-before-seen object instances. The robot scans the object from multiple viewpoints, building a 3D point cloud representation of it. Our system then segments this point cloud into parts and infers the kinematic hierarchy that couples them, allowing for the robot to plan and execute articulated motions of the object.

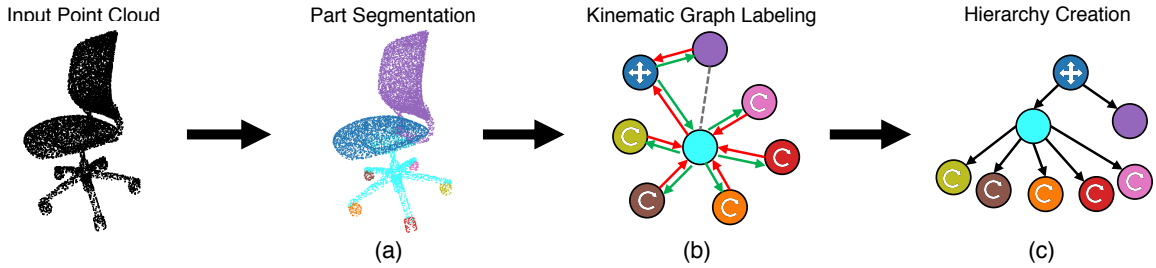


Figure 4.2: Overview of our system. Given an input 3D point cloud representation of an object, our system (a) segments the point cloud into parts using an instance segmentation network, (b) constructs a part graph and uses a graph neural network to label its nodes and edges with kinematic properties, and (c) constructs a kinematic hierarchy from this graph which can be used for robot manipulation.

4.2.2 Kinematic Graph Labeling

Given a part-segmented point cloud, our system next infers the articulation properties of those parts: the type of articulated motion they support (if any) and how the motions of different parts are

coupled. To do this, it converts the segmented point cloud into a graph and phrases the problem as one of *graph labeling*: nodes in the graph represent parts, and edges between them represent potential kinematic connections (e.g. joints) between parts. A graph neural network predicts whether each node is static, rotating, or translating, as well as whether each edge should exist (i.e. whether two parts should be kinematically connected), the probability of each node being the root of the tree, and the direction of kinematic dependency (Fig. 4.2b).

4.2.3 Hierarchy Creation

Given the labeled graph, we construct a kinematic hierarchy by converting the bidirectional graph into a n-ary tree. Xu *et al.* address this problem by constructing a pairwise matrix where each entry represents the negative log probability of the corresponding parts possessing an edge connection (Xu et al., 2020b). The pairwise matrix is traversed from a predicted root node and a n-ary tree is extracted via a Minimum Spanning Tree (MST) algorithm (Prim, 1957). We adopt this method to convert our predicted bidirectional graphs to an n-ary tree.(Fig. 4.2c).

4.3 Part Segmentation

To segment point clouds into parts, we use a neural network based on that of PartNet (Mo et al., 2019). This architecture uses a PointNet++ network (Qi et al., 2017b) to compute per-point embeddings, each of which is then fed to a semantic segmentation branch (assigns semantic labels to each point) and an instance segmentation branch (assign each point to one of N possible instance masks; we use $N = 24$). As our system does not assume parts are semantically labeled, we could remove the semantic segmentation branch. However, PartNet showed that semantic segmentation helps improve the quality of instance segmentation. Thus, we treat the semantic segmentation branch as a "motion type prediction" branch (static, rotating, translating, rotating and translating)—labels which we do assume of our data.

We train the segmentation network on 3D objects from the PartNet-Mobility dataset (Xiang et al., 2020). For augmentation, we use multiple poses for each moving part of each object, as described in more detail in Section 4.5. To simulate a robot's onboard depth sensor, we render point clouds from these objects using a software simulator of the Kinect sensor (Landau, 2017). We train the network using the Adam optimizer (Kingma and Ba, 2015) with learning rate 10^{-4}. Training

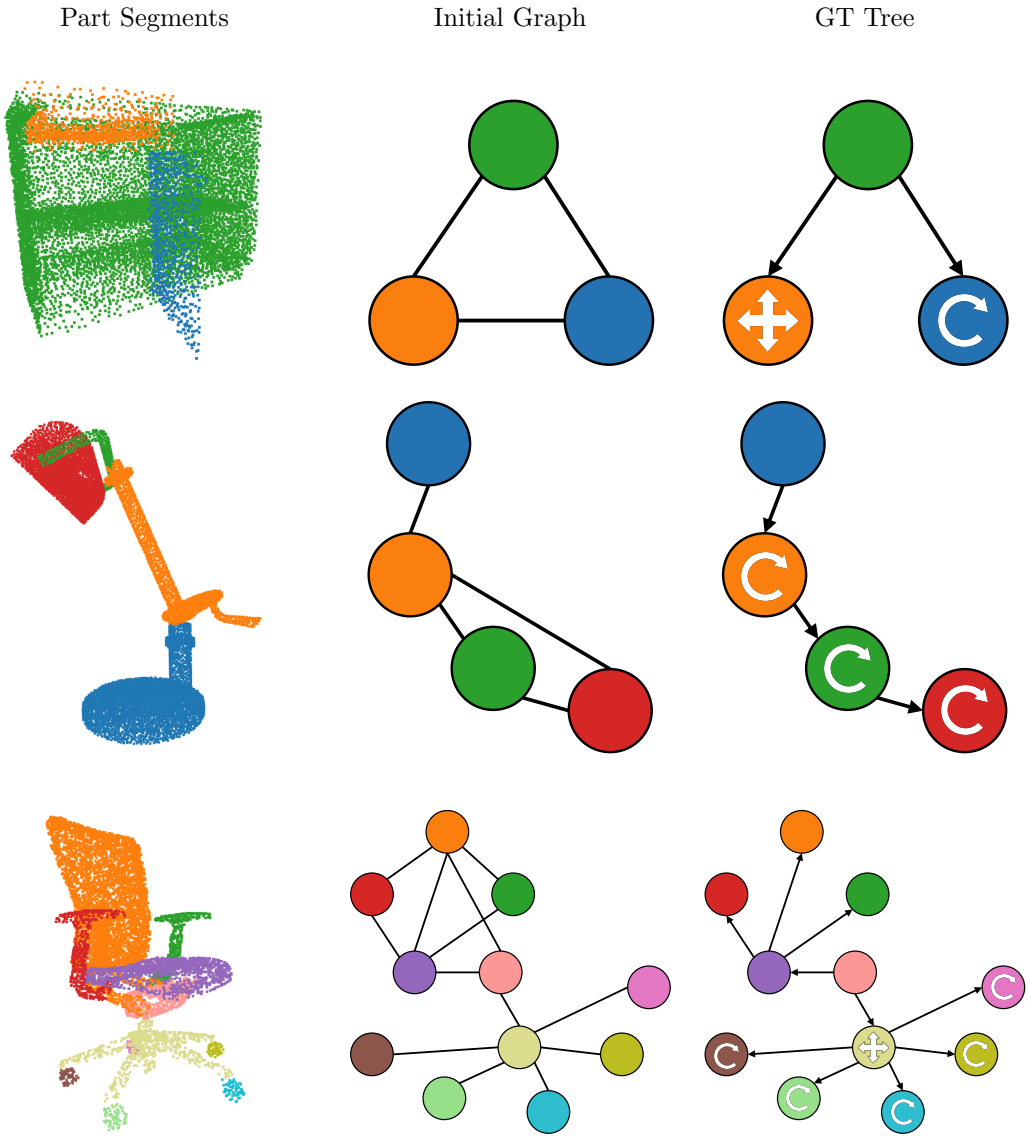


Figure 4.3: Segmented point clouds, their initial overcomplete part graphs, and their ground truth part graphs (as determined by a human labeler).

was run for at most 8 hours per category.

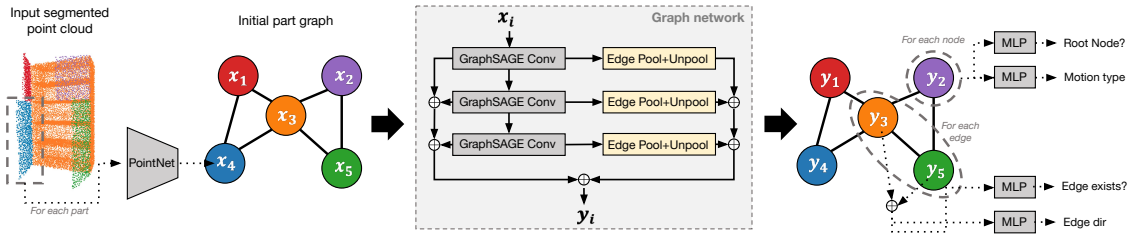


Figure 4.4: Neural network architecture for kinematic graph labeling. A PointNet (Qi et al., 2017a) converts each part point cloud into an initial node feature $\mathbf{x}$. These features go through graph convolutional + pooling layers to produce features $\mathbf{y}$ which are passed to MLPs for predicting node and edge attributes.

4.4 Kinematic Graph Annotation

Here we describe how our system takes a segmented point cloud, converts it to a graph, and labels this graph with the information required to construct a kinematic hierarchy.

4.4.1 Point Cloud to Graph Conversion

The kinematic hierarchy for a segmented object is a directed tree: tree nodes correspond to parts, nodes are labeled with a motion type (rotation and/or translation) and edge directions indicate kinematic couplings (i.e. edge $A \rightarrow B$ means part B moves with part A). To infer this tree from a segmented point cloud, our system first constructs an overcomplete, undirected, unlabeled graph over the parts, and then prunes edges, determines edge directions, and labels nodes using a graph neural network. Our system constructs the initial graph by adding an undirected edge between any two parts whose Euclidean distance is below a small threshold. Fig. 4.3 shows some example segmented point clouds and their initial graphs.

4.4.2 Graph Labeling Network

Given a graph constructed by the above procedure, we use a graph neural network to label its nodes and edges with the information required to construct a kinematic hierarchy.

Input features: Every node corresponds to one part, i.e. one subset of points from the object point cloud. We encode these point subsets into per-node vectors $\mathbf{x}$ using a PointNet (Qi et al., 2017a).

Graph network: To make per-node and per-edge predictions, our system must understand the

context of each node or edge within the object. To satisfy this goal, we use a graph convolutional network to convert node features $\mathbf{x}$ into context-aware features $\mathbf{y}$ (Fig. 4.4). Our network passes all nodes through three GraphSAGE convolution layers (Hamilton et al., 2017), which compute new node features based on learned aggregations of neighbor node features. Each convolution layer is followed by a pooling and unpooling layer based on learned edge collapses (Diehl, 2019). In early experiments, we found this pooling + unpooling scheme to perform better than architectures that used only convolution. Finally, the outputs of all convolution layers and all pooling layers are concatenated to form a multi-scale feature vector $\mathbf{y}$ for each node.

Node and edge attribute prediction: The per-node features $\mathbf{y}$ are fed to multi-layer perceptrons (MLPs) for predicting node attributes. For edge attributes, the $\mathbf{y}$'s for both edge endpoints are concatenated and fed to the MLP.

4.4.3 Training

We train the graph labeling network on synthetic 3D models from PartNet-Mobility (Xiang et al., 2020). We first pre-train the node feature PointNet by training it as the encoder in an autoencoder framework; this network trains for 500 epochs using Nesterov SGD with a learning ratte of 10^{-3}. The graph labeling network trains for 30 epochs using Adam with a learning rate of 10^{-3}. Training both networks takes 1.5 - 3 hours on a machine with a NVIDIA RTX 2080 Ti, 6-core Intel i7 CPU, and 32GB of RAM

4.5 Results

Here we evaluate our method's ability to infer kinematic hierarchies via controlled experiments and a demonstration on a real-world robot. For experiments on synthetic PartNet-Mobility objects, we use the same train/test splits introduced by Mo et al. (2019), restricted to models with valid kinematic trees. We augment the training and test datasets by uniformly sampling articulated poses within each part's ground truth range of motion. We train on 249 storage furniture models, 22 lamp models, and 53 chair models; we test on 69, 8, and 12 models, respectively. We use 18 pose augmentations for training and testing.

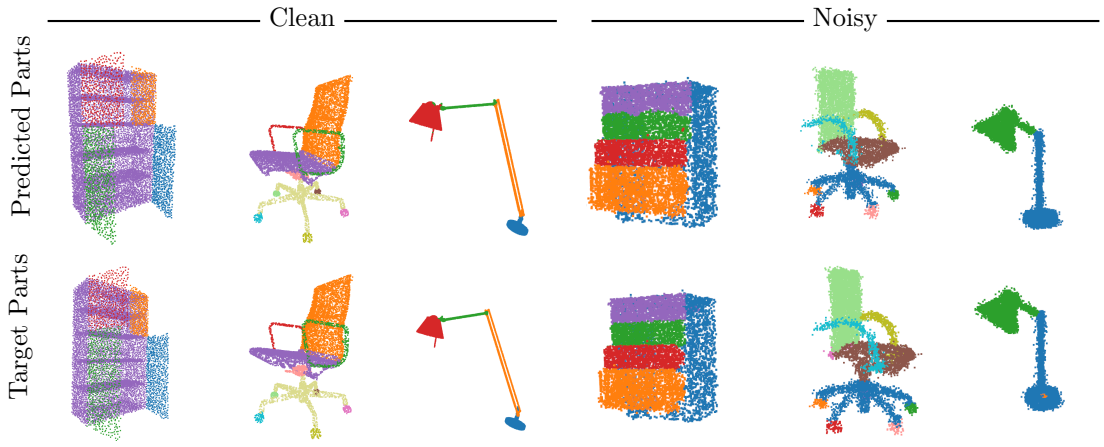


Figure 4.5: Examples of part segmentation for both Clean and Noisy point clouds.

4.5.1 Part Segmentation

Table 4.1 summarizes the performance of our part segmentation method on synthetic 3D objects. We use an established instance segmentation metric (Mo et al., 2019): we compute the intersection over union (IoU) of each predicted part segment with its closest ground truth part segment, classify it as a "correct" prediction if the IoU is over 0.5, and then compute the mean average precision of these classifications across all parts in all test set objects. We compute this metric under two conditions: (1) *Clean* point clouds sampled directly from the surface of PartNet-Mobility 3D meshes, (2) *Noisy* point clouds generated by simulated Kinect scans of PartNet-Mobility models as described in Section 4.3. Fig. 4.5 shows some qualitative examples.

The network reliably segments parts in clean point clouds, and this performance degrades slightly in the noisy depth scans. The lamp category is especially challenging due to the small number of models present in the training data.

Table 4.1: Part segmentation on PartNet-Mobility (mAP @ 0.5 IoU)

Category	**Clean**	**Noisy**
Storage Furniture	0.922	0.907
Lamp	0.695	0.593
Chair	0.824	0.811

4.5.2 Kinematic Structure Prediction

We next evaluate our graph neural network module's ability to infer the structure of an object's kinematic hierarchy. To assess this performance, we use four metrics:

- $\mathbf{E}_{\text{type}}$: % of nodes whose motion type is mis-predicted.
- $\mathbf{E}_{\text{exist}}$: % of edges whose existence is mis-predicted.
- $\mathbf{E}_{\text{dir}}$: % of edges whose direction is mis-predicted.
- $\mathbf{E}_{\text{root}}$: % of models whose root node is mis-predicted.
- **Tree F1**: the F1 score of the predicted kinematic hierarchy with respect to the ground truth one. Precision is computed via top-down traversal of the predicted tree, counting the fraction of nodes and edges which match their counterparts in the ground truth tree. Recall computation uses the same procedure, with the roles of the predicted and ground-truth trees reversed.

Table 4.2 summarizes the performance of our network on Clean and Noisy synthetic data. As expected, the GNN evaluated on clean data with ground truth instance segmentation preforms well on all classes. Even in the case where $\mathbf{E}_{\text{dir}}$ is relatively high for the lamp and chair class the negative log probability pairwise matrix complemented with MST is able to construct trees with a negligible amount of error. On noisy data, performance again degrades slightly. Fig. 4.6 shows some qualitative examples.

Table 4.2: Kinematic structure error for different data types

Category	**Data Type**	$\mathbf{E}_{\text{type}} \Downarrow$	$\mathbf{E}_{\text{exist}} \Downarrow$	$\mathbf{E}_{\text{dir}} \Downarrow$	$\mathbf{E}_{\text{root}} \Downarrow$	**Tree F1** $\Uparrow$
Storage Furniture	Clean	1.16	1.2	2.22	0	99.73
	Noisy	0.39	2.03	29.04	0	99.02
Lamp	Clean	0.47	4.63	31.11	0.23	98.24
	Noisy	0	0.98	17.24	0.92	97.82
Chair	Clean	0.08	0.1	39.64	0	99.77
	Noisy	0.24	0.99	41.73	0	98.67

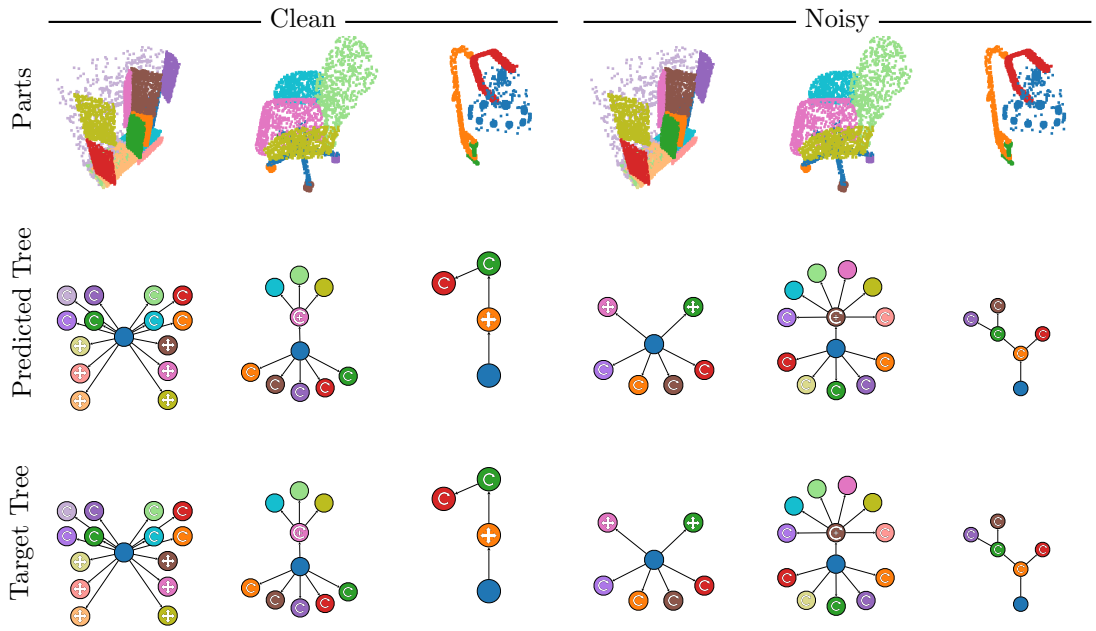


Figure 4.6: Examples of predicted kinematic hierarchies for both Clean and Noisy point clouds. All examples shown use ground truth part segmentations.

4.5.3 Robot Demo

Finally, we demonstrate that the model enables a Kinova Movo robot to manipulate novel object instances. A storage furniture object was densely scanned using the robot's Kinect sensor by driving the robot around the object. Pointclouds were merged using RTAB-Map (Labbé and Michaud, 2019), which performed SLAM using the robot's odometry, base laser scans, and visual data. The object was manually segmented from the scene and assigned a frame of reference. Our model successfully performed part segmentation and kinematic hierarchy inference; predictions are visualized in Figure 4.1. Motion model predictions, i.e. the axes of rotation, were generated using the motion type predictions from the estimated kinematic tree and a heuristic: the axes of rotation were determined from the intersection of part bounding boxes and oriented in the world vertical direction.

Using the predicted tree and motion models, the agent is able to plan and execute interactions with the object. We fixed a grasp on the relevant object part, and ran a simple controller which moves the end-effector along the predicted direction of motion toward a goal configuration. Fig. 4.7 shows frames from a video of the robot scanning the cabinet and closing one of its doors.

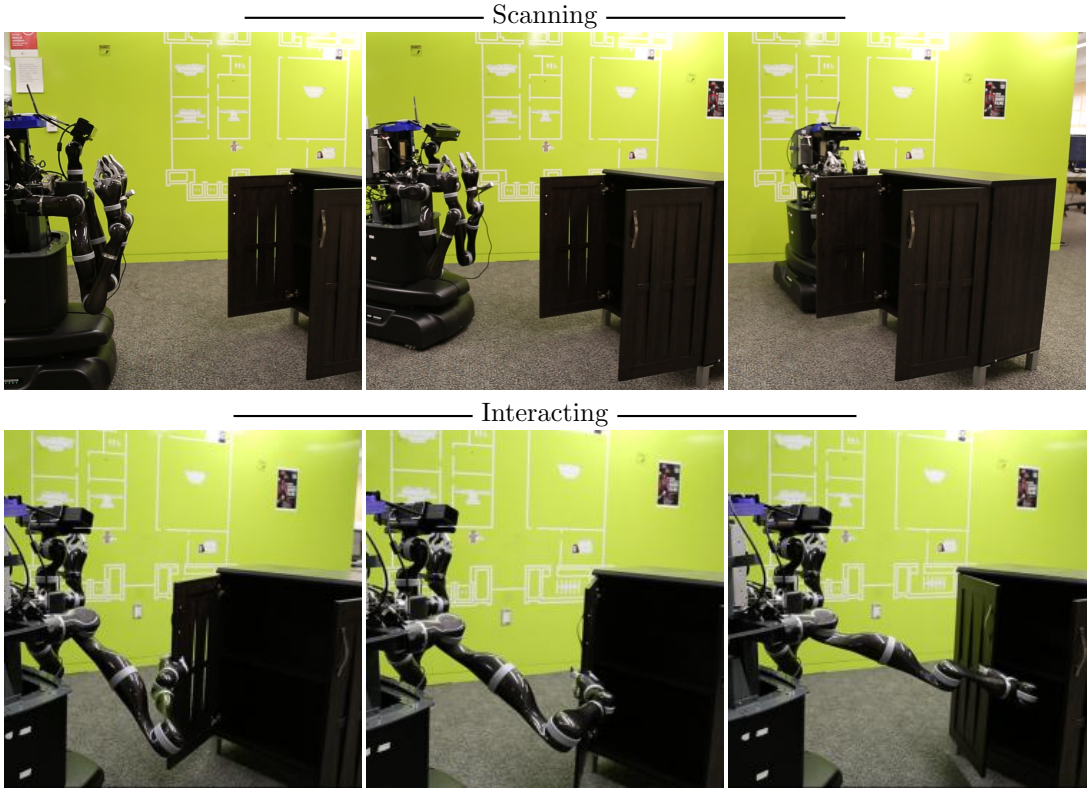


Figure 4.7: Frames from a video demonstrating a robot scanning and then interacting with an articulated object.

4.6 Conclusion

We presented a perception system that infers kinematic hierarchies for never-before-seen object instances. Our system infers the moving parts of an object and the kinematic couplings between them. To infer parts, it uses a point cloud segmentation neural network. To infer kinematic couplings, it uses a graph neural network to predict the existence, direction, and type of edges (i.e. joints) that relate the inferred parts. We train these networks using simulated scans of synthetic 3D models. In experiments, our system inferred accurate kinematic hierarchies for simulated scans of 3D objects and scans of real-world objects gathered by a mobile robot.

Training our neural networks requires collections of part-segmented, kinematically-annotated 3D models. Such data is not widely available, as it requires nontrivial human annotation effort. Recent computer vision research has demonstrated the possibility of segmenting 3D models with limited or even no human supervision (Chen et al., 2019; Zhu et al., 2020a). The development of

similar techniques for identifying articulated motion within 3D shapes would allow our methods to be applied to any object type for which unsegmented 3D models are available (Xu et al., 2020a).

Our approach treats segmentation and graph labeling as separate sub-problems, but one can argue that they are coupled: parts determines what motions are possible, and whether a motion is possible determines whether a proposed part is a good one. Accordingly, the Shape2Motion system found benefits in jointly segmenting and predict motion parameters for parts (Wang et al., 2019b). It is possible that jointly segmenting and predicting kinematic hierarchy could confer similar benefits.

We focused on inferring the structure of an object's kinematic hierarchy. Our system could be combined with one that focuses on kinematic motion parameter prediction (Wang et al., 2019b; Yan et al., 2019; Yi et al., 2018) to produce a complete mobility perception system.

In the preceding chapters, we have developed systems capable of inferring kinematic hierarchies and their associated models. We used these models and simple motion planning algorithms to compute approximate policies for interacting with the objects. However, these policies are strictly kinematic: they fail to reason about object dynamics or high-dimensional observations like vision or touch data. To develop more performant manipulation controllers for these sorts of tasks, we now turn our attention to the problem of learning motor control, leveraging these kinematic models to perform intelligent exploration.

Chapter 5

Bootstrapping Motor Skill Learning with Motion Planning

The following chapter appeared as a conference paper in the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, co-authored with Eric Rosen.

5.1 Introduction

Robots require motor policies for interacting with objects in their environment. Reinforcement learning (RL) provides a framework for acquiring motor policies without explicitly modeling the unknown world, but model-free RL methods like policy search (Deisenroth et al., 2013) have high sample-complexity, and often fail to learn a reasonable policy from random initialization. Supervised approaches for policy learning like Learning From Demonstration (LfD) (Argall et al., 2009) can encode human prior knowledge by imitating expert examples, but do not support optimization in new environments. Combining RL with LfD is a powerful method for reducing the sample complexity of policy search, and is often used in practice (Levine and Koltun, 2013; Rajeswaran et al., 2017; Cheng et al., 2018; Schaal, 1999). However, this approach typically requires a human demonstrator for initialization, which fundamentally limits the autonomy, and therefore utility, of a robot that may need to acquire a wide range of motor skills over its operational lifetime. More recently, model-based control techniques (including Model Predictive Control (Pan et al., 2017) and LQR (Levine

and Koltun, 2013)) have been proposed as exploration methods for policy search; these methods still require human demonstrations or complete dynamic models of both the robot and every object in the scene.

We propose the use of kinematic motion planning to initialize motor skill policies. While previous work has leveraged sample-based motion planners for learning motor skills (Tosun et al., 2019; Jurgenson and Tamar, 2019; Jiang et al., 2019), they only focus on either free-space motions or do not learn a closed-loop controller. To our knowledge, this is the first use of motion planning to provide initial demonstrations for learning closed-loop motor skill policies by leveraging estimated object kinematics.

We show that given a (potentially approximate, and readily estimated) kinematic description of the environment and the robot, off-the-shelf motion planning algorithms can generate feasible (potentially successful but inefficient) initial trajectories (Figure 5.5a) to bootstrap an object-manipulation policy that can subsequently be optimized using policy search (Figure 5.5b). This framework enables the robot to automatically produce its own demonstrations for effectively learning and refining object manipulation policies. Our work enables the robot to exploit kinematic planning to realize the benefits of an initial demonstration fully autonomously.

To evaluate our method, we used two different motor policy classes (Dynamic Movement Primitives (DMPs) (Ijspeert et al., 2002) and deep neural networks (Levine et al., 2016)). We compared bootstrapping with motion planning against learning from scratch in three simulated experiments, and against human demonstrations in real hardware experiments. We show that motion planning using a kinematic model produces a reasonable, though suboptimal, initial policy compared to a supervised human demonstration, which learning adapts to generate efficient, dynamic policies that exploit the dynamics of the object being manipulated. Our method is competitive with human-demonstrated initialization. It serves as a suitable starting point for learning, and significantly outperforms starting with a random policy. Taken together, these results show that our method is competitive with human demonstrations as a suitable starting point for learning, enabling robotics to efficiently and autonomously learn motor policies for dynamic tasks without human demonstration.

5.2 Bootstrapping Skills with Motion Planning

Our methodology is inspired by how humans generate reasonable first attempts for accomplishing new motor tasks. When a human wants to learn a motor skill, they do not start by flailing their arms around in a random fashion, nor do they require another person to guide their arms through a demonstration. Instead, they make a rough estimate of how they want an object to move and then try to manipulate it to that goal. For example, before being able to drive stick shift, a human must first learn how to manipulate a gear shifter for their car. Just by looking at the gear shifter, humans can decide (1) what they should grab (the shaft), (2) where they want the shaft to go (positioned in a gear location), and (3) how the shaft should roughly move throughout the action (at the intermediate gear positions). Similarly, a robot that has a good kinematic model of itself, and a reasonable kinematic model of the object it wishes to manipulate, should be able to form a motion plan to achieve the effect it wishes to achieve.

That plan may be inadequate in several ways: its kinematic model may be inaccurate, so the plan does not work; object dynamics (like the weight of a door, or the friction of a joint) may matter, and these are not represented in a kinematic model; or a feasible and collision-free kinematic trajectory may not actually have the desired effect when executed on a robot interacting with a real (and possibly novel) object. These are all the reasons why a novice driver can immediately shift gears, but not very well. But such a solution is a *good start*; we therefore propose to use it to bootstrap motor skill learning.

Our approach, outlined in Figure 5.1, leverages the (partial) knowledge the robot has about its own body and the object it is manipulating to bootstrap motor skills. Our method first assumes access to the configuration space of the robot, denoted as $\mathcal{C}_R$, as well as its inverse kinematics function f_R^{-1}. This assumption is aligned with the fact that the robot often has an accurate description of its own links and joints and how they are configured during deployment. However, the world is comprised of objects with degrees of freedom that can only be inferred from sensor data. Therefore, our approach only assumes access to estimated kinematics of the object to be manipulated, in the form of configuration space $\mathcal{C}_O$ and forward kinematics f_O. Recent work has shown that estimating these quantities for novel objects from sensor data in real environments is feasible (Abbatematteo et al., 2019; Li et al., 2020), though state-of-the-art estimates still include noise.

Finally, our approach assumes that the task goal can be defined in terms of kinematic states

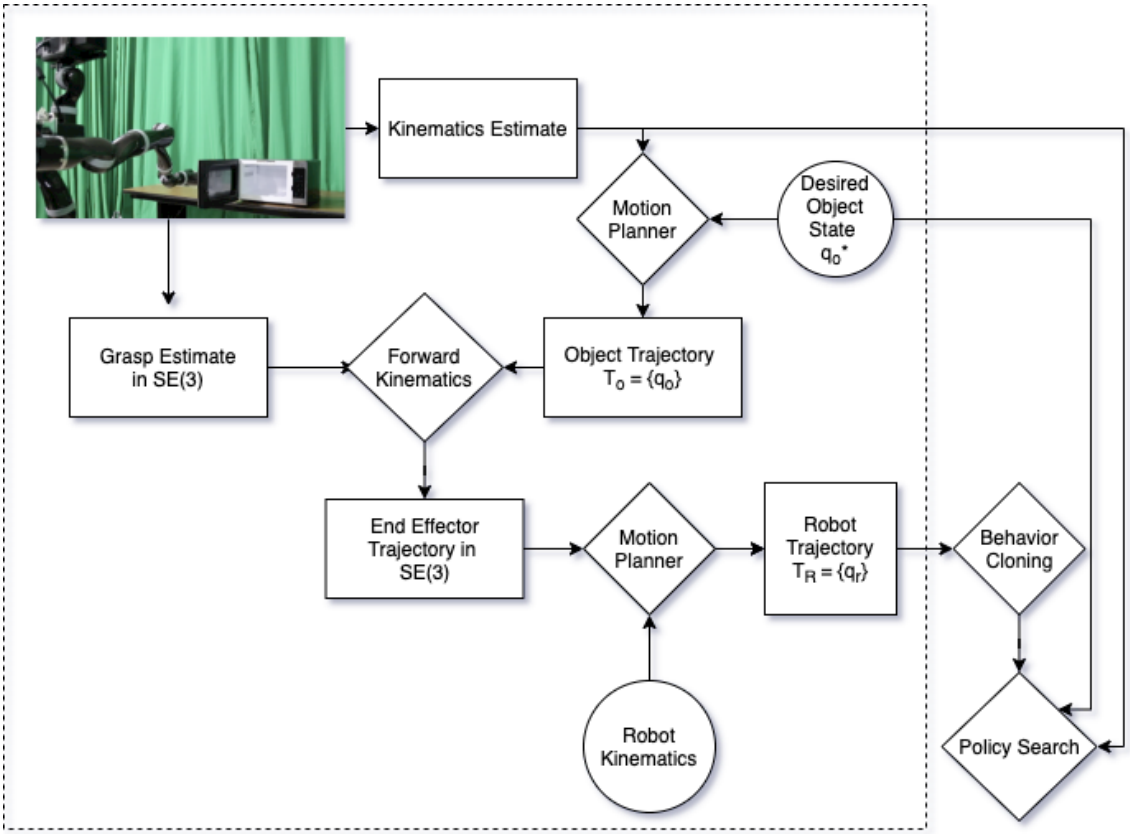


Figure 5.1: **System overview** illustrating our proposed framework for generating demonstrations with a motion planner and subsequently performing policy search. The dashed box contains the steps from Algorithm 2.

of the robot and environment. Examples of such tasks include pick-and-place, articulated object manipulation, and many instances of tool use. (Note that this requirement cannot capture reward functions defined in terms of force, for example exerting a specific amount of force in a target location.) Such a goal, together with object and robot kinematics, enables us to autonomously generate useful initial trajectories for policy search.

Our approach is outlined in Algorithm 1, and can broken down into five main steps: 1) collect initial trajectories from a motion planner using estimated object kinematics, 2) fit a policy with these initial trajectories, 3) gather rollouts to sample rewards for the current policy based on the kinematic goal, 4) update the policy parameters based on the actions and rewards, 5) repeat steps 3-4.

Algorithm 1 Planning for Policy Bootstrapping

procedure PPB($C_R, f_R^{-1}, C_O, f_O, q_O^*$)
  $D \leftarrow \varnothing$
  for 0 to N **do**
    $D \leftarrow D \cup$ InitialMPDemos($C_R, f_R^{-1}, C_O, f_O, q_O^*$)
  end for
  $\theta \leftarrow$ FitPolicy($D_0, ..., D_N$)
  for 0 to E **do**
    $T_0, .., T_n \leftarrow$ Rollout(π, θ, q_O^*)
    $\theta \leftarrow$ UpdatePolicy($T_1, .., T_n, \theta$)
  end for
end procedure

Algorithm 2 Initial Motion Plan Demos

procedure InitialMPDemos($C_R, f_R^{-1}, C_O, f_O, q_O^*$)
  $T_O \leftarrow$ MotionPlanner(C_O, q_O^*)
  $g \leftarrow$ EstimateGrasp(C_O,f_O)
  $eepath \leftarrow$ GraspPath(T_O, C_O, f_O, g)
  $T_R \leftarrow$ MotionPlanner(C_R, $eepath$, f_R^{-1})
  return T_R
end procedure

5.2.1 Fitting a Policy to a Demonstration

After collecting initial demonstrations from the motion planner, D, we can bootstrap our motor policy by initializing the parameters to the policy θ using any behavioral cloning technique; in practice, we use Locally Weighted Regression (Schaal and Atkeson, 1998) for DMPs, and maximize the likelihood of the demonstration actions under the policy for neural networks.

5.2.2 Policy Search with Kinematic Rewards

To improve the motor policies after bootstrapping, we can perform policy search based on the given (kinematic) reward function. Specifically, we choose a number of epochs E to perform policy search for. For each epoch, we perform an iteration of policy search by executing the policy and collecting rewards based on the goal q_O^*. We define our reward functions using estimated object states q_O and goal states q_O^*, and add a small action penalty.

5.3 Experiments

The aim of our evaluation was to test the hypothesis that motion planning can be used to initialize policies for learning from demonstration without human input. We tested this hypothesis in simulation against learning from scratch, and on real hardware, against human demonstrations, on three tasks: microwave-closing, drawer-opening, and t-ball. We note that we do not show asymptotic performance because our emphasis is on learning on real hardware from a practical number of iterations. All the elements of the motion planner—state sampler, goal sampler, distance metrics, etc.—are reused between problems without modification. We chose these two different motor policy classes because they represent opposite ends of the policy spectrum: deep neural networks are extremely expressive in what policies they can represent, but are extremely sample inefficient compared to structured motor primitives like DMPs, which are more structured and less expressive.

5.3.1 Simulation Experiments

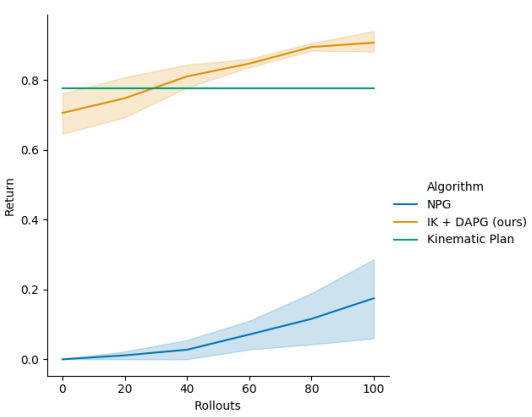
We used PyBullet (Coumans et al., 2013) to simulate an environment for our object manipulation experiments. We used URDFs to instantiate a simulated 7DoF KUKA LBR iiwa7 arm and the objects to be manipulated, which gave us ground-truth knowledge of the robot and object kinematics. For all our simulated experiments, we compared implementations of our method against starting with a random policy.

For all three tasks, the state was represented as $s_t = [q_R, q_O]^T$ where q_R denotes robot configuration and q_O denotes object configuration. The action space $\mathcal{A}$ was commanded joint velocity for each of the 7 motors. The reward at each timestep r_t was given as:

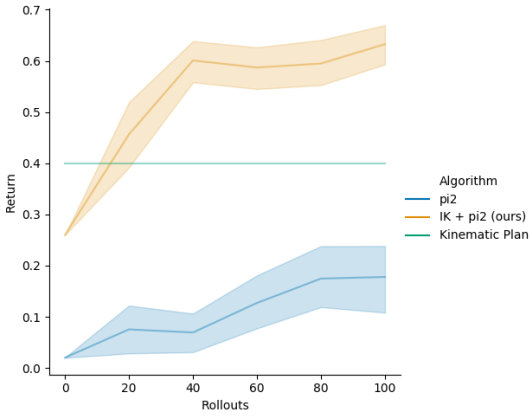
$$r_t = -c \; ||q_O^* - q_O||_2^2 - a_t^T R a_t, \tag{5.1}$$

where q_O denotes the object state at time t, q_O^* denotes desired object state, and a_t denotes the agent's action. We set $c = 60$ and $\mathrm{R} = I \times 0.001$ for all experiments. As such, maximum reward is achieved when the object is in the desired configuration, and the robot is at rest.

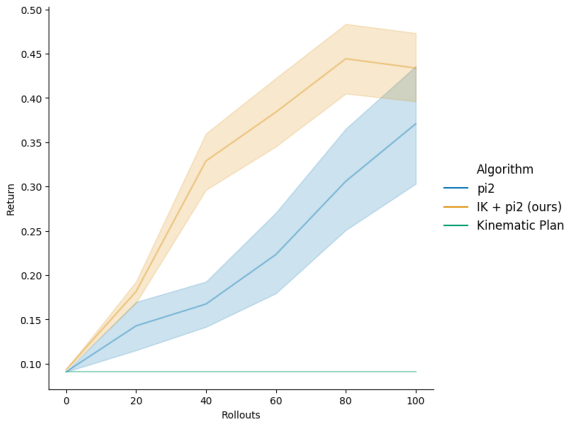
Our first simulated task was to close a microwave door, which consisted of three parts: a base, a door, and a handle. The pose of the handle was used for the EstimateGrasp method in Algorithm 2. The robot was placed within reaching distance of the handle when the microwave door was in an



(a) Microwave closing (MLP)



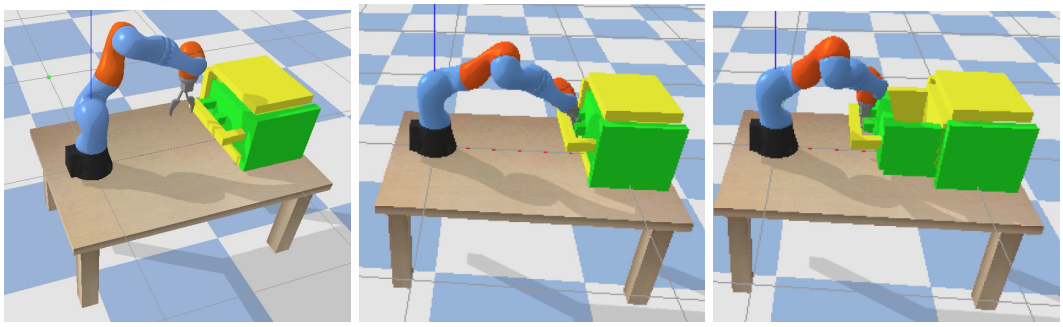
(b) Drawer opening (DMP)



(c) T-ball (DMP)

Figure 5.2: **Simulation Results.** a) Comparison of our method optimized with DAPG against Natural Policy Gradient starting with a random policy in a microwave closing task using Gaussian multi-layer perception policies. b) Comparison of our method against PI^2-CMA starting with a random policy in a drawer opening task with DMP policies. c) Our method compared with PI^2-CMA with a initially random policy in t-ball with DMP policies. Results are shown as mean and standard error of the normalized returns aggregated across 20 random seeds.

open position, but was too far to reach the handle in its closed configuration. Thus, the agent was forced to push the door with enough velocity to close it. We used Gaussian policies represented as multi-layer perceptrons with two hidden layers of sizes (32,32) in this experiment. The randomly initialized policy was optimized with natural policy gradient (Kakade, 2002). Ten demonstrations were generated by perturbing the start state and initial kinematic plan with Gaussian noise. The behavior cloning was performed by maximizing likelihood over the demonstration dataset for 10 epochs. Our pretrained policy was optimized using Demo Augmented Policy Gradient (Rajeswaran et al., 2017), which essentially adds the behavior cloning loss to the natural policy gradient loss,



(a) (b) (c)

Figure 5.3: **Opening a Drawer** experiment in simulation, where the robot needs to apply enough force on the handle to slide the drawer open. (a) An image of the starting pose of the robot arm and drawer. When learning from scratch, the robot random explores for many steps before grasping the handle. (b) An image of the robot using our method to produce an initial demonstration from a motion planner based on the drawer's kinematics. This demonstration guides the robot to the handle, but ignores the dynamics of the heavy drawer which leads to failure. (c) An image after the robot has bootstrapped a skill with our method. The final policy learns to leverage the dynamics to precisely grasp the handle and then produce a strong pulling force to open the drawer completely.

annealing it over time. This ensures that the agent remains close to the demonstrations early in learning, but is free to optimize reward exclusively as learning progresses. Results are shown in Figure 5.2a.

The second simulated task was to open a drawer (Figure 5.3). This task required the agent to grasp the drawer's handle and pull the drawer open.

Again, the pose of the object's handle was used for EstimateGrasp method in our algorithm. In this experiment, we used DMP policies. The weights, goals, and speed parameters of the policies were optimized using PI^2-CMA (Stulp and Sigaud, 2012). We used 32 basis functions for each of the DMPs. The pretrained policy was initialized using Locally Weighted Regression (LWR) (Schaal and Atkeson, 1998) with a single demonstration. The results of this experiment are shown in Figure 5.2b.

The third simulated task was to hit a ball off a tee. The ball started at rest on top of the tee. The pose of the ball was used in the EstimateGrasp method. The object state was defined as the object's y position relative to its initial pose. This is a poor initialization for a hitting task because it is based only on the ball's kinematics and ignores the dynamics involved in swinging, resulting in low-return, but it is effective for bootstrapping policy search. This experiment again used DMPs initialized with LWR and optimized with PI^2-CMA. Results of this experiment are visualized in Figure 5.2c.

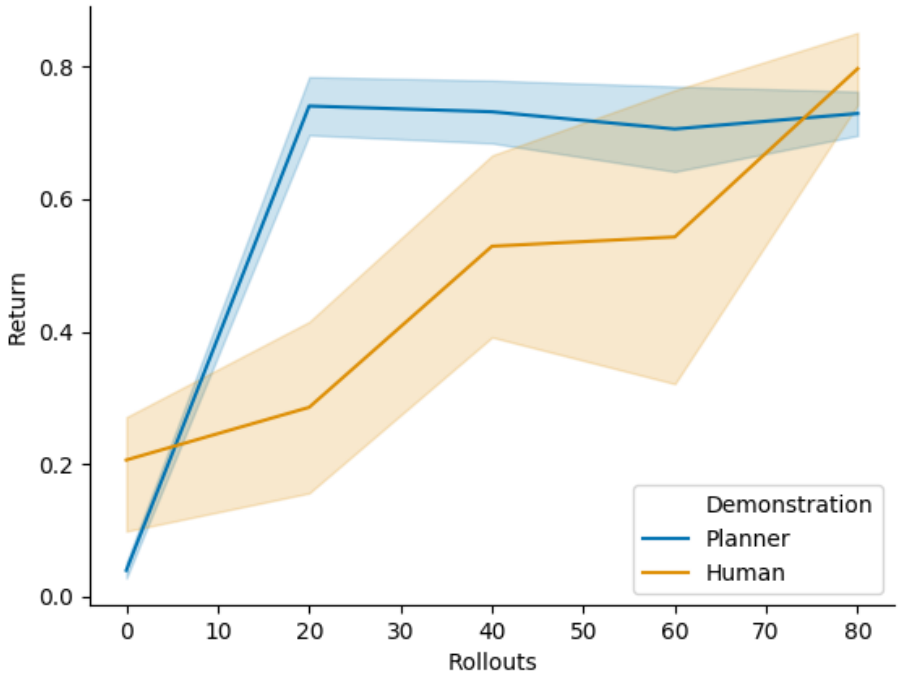


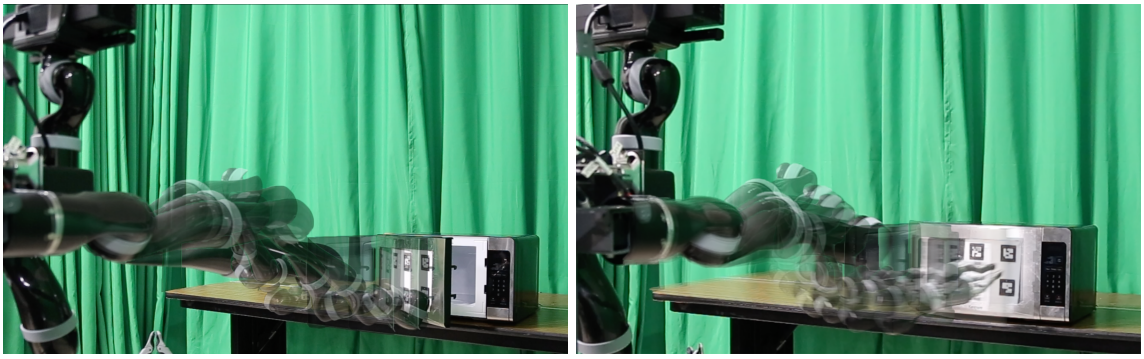
Figure 5.4: **Hardware experiment** comparing our initialization scheme with human demonstration. Results are shown as mean and standard error, aggregated across three random seeds.

Across all three tasks, we observe that policies initialized with our method dramatically outperform starting learning with a random policy. This confirms our hypothesis that using motion planning to generate demonstrations significantly speeds the acquisition of motor skills in challenging tasks like articulated object manipulation and t-ball.

5.3.2 Real-world Experiments

For all our real-world experiments, we used a 7DoF Jaco arm (Campeau-Lecours et al., 2019) to manipulate objects (Figure 5.5). We used ROS and MoveIt! (Coleman et al., 2014) as the interface between the motion planner (RRT* (Karaman et al., 2011) in our experiments) and robot hardware. For all real-world experiments, we compared implementations of our method against bootstrapping with a human demonstration, which we supplied.[1] To collect human demonstrations, we had an expert human teleoperate the robot with joystick control to perform the task. For all tasks, the state

[1]We acknowledge this potential bias in expert trajectories, and qualify our decision by only training on human demonstrations that at least accomplished the task.



(a) (b)

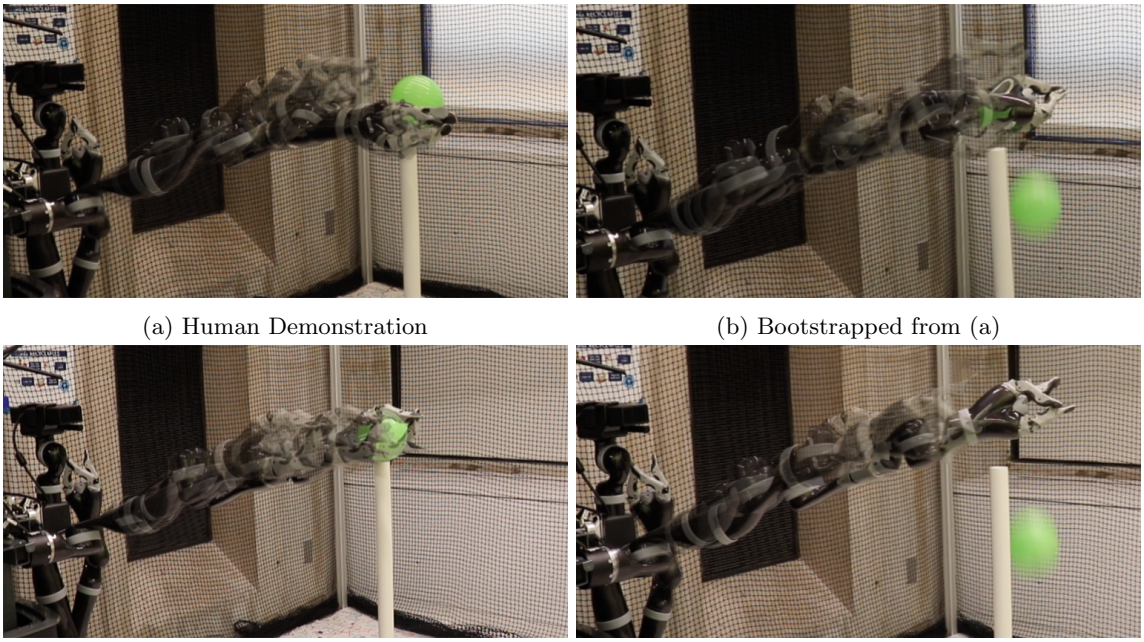
Figure 5.5: A robot using our method to autonomously learn to close a microwave that is out of reach. (a) The robot uses a motion planner to generate an initial attempt at closing the microwave door using a kinematic model of the microwave. The resulting plan is unable to fully close the microwave door because of the robot's limited reach. (b) After bootstrapping a motor skill with the trajectory from (a), the robot learns a motor skill that gives the door a push, exploiting its dynamics to fully close the microwave.

space, action space, and reward were defined in the same way as in our simulated results (Section 5.3.1). Both experiments used DMP policies initialized with LWR (Schaal and Atkeson, 1998) and optimized with PI^2-CMA (Stulp and Sigaud, 2012) with 10 basis functions for each of the DMPs.

Our first real-world task was to close a microwave door, similar to the one described in our simulated domain (Section 5.3.1). As in the simulated microwave task, we used the pose of the handle for the EstimateGrasp method in Algorithm 2, and also the robot was similarly placed such that it was forced to push the door with enough velocity to close. We placed an AR tag on the front-face of the microwave to track the microwave's state using a Kinect2. Results are shown in Figure 5.4. The behavior is visualized in Figure 5.5.

We observe that the human demonstration is better than the one produced by the motion planner, which we credit to the fact that the motion of the door was heavily influenced by the dynamics of the revolute joint which the motion planner did not account for. Nonetheless, both policies converge to a similar final performance, with our method converging slightly faster. Note the importance of the policy search: the motion planner alone is insufficient for performing the task efficiently.

Our second real-world task was to hit a ball off a tee as far as possible (Figure 5.6). Similar to our simulated task, the ball started at rest on top of the tee. The pose of the ball was used in the EstimateGrasp method. The object state was defined as the object's y position relative to its initial pose. We placed scotchlite-reflective tape on the surface of the ball and conducted our experiments



(a) Human Demonstration (b) Bootstrapped from (a)

(c) Motion Plan Demonstration (d) Bootstrapped from (c)

Figure 5.6: **Real-world Ball Hitting.** Images comparing bootstrapping motor skills with a human demonstration vs. a motion planner on a real-world robot hitting a ball off a tee. In both cases, the bootstrapped motor skill outperforms the initial demonstration. (a) A demonstration provided by a human teleoperating the robot. (b) A motor skill bootstrapped by the human demonstration. (c) A demonstration provided by a motion planner. (d) A motor skill bootstrapped by the motion planner demonstration.

within an OptiTrack motion-capture cage to track the object pose. We observe that when using a motion planner to hit the ball, it moves the bat in a linear motion to make contact, therefore transferring only horizontal motion to the ball. We qualitatively observe that during policy search, the robot learns a dynamic policy that accounts for the dynamics of the ball by applying force under the ball to "scoop" the ball upwards and forwards.

5.4 Related Work

To our knowledge, our method is the first to use an object's estimated kinematics in conjunction with a known robot model to bootstrap motor policy learning.

While classic robot motor learning papers (Atkeson and Schaal, 1997) leverage the known kinodynamics of the robot, they do not discuss kinematics of external objects or grasp candidates to bootstrap motor policies for object manipulation. We emphasize that we cannot form dynamic plans

in the problem setting we are interested in: objects with unknown a priori dynamics.

Recently, Model-Predictive Control (MPC) has been used in the context of imitation learning and reinforcement learning to address the high sample complexity of policy search (Pan et al., 2017). These approaches require a priori object dynamics, or human demonstrations to fit learned models; in contrast, our approach requires only object kinematics, which are much more readily estimated from visual data at runtime (Abbatematteo et al., 2019; Li et al., 2020). As such, our approaches enables the learning of manipulation skills to be more autonomous than existing MPC-based methods. Tosun et al. (2019) proposed a neural network model for generating trajectories from images, using a motion planner during training to enable the robot to generate a trajectory with a single forward pass at runtime. While this approach uses a motion planner for behavior cloning, it stops short of optimization to improve the resulting policy. By contrast, our method uses object kinematics to produce initial trajectories, while Tosun et al. (2019) only use the robot's kinematic model, which is insufficient when the task is to manipulate an object to a specific joint configuration.

Kurenkov et al. (2019) proposed training an initially random RL policy with an ensemble of task-specific, hand-designed heuristics. This improves learning but the initial policy is still random, yielding potentially unsafe behavior on real hardware, and delaying convergence to a satisfying policy. By contrast, we choose to initialize the policy with demonstrations from a kinematic planner, ensuring feasibility, safety, and rapid learning. Moreover, we argue that motion planning is the principled heuristic to use to accelerate learning, as it is capable of expressing manually programmed heuristics like reaching and pulling. Finally, our approach can use the existing estimated object kinematics to provide a principled reward signal for model-free reinforcement learning.

Recently, residual reinforcement learning approaches have been developed which learn a policy superimposed on hand-designed or model-predictive controllers (Silver et al., 2018; Johannink et al., 2019). Our method is compatible with these approaches, where demonstrations from the motion planner can be used as a base policy on top of which a residual policy can be learned based on kinematic rewards. These methods typically suffer from the same limitations as MPC-based methods mentioned above.

Guided Policy Search (GPS) (Levine and Koltun, 2013) uses LQR to guide policy search into high-reward regions of the state-space. The models employed are fundamentally local approximations, and thus would benefit greatly from a wealth of suboptimal demonstrations from the outset (as

made evident by Chebotar et al. (2017)). GPS is one of the state-of-the-art algorithms we expect to be used within our framework as the policy search implementation (Section 5.2.2). A critical distinction between our work and GPS is the notion of planning trajectories in object configuration spaces and reasoning about grasp candidates to achieve a desired manipulation. This is done using information available a priori, and thus is immediately capable of generating high-value policies, whereas GPS estimates dynamics models given observed data (obtained either from demonstration or random initialization). In the absence of a human demonstrator, our method would provide far more useful data at the outset of learning than running a naively initialized linear-Gaussian controller (as evidenced by our comparisons to random initialization).

Most similar to our line of work are those that use sample-based motion planners for improved policy learning. Jurgenson and Tamar (2019) harness the power of reinforcement learning for neural motion planners by proposing an augmentation of Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) that uses the known robot dynamics to leverage sampling methods like RRT* to reduce variance in the actor update and provide off-policy exploratory behavior for the replay buffer. However, Jurgenson and Tamar (2019) are only able to address domains where they can assume good estimates of the dynamics model, such as producing free-space motions to avoid obstacles. Our setting, in contrast, focuses on object manipulation, where dynamics are not readily available, but are critical for learning good policies. Jiang et al. (2019) address learning to improve plans produced by a motion planner, but do not bootstrap closed-loop policies. Motion planners are insufficiently expressive to leverage the dynamics in object-manipulation tasks, especially in the presence of unknown dynamics, and traditionally are unable to handle perceptual data like RGB images. Our method, on the other hand, enables motion planning to bootstrap policies that are more expressive than the original planner.

5.5 Conclusion

We have presented a method that uses kinematic motion planning to bootstrap robot motor policies. By assuming access to a potentially noisy description of the object kinematics, we are able to autonomously generate initial demonstrations that perform as well as human demonstrations—but do not require a human—resulting in a practical method for autonomous motor skill learning.

Our methodology is agnostic to the motion planner, motor policy class, and policy search algorithm, making it a widely applicable paradigm for learning robot motor policies. We demonstrate the power of our methodology by bootstrapping different policy classes with demonstrations from humans and a motion planner, and learn motor policies for three dynamic manipulation tasks: closing a microwave door, opening a drawer, and hitting a ball off a tee. Our framework is the first to enable robots to autonomously bootstrap and improve motor policies with model-free reinforcement learning using only a partially-known kinematic model of the environment.

The preceding chapters have developed intelligent exploration methods for autonomous mobile robots by exploiting structure in perceptual spaces. We now shift our focus to exploiting structure present in behavior itself, aiming to develop a policy class for contact-rich manipulation skill learning that builds on the impedance control scheme described previously. The next chapter explores Riemannian Motion Policies as a potential candidate for a safer policy class. Chapter 7 explores additional structure in skill learning that we can exploit for efficiency and compositionality, and proposes a different formulation of a safety controller.

Chapter 6

RMPs for Safe Impedance Control in Contact-Rich Manipulation

The following appeared as a conference paper in the 2022 IEEE International Conference on Robotics and Automation in Philadelphia, PA. The work was primarily authored by Seiji Shaw under my supervision.

6.1 Introduction

Learning autonomous contact-rich manipulation behavior is a critical challenge for robotics; indeed, the very purpose of a robot is often to make contact with the environment in order to manipulate it. However, contact-rich behavior is challenging—robots must be able to reason about sudden constraints when contacting target objects and model the unknown dynamics of the objects they are manipulating, all the while respecting joint limit and collision constraints.

To address these issues, Martín-Martín et al. (2019) discuss the advantages of formulating the task in the robot's end-effector space. They propose the use of variable impedance control in end-effector space (VICES), which defines the robot's actions in terms of displacements and compliance of the end-effector. The VICES commands are then translated back as lower-level torque commands to the joints in the arm by an operational-space impedance controller (OSC). When training an agent using reinforcement learning, the policy is trained to output actions in the VICES action-space, directly

controlling end-effector behavior, and thus can manage the discontinuous mode-changes made at contact using the compliance component of the space.

While the VICES modality serves as a useful layer of abstraction for learning and transfer, the agent loses the ability to reason about the configuration of the rest of its arm in relation to the environment. This can lead to routine hyperextension and collisions during and after training, which are undesirable and potentially dangerous. The problem of *safe* variable impedance control has therefore not yet been solved robustly (Abu-Dakka and Saveriano, 2020).

In this chapter, we study the problem of safely learning contact-rich manipulation behavior. We present RMP-VICES, a principled way to incorporate well-defined collision and joint-limit avoidance behavior into the robot's control system prior to the start of learning. We leverage Riemannian motion policies (RMPs), which allow a user to specify different behaviors in more convenient 'task' manifolds and then synthesize them (Ratliff et al., 2018; Cheng et al., 2021). Most importantly, RMPs have a mechanism (the Riemannian metric of these task manifolds) for designating the priority of these behaviors, which can vary based on the current position and velocity of the arm. We reformulate the VICES impedance controller as an attraction-type Riemannian motion policy, and synthesize it with pre-defined obstacle avoidance and a joint limit behavior (Cheng et al., 2021).

We verify the efficacy of our approach by demonstrating that RMP-VICES is comparably performant to state-of-the-art learning-based manipulation algorithms while largely preventing unsafe behavior. We do so by training a 7DOF Kinova Jaco 2 on three different simulated domains and analyzing the frequency and severity of collision and joint limit events that occur over the course of learning. Afterwards, we run the trained agents in the same domains with randomly-placed obstacles to evaluate RMP-VICES's ability to adapt to obstacles not present in training. RMP-VICES reduces the number of collision by up to $\sim 50\%$ and joint limit events by up to $\sim 90\%$ in all of these domains with only a moderate impact on task performance.

6.2 Background

6.2.1 Variable Impedance Control in End-Effector Space

Recall the action space VICES (Martín-Martín et al., 2019) that enables us to use end-effector space ($\mathsf{SE}(3)$) and compliance of the end-effector as the action space of the agent. VICES defines an action $a = (\Delta x, k^p_{pos}, k^p_{ori}) \in \mathcal{A} = \mathsf{SE}(3) \times \mathbb{R}^3 \times \mathbb{R}^3$, where k^p_{pos} and k^p_{ori} are vector representations of stiffness

matrices in an impedance controller.

In a single timestep, Δx is composed with the current end-effector position to produce desired set-points p_d and R_d. The corresponding joint torques τ are then computed using Khatib's formulation of operation space control (Khatib, 1995, 1987):

$$\tau = J_{pos}^T[\Lambda^{pos}[k_p^{pos}(p_d - p) - k_v^{pos}v]] + \quad J_{ori}^T[\Lambda^{ori}[k_p^{ori}e_r(R_d, R) - k_v^{ori}\omega]], \tag{6.1}$$

where J_{pos} and J_{ori} are the Jacobians associated with the FK map to end-effector position p and orientation R, Λ^{pos} and Λ^{ori} the corresponding end-effector inertia matrices, and v and ω the velocity and angular velocity of the end-effector. We compute damping matrices k_v^{pos} and k_v^{ori} so that the system is critically damped relative to given k_{pos}^p and k_{ori}^p as chosen by the agent. e_r is an error on $\mathsf{SO}(3)$ that can be used for impedance controllers (Luh et al., 1980); see section 6.3.2.

The greatest strength and weakness of VICES is that the control signal (actions) a and states are restricted to the end-effector space, $\mathsf{SE}(3)$. While this means that the agent will learn policies that can be transferred from one robot to another using $\mathsf{SE}(3)$ as a layer of abstraction, the policy is not able to reason about its joint states (and by proxy, the pose of the robot's links) that lead to its end-effector configuration to avoid collision.

6.2.2 Riemannian Motion Policies

The Riemannian motion policies framework (RMPs) is a mathematical formalism that decomposes complex robot motion behavior into individual behaviors specified in more interpretable task spaces (Ratliff et al., 2018). RMPs enable the controllers to be expressed in their appropriate task spaces and then manage the transforms and flow of control between those spaces to the robot's configuration space, which is where control must ultimately occur. Additionally, they allow us to blend multiple such controllers to, for example, reach to a point (an attractive controller between the robot's end-effector and a target location) while avoiding obstacles (a repulsive or dissipative controller between the robot's arm and obstacles in its 3D environment).

RMPs are organized in a tree-like structure—the RMP-Tree—where the root represents the robot's configuration manifold $\mathcal{Q}$, and every node represents a task manifold (e.g. $\mathsf{SE}(3)$). The edges of the tree represent maps from the positions in the parent task space to the child task space (for an example, see Fig. 6.1). Here, we denote a parent space as $\mathcal{M}$ and the ith child space as $\mathcal{N}_i$.

We will write the map from $\mathcal{M}$ to $\mathcal{N}_i$ as ϕ_i.

Individual component behaviors to be synthesized are represented by maps from position and velocities to forces at the leaf nodes. If $\mathcal{N}$ is a leaf task-space, we notate the behavior map as $f(x, \dot{x})$. The priority of each of these behaviors is specified by a positive-semidefinite tensor that resembles the Riemannian metric of these task manifolds, which can vary based upon the position and velocity in that task space. We denote this metric as $M(x, \dot{x}) \in \mathbb{R}^{n \times n}$, where $n = \dim \mathcal{N}$, and $x \in \mathcal{N}$.

RMP-trees are evaluated in two phases: pushforward and pullback. In pushforward, the current state of the arm $(q, \dot{q})$ is propagated through the tree to compute the state in each child manifold. In pullback, forces f_i are evaluated at each child manifold and then synthesized back to compute the joint accelerations $\ddot{q}$ at that timestep.

Pushforward

Let $\mathcal{M}$ be a parent space, and let $\mathcal{N}_i$ be one of its child spaces, and let $x, \dot{x}$ be a computed position and velocity in $\mathcal{M}$. In pushforward, the corresponding position y and velocity $\dot{y}$ in $\mathcal{N}$ are computed as $y = \phi_i(x)$ and $\dot{y} = J_{\phi_i}\dot{x}$ respectively, where J_{ϕ_i} is the Jacobian of task-mapping ϕ_i.

Pullback

After pushforward is complete, we have position and velocity in each of the leaf task manifolds. We then must evaluate each policy f_i and propagate the force signal back to the configuration manifold. Given parent manifold $\mathcal{M}$ and child leaf manifold N_i, with position y_i and velocity $\dot{y}_i$ in $\mathcal{N}_i$ we first evaluate $\ddot{y}_i = f_i(y_i, \dot{y}_i)$ to find the corresponding leaf's acceleration, and $M_i(y_i, \dot{y}_i)$ to find the Riemannian-metric priority tensor. We then compute the corresponding acceleration $\ddot{x}$ and metric M in $\mathcal{M}$ as follows:

$$\ddot{x} = \sum_{i=1}^{n} J_{\phi_i}^T (\ddot{y}_i - M_i(y_i, \dot{y}_i)\dot{J}_{\phi_i}\dot{x}),$$
$$M = \sum_{i=1}^{n} J_{\phi_i}^T M_i(y_i, \dot{y}_i) J_{\phi_i}.$$

We then recursively pullback from the leaf nodes to the root node and find the joint accelerations by computing $\ddot{q} = M^{\dagger} f$, where $M^{\dagger}$ is the Moore-Penrose pseudo-inverse of M. This process solves the least-squares problem that trades off policy outputs with respect to each metric M_i.

6.2.3 Related Work on RMPs

RMPs build on many earlier methods that decompose behavior into a number of different task spaces. Nakamura et al. (Nakamura et al., 1987), Sentis et al. (Sentis and Khatib, 2005), and Coelho and Grupen (Coelho Jr and Grupen, 1997) all propose similar frameworks that decompose the robot task space in recursively-defined nullspaces. RMPs are formulated in a way that subsume these methods.

RMPs have been used primarily to guide robot arms through free space in a variety of reaching and manipulation-based tasks (Li et al., 2021; Mukadam et al., 2020; Lee et al., 2020; Handa et al., 2020; Kappler et al., 2018). However, none of them perform contact-rich manipulation by training an agent whose actions are translated into attraction-type behavior in the RMP-tree itself. While not addressing contact-manipulation problems, Li et al. (Li et al., 2021) introduces RMP^2, a framework that allows an agent to learn with RMPs to accomplish 2D reaching tasks safely using state-of-the-art deep RL methods.

6.3 RMP-VICES

A naive operational-space controller (e.g. as in VICES (Khatib, 1987)) fails to reason about collision with the environment and violation of the arm's joint limits. We propose RMP-VICES, an operational-space controller which translates $\mathsf{SE}(3)$ impedance commands from a learned policy to low level torques that accounts for the geometry of the configuration space and surrounding environment. We formalize variable-impedance control as an attractor-type RMP, and fuse it with repulsion-type RMPs based at points sampled from the robot's arm. The resulting controller blends the desired policy control with collision avoidance behavior in training and in deployment.

In this chapter, we assume perfect knowledge of the robot arm dynamics and the pose and geometry of all obstacles in the environment, but neither object nor contact dynamics. The goal of RMP-VICES is to solve manipulation tasks with comparable performance to VICES but with fewer obstacle collisions and joint-limit extensions over training and deployment.

6.3.1 Tree Structure

The forward-kinematics (FK) map to the last link of the arm is used to compute the pose of the end-effector in $\mathsf{SE}(3)$. We then perform a selection mapping to decompose $\mathsf{SE}(3)$ into $\mathbb{R}^3$ and $\mathsf{SO}(3)$,

and define the variable impedance behavior in those spaces. We perform an identity map from the robot's configuration space to itself, where we define the joint-limiting behavior as shown in Cheng et al. (Cheng et al., 2021) and was tuned via experimental validation.

For the purpose of collision avoidance, we sample control points on the links of the arm, and define distance spaces to each obstacle. Since we conducted training with a 7DOF arm, we need seven different FK maps from the configuration space to $\mathsf{SE}(3)$ for the pose of each individual link of the arm. From each of the sampled control points, we compute a selection map to $\mathbb{R}^3$ and then a distance map to $\mathbb{R}$ for each obstacle, and define collision-avoidance behavior there. A schematic of the RMP-tree can be seen in Fig. 6.1.

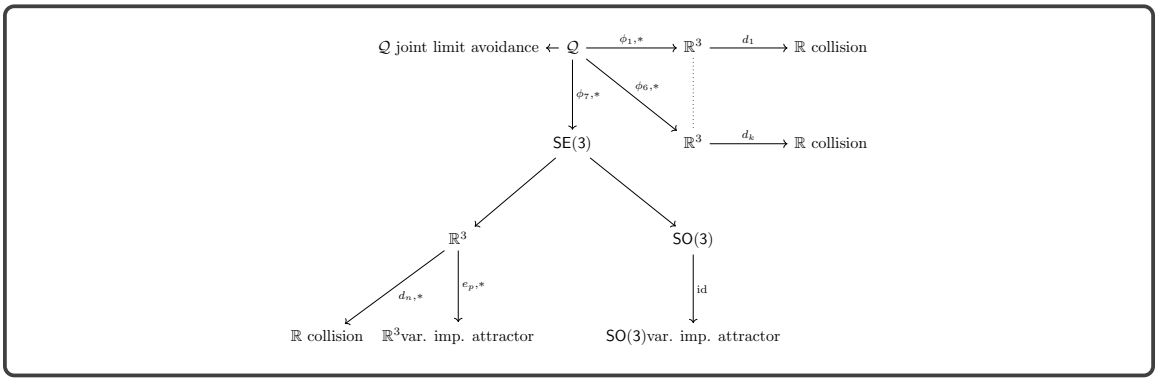


Figure 6.1: The RMP-tree structure used for RMP-VICES. $\phi_1, ..., \phi_7$ denote FK maps from $\mathcal{Q}$ to each link of the arm, where VICES sits in $\mathsf{SE}(3)$. The rest of the RMP-tree is made up to control points sampled from the link meshes, which are mapped to the shortest-distance space each obstacle ($d_1, ..., d_n$ act as signed-distance functions) where collision avoidance is defined. The selection map from $\mathsf{SE}(3)$ to $\mathbb{R}^3$ has been omitted.

6.3.2 Impedance Control as Attractor-type RMPs

We formulate impedance controllers as RMP leaves in the $\mathbb{R}^3$ and $\mathsf{SO}(3)$ task spaces. Both of these controllers are formulated as attraction-type policies (Ratliff et al., 2018).

Let $x \in \mathbb{R}^3$ be a position of the end-effector in free space and $\dot{x}$ be its associated velocity as computed by the pushforward operation. Let $x_g \in \mathbb{R}^3$ be a desired position. Then the motion behavior is formulated as a spring-damper attractor-type policy in $\mathbb{R}^3$:

$$f(x, \dot{x}) = k_p(x - x_g) - k_d(\dot{x}), \tag{6.2}$$

with the associated metric being the identity matrix I. The diagonal matrices k_p^{pos} and k_d^{pos} specify the stiffness of this controller and are set by the trained policy at every timestep.

The attractor in $\mathsf{SO(3)}$ is written in a very similar way to the RMP above, except we also must account for the non-Euclidean topology of $\mathsf{SO(3)}$ in the error term. Let $r \in \mathsf{SO(3)}$ and ω be the end-effector's current orientation and angular velocity, and r_g be the desired orientation. We represent $r, r_g \in \mathsf{SO(3)}$ as rotation matrices, i.e. $r = [r_x, r_y, r_z]$ and $r_g = [r_{gx}, r_{gy}, r_{gz}]$. We then use the same error term on $\mathsf{SO(3)}$ from VICES:

$$e_r(r, r_g) = \frac{1}{2}\left(r_x \times r_{xg} + r_y \times r_{yg} + r_z \times r_{zg}\right). \tag{6.3}$$

This error is equivalent to the sine of the angle to rotate r to r_g about a single axis (Luh et al., 1980). We then define the impedance controller using this error:

$$f(r, \omega) = k_p^{ori} \cdot e_r(r, r_g) - k_d^{ori} \cdot \omega, \tag{6.4}$$

where k_p, k_d are the stiffness and damping coefficients respectively. As before, the metric associated with this attraction-type policy is the identity I.

It is important to note that removing the obstacle-avoidance and joint-limit policies from the RMP-tree reduces the controller to VICES (Ratliff et al., 2018; Martín-Martín et al., 2019).

6.3.3 Collision Avoidance RMPs

For both types of obstacles, we use the signed-distance function to map the position of the control point sampled on the arm to the distance space between the point and the obstacle. We define the Riemannian metric $m(x, \dot{x}) \in \mathbb{R}$ of this 1-dimensional space to be the following expression:

$$m(x, \dot{x}) = \frac{(\max\{\dot{x}, 0\})^2}{x^4}. \tag{6.5}$$

This metric was derived via experimental validation. As described in (Ratliff et al., 2018), the collision policy will only activate and strengthen if the control point is moving towards the obstacle. For both planar and spherical collision-avoidance behaviors, we do not provide a repulsive signal but a damping term ($f(x, \dot{x}) = \eta\dot{x}$, for some damping coefficient η). As Bylard et al. (2021) observe, using only the Riemannian metric and a dissipation function to reduce the movement towards the

obstacle to generate collision-avoidance behavior reduces the likelihood of the arm to trap itself in local-minima.

6.3.4 Integrating Policy Actions in the RMP-VICES Controller

In an MDP formulation of a manipulation task, our state space will always be a superset of $\mathsf{SE}(3) \times S_{obj}$, where $\mathsf{SE}(3)$ is the space of end-effector pose and S_{obj} is the space of states of the manipulated object. An action will be a tuple $(\Delta x, k_p^{pos}, k_p^{ori}) \in \mathsf{SE}(3) \times \mathbb{R}^3 \times \mathbb{R}^3$, where Δx is a displacement in $\mathsf{SE}(3)$ for the next goal pose and k_p^{pos} and k_p^{ori} are the stiffness coefficients for the impedance controllers (k_d is computed so the impedance controllers are critically-damped). The policy π will be sampled in pushforward and used to update k_p^{pos}, k_p^{ori}, k_d^{pos}, k_d^{ori}, and r_g, and x_g in eqs. 6.2 and 6.4 in the RMP-tree on every timestep.

After the pullback stage of the RMP, we obtain a corresponding joint acceleration $\ddot{q}$ given by the synthesis of the end-effector impedance control and collision avoidance behavior. To convert $\ddot{q}$ into joint torques, we multiply by the inertia matrix of the arm in joint space and compensate for gravity and Coriolis forces.

6.4 Experiments

We test and verify the efficacy of RMP-VICES on the trajectory-following and door-opening domains constructed in Martin-Martin Martín-Martín et al. (2019) and an additional block pushing domain we developed. In each simulated domain, we first train VICES and RMP-VICES in an environment with no additional obstacles to evaluate any negative impact the RMP-tree has on learning efficiency. We also record the force of all collisions of the arm with objects in the workspace to understand the severity of collisions events using VICES and RMP-VICES. Collision and joint-limit events incur a penalty in the reward function and termination of the episode (but an ablation on these two properties showed that they were not essential for effective training). Afterwards, we perform 100 rollouts with the learned policy in each domain with two randomly placed spherical obstacles and record the force of any collision events to verify the adaptability of each algorithm to obstructions not seen during training.

All simulated experiments were conducted in Robosuite (Zhu et al., 2020b). The agent was given a stochastic policy parameterized by two fully-connected layers of 64 nodes with tanh activations

and optimized using PPO, as done in VICES. Our PPO implementation is based on the code written for OpenAI's SpinningUp (Achiam, 2018). In each domain, each episode was given a length of 1024 steps, with 4096 steps per epoch. The policy network was initialized with random weights (10 seeds) and was trained for 367 epochs (1.6×10^6 training steps).

6.4.1 Environments

Trajectory-Following

We randomly place four via points in the workspace, and specify an order in which the robot end-effector must traverse through them. As in VICES, the state of the agent is represented by the pose and the velocity of the end-effector, the position of each via-point, and whether each via-point has been checked. For the rollouts after training, we randomly generate two spherical obstacles with a radius of 5 cm in the bounding volume that is used to generate the via-points. We ensure that no obstacles overlap with a via-point so that the task still has a guaranteed solution.

Door-Opening

The RMP-tree is initialized with two plane collision avoidance policies to avoid the panel of the door and the surface of the table. For the rollouts after training, we generate two spherical obstacles in a bounding volume placed at a distance of front of the door to ensure that it can still swing open for task completion.

Block-Pushing

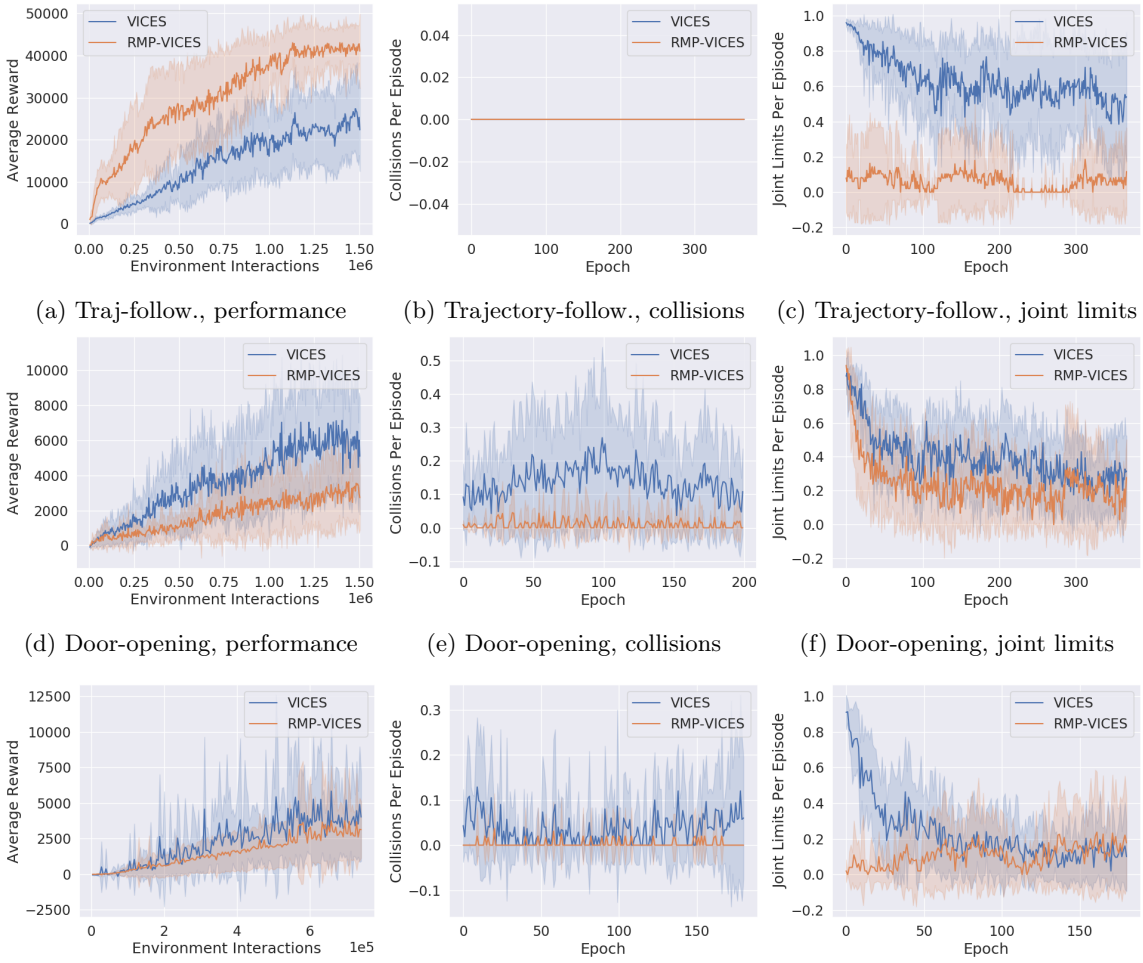
We construct a domain where the robot must push a block across a table to a goal position. The state includes the end-effector's pose and distance to the block as well as the distance between the block and the goal position. The reward function gives a small shaped reward that depends on the distance between the end-effector and the goal, and a larger shaped reward between the cube and the goal:

$$\begin{aligned} r_{main}(d_{h2c}, d_{c2g}) =& r_{h2c} * (1 - \tanh(20 d_{h2c})) + \\ & r_{c2g} * (1 - \tanh(20 d_{c2g})), \end{aligned} \tag{6.6}$$

where d_{h2c}, d_{c2g} are the distance form the hand to the cube, and distance from the cube to the goal respectively. r_{h2c} and r_{c2g} are the maximum rewards made to provide incentive for the agent to

bring the end-effector to the cube, and the cube to the goal, respectively.

6.5 Results



(a) Traj-follow., performance (b) Trajectory-follow., collisions (c) Trajectory-follow., joint limits

(d) Door-opening, performance (e) Door-opening, collisions (f) Door-opening, joint limits

(g) Block-pushing, performance (h) Block-pushing, collisions (i) Block-pushing, joint limits

Figure 6.2: Average reward, collision, and joint limit performance of VICES and RMP-VICES over the course of training on the trajectory-following (6.2a,6.2b,6.2c), door-opening (6.2d,6.2e,6.2f), and block-pushing domains (6.2d,6.2e,6.2f). When integrating the average curves for joint limits and collision avoidance behaviors, we see that RMP-VICES outperforms VICES in every domain. Percent decrease in collision and joint limit rates, respectively: trajectory-following: N/A%, 90.5%; door-opening: 39.8%, 90.7%; block-pushing: 54.6%, 93.2%.

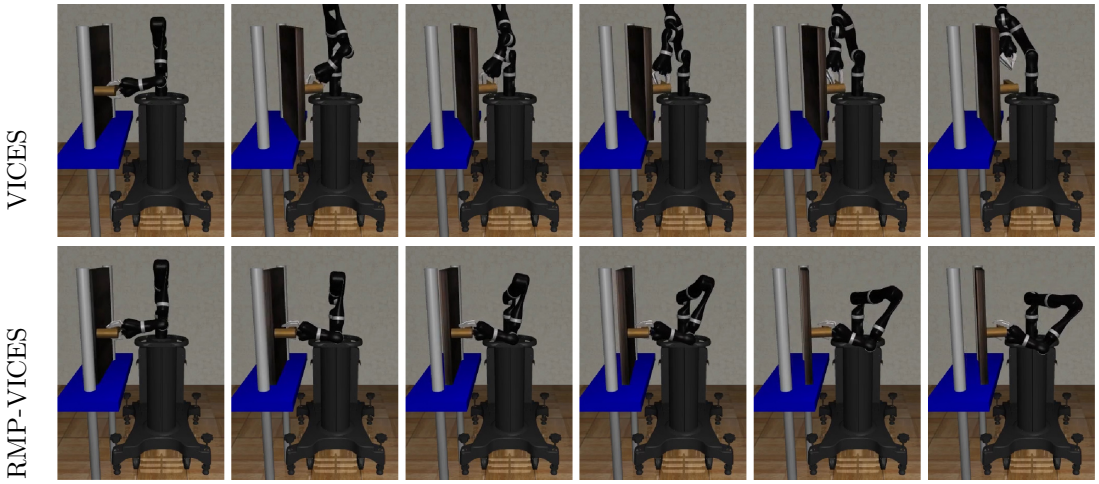


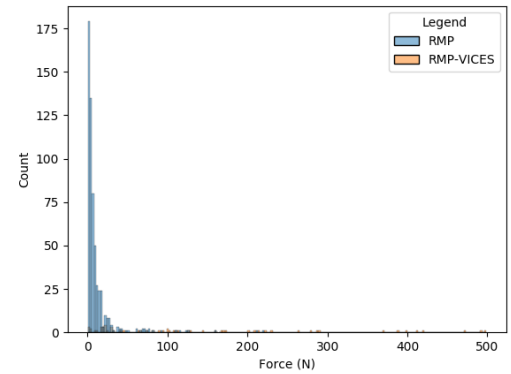
Figure 6.3: Regularly-timed stills of a single rollout of VICES (top) and RMP-VICES (bottom) after training is complete.

6.5.1 Safety During Training

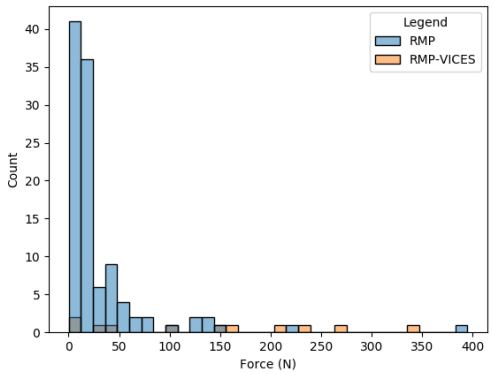
In the trajectory-following task, RMP-VICES has far fewer joint-limit events than VICES (Fig. 6.2c) throughout training. Furthermore, RMP-VICES learns more sample-efficiently than just VICES alone (Fig. 6.2a). Since violating a joint limit causes episode termination, RMP-VICES has access to more viable data earlier in training, which accelerates the agent's ability to learn the task.

In the door-opening task, we see that while RMP-VICES avoids collision better than VICES (figs. 6.2e, 6.4a), the agent trained with RMP-VICES learns with worse overall performance (Fig. 6.2d). However, RMP-VICES is significantly safer in guiding the robot away from collision avoidance behavior. After reviewing several rollouts of each policy (Fig. 6.3), we see that the policy learns a behavior that keeps the robot's pose largely the same with the elbow pointing straight up, which the RMP-VICES joint-limit and collision-avoidance policy represses. As a result, the RMP-VICES controller first pitches the elbow downward, and then opens the door. Since we give a dense reward that increases with door angle, the RMP-VICES agent obtained a lower average reward in VICES. However, the RMP-VICES policy is a safer behavior than the one learned by VICES alone, since the VICES behavior is prone to collision with the door (Fig. 6.3). This problem can be addressed with better tuning of the joint-limit policy or by learning the Riemannian metric (Li et al., 2021).

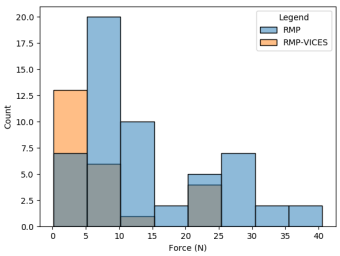
In the block-pushing task, we see that RMP-VICES starts with better joint-limiting performance at the beginning of training figs. 6.2h 6.2i. While VICES performs slightly better than RMP-VICES



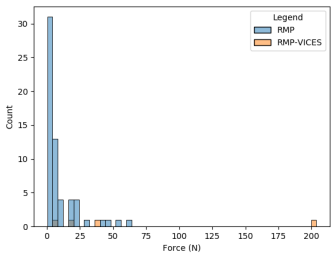
(a) Door-open.



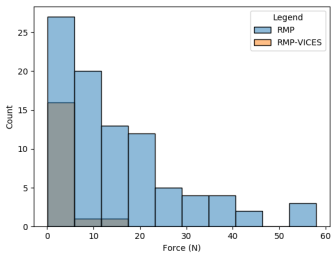
(b) Block-push.



(c) Traj-follow.



(d) Door-open.



(e) Block-push.

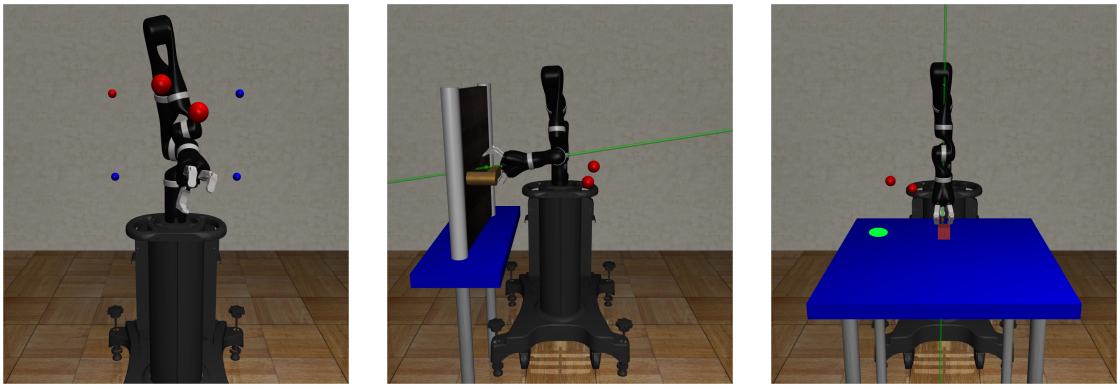
Figure 6.4: Severity of collision events in training (a,b), and in environments with randomly generated obstacles (c-e).

in efficacy of learning, RMP-VICES is much more performant in collision avoidance throughout training.

6.5.2 Deployment in Environments with Generated Obstacles

For each experiment, we choose the best performing seeds from the VICES and RMP-VICES agent from each domain, and played through 100 episodes with randomly-generated spherical obstacles in the environment. The size and location of the bounding volumes used for each domain were chosen as areas where the arm would frequently pass through to the solve the task. Table 6.1 gives an aggregation of the performance of VICES and RMP-VICES from this set of experiments. See Fig. 6.5 in the for visualizations of all three environments for a single rollout instance.

Relative to performance in training, we see a drop in performance in all tasks. This makes intuitive sense because neither the VICES nor RMP-VICES policies experienced these obstacles before training. RMP-VICES, in all tasks, collides less frequently than VICES. While we do see a



(a) Traj-follow. env. (b) Door-open. env. (c) Block-push. env.

Figure 6.5: Instances of task environments with randomly-generated obstacles (red spheres).

slight drop in performance in average reward, it is important to note that the success of the agent must be measured by its efficacy its ability to remain safe while completing the task.

It is clear that the agent trained on RMP-VICES exploits the geometric prior given by the collision-avoidance RMPs for more safe behavior. An agent trained with VICES alone has no obvious way to incorporate this information, and the arm moves through the space blindly without adjusting its trajectory to avoid the new obstacles in its path.

Furthermore, we see that RMP-VICES reduces the impact of collision between the arm and its surrounding environment. Here, we see that RMP-VICES in these domains exhibits a sharper distribution about 0N (Fig. 6.4).

	Task	Avg. Reward	Coll.	Jnt Lim.
VICES	Traj-follow.	$7864 \pm 1.167 \times 10^4$	56	44
	Door-open.	$1703 \pm 1.948 \times 10^3$	61	36
	Block-push.	$1755 \pm 2.256 \times 10^3$	93	0
RMP-VICES	Traj-follow.	$6847 \pm 3.723 \times 10^3$	25	22
	Door-open.	$1051 \pm 6.96 \times 10^2$	5	11
	Block-push.	$1740. \pm 8.54 \times 10^2$	19	0

Table 6.1: Results from rollouts with randomly-generated obstacles. As collision or passing a joint limit triggers the episode to end, the maximum number of collision and joint events for a single domain for a single agent is 100.

6.6 Conclusion

RMP-VICES is a geometry-aware operational-space controller for contact-rich manipulation. Our formulation synthesizes learned impedance control in $\mathsf{SE}(3)$ with predefined collision and joint limit avoidance behaviors, yielding a safer framework for reinforcement learning of manipulation skills. Our evaluation shows reduced frequency of collision and joint limit violation on the majority of tasks studied, especially early in learning. The main drawbacks of our proposed approach are the intensive tuning efforts required to weight leaf policies appropriately in the RMP-tree. Our framework also lacks Lyapunov or control barrier-certificates that guarantee stability or safety set invariance, respectively. Future work is warranted to investigate the design and tuning of provably safety behaviors and their interaction with policy optimization. In the next chapter, we propose Compositional Interaction Primitives (CIPs), a policy class for achieving safe, efficient motor skill learning. While RMPs showed some improvement in safety, ultimately the tuning effort proved to be prohibitively challenging. We used the experience in developing the system of the present chapter to develop a simpler safety controller for CIPs.

Chapter 7

Compositional Interaction Primitives

The following chapter is derived from a conference paper under review. The work was co-authored with Eric Rosen, Skye Thompson, Tuluhan Akbulut, Sreehari Rammohan, and George Konidaris.

7.1 Introduction

The unique potential of robots lies in their ability to do physical work in the world—every process that currently requires a human to meaningfully interact with a physical object can only be automated by a robot. Despite this immense potential value, only a tiny fraction of the physical manipulation tasks that can be automated currently are (Kroemer et al., 2021). There are multiple causes of this failure, but one of the most acute is that robots are currently not as flexible as humans in their ability to learn to interact with objects around them. A factory worker can be trained to basic proficiency in an unfamiliar task in a day; skillful and reliable execution of rote manual labor tasks rarely requires more than a few weeks. Achieving the same level of flexibility, reliability, and skill in robots requires major advances in their learning capabilities, so that a robot can be trained to solve a new task, and subsequently improve its own performance, in reasonable time and without the support of expert programmers.

There are two families of approaches to learning manipulation skills. The first combines end-to-end deep neural networks with reinforcement learning (RL) (Sutton and Barto, 1998; Kober and Peters, 2010) to learn "pixel to torque" controllers (Levine et al., 2016) that directly map sensor input to motor output. Such approaches couple RL's promise of flexibility, generality, and autonomy, with the opportunity to exploit the power of deep networks. However, they have dauntingly high sample complexity and face difficulty incorporating other techniques from robotics such as forward and inverse kinematics, motion planning, wrench closure, and feedback control. The second family aims to develop carefully designed and highly structured policy classes (Ijspeert et al., 2002; Cheng et al., 2021; Ratliff et al., 2018) to achieve sample-efficient learning, thereby trading design effort, flexibility, and generality for sample efficiency. Such approaches have learned an impressive range of dynamic behaviors (Kober and Peters, 2010; Muelling et al., 2013) in a feasibly low number of interactions, but are best suited for targeting a restricted class of motor skills where there is structure to be exploited and sample efficiency is paramount.

We focus on one such class, *sustained contact manipulation skills*—where a robot must establish stable contact (in the form of a grasp) with an object in order to change its state, and sustain that contact throughout execution. Examples of such tasks include opening a drawer, pulling a lever, turning a doorknob, opening a door, turning a wheel, or shifting gears. We introduce a new policy class, *Composable Interaction Primitives* (or CIPs), that draws from the best of both motor skill learning approaches: it exploits the structure present in sustained contact tasks, resulting in a policy class that is structured, safe, and highly parameter- (and therefore data-) efficient; and then applies deep networks to the components where learning from high-dimensional input is unavoidable. Additionally, CIPs are sequentially composable by construction, so that learned skills can be sequenced to solve new tasks in an order determined at runtime by a task-level planner. Using an ablation experiment in four simulated manipulation tasks, we experimentally explore the role of structure in manipulation skill learning, and show each of the components of CIPs substantially improves learning efficiency and safety. We then demonstrate the use of CIPs to efficiently learn, and subsequently sequence on-demand, sustained-contact manipulation skills on real robot hardware.

7.2 Composable Interaction Primitives

The success of DMPs suggests that one approach to achieving successful motor skill learning is to match a restricted but important class of motor skills with a representation designed to exploit its structure (Schaal, 2006; Ijspeert et al., 2002). We address learning motor skills in the *sustained contact* regime: skills where a robot must establish and maintain contact with an object while exerting force on that object to successfully manipulate it, such as when a human opens a door, pulls a lever, wipes a surface, or shifts gears. Such motor skills are common, complex, and important: much of the work that a robot with a gripper will be tasked with performing in the world—all except pick-and-place and instantaneous contact skills like pushing a button—will require sustained interaction with an unmodeled (or partially modeled) object. They are also highly structured (e.g., including making and breaking contact via a pre-grasp pose), necessitate complex safety constraints such as joint position and torque limits, and demand precise control driven by policies that must be learned from noisy and high-dimensional tactile and force feedback. Finally, they should be designed to support composition: suitable for sequential execution to address new tasks in an order determined at runtime, a capability that is unlikely to occur by chance and can only be achieved through design. All these reasons make sustained-contact motor skills excellent candidates for a specialized motor policy class.

We identify four important properties present in sustained-contact motor skills. First, *skill execution can be decomposed into phases*: the robot first moves through free-space to reach a pre-grasp pose, then achieves a stable grasp, then manipulates the object, then releases its grasp, and finally controls its gripper back into free-space. Second, *most phases involve little or no per-task learning*: motion through free-space and to achieve or release a grasp can be computed using motion planning and feedback control, respectively; the choice of where to grasp the object is a supervised learning task that can be resolved (or at least bootstrapped) using a generic grasp detector (Bohg et al., 2013). Only the sustained-contact controller itself need be largely learned on a per-task basis, though it could be bootstrapped using learning from demonstration (Argall et al., 2009) or kinematic motion planning (Abbatematteo et al., 2021). Third, *the sustained-contact controller itself naturally suggests structure*: the controller must be a function of force- and tactile-feedback, learned using reinforcement learning; the goal of learning should be to reach a task-specific goal (e.g., opening a door, or switching a light on) while avoiding task-general failure modes (like losing contact with

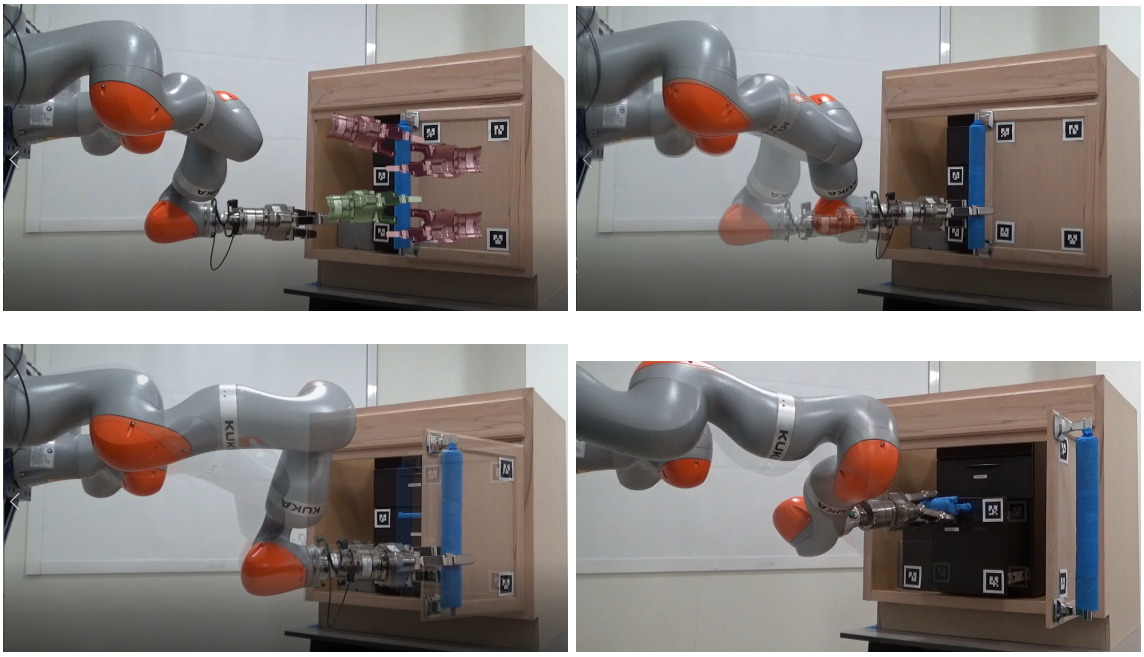


Figure 7.1: **Composable Interaction Primitives (CIPs)** are a structured policy class that enables safer, sample-efficient learning of contact-rich manipulation skills by incorporating model-based priors. Our approach (a) identifies promising grasp poses for initiating manipulation (b) uses motion planning to move in free space rather than learning to reach (c) learns model-free policies for interaction where necessary (d) enables composition by construction to support task-level planning.

the object or becoming stuck); and during learning the policy should be able to explore while being position- and torque-constrained so as to never damage the robot or the object. Here, task-specific structures are components that are either learned or hand-specified for a specific object manipulation skill, whereas task-general structures are components that may be specific to the robot but can be reused across different object manipulation skills. Finally, *a natural means of composition is through free-space motion planning*: motor skills can be sequenced by simply motion planning from one skill's release point to another skill's grasp point.

We therefore propose *Composable Interaction Primitives* (CIPs), a new policy class structured by these insights and aimed at learning composable sustained-contact manipulation skills in tens, rather thousands, of real-world interactions. CIPs are structured as a tuple, where components subscripted by c are specific to the task, and the remainder are specific to the robot but generic across tasks:

$$C = (\pi_c, \sigma, \beta_c, I_c, h, t, \Gamma, B)\text{, where:}$$

- $\pi_c : \phi \rightarrow \tau$ is a motor control policy that maps tactile sensor signals, proprioceptive data, and object state information to joint torques τ and gripper commands τ_g, and is parameterized by ψ.

- Policy π_c is constrained by σ, a safety envelope specific to the robot but not to the task. Execution is constrained to obey σ so that the agent does not damage the object it is interacting with or itself.

- $\beta_c : \phi \rightarrow \{0, 1\}$ is a task-specific success indicator that maps the robot's observations ϕ to a boolean indicating whether the interaction primitive has achieved its goal.

- B is a task-general classifier indicating interaction failure (e.g., that contact has been lost, the interaction has timed out, or execution cannot continue without a safety constraint being violated). Once initiated, π_c continues execution until either β_c indicates success or B indicates failure. The resulting signal informs a policy search algorithm to optimize π_c.

- $I_c : v,\ g,\ \psi_g \rightarrow [0, 1]$ is the grasp initiation set, a probabilistic classifier conditioned on visual data v that maps end-effector poses g and grasp parameters ψ_g to the probability with which executing π_c from grasp g terminates in β_c (success) as opposed to B (failure). During learning, selecting promising grasps is formulated as a bandit problem.

- h and t are the head and the tail, motion planners that control the robot through free space to achieve a grasp generated by I_c, and extract it from contact back into free space—or into the head of another skill—after termination. These serve to establish and break contact, and to sequence skills: the tail of one skill simply becomes the head of another.

- $\Gamma : g,\ \psi_g \rightarrow \tau_g$ is a grasp controller parameterized by grasp pose g, sampled from the initiation set, and grasp parameters ψ_g (e.g. grasp type, force), and outputting motor commands for the gripper τ_g.

For most tasks, we envision that all the skill components are given or designed except π_c and I_c, which leads to a problem of jointly learning a policy and affordance model for functional grasping. The CIP model structures the motor skill learning problem so that: only motor control involving contact with the object is learned, and free-space motion is generated using a planner; interaction with an object is always safe; and motion planning is used for the remainder of motor control, especially to stitch motor skills together. At the same time, the components that must be learned offer

natural opportunities for incorporating powerful deep network methods to learn rich sensorimotor policies. The result will be small, isolated pockets of motor skill learning connected by much longer trajectories generated by a motion planner. Note that the CIP model does not assume access to a simulator or dynamics model of the environment.

7.2.1 Instantiating CIPs

One benefit of the CIP framework is that its different components may be chosen to match the robot hardware it is being instantiated on. We now detail our specific choices of component instantiations used in the experiments (described in Section 7.3) as an illustrative example.

Motor control policy π_c Sensor input from the touch sensors on the robot's grippers, the joint and Cartesian state of the robot, and object joint state are fed into a neural network policy. For the action space, we chose to have the robot command the end-effector in Cartesian space while remaining compliant to promote sample-efficiency and safety during sustained contact. We therefore selected the Variable Impedance Control in End-Effector Space (Martín-Martín et al., 2019) scheme as our action space. Recall motor policy π_c maps sensor readings ϕ to a desired delta end-effector position $\Delta^{pos} = (p_d - p)$ and rotation $\Delta^{ori} = R_d \ominus R$, as well as commanded stiffness terms $k^p_{pos} \in \mathbb{R}^{3\times3}$ and $k^p_{ori} \in \mathbb{R}^{3\times3}$ for position and rotation respectively. These terms are then used to directly map to joint torques τ as before. The resulting action space therefore consists of 13 dimensions: six for end-effector pose, six parameterizing the diagonal of the stiffness matrices, and one for the state of the parallel-jaw gripper.

Safety envelope σ We limit the maximum value of stiffness parameters k^{pos}_p and k^{ori}_p, so that the robot remains compliant and does not generate high torque values when it contacts the object. In addition, the torques are clipped if they exceed the allowed range. We use a two-fold strategy to prevent joint limit violations, with two threshold parameters, σ_1 and σ_2 ($\sigma_1 > \sigma_2$), that check how close the robot joints are to its limits. If a joint position θ_i exceeds its threshold σ_1, we switch to a null-space controller (Khatib, 1987) that attempts to move θ_i away from its limit without changing the end-effector pose. If the robot nonetheless exceeds σ_2 at joint index i (e.g. due to a high enough initial velocity to overcome the null-space controller), the controller generates a torque in the opposite direction for θ_i until it returns to a safe configuration.

Task-specific success indicator β_c These were designed by hand for each task, and return true when the object's joint states are above a threshold position. In principle, they could be learned from data.

Task-general failure classifier B In our case, B simply served as a joint limit safety check: if the robot is within 5 degrees of its joint limits, the classifier returns true and ends the learning episode. Episodes are also terminated if the agent loses contact with the object for several timesteps.

Grasp initiation set I_c In each case, the visual data v is represented as a point cloud of the scene, which is segmented to only include the part of the object that the robot should manipulate. An existing task-general grasp generator by ten Pas et al. (2017) is used to sample a set of grasp poses G based on the normals calculated from the point cloud. Each grasp $g \in G$ is then checked for reachability and collision. Grasps g which pass these checks are added to a list of acceptable grasp poses that define the domain of I_c.

To sample a grasp pose $g \in I_c$ for the head h during learning, we cast grasp sampling as a bandit problem that is solved with Upper Confidence Bounds (UCB) (Kocsis and Szepesvári, 2006) where Q-values are task success rates. We therefore treat learning I_c and sampling grasp poses as an active learning problem, and use UCB since it appropriately balances exploration and exploitation.

Once a grasp pose g is sampled, we obtain a suitable joint configuration θ by optimizing manipulability. A manipulability score is computed for a joint configuration θ as the product of two values: 1) the manipulability index introduced by Yoshikawa (1985) that analyses the volume of the manipulability ellipsoid: $w = \sqrt{det(JJ^T)}$ where J is the Jacobian for a particular joint configuration θ, and 2) a penalization term introduced by Tsai (1986) based on the distance to the upper and lower joint limits for a particular joint configuration θ,

$$P(\theta) = 1 - exp(-k \prod_{j=1}^{n} \frac{(\theta_j - l_j^-)(l_j^+ - \theta_j)}{(l_j^+ - l_j^-)^2}), \tag{7.1}$$

where l_j^- and l_j^+ are the lower and upper joint limits for joint j. These two metrics capture for a joint configuration θ how close the robot's end-effector is to a singularity and how close the robot's joints are to joint limits, respectively.

Motion planners h and t These were instantiated for each domain using the IKFlow inverse kinematics solver (Ames et al., 2022) and MoveIt! (Coleman et al., 2014) to move the robot to the grasp pose sampled from the grasp initiation set I_c.

Grasp Controller Γ The implementation of the grasping controller depends heavily on the morphology of the gripper. For simple parallel jaw grippers the parameterization is simply the opening state of the gripper. For more complex hands, e.g. with three or more fingers, the grasping controller may select among power and pinch grasps (Corsaro et al., 2021).

Algorithm 3 below describes the process of learning a CIP for an object manipulation task.

Algorithm 3 Learning a CIP

Require: CIP $C = (\text{policy } \pi_c \text{ with parameters } \psi, \text{safety envelope } \sigma, \text{termination classifier } \beta_c, \text{failure classifier } B, \text{initiation set classifier } I_c, \text{head } h, \text{tail } t, \text{grasp controller } \Gamma)$, visual observation v, initial joint configuration θ_0

$v \leftarrow$ observe_scene()
$G \leftarrow$ get_grasp_candidates(v)
$I_c(g) \leftarrow 0 \quad \forall g \in G$ ▷ initialize initiation classifier
$D \leftarrow \{\}$ ▷ initialize policy replay buffer D
while not converged **do**:
    $g, \psi_g \leftarrow$ select_grasp(I_c, G) ▷ UCB for grasp selection
    $\Theta \leftarrow$ get_ik_solns(g)
    $\theta \leftarrow$ optimize_manipulability(Θ)
    execute_motion_plan(θ, θ_0) ▷ head of CIP h
    execute_grasp(Γ, g, ψ_g)
    obs, actions, rewards, success $\leftarrow$ rollout(π_c, ψ, β_c, B)
    $D \leftarrow D \cup$ (obs, actions, rewards)
    $\psi \leftarrow$ improve_policy(D, π_c, ψ)
    $I_c(g, \psi_g) \leftarrow$ update_success_rate(g, ψ_g, success)
end while

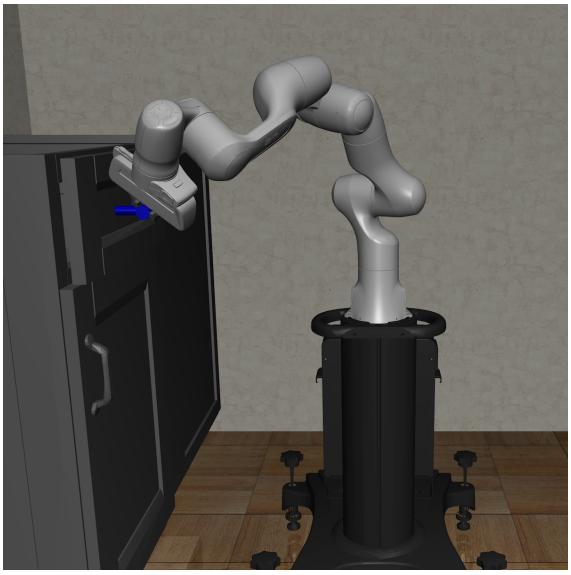
7.3 Experiments

We evaluate the CIP framework in simulation using Robosuite (Zhu et al., 2020b). We conducted experiments on four different articulated object tasks: opening a door, opening a drawer, sliding a knob, and lifting a lever. The position and orientation of the object in each episode is randomized over a small range as in the original benchmark.

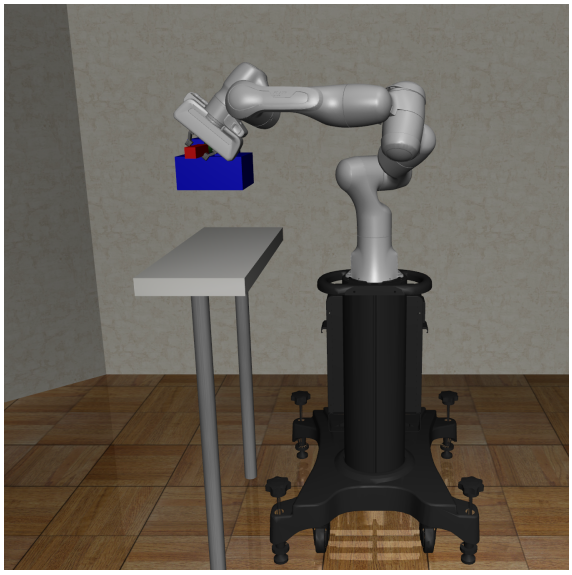
The observations consist of the state of the object, position and velocities of the robot's joints, end-effector pose, and tactile readings from the force sensors at the robot's gripper. We use TD3 (Fujimoto et al., 2018) as our actor-critic method. The reward functions are dense as a function

(a) Open door task

(b) Slide knob task



(c) Open drawer task



(d) Flip lever task

Figure 7.2: Simulation Task Environments.

of progress toward the object goal joint state, which leverages potential-based reward shaping (Ng et al., 1999) to ensure the optimal policy is not changed compared to the sparse reward setting based on success.

To incorporate exploration during learning, we add Gaussian noise parameterized with 0 mean and 0.05 standard deviation to the policy. We use the Adam optimizer with a learning rate set to 0.0001. The neural network policies are two-layered multi-layered perceptrons (MLP) with 64

hidden nodes.

Training episodes are ran for a maximum of 250 steps each, and trained for a total of 500 training episodes. We evaluate performance every 10 training episodes with 20 policy roll-outs.

The safety envelope parameters σ_1, σ_2 were set to 0.3, 0.2 rad respectively and were determined empirically (relative to the joint limit violation threshold 0.1 rad).

We consider two evaluation metrics: 1) **Task Success Rate**, which measures how successful the policy is at manipulating the object, and 2) **Joint Limit Violation Rate**, which is a proxy measure for how safe the policy is. To analyze how each of the structures of CIP impact these metrics, we run ablations that incrementally include structure described in Section 7.2.1:

1. **E2E**: This setting is an end-to-end baseline that incorporates none of the CIP structure. The robot begins in a home pose with no contact to the object, and must learn a complete policy for moving to the object and manipulating it.

2. **Head**: This baseline incorporates the head h structure of the CIP, but no initiation set learning—the robot samples grasp poses randomly and uses naive IK to reach them.

3. **Safety**: This baseline extends **Head** to additionally incorporate the safety envelope.

4. **Manipulability Value (MV)**: This baseline extends the **Safety** setting, and additionally incorporates the manipulability value into the sampling approach for I_c, which adds additional structure on top of the **Head** approach. After sampling a random grasp, we sample a set of inverse kinematics solutions and select the one with the highest manipulability value.

5. **CIP**: Incorporates all the structure of the CIP; extends the **MV** approach to additionally perform active learning with UCB over grasp poses.

6. **CIP+BC**: Ten demonstrations from an expert policy are provided to the **CIP** learner and incorporated into policy search using the behavior cloning loss and Q-Filtering (Nair et al., 2018).

7. **E2E+BC**: Ten demonstrations from an expert policy are provided to the **E2E** learner.

Results The results for all our experiments can be found in Figures 7.3 and 7.4, where we show the best-to-date performance for both metrics across all the tasks. Across all the tasks, the **E2E**

baseline is unable to learn any meaningful policy, and also has many joint violation rates throughout learning. This is expected as exploration is extremely challenging in the absence of a strong reward signal for reaching the object and making contact. We also see that once the head of the CIP is incorporated (**Head**), the agent is able to start achieving some amount of task success, but still encounters many joint state violations throughout the learning process. The **Safety** baseline is able to achieve a task success rate on par with **Head**, but significantly reduces the joint state violation rates during learning. The **MV** baseline has improved task success over the **Head** baseline, which demonstrates the usefulness of incorporating the manipulability value when selecting joint configurations for sustained-contact manipulation tasks, but still has trouble learning an effective policy for the Lever and Drawer task in a small number of training episodes. Once the full structure of the CIP is incorporated (**CIP**), the agent is able to rapidly learn a policy with a high success rate (at least an average of 80%) within hundreds of training episodes. When demonstrations are available (**CIP+BC**) we see rapid, safe learning within tens of episodes. Note that **CIP+BC** outperforms the end-to-end agent with demonstrations (**E2E+BC**). We hypothesize that the Drawer task is challenging due to the relatively low manipulability of grasps on the object. These results demonstrate that each structural component of the CIP is useful for promoting safe and efficient learning across a diverse set of sustained-contact manipulation tasks.

7.4 Skill Composition Demonstration on Hardware

One of the advantages of the CIP structure is it enables zero-shot composition by construction. The motion planning performed via the head h and tail t, together with learned initiation sets I_c, enable a robot to learn sustained-contact manipulation skills in isolation, and then sequentially execute the skills with no additional learning necessary. We validate our approach by learning and sequencing two manipulation skills—opening a cabinet door and pulling a drawer open—on a KUKA LBR iiwa7 with a Schunk Dexterous Hand 3-fingered gripper as shown in Figure 7.5. The sequence is determined a priori by an expert but could be computed using off-the-shelf task planning methods. First, the the cabinet door is opened by sampling a grasp, planning to it, and running the policy. The robot releases the cabinet handle, then plans to the drawer grasp. Then the drawer pulling skill is used to open the drawer.

For each skill, we produce a set of possible pinch grasps by collecting a set of grasp proposals on

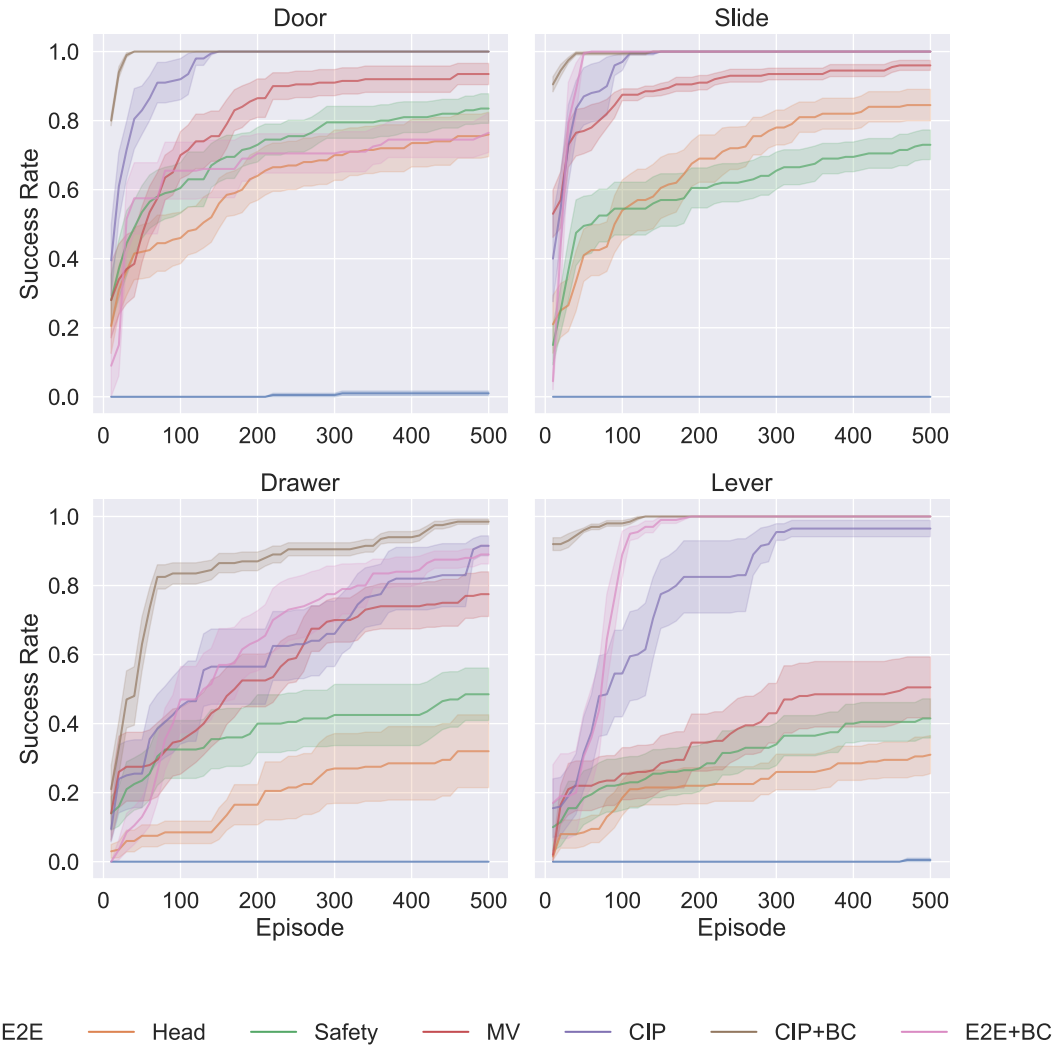


Figure 7.3: Task success rates vs. the number of training episodes. The shaded region around the average shows the standard error over 10 seeds.

the object's handle using the grasp pose generator of ten Pas et al. (2017). This set of grasps is filtered for collision, IK feasibility, and successful contact with the handle. We provide 3-5 demonstrations by manually guiding the arm through the task on different grasps, which we use to train the skill on the robot using behavior cloning and policy search.

Given a grasp proposed by the UCB sampler, we sample multiple possible IK solutions and select the one with the highest manipulability index as described in section 7.2.1. The grasp is executed and the policy execution begins. The observation space consists of the Cartesian position and orientation of the robot end effector, tactile readings from each of 6 touch sensors on the gripper's fingers, and

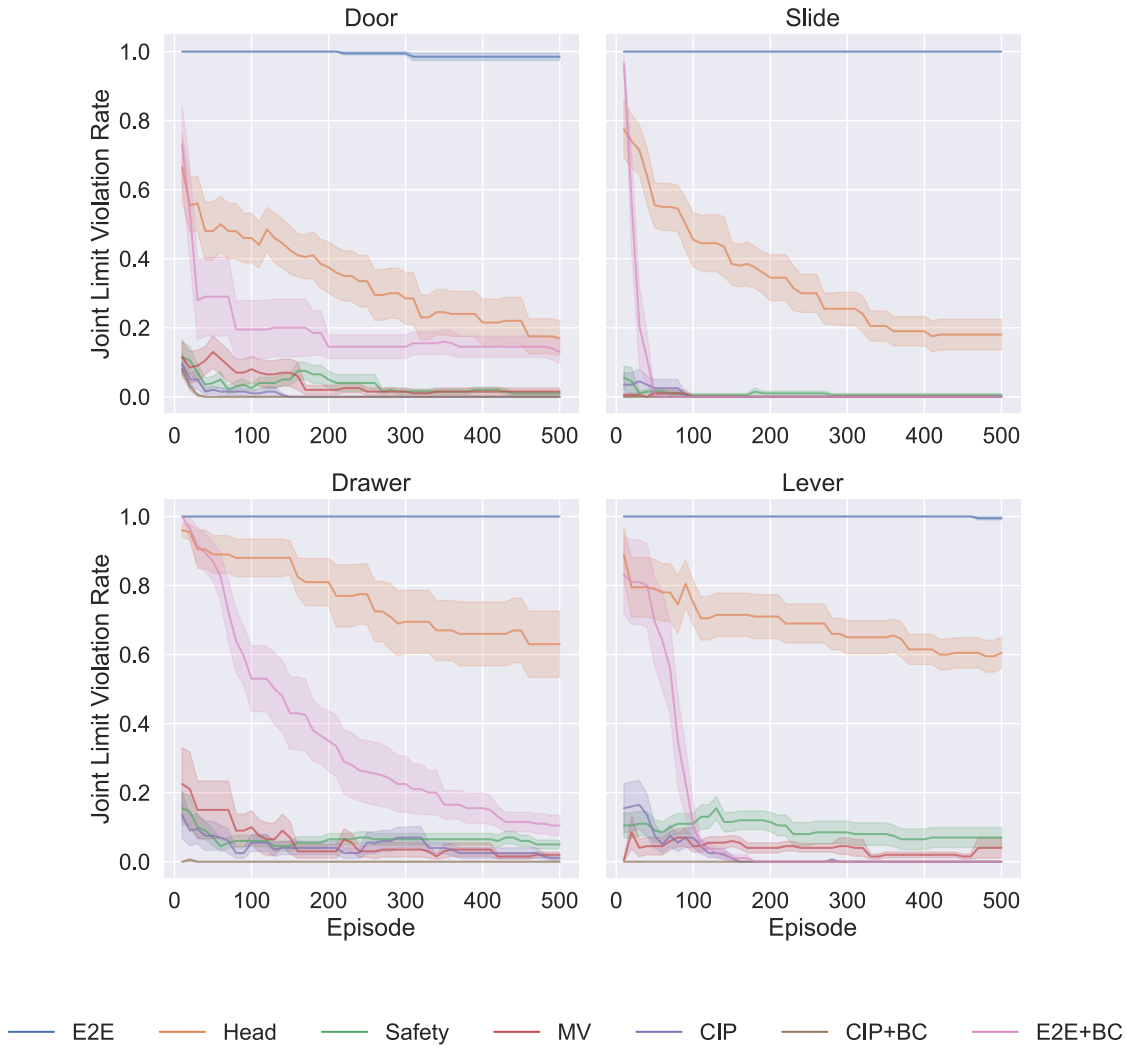


Figure 7.4: Joint limit violation rates for simulated tasks.

the state of the door or drawer. The actions consist of Cartesian displacements in position and are executed using the iiwa's Cartesian impedance functionality. A shaped reward is provided as in the simulation experiments as a function of object state tracked using `ar_track_alvar`.

We used the iiwa's native safety controller for detecting collisions, constraining joint limits, and limiting forces exerted on or by the arm, in conjunction with an additional orientation constraint limiting the gripper from large rotations along the plane of the wrist after grasping, to protect the gripper and objects.

As shown in Figure 7.5, when trained with behavior cloning and policy search, the robot is able to successfully learn and execute each skill, and to compose the two skills and execute them in

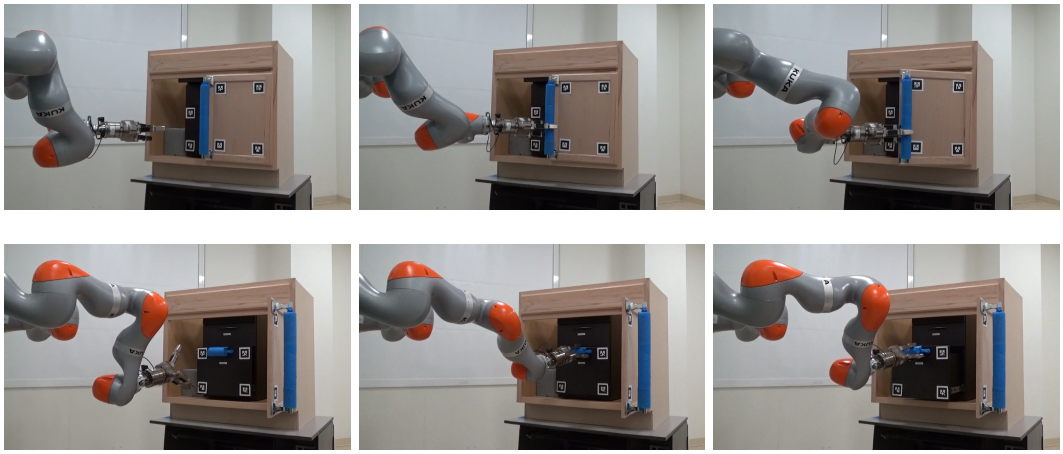


Figure 7.5: Two CIPs executed in succession on robot hardware.

sequence.

7.5 Conclusion

We propose a new policy class for sustained-contact manipulation skills: Composable Interaction Primitives (CIPs). CIPs are designed to exploit readily-accessible structure in the world and structure in the robot to enable sample-efficient and safe policy learning, and be easily leveraged by high-level planners due to their sequential composability via motion planning. Future work will investigate efficient methods to learn effect models to autonomously construct a symbolic vocabulary to support integrating CIPs with a high-level task planner.

Chapter 8

Conclusion

To build useful robots, and to solve the grand challenge of artificial intelligence, we must equip our robots with the ability to quickly learn new tasks on the fly. The novelty and complexity of unstructured home and industry environments preclude us from pre-programming motor skills; the scarcity of real robotic data forces us to cleverly scope learning problems of embodied agents for maximum efficiency. In this thesis, we have sought to identify and exploit structure – regularities across the tasks faced by a robot – to build systems capable of learning motor skills in the real world.

First, we introduced the problem of category-level articulated object manipulation, and built a machine learning model capable of inferring object models from individual observations. This enabled a robot to interact with novel articulated objects despite encountering them for the first time. This system exploits structure present in articulated objects as well as patterns in object categories. We then extended this system to more dynamically infer kinematic graphs by leveraging advances in graph neural networks and part segmentation, resulting in a more broadly applicable system for category-level kinematic model estimation. Both of these approaches were the first of their kind. Both represent an example of learning that we can perform prior to sending robots out into the world – an instance of learning deployed at design time in order to make runtime learning of motor skills easier.

We then built a system which did just that – learned motor skills by first using approximate, kinematic models of the world, emulating the process humans employ when faced with a new task. Initial attempts are computed via planning which are then used to initialize motor skill policies for feedback control. This system enables intelligent exploration in a broad class of manipulation

problems, combating the complexity of these contact-rich interactions and autonomously obtaining high-quality data for skill learning.

Finally, we sought to exploit the modularity and compositionality of behavior, designing a novel motor primitive that is efficiently learnable and readily composed to generate abstract plans. We investigated multiple safety controller designs in Chapters 6 and 7, ultimately arriving at a simple solution that enables real world skill learning with minimal damage to the robot and environment.

The experimental evidence presented throughout give credence the main claim of the thesis:

> *By exploiting structure present across tasks faced by a robotic agent, we may imbue our learning systems with greater data efficiency and generalization.*

8.1 Future Directions

In the era of multi-billion-parameter language models, this dissertation advocates for the view that not all of behavior will emerge from big data and big models. Scale alone has revolutionized the fields of computer vision and natural language, and it is tempting to believe the same will be true of robotics. But obtaining data of real robots interacting with the world, at the scale required to reliably learn behavior, is impossible. This thesis instead suggests that we need to build upon decades of research in robotics and AI, interweaving mature techniques with machine learning and the data that we can generate.

Many such questions of structure remain. One is the development of more general approximate models of the world — a structural ingredient necessitated by the novelty of life in human environments. Our work in Chapters 3, 4, and 5 illustrates the utility of such models for manipulating articulated rigid bodies which can be modeled kinematically. But, of course, many classes of objects do not fit this mold. Deformable objects, liquids, and particulate matter evade such a description; tool use is typically forceful and would be difficult to capture with a purely kinematic model. As such, there is a need for a broader class of dynamics models that capture forces required to manipulate the world, rather than just kinematics, unifying these seemingly disparate object categories. Here, there is an opportunity to leverage scale in the form of video data; the critical questions are how one might repurpose that data for control, and what role these kinematic models might play. Should these models include the notions of objects? Object parts? Or will these concepts emerge from sufficient data? Articulated object modeling itself will undoubtedly continue to be explored

because of its practical utility; interesting directions include the incorporation of neural fields to recover geometry and the unification of affordance-based and traditional graph-based kinematics estimation.

On the control side, there is still much work to do in order to rein in the power of deep learning. The critical structural question of *policy representation* still largely persists. Chapters 6 and 7 illustrate the role of traditional robotics primitives like motion planning, grasping, and collision avoidance; the challenge still remains to find appropriate representations of the feedback policy responsible for control when in contact with the world. Human motions are generally smooth and stable, but generically applying deep networks by no means provides these properties. It remains an open challenge to rectify dynamical systems and lower-dimensional action spaces with the expressivity of neural networks necessary to handle high dimensional inputs like vision and tactile data. There are many exciting works in this area to expand upon and integrate with the CIPs framework, e.g. Sharma et al. (2021); Allshire et al. (2021); Xie et al. (2023); Wang et al. (2023). Finally, integrating these systems with task-level planning remains an open challenge. The CIPs framework illustrates that the structure of reaching and grasping should be reflected in our motor primitives. How we reconcile this with approximate forward models, discussed above, and the role of *skills* in such a context, are still unclear. Moreovever, how we should approach the daunting complexity of task-and-motion planning remains to be pinned down. Here, common sense reasoning displayed by large language models may well serve as a useful heuristic.

Such questions are suggested and made possible by the work presented in this thesis. Even in the era of scale, the future of exploiting structure in robotic manipulation is bright, indeed.

Appendix A

Synthetic Dataset

A.1 Synthetic Dataset

Our simulated dataset of procedurally generated articulated objects (available here) consists of 6 categories (cabinet, drawer, microwave, toaster oven, two-door cabinet, refrigerator). The quantities which were randomized during data generation and their ranges are displayed in the tables below.

Geometric parameters were sampled from independent, uniform distributions over depth, width, and height. Cabinets that open clockwise/counterclockwise were generated with equal probability.

Table A.1: Randomized Geometric Parameters

	Depth (cm)	Width (cm)	Height (cm)
Cabinet	56 - 64	60 - 140	60 - 140
Drawer	45 - 60	30 - 76	10 - 30
Microwave	25 - 56	40 - 76	22 - 45
Toaster	20 - 50	40 - 60	20 - 50
Cabinet2	56 - 64	60 - 140	60 - 140
Refrigerator	60 - 80	60 - 90	82 - 88

Pose parameters were sampled similarly for each object. The camera was positioned looking down the positive x-axis with the positive z-axis upward. The x,y,z values refer to the center of the base of the object. Rotation denotes angle about the z-axis. Configuration denotes articulated pose, reported in radians for revolute mechanisms and centimeters for prismatic joints. For 2-DoF objects, configuration was randomized to cover the full 2-dimensional configuration space in the

range provided.

Table A.2: Randomized Pose Parameters

	x (m)	y (m)	z (m)	Rotation (rad)	Configuration
Cabinet	[1.0, 2.0]	[-0.5, 0.5]	[-0.7, 0.3]	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$[0, \frac{\pi}{2}]$ rad
Drawer	[1.0, 2.0]	[-0.5, 0.5]	[-0.8, 0.0]	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	[0, 40] cm
Microwave	[1.0, 2.0]	[-0.5, 0.5]	[-0.7, 0.3]	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$[0, \frac{\pi}{2}]$ rad
Toaster	[1.0, 2.0]	[-0.5, 0.5]	[-0.7, 0.3]	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$[0, \frac{\pi}{2}]$ rad
Cabinet2	[1.0, 2.0]	[-0.5, 0.5]	[-0.7, 0.3]	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$[0, \frac{\pi}{2}]$ rad
Refrigerator	[1.5, 3.5]	[-1.0, 1.0]	[-1.5, -0.7]	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	$[0, \frac{\pi}{2}]$ rad

Bibliography

B. Abbatematteo, S. Tellex, and G. Konidaris. Learning to generalize kinematic models to novel objects. In *Proceedings of the Third Conference on Robot Learning*, 2019.

B. Abbatematteo, E. Rosen, S. Tellex, and G. Konidaris. Bootstrapping motor skill learning with motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4926–4933. IEEE, 2021.

F. J. Abu-Dakka and M. Saveriano. Variable impedance control and learning—a review. *Frontiers in Robotics and AI*, page 177, 2020.

J. Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

M. T. Akbulut, U. Bozdogan, A. Tekden, and E. Ugur. Reward conditioned neural movement primitives for population-based variational policy optimization. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10808–10814, 2021a.

M. Akbulut, E. Oztop, M. Y. Seker, H. Xue, A. Tekden, and E. Ugur. Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing. In J. Kober, F. Ramos, and C. Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1896–1907. PMLR, 16–18 Nov 2021b.

A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, and A. Garg. Laser: Learning a latent action space for efficient reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6650–6656. IEEE, 2021.

B. Ames, J. Morgan, and G. Konidaris. Ikflow: Generating diverse inverse kinematics solutions. *IEEE Robotics and Automation Letters*, 7(3):7177–7184, 2022.

O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 12–20, 1997.

P. R. Barragän, L. P. Kaelbling, and T. Lozano-Pérez. Interactive bayesian identification of kinematic mechanisms. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2013–2020. IEEE, 2014.

M. Baum, M. Bernstein, R. Martin-Martin, S. Höfer, J. Kulick, M. Toussaint, A. Kacelnik, and O. Brock. Opening a lockbox through physical exploration. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 461–467, Nov 2017. doi: 10.1109/HUMANOIDS.2017.8246913.

D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner. Manipulation planning on constraint manifolds. In *2009 IEEE international conference on robotics and automation*, pages 625–632. IEEE, 2009.

F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

C. M. Bishop. Mixture density networks. 1994.

J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on robotics*, 30(2):289–309, 2013.

J. Brookshire and S. Teller. Articulated pose estimation using tangent space approximations. *The International Journal of Robotics Research*, 35(1-3):5–29, 2016.

D. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

B. Burchfiel and G. Konidaris. Hybrid bayesian eigenobjects: Combining linear subspace and deep network methods for 3d robot vision. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6843–6850. IEEE, 2018.

R. Burridge, A. Rizzi, and D. Koditschek. Sequential composition of dynamically dextrous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555, 1999.

A. Bylard, R. Bonalli, and M. Pavone. Composable geometric motion policies using multi-task pullback bundle dynamical systems. In *Proc. 2021 IEEE International Conference on Robotics and Automation*, pages 7464–7470, 2021. doi: 10.1109/ICRA48506.2021.9561320.

A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L.-J. C. L'Ecuyer. Kinova modular robot arms for service robotics applications. In *Rapid Automation: Concepts, Methodologies, Tools, and Applications*, pages 693–719. IGI Global, 2019.

Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine. Path integral guided policy search. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3381–3388. IEEE, 2017.

X. Chen and S. Sankaranarayanan. Reachability analysis for cyber-physical systems: Are we there yet? In *NASA Formal Methods Symposium*, pages 109–130. Springer, 2022.

Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang. Bae-net: Branched autoencoder for shape co-segmentation. *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff. Rmpflow: A geometric framework for generation of multitask motion policies. *IEEE Transactions on Automation Science and Engineering*, 18(3):968–987, 2021.

C.-A. Cheng, X. Yan, N. Wagener, and B. Boots. Fast Policy Learning through Imitation and Reinforcement. *arXiv e-prints*, art. arXiv:1805.10413, May 2018.

R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proc. AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

J. A. Coelho Jr and R. A. Grupen. A control basis for learning multifingered grasps. *Journal of Robotic Systems*, 14(7):545–557, 1997.

D. Coleman, I. Sucan, S. Chitta, and N. Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *Journal of Software Engineering for Robotics*, 5(1):3–16, May 2014.

M. Corsaro, S. Tellex, and G. Konidaris. Learning to detect multi-modal grasps for dexterous grasping in dense clutter. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4647–4653. IEEE, 2021.

E. Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 15(49):5, 2013.

A. F. Daniele, T. M. Howard, and M. R. Walter. Learning articulated object models from language and vision. 2017.

M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

B. Deng, J. P. Lewis, T. Jeruzalski, G. Pons-Moll, G. Hinton, M. Norouzi, and A. Tagliasacchi. Nasa: neural articulated shape approximation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 612–628. Springer, 2020.

K. Desingh, S. Lu, A. Opipari, and O. C. Jenkins. Efficient nonparametric belief propagation for pose estimation and manipulation of articulated objects. *Science Robotics*, 4(30), 2019. doi: 10.1126/scirobotics.aaw4523. URL `https://robotics.sciencemag.org/content/4/30/eaaw4523`.

F. Diehl. Edge contraction pooling for graph neural networks. *arXiv preprint arXiv:1905.10990*, 2019.

C. Diuk, A. Cohen, and M. L. Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247. ACM, 2008.

D. Driess, I. Schubert, P. Florence, Y. Li, and M. Toussaint. Reinforcement learning with neural radiance fields. *Advances in Neural Information Processing Systems*, 35:16931–16945, 2022.

B. Eisner, H. Zhang, and D. Held. Flowbot3d: Learning 3d articulation flow to manipulate articulated objects. In *Robotics: Science and Systems (RSS)*, 2022.

P. Englert and M. Toussaint. Kinematic morphing networks for manipulation skill transfer. In *Proceedings of the IEEE International Conference on Intelligent Robotics Systems*, 2018.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017.

T. Flash and B. Hochner. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology*, 15(6):660–666, 2005.

S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

S. Y. Gadre, K. Ehsani, and S. Song. Act the part: Learning interaction strategies for articulated object part discovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15752–15761, 2021.

J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in neural information processing systems*, 31, 2018.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

A. Handa, K. Van Wyk, W. Yang, J. Liang, Y. W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *Proc. 2020 IEEE International Conference on Robotics and Automation*, pages 9164–9170, 2020. doi: 10.1109/ICRA40945.2020.9197124.

K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme. Active articulation model estimation through interactive perception. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3305–3312. IEEE, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

N. Heppert, T. Migimatsu, B. Yi, C. Chen, and J. Bohg. Category-independent articulated object tracking with factor graphs. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3800–3807. IEEE, 2022.

J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

N. Hogan. Impedance control: An approach to manipulation: Part ii—implementation. 1985.

C.-C. Hsu, Z. Jiang, and Y. Zhu. Ditto in the house: Building articulation models of indoor scenes through interactive perception. *arXiv preprint arXiv:2302.01295*, 2023.

E. Huang, X. Cheng, Y. Mao, A. Gupta, and M. T. Mason. Autogenerated manipulation primitives. *The International Journal of Robotics Research*, page 02783649231170897, 2023.

L. Huo, Z. Wang, M. Xu, and Y. Song. Learning diverse sub-policies via a task-agnostic regularization on action distributions. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3932–3936. IEEE, 2020.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems*, 15, 2002.

A. Jain and S. Niekum. Learning hybrid object kinematics for efficient hierarchical planning under uncertainty. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5253–5260. IEEE, 2020.

A. Jain, R. Lioutikov, C. Chuck, and S. Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13670–13677. IEEE, 2021.

A. Jain, S. Giguere, R. Lioutikov, and S. Niekum. Distributional depth-based estimation of object articulation models. In *Conference on Robot Learning*, pages 1611–1621. PMLR, 2022.

Y. Jiang, F. Yang, S. Zhang, and P. Stone. Task-motion planning with reinforcement learning for adaptable mobile service robots. In *IROS*, pages 7529–7534, 2019.

Z. Jiang, C.-C. Hsu, and Y. Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5616–5626, 2022.

T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

T. Jurgenson and A. Tamar. Harnessing reinforcement learning for neural motion planning. *arXiv preprint arXiv:1906.00214*, 2019.

S. M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal. Learning force control policies for compliant manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4639–4644. IEEE, 2011.

D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018. doi: 10.1109/LRA.2018.2795645.

S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

D. Katz and O. Brock. Manipulating articulated objects with interactive perception. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 272–277. IEEE, 2008.

Y. Kawana, Y. Mukuta, and T. Harada. Unsupervised pose-aware part decomposition for man-made articulated objects. In *European Conference on Computer Vision*, pages 558–575. Springer, 2022.

S. A. Khader, H. Yin, P. Falco, and D. Kragic. Stability-guaranteed reinforcement learning for contact-rich manipulation. *IEEE Robotics and Automation Letters*, 6(1):1–8, 2021. doi: 10.1109/LRA.2020.3028529.

O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. doi: 10.1109/JRA.1987.1087068.

O. Khatib. Inertial properties in robotic manipulation: An object-level framework. *The International Journal of Robotics Research*, 14(1):19–36, 1995. doi: 10.1177/027836499501400103.

A. Khazatsky, A. Nair, D. Jing, and S. Levine. What can i do here? learning new skills by imagining visual affordances. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14291–14297. IEEE, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2): 171–203, 2010.

L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, pages 282–293, 2006.

G. Konidaris and A. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems 22*, pages 1015–1023, 2009.

L. Koutras and Z. Doulgeri. A correct formulation for the orientation dynamic movement primitives for robot control in the cartesian space. In *Conference on robot learning*, pages 293–302. PMLR, 2020.

O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 22(30):1–82, 2021.

J. Kulick, S. Otte, and M. Toussaint. Active exploration of joint dependency structures. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2598–2604. IEEE, 2015.

A. Kurenkov, A. Mandlekar, R. Martin-Martin, S. Savarese, and A. Garg. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. *arXiv preprint arXiv:1909.04121*, 2019.

M. Labbé and F. Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36 (2):416–446, 2019.

M. Landau. Kinect Infrared (IR) and Depth Image Simulator. MATLAB Central, 2017.

S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

M. A. Lee, C. Florensa, J. Tremblay, N. Ratliff, A. Garg, F. Ramos, and D. Fox. Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient

policy learning. In *Proc. 2020 IEEE International Conference on Robotics and Automation*, pages 7505–7512, 2020. doi: 10.1109/ICRA40945.2020.9197125.

J. Lei, C. Deng, B. Shen, L. Guibas, and K. Daniilidis. Nap: Neural 3d articulation prior. *arXiv preprint arXiv:2305.16315*, 2023.

S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

A. Li, C.-A. Cheng, M. A. Rana, M. Xie, K. Van Wyk, N. Ratliff, and B. Boots. RMP2: A Structured Composable Policy Class for Robot Learning. In *Proc. Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.092.

X. Li, H. Wang, L. Yi, L. Guibas, A. L. Abbott, and S. Song. Category-level articulated object pose estimation. In *CVPR*, 2020.

J. Liang, X. Cheng, and O. Kroemer. Learning preconditions of hybrid force-velocity controllers for contact-rich manipulation. *arXiv preprint arXiv:2206.12728*, 2022.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

T. Lozano-Perez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(10):681–698, 1981.

T. Lozano-Perez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, 1984.

J. Luh, M. Walker, and R. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, 1980. doi: 10.1109/TAC.1980.1102367.

J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

Z. Manchester and S. Kuindersma. Variational contact-implicit trajectory optimization. In *Robotics Research: The 18th International Symposium ISRR*, pages 985–1000. Springer, 2020.

R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1010–1017. IEEE, 2019.

R. Martín-Martin and O. Brock. Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2494–2501, Sept 2014. doi: 10.1109/IROS.2014.6942902.

M. T. Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28, 2018.

M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg. Articulated object interaction in unknown scenes with whole-body mobile manipulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1647–1654. IEEE, 2022.

K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

H. Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

J. Mu, W. Qiu, A. Kortylewski, A. Yuille, N. Vasconcelos, and X. Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021.

K. Muelling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 411–416. IEEE, 2010.

K. Muelling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

M. Mukadam, C.-A. Cheng, D. Fox, B. Boots, and N. Ratliff. Riemannian motion policy fusion through learnable lyapunov function reshaping. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proc. Conference on Robot Learning*, volume 100 of *Proc. Machine Learning Research*, pages 204–219. PMLR, 30 Oct–01 Nov 2020.

O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2146–2153. IEEE, 2017.

A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987. doi: 10.1177/027836498700600201.

A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287, 1999.

N. Nie, S. Y. Gadre, K. Ehsani, and S. Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arXiv preprint arXiv:2207.08997*, 2022.

S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246. IEEE, 2012.

S. Niekum, S. Osentoski, C. G. Atkeson, and A. G. Barto. Online bayesian changepoint detection for articulated motion models. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1468–1475. IEEE, 2015.

Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017.

A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

S. Pillai, M. R. Walter, and S. Teller. Learning articulated motions from visual demonstration. *arXiv preprint arXiv:1502.01659*, 2015.

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

R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017a.

C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Neural Information Processing Systems*, 2017b.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

N. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

F. Sadeghi and S. Levine. CAD2RL: Real single-image flight without a single real image. In *Robotics: Science and Systems XIII*, 2016.

M. Saggar, T. D'Silva, N. Kohl, and P. Stone. Autonomous learning of stable quadruped locomotion. In *RoboCup 2006: Robot Soccer World Cup X 10*, pages 98–109. Springer, 2007.

J. K. Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *1980 19th IEEE conference on decision and control including the symposium on adaptive processes*, pages 95–100. IEEE, 1980.

M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel. Dynamic movement primitives in robotics: A tutorial survey. *ArXiv*, abs/2102.03861, 2021.

S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6): 233–242, 1999.

S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural computation*, 10(8):2047–2084, 1998.

S. Schaal, C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

G. Schiavi, P. Wulkop, G. Rizzi, L. Ott, R. Siegwart, and J. J. Chung. Learning agent-aware affordances for closed-loop interaction with articulated objects. *arXiv preprint arXiv:2209.05802*, 2022.

M. Y. Seker, M. Imre, J. Piater, and E. Ugur. Conditional neural movement primitives. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

L. Sentis and O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(04):505–518, 2005.

M. Sharma and O. Kroemer. Efficiently learning manipulations by selecting structured skill representations. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1039–1046. IEEE, 2022.

M. Sharma, J. Liang, J. Zhao, A. Lagrassa, and O. Kroemer. Learning to compose hierarchical object-centric controllers for robotic manipulation. In *Conference on Robot Learning*, pages 822–844. PMLR, 2021.

S. Shaw, B. Abbatematteo, and G. Konidaris. Rmps for safe impedance control in contact-rich manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2707–2713, 2022.

T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic skills for bilevel planning. In *Conference on Robot Learning*, pages 701–714. PMLR, 2023.

F. Stulp and O. Sigaud. Path integral policy improvement with covariance matrix adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 0–0, 2012.

J. Sturm, A. Jain, C. Stachniss, C. C. Kemp, and W. Burgard. Operating articulated objects based on experience. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2739–2744. IEEE, 2010.

J. Sturm, C. Stachniss, and W. Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, 41:477–526, 2011.

W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3309–3318. JMLR. org, 2017.

J. Sung, S. H. Jin, and A. Saxena. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Robotics Research*, pages 701–720. Springer, 2018.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

R. Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. In *Robotics: Science and Systems V*, pages 18–24, 2009.

R. Tedrake. *Robotic Manipulation*. 2022. URL `http://manipulation.mit.edu`.

A. ten Pas and R. Platt. Using geometry to detect grasp poses in 3D point clouds. In *Proceedings of the International Symposium on Robotics Research*, 2015.

A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

K. A. Thoroughman and R. Shadmehr. Learning of action through adaptive combination of motor primitives. *Nature*, 407(6805):742–747, 2000.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

F. Torabi, G. Warnell, and P. Stone. Recent advances in imitation learning from observation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6325–6331. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/882. URL `https://doi.org/10.24963/ijcai.2019/882`.

T. Tosun, E. Mitchell, B. Eisner, J. Huh, B. Lee, D. Lee, V. Isler, H. S. Seung, and D. Lee. Pixels to plans: Learning non-prehensile manipulation by imitating a planner. *arXiv preprint arXiv:1904.03260*, 2019.

M.-J. Tsai. *Workspace Geometric Characterization and Manipulability of Industrial Robots (Kinematics)*. The Ohio State University, 1986.

W.-C. Tseng, H.-J. Liao, L. Yen-Chen, and M. Sun. Cla-nerf: Category-level articulated neural radiance field. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8454–8460. IEEE, 2022.

A. Ude, B. Nemec, T. Petrić, and J. Morimoto. Orientation in cartesian space dynamic movement primitives. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2997–3004. IEEE, 2014.

K. P. Wabersich and M. N. Zeilinger. Linear model predictive safety certification for learning-based control. In *Proc. 2018 IEEE Conference on Decision and Control*, pages 7130–7135, 2018. doi: 10.1109/CDC.2018.8619829.

H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019a.

X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019b.

Y. Wang, N. Figueroa, S. Li, A. Shah, and J. Shah. Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations. In *Conference on Robot Learning*, pages 94–105. PMLR, 2023.

Y. Wang, R. Wu, K. Mo, J. Ke, Q. Fan, L. J. Guibas, and H. Dong. Adaafford: Learning to adapt manipulation affordance for 3d articulated objects via few-shot interactions. In *European Conference on Computer Vision*, pages 90–107. Springer, 2022.

Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40 (6-7):866–894, 2021.

S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

T. Wei and C. Liu. Safe control algorithms using energy functions: A united framework, benchmark, and new directions. In *Proc. 2019 IEEE 58th Conference on Decision and Control*, pages 238–243, 2019. doi: 10.1109/CDC40024.2019.9029720.

Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas. Captra: Category-level pose tracking for rigid and articulated objects from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13209–13218, 2021.

R. Wu, Y. Zhao, K. Mo, Z. Guo, Y. Wang, T. Wu, Q. Fan, X. Chen, L. Guibas, and H. Dong. Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects. *arXiv preprint arXiv:2106.14440*, 2021.

F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

M. Xie, A. Handa, S. Tyree, D. Fox, H. Ravichandar, N. D. Ratliff, and K. Van Wyk. Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration. In *Conference on Robot Learning*, pages 1355–1367. PMLR, 2023.

X. Xu, D. Charatan, S. Raychaudhuri, H. Jiang, M. Heitmann, V. Kim, S. Chaudhuri, M. Savva, A. X. Chang, and D. Ritchie. Motion annotation programs: A scalable approach to annotating kinematic articulations in large 3d shape collections. In *2020 International Conference on 3D Vision (3DV)*, pages 613–622. IEEE, 2020a.

Z. Xu, Y. Zhou, E. Kalogerakis, C. Landreth, and K. Singh. Rignet: Neural rigging for articulated characters. *ACM Trans. on Graphics*, 39, 2020b.

Z. Xu, H. Zhanpeng, and S. Song. Umpnet: Universal manipulation policy network for articulated objects. *IEEE Robotics and Automation Letters*, 2022.

J. Yan and M. Pollefeys. Automatic kinematic chain building from feature trajectories of articulated objects. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 712–719. IEEE, 2006.

Z. Yan, R. Hu, X. Yan, L. Chen, O. van Kaick, H. Zhang, and H. Huang. Rpm-net: Recurrent prediction of motion and parts from point cloud. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA 2019)*, 38(6):240:1–240:15, 2019.

L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas. Deep part induction from articulated object pairs. *Transactions on Graphics*, 37(6), December 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275027. URL `https://doi.org/10.1145/3272127.3275027`.

T. Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985.

A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

V. Zeng, T. E. Lee, J. Liang, and O. Kroemer. Visual identification of articulated object parts. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2443–2450. IEEE, 2021.

G. Zhang, O. Litany, S. Sridhar, and L. Guibas. Strobenet: Category-level multiview reconstruction of articulated objects. *arXiv preprint arXiv:2105.08016*, 2021.

H. Zhang, B. Eisner, and D. Held. Flowbot++: Learning generalized articulated objects manipulation via articulation projection. *arXiv preprint arXiv:2306.12893*, 2023.

C. Zheng, W. Wu, C. Chen, T. Yang, S. Zhu, J. Shen, N. Kehtarnavaz, and M. Shah. Deep learning-based human pose estimation: A survey. *ACM Computing Surveys*, 2020.

C. Zhu, K. Xu, S. Chaudhuri, L. Yi, L. J. Guibas, and H. Zhang. AdaCoSeg: Adaptive shape co-segmentation with group consistency loss. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2020a.

Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020b.