BROWN UNIVERSITY

DOCTORAL DISSERTATION

Abstraction for Autonomous Human-Robot Interaction

Author:

Eric Rosen

Advisor:

Stefanie Tellex and

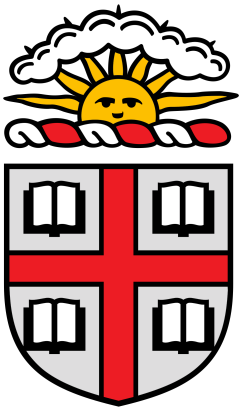
George Konidaris

A dissertation submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

Department of Computer Science



February, 2024

© Copyright 2023 by Eric Rosen

This dissertation by Eric Rosen 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 09/08/2023

Stefanie Tellex, Advisor

Date 09/08/2023

George Konidaris, Advisor

Recommended to the Graduate Council

Date 2023-08-09

James Tompkin, Reader

Date 08-09-2023

Nicholas Roy, Reader

Approved by the Graduate Council

Date _____

Thomas A. Lewis, Dean of the Graduate School

Chapter 1

Curriculum Vitae

Eric Rosen attended Brown University, where he received a Bachelor of Science in Computer Science and Applied Mathematics. At Brown Univerity, he co-created and taught Chore-orobotics 101, the first course investigating the intersection of robotics and choreography. Eric also spent time as a research intern at Uber ATG, Meta Reality Labs, and MERL.

During his PhD, Eric published the following papers: *Synthesizing Navigation Abstractions for Planning with Portable Manipulation Skills* , *Norm Learning with Reward Models from Instructive and Evaluative Feedback*, *Bootstrapping Motor Skill Learning with Motion Planning*, *Mixed Reality as a Bidirectional Communication Interface for Human-Robot Interaction*, *A General Methodology for Teaching Norms to Social Robots*, and *Building Plannable Representations with Mixed Reality*.

Acknowledgements

I would like to thank my committee members, Stefanie Tellex, George Konidaris, James Tompkin, and Nicholas Roy, for all the help and support they have given me over the years on advising my research. I would like to also thank my family (my mother, fathers, sister, and grandparents especially) for all their love and support.

Contents

Abstract **v**

1 Introduction **1**

2 Background **4**

2.1 Robot Motor Skills 4

2.1.1 Portable Skills 5

2.1.2 Motion Planning 8

2.1.3 State Abstractions for Planning 9

2.2 Semantic Maps 10

2.3 Mixed Reality for Human-Robot Interaction 11

3 Composable Interaction Primitives for Sustained-Contact Manipulaton **13**

3.1 Introduction 13

3.1.1 Related Work 14

3.2 Approach 17

3.2.1 Instantiating CIPs 19

3.3 Experiments 21

3.4 Results 23

3.4.1 CIPs for multi-step plan execution 24

3.5 Conclusion 25

4 Bootstrapping Motor Skills with Motion Planning **26**

4.1 Introduction 26

4.1.1 Related Work 28

4.2 Bootstrapping Skills with Motion Planning . . . 29
4.2.1 Fitting a Policy to a Demonstration . . . 33
4.2.2 Policy Search with Kinematic Rewards . . . 33
4.3 Experiments . . . 33
4.3.1 Simulation Experiments . . . 33
4.3.2 Real-world Experiments . . . 36
4.4 Conclusion . . . 38

5 Learning Navigation Abstractions for Planning with Manipulation Skills 39
5.1 Introduction . . . 39
5.2 Related Work . . . 41
5.3 Exploiting Spatial Independence for Learning Abstractions . . . 42
5.4 Simulation and Hardware Experiments . . . 48
5.5 Conclusion . . . 54

6 Constructing Abstractions for Robotic Planning with Mixed Reality 55
6.1 Introduction . . . 55
6.2 Action-Oriented Semantic Maps . . . 57
6.2.1 Defining Action-Oriented Semantic Maps . . . 57
6.2.2 Instantiating AOSMs with Mixed Reality . . . 59
6.3 Iterative Design Study . . . 60
6.3.1 Study Task . . . 60
6.3.2 Mixed Reality Interface . . . 62
6.3.3 Rapid Iterative Testing and Evaluation . . . 63
System V0 . . . 64
System V1 . . . 65
System V2 . . . 65
System V3 . . . 65
6.3.4 Overall Impressions of System . . . 65
6.4 Results . . . 66

6.5 Conclusion 68

7 Conclusion **69**

List of Figures

3.1 Simulation Tasks for Composable Interaction Primitives 21
3.2 Simulation Results for Composable Interaction Primitives 23
3.3 Executing Two Composable Interaction Primitives 24

4.1 Learning to close a microwave by bootstrapping with motion plans 27
4.2 System diagram for bootstrapping skills with motion planners 31
4.3 Real-world ball hitting with learned skills 32
4.4 Learning to open a drawer in simulation 35
4.5 Simulation results for bootstrapping with motion plans 36
4.6 Hardware results for learning to close microwave 37

5.1 An AOSM for a coffee preparation task 40
5.2 Iteratively constructing an AOSM 43
5.3 Transferring learned abstractions results 47
5.4 Spatial independence graph 49
5.5 Examples of learned operators 52
5.6 Hardware demonstration of AOSM on Spot 52

6.1 Hardware demonstration of MR for teaching abstractions 56
6.2 User perspective with MR interface 57
6.3 Images of household cleaning tasks for robot hardware experiment 60

BROWN UNIVERSITY

Abstract

Department of Computer Science

Doctor of Philosophy

Abstraction for Autonomous

Human-Robot Interaction

by Eric Rosen

Humans are able to solve complex problems by distilling their knowledge of the world into simplified task-relevant representations and creating plans to achieve their goals. In addition, central to effective human-human collaboration is the ability to teach these concise models of the world to situated partners with ease. Motivated by these properties, this thesis develops methods that enable mobile manipulators to learn action and state abstractions for task planning, and to effectively communicate and learn relevant abstractions from humans via Mixed Reality (MR) communication channels.

First, we focus on autonomously learning action abstractions. We describe a novel policy class for efficiently learning sustained-contact manipulation skills, and a method for bootstrapping learning of dynamic motor skills with motion planning. Next, we focus on autonomously learning state abstractions. We describe research on learning symbolic representations for navigation to support task planning on a mobile manipulator platform. Lastly, we describe research on learning action and state abstractions from end-users via MR. Our MR system enables humans to easily teach robots how to manipulate objects as well as label scene information to support planning. Collectively, these works lay the groundwork for enabling mobile manipulators to solve tasks in complex environments by learning state and action abstractions from interacting with the world or human teachers.

Chapter 1

Introduction

The long-standing promise of robotics is to enable autonomous robots to enter human-centered environments and help people in their daily lives. Imagine a scenario where a mobile manipulator enters a person's home for the first time, and is able to rapidly learn how to tidy rooms to their liking. For this to be a reality, the robot must be able to navigate the space, interact with objects, and plan how to sequence these behaviors to achieve its intended goals. In addition, the human may be able to intuitively communicate with the robot, so that they can easily specify goals to the robot and act as a teacher when new behaviors need to be programmed. A robot with a fluid ability to autonomously interact with the environment and the humans situated within it would be of immense value, and be a step towards moving robots out of structured environments and into society at large.

To solve this problem, we suggest that a robot must construct a world model, or set of abstractions, that can be used to support task planning. Robotic abstractions come in two forms: action abstractions, which are temporally-extended actions (motor skills) that can be sequentially composed, and perceptual abstractions (symbols), which are functions over sensor data that can represent goals and construct executable sequences of actions (plans) that achieve those goals. While the use of abstractions for generally intelligent robot behavior is well-studied, a long-standing problem has been how to acquire a relevant and useful set of abstractions for the task and environment at hand.

This thesis investigates two relevant sources of these two types of abstractions: autonomously (by interacting with the environment), or from humans from users (via mixed reality communication channels). We decompose this down into four related subproblems: 1) learning skills

autonomously, 2) learning symbols autonomously, 3) learning skills from humans, and 4) learning symbols from humans. Each chapter of this thesis addresses these subproblems, and all the solutions share a common theme: they leverage 3D spatial structure. Leveraging 3D spatial structure is a rich area for these four subproblems because a) all tasks a general mobile manipulator faces takes place in 3D space and b) Mixed Reality technologies for human-robot interaction benefit from 3D spatial modeling techniques. This leads to my thesis statement: *Leveraging 3D scene geometries enables efficient robot skill and symbol learning, either autonomously interaction or intuitive human teaching via mixed-reality channels.* We now provide an overview of how each of our solutions leverage the 3D spatial structure present in robotic domains to effectively learn abstractions autonomously and from humans.

We first describe our research on learning skills autonomously in Chapters 3 and 4. In Chapter 3, we present a structured policy class for sustained-contacted manipulation with objects: Composable Interaction Primitives (CIPs). The robot can be put in a scene with multiple objects that each require durative contact to manipulate, such as opening a door, sliding a knob, lifting a lever, or opening a drawer, and is able to efficiently and safely learn a set of motor skills to manipulate each of the objects. CIPs are designed so that after learning, given a specified sequence of objects to interact with, it can compose the motor skills in any order without any additional learning necessary, making them easy to integrate with task planning.

In Chapter 4, we discuss our work on bootstrapping motor skill learning using motion planning. The robot can use readily-estimated (but potentially noisy) kinematic models of articulated objects to generate motion trajectories that manipulate the object into a desired pose. By leveraging a motion planner, we can plan kinematic paths in the object's configuration space, which defines a constrained motion planning problem for the manipulator based on a grasp pose. Since we only use the kinematic model of the object and ignore dynamics, this trajectory is sub-optimal and may not even accomplish the task in full, but it provides an initial demonstration without any human intervention that can be used to bootstrap a policy search algorithm to enable more efficient learning. These two works jointly address how 3D scene geometries can be leveraged to enable robots to autonomously learn motor skills for manipulation that can be effectively ported to new environments.

In Chapter 5, we discuss our work on learning perceptual abstractions for task planning on a mobile manipulator. Given that a robot has a set of portable manipulation skills, we propose a hierarchical planning representation, Action-Oriented Semantic Maps (AOSMs), that can be learned from the agent automatically interacting with its environment. Perceptual symbols that involve spatial components are learned in an object-centric frame and navigation behaviors are generated on-the-fly using off-the-shelf path planners so that high-level manipulation knowledge can be rapidly transferred to new domains. This work addresses how mobile manipulators can leverage the 3D spatial structure present in robot domains to autonomously learn symbols for task planning.

In Chapter 6, we discuss work on enabling users to teach robots action and perceptual abstractions for planning using Mixed-Reality Head-Mounted Displays (MR-HMDs). This work involves users wearing MR-HMDs and labeling both action information relevant to manipulation and semantic information about the objects in the scene. We demonstrate that humans can rapidly use our interface to enable a robot to plan actions to manipulate different objects in a scene.

Taken together, this work demonstrates how 3D spatial structure can be leveraged to enable an interface for humans to rapidly specify complex action and perceptual abstractions when situated in the same environment as the robot. The culmination of this research is a significant step in enabling robots to acquire state and action abstractions for task planning, either via autonomous interaction with the environment or from a situated teacher.

Chapter 2

Background

2.1 Robot Motor Skills

Motor skills are typically learned using reinforcement learning (RL) [118], where tasks are typically 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*).

One popular approach for learning policies are policy search methods [31], which 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 $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, the objective of policy search is to maximize the expected return of the policy π_θ:

$$\max_\theta \mathop{\mathbb{E}}_{\mathcal{M}, \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t \right]. \tag{2.1}$$

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, which often makes them infeasible to apply directly to robot learning problems.

Behavioral cloning methods [11, 95, 45] are another approach for learning policies, but directly from supervised data. These methods attempt to directly learn a policy that reproduces the demonstrated policies. Given a dataset of expert demonstrations D, the objective of behavioral cloning is: $\max_\theta \sum_{(s,a)\in D} \pi_\theta(a|s)$.

In many cases, the 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* [119], 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. When a motor skill is modeled this way, the skill's objective is typically specified by a reward function R_o, and the robot's task is to learn option policy π_o that maximizes return in the usual way. 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; composable options therefore have small termination conditions that are highly likely to lie within many other options' (ideally large) initiation sets [82, 21, 120, 65].

2.1.1 Portable Skills

By considering an abstract action set, the decision-making process can select between much higher-level motor skills, effectively reducing the problem diameter while leaving lower-level control to resolve the much more limited-scope task of realizing each individual high-level action. The options framework [117] is the most popular abstract action framework. An *option* o is a tuple with three components: an option policy π_o, executed when the option is invoked and mapping low-level states $s \in S$ to low-level actions $a \in A$; an initiation set $I_o \subseteq S$ that

identifies which low-level states the option policy can be executed from; and a termination condition $\beta_o(s) \to [0, 1]$ that determines the low-level states in which execution will cease.

An additional advantage of using modular higher-level motor skills is that they need not necessarily be functions of the full problem state. For example, a motor skill to walk towards a door can be defined using just the robot's local perception, rather than an entire map. In such cases we model the option components as depending on some observation space D obtained using sensor model $\phi(S) \to D$. Options with all three components defined in a portable observation space are themselves portable, and can be reused in several places in a task, and in new tasks [67, 51].

Robotic state and action spaces are typically real-valued and high-dimensional. Consequently, the most popular family of approaches in this setting are policy search methods [32, 63], where the mapping from states to actions is directly represented as $\pi(a|s, \psi)$, controlled by parameter vector ψ. This form offers the opportunity to *structure the policy*: π need not be a simple learned mapping, but can instead include features such as safety constraints, stabilizing feedback, highly specified policy programs with just a few parameters, actuation limits, structured phases, and even motion planning. Additionally, in robotics there is the question of which state and action spaces the policy should use—the policy designer must choose where to place the policy on a spectrum ranging from directly mapping raw sensor input to motor torques, to wrapping the policy in highly processed input and output spaces (e.g., mapping object-level features extracted via computer vision to operation-space control). All of these choices are critical to effective learning.

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 [77, 62]. These methods offer flexibility, generality, and autonomy by exploiting recent advances in learning deep networks. 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 [36, 106, 8], collect experience from multiple robots in parallel [42], or include human demonstration [107, 9, 89]. 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.

An alternative approach 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 properties of a broad class of target tasks to support sample-efficient learning. The most historically important such policy class is *Dynamic Movement Primitives*, or DMPs [109, 48], which have been used to learn an impressive range of dynamic behaviors [63, 88] in tens or low hundreds of interactions, though they must typically be bootstrapped by an expert demonstration trajectory [9]. The key assumption underlying DMPs is that dynamic motions can be represented largely as a trajectory shape—represented separately for each joint, as a linear combination of learned weights with basis functions over time—coupled with a second-order dynamical system that safely and stably controls the robot towards the shape trajectory.

Other important policy classes overcome the standard shortcomings of DMPs. For example, Probabilistic Movement Primitives [93] learn a distribution over basis functions, so that variability across demonstrations and different DoFs are captured. Conditional Movement Primitives [111] encode demonstrations as a whole with high dimensional task parameters by a deep network, and in RL setting, safety and stability are achieved by encoding exploration trajectories into the same latent space [6] or coupling it with external controllers [5]. However, these approaches rely on demonstrations. Riemannian Motion Policies (RMPs) [25, 102] support combining multiple (second-order dynamical system) controllers defined in potentially different task spaces in a natural way to obtain a single controller that combines their effects in the joint space. RMPs provide a principled approach for integrating multiple concurrent controllers but have yet to see wide success in contact-rich manipulation tasks [112, 130].

An important consideration for motor learning is at what level of control and in which space the policy's action space operates at. Approaches that learn low-level controllers by directly mapping to joint torque or velocities [77] can optimize across the entire space of possible motor behaviors resulting in highly efficient behaviors, but can require a large number of samples because of the large policy space, as well as potentially exploring behaviors with

undesirable properties. A popular choice of high-level action spaces for contact-rich manipulation are those that combine end-effector control (for position control) with impedance control (for compliant force control), which enable the robot to map to actions in the task-relevant space (e.g. $SE(3)$) and explore safe and useful policies for interaction. Examples of such action spaces include Variable Impedance Control in End-Effector Space (VICES) [85] and hierarchical approaches like Cartesian Adaptive Force-Impedance Control (AFORCE) [124].

2.1.2 Motion Planning

The pose of an articulated rigid body can defined by the state of each of its movable joints. The space of these poses is called the configuration space $\mathcal{C}$ [83]. Motion planning is the problem of finding a path (sequence of poses) through configuration space such that the articulated object is moved to a desired goal configuration, without encountering a collision.

While there exist many different families of motion planning algorithms, such as geometric, grid-based, and probabilistic road maps [76], 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 valid path of joint configurations $\{q_t\}_{t=0}^{T}$ between the start and end configurations. We focus on sample-based motion planning approaches.

Probabilistic motion planners provide a principled approach for quickly generating collision-free robot trajectories. However, online replanning is expensive, and kinematic motion planners are only as effective as their kinematic models are accurate: they generate trajectories directly, and thus cannot be improved through subsequent interaction and learning. Furthermore, kinematic planners 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 effectively performing contact-rich, dexterous manipulation.

The process of computing the position and orientation $p \in SE(3)$ of a link in a kinematic chain for a given joint variable setting (a point in configuration space) is termed *forward kinematics*. Inversely, computing a configuration to attain a specific end effector pose p is termed *inverse kinematics*. We denote the forward kinematics functions $p = f(q)$.

2.1.3 State Abstractions for Planning

We are interested in learning an abstract representation that facilitates planning. A probabilistic plan p is sequence of (potentially abstract) actions to execute from states sampled from a distribution Z: $p = \{o_1, ..., o_{p_n}\}$. A suitable representation for planning must enable the agent to correctly evaluate the probability of a plan. Since a plan is a sequence of options to be executed, it is necessary and sufficient to learn when an option can be executed (known as the preconditions, which represent the probability the agent can execute the option from a given state) and what the result of executing an option is (known as the image operator, which represents the distribution over states the agent will be in after executing the option from a starting state set). Computing the image operator for arbitrary options is challenging for continuous state spaces; however, it is tractable for a subclass of *subgoal* options [98]: A subgoal option's resulting state distribution after executing the policy is independent of the starting state, so $Pr(s^{'}|o, s) = Pr(s^{'}|o)$, so computing the entire image operator can be substituted with representing the *effect* of executing the option (the distribution over states the agent will be in after executing the option), Effect(o). If an option does not satisfy the subgoal condition, the state space can be partitioned into classes C such that $P(s^{'}|o, s, c) \approxeq P(s^{'}|o, c) \forall c \in C$, which corresponds to $|C|$ subgoal options.

If all options modify all state variables simultaneously, then the resulting abstraction can be represented as a graph. However, if an option only modify a subset of state variables—the option's *factors*—then the abstract state space is expressible using a factored classical planning representation like PDDL [66]. In this formulation, preconditions and effects can be represented by propositional symbols (which constitute an abstract state space), and actions are expressed as operators on the propositional symbols.

When the options are portable, the precondition and effect symbols are defined over the observation space D instead of the state space S. While this enables the options and symbols to be ported across different tasks, there is the issue that goals specified in the state space may be aliased due to symbols defined in the observation space, making it infeasible to correctly evaluate the probability of a plan using the abstract model. To address this issue, a two stage approach is used to learn a portable symbolic vocabulary and generate a forward model: first,

propositional symbols of the options over the observation space D are learned in a training environment (resulting in parameterized symbols), then in a test environment, the portable options are partitioned based on the effects in the state-space to make them subgoal in both S and D. We defer the details of this process to James et al. [51, 52], and note that we use a similar approach for constructing our portable symbolic vocabulary.

This formulation for learning abstractions has been used by several real robots to perform complex tasks [66, 7, 41], but is generic and does not exploit any structure present across the family of tasks a real robot faces. As a consequence, it takes too long to learn the abstractions to be practical (e.g., several hours and over a hundred skill executions to learn a representation for a single small room [66]), and it assumes the robot is already equipped with a complete set of action abstractions for the particular environment it faces.

2.2 Semantic Maps

Numerous previous works have combined low-level metric maps with high-level topological and semantic information [126, 99, 68, 91, 13, 114, 43], but with a focus on navigational tasks. Various works have made semantic map representations that use a hybrid of metric, topological, and conceptual representations [126], and incorporated human input to improve and teach these representations for the purpose of navigation [114, 13, 99]. Most notably, Pronobis and Jensfelt [99]'s semantic map representation has place appearance and geometry, object information, topology, human input, segmentation, conceptual maps, uncertain concepts, inferred properties, and autonomously acquired concepts. However, Pronobis and Jensfelt [99] do not learn object manipulation requisites, such as grasp points, termination sets, and motor policy representations, and only focus on information necessary for effective localization and navigation.

Previous works have learned object representations that do contain information that is used for object manipulation planning [129, 71, 15, 101, 97], but do not consider learning semantic map representations of their environment in the process. Object-Action-Complexes (OACs) [129, 71], which consider objects and action representations to be intertwined by capturing interactions between objects and associated actions, allow the agent to acquire object

knowledge about the world through predicting changes in the world via agent interaction. While OACs provide a symbolic representation of sensormotor experience for objects, they do not have sufficient information about the environment to generate maps for the use of navigation. Beetz et al. [15] present an impressive knowledge-base, KNOWROB2, which incorporates components like perception, learning, and reasoning to achieve complicated manipulation tasks like making a pizza. While KNOWROB2 is able to learn what robot poses in a map of the environment are useful for actions like suitable grasps, it requires access to an "inner world" of the environment with symbolically-annotated objects with the map, and does not address how to learn such a detailed and accurate semantic map representation of the environment. More importantly, KNOWROB2 does not represent actions in local object frames, which is crucial for leveraging the MR interface and teacher input.

2.3 Mixed Reality for Human-Robot Interaction

MR-HMDs show great promise for facilitating human-robot interaction, and have been used for communicating robot motion trajectories [103, 125, 23] and specifying robot commands [46]. Beyond their improvements to speed, accuracy, and mental workload over baselines [103], MR-HMDs also enable the human to share the same space as the robot and interact with a virtual environment instead of having to interact with the real, physical robot [38]. While projector-based approaches are also a powerful tool for facilitating human-robot interaction [20], they require structured environments and are unable to highlight free 6D space because they must project onto a surface, which is limiting in the case where a human must teach spatial attributes (like a grasp pose) for planning.

While some previous work has used MR-HMDs to have robots learn from humans, it has only focused on simple pick and place tasks, and not on using the MR interface to learn requisite information needed for complex object manipulation and navigation [38, 72]. Gadre et al. [38] designed an MR interface to enable end-users to program robot motions via waypoint specification for the purpose of pick and place, and Krupke et al. [72] designed a MR interface for a similar task, but instead manipulated virtual items in the workspace to specify place

locations. While these works demonstrate the capability of learning with MR, they focus on how such an interface compares to other modalities (like 2D monitor interfaces).

Chapter 3

Composable Interaction Primitives for Sustained-Contact Manipulaton

3.1 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 [69]. 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 human factory worker can be trained to basic proficiency in an unfamiliar new task in a day; skillful and reliable execution of rote manual labor tasks rarely requires longer 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 a reasonable amount of time without the support of expert programmers. How can robots efficiently learn action abstractions for object manipulation in an autonomous manner?

To answer this question, this chapter investigates designing a highly structured policy class [48, 25, 102] to achieve sample-efficient learning, thereby trading design effort, flexibility, and generality for sample efficiency. Such approaches have been used to learn an impressive range of dynamic behaviors [63, 88] 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.

This chapter's focus is 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 that the components of CIPs substantially improve learning efficiency and safety. We then demonstrate the use of CIPs to efficiently learn, and subsequently sequence on-demand, two real sustained-contact manipulation skills.

3.1.1 Related Work

To our knowledge, our method is the first to use an object's estimated kinematics in conjunction with a known robot dynamics model to bootstrap motor policy learning, and we discover and discuss important problems that are only introduced when leveraging policy-learning algorithms, behavioral-cloning, and motion planning algorithms to do so. In this section, we discuss relevant approaches to motor skill learning.

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 [58, 92]. These approaches require a priori object dynamics, or human demonstrations to fit learned models; in constrast, our approach requires only object kinematics, which are much more readily estimated from visual data at runtime [2, 80]. As such, our approaches enables the

learning of manipulation skills to be more autonomous than existing MPC-based methods. Tosun et al. [122] 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. In constrast, our method uses object kinematics to produce initial trajectories, while Tosun et al. [122] only use the robot's kinematic model, which is insufficient when the task is to manipulate an object to a specific joint configuration.

While classic robot motor learning papers [11] 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.

Kurenkov et al. [74] 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 [113, 55]. 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) [78] 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. [24]). GPS is one of the state-of-the-art algorithms we expect to be used within our framework as the policy search implementation (Section 4.2.2). 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 apriori, and thus is immediately capable of generating high-value policies, whereas GPS is estimating 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). The ideas proposed in our paper are distinct from those put forth in GPS: we present a method for obtaining demonstrations under certain conditions in the absence of a human.

Most similar to our line of work are those that use sample-based motion planners for improved policy learning. Jurgenson and Tamar [56] harness the power of reinforcement learning for neural motion planners by proposing an augmentation of Deep Deterministic Policy Gradient (DDPG) [81] 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 [56] 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. [54] address learning to improve plans produced by a motion planner, but do not bootstrap closed-loop policies. Motion planners aren't expressive enough 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.

3.2 Approach

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. Only the sustained-contact controller itself need be largely learned on a per-task basis, though it could be bootstrapped using learning from demonstration [9] or kinematic motion planning [3]. Third, *the sustained-contact controller itself requires 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 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. 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:

$$C = (\pi_c, \sigma, \beta_c, I_c, h, t, B)\,,$$

where:

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

- Policy π_c is constrained by σ, a safety envelope specific to the robot joint space but not to the task. Learning and execution are 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.

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

- 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 the robot from contact back into free space—or into the head of another skill—after the skill terminates, respectively. These serve to establish and break contact, and to sequence skills: the tail of one skill simply becomes the head of another.

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 is small, isolated pockets of motor skill learning connected by much longer trajectories generated by a motion planner.

3.2.1 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 3.3) as an illustrative example.

Motor control policy π_c. Our examples use sensor input from the touch sensors on the robot's grippers, the joint and Cartesian state of the robot, and object joint state, which are fed into a two-layered multi-layered perceptron (MLP) with 64 hidden nodes. For the action space, we chose to have the robot command the end-effector in Cartesian space while maintaining compliance with external forces, to promote ease-of-learning and safety during sustained contact. We therefore selected the Variable-Impedance Control in End-Effector Space [85] scheme as our action space. Motor policy π_c maps sensor readings ϕ to a desired delta end-effector position p_d and desired rotation R_d, as well as commanded stiffness terms k_p^p and k_p^R for position and rotation respectively. These terms are then use to directly map to joint torques τ via:

$$\tau = J_p[\Lambda_p[k_p^p(p_d - p) - k_d^p v]] + J_R[\Lambda_R[k_p^R(R_d \times R) - k_d^R \omega]], \tag{3.1}$$

where Λ_p and Λ_R are the position and orientation components of the inertia matrix $\Lambda \in \mathbb{R}^{6\times 6}$ in the end-effector frame, J_p and J_R are the position and orientation components of the end-effector Jacobian J, and $R_d \times R$ corresponds to subtraction in $SO(3)$. k_d^p and k_d^R are the damping values for position and rotation respectively, and are set with a damping ratio of 1 (critically damped). We also map directly to the gripper state $g \in \mathbb{R}$, with -1 being fully opened at 1 being fully closed. The resulting action space is therefore 13 dimensional.

Safety envelope σ. We limit the maximum value of stiffness parameters k_p^p and k_p^R, 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. In order to prevent joint limit violations, we use a two-fold strategy 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 [61] 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 to attempt to return to a safe configuration.

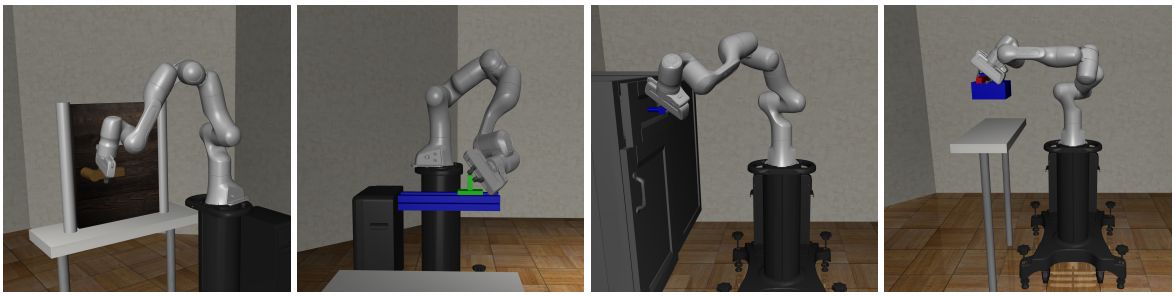
Task-specific success indicator β_c. These were designed by hand for each task, and return true when the object's joints are above a threshold.

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 the learning episode ends early. Episodes are also terminated early if the agent loses contact with the object for sufficiently many 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. [121] 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, and the stability of the grasp for sustained-contact manipulation is evaluated by using a random noise policy to jiggle the gripper at the grasp pose g, and then the gripper is checked to still be in contact with the object. Grasps g which pass all these checks are added to a list of acceptable grasp poses that define the domain of I_c.

For sampling a grasp pose $g \in I_c$ for the head h during learning, we propose treating grasp pose sampling as a bandit problem that is solved with Upper Confidence Bounds (UCB) [116] where Q-values are task success rates. We therefore treat learning I_c and sampling grasp poses as an active-learning problem, and purposely choose UCB since it is particularly well-suited to balance exploration and exploitation.

Once a grasp pose g is sampled, we repeatedly solve inverse kinematics to obtain a joint configuration θ with high manipulability. A manipulability score is computed for a joint



(A) Open door task (B) Slide knob task (C) Open drawer task (D) Flip lever task

FIGURE 3.1: Simulation Task Environments

configuration θ as the product of two values: 1) the manipulability index introduced by Yoshikawa [131] that analyses the size 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 [123] based on the distance to the upper and lower joint limits for a particular joint configuration θ:

$$P(\theta) = 1 - exp(-k \prod_{j=1}^{n} \frac{(\theta_j - l_j^-)(l_j^+ - \theta_j)}{(l_j^+ - l_j^-)^2}), \tag{3.2}$$

where l_j^- and l_j^+ are the lower and upper joint limits for joint j. When these two metrics are multiplied together, they 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, which is termed the manipulability value.

Motion planners h and t. These were instantiated for each domain using the TRAC-IK inverse kinematics solver [14] and a basic grasping controller for establishing contact at the grasp pose sampled from the grasp initiation set I_c.

3.3 Experiments

We evaluate the CIP framework in simulation using Robosuite [132]. We conducted experiments on four different articulated object tasks: opening a door, opening a drawer, sliding a knob, and lifting a lever.

The state space for our policy is the state of the object, the position and velocities of the robot's joints, and tactile readings from the force sensors at the robot's grippers. We use TD3

[37] as our actor-critic method. 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.

Our reward function is a dense reward based on the state of the object and how much its joint has progressed towards the goal, which leverages potential-based reward shaping [90] to ensure the optimal policy is not changed compared to the sparse reward setting based on success. Training episodes are ran for a maximum of 250 steps each, and trained for a total of $25,000$ training steps, which result in $100 - 1000$ training episodes depending on the task and experiment conditions. We evaluate the policies performance every 10 training episodes with 10 policy roll-outs.

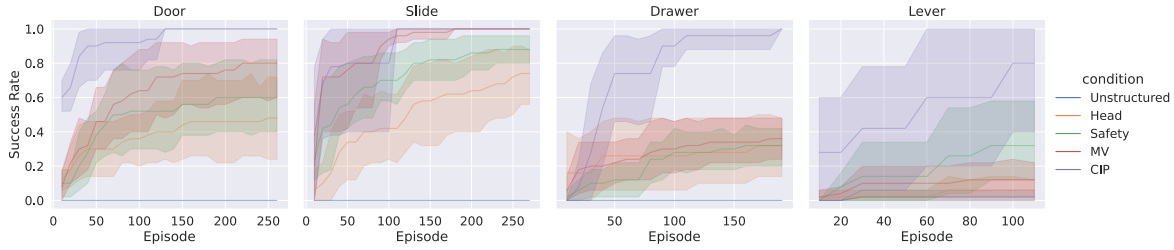
We consider two evaluation metrics: 1) **Task Success Rate**, which measures how successful the policy is at manipulating the object, and 2) **Joint Limit Violation Rate**, which is a proxy measure for how safe the policy is. To analyze how each of the structures of CIP impact these metrics, we run 5 ablations that incrementally include structure described in Section 3.2.1:

1. **Unstructured**: This setting is a 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.

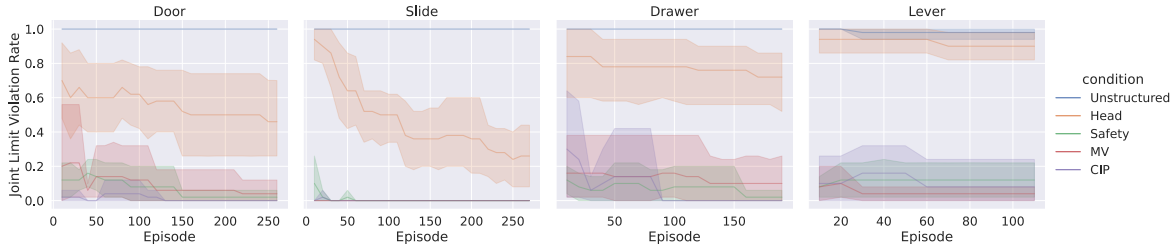
2. **Head**: This setting is a baseline that incorporates the head h_c structure of the CIP. The agent has access to the domain of the grasp initiation set I_c, but samples grasp poses randomly and chooses random valid joint configurations.

3. **Safety**: This baseline extends the **Head** setting 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.



(A) Success rates for simulated tasks (Door, Slide, Drawer, Lever).



(B) Joint limit violation rates for simulated tasks (Door, Slide, Drawer, Lever).

FIGURE 3.2: Task success rates and joint limit violation rates vs. the number of training episodes. The shaded region around the average is the 95% confidence interval over 5 seeds.

5. **CIP**: This setting incorporates all the structure of the CIP. Specifically, it extends the **MV** approach to additionally perform active learning with UCB over grasp poses.

3.4 Results

The results for all our experiments are in Figure 3.2, where we show the best-to-date performance for both metrics across all the tasks. Across all the tasks, the **Unstructured** baseline performs the worst and 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 (the baseline with minimal additional structure being **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 is able to significantly reduce the number of 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 100 training episodes. These results demonstrate that each structural component of the CIP is useful for ensuring that the agent is able to safely and efficiently learn across a diverse set of sustained-contact manipulation tasks.

3.4.1 CIPs for multi-step plan execution

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 enable a robot to learn sustained-contact manipulation skills in isolation using a model-free learning algorithm, and then sequentially execute the skills when multiple objects are in the scene with no additional learning necessary. This is particularly useful when an agent performs high-level planning with the skill repertoire, since a plan involves composing actions together in sequence. We show a demonstration of this behavior in Figure 3.3, where the agent has been trained to separately slide a knob and open a door, and is now placed in a scene that has both. When tasked with a plan that involves first sliding the knob and then opening the door, the robot is able to use the head h to first motion plan to grasp the slide knob, execute the slide knob policy, and then motion plan to the grasp the door handle to finally execute the open door policy before returning to free space using the tail t.

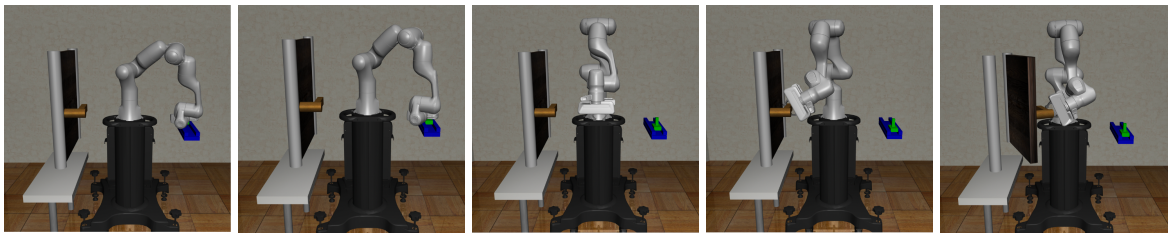


FIGURE 3.3: Two CIPs executed in succession.

3.5 Conclusion

We proposed 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 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. By exploiting the spatial structure, we enable robots to autonomously learn action abstractions for object manipulation in a safe and sample effecient manner.

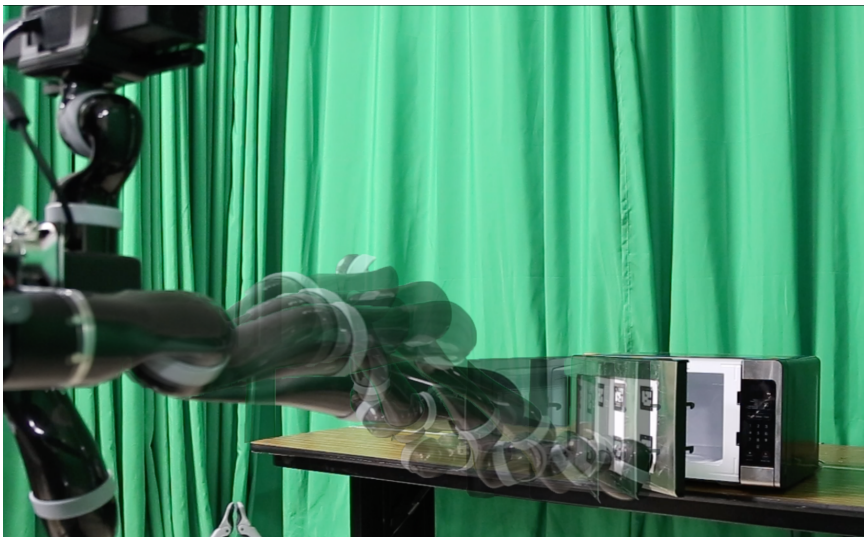
Chapter 4

Bootstrapping Motor Skills with Motion Planning

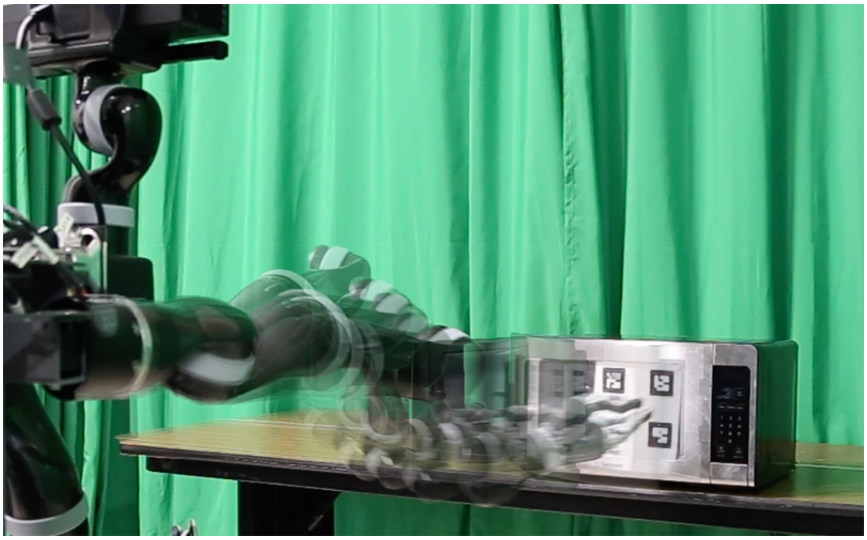
4.1 Introduction

Using RL to enable robots to autonomously acquire motor skills is important, but is a much more challenging problem than the supervised learning regime. Supervised approaches for policy learning like Learning From Demonstration (LfD) [10] 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 [78, 100, 26, 108]. 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 [92] and LQR [78]) 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. This chapter addresses the question: how can robots leverage supervised learning techniques to improve the abilities of robots to autonomously acquire manipulation skills?

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 [122, 56, 54], they only focus on either free-space motions or do not learn a closed-loop controller. To



(A)



(B)

FIGURE 4.1: 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.

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 4.1a) to bootstrap an object-manipulation policy that can subsequently be optimized using policy search (Figure 4.1b). 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) [49] and deep neural networks [79]). 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.

4.1.1 Related Work

An Action-Oriented Semantic Map is a spatial data structure that provides sufficient information for synthesizing a navigation stack with a motion planner to support task planning with manipulation skills. Because of this, our work is related to the Task and Motion Planning (TAMP) literature.

TAMP solutions integrate high-level discrete task planning with low-level continuous multimodal motion planning to induce a planning hierarchy where different specialized planning and learning algorithms can advantage of the structure present at each level [57]. Garrett et al. [40] introduced the concepts of *transit modes* (when the robot is not in contact with any object) and *transfer modes*, (when a robot is in contact with an object), which is closely related to our intuitive notions of the distinction between navigation and manipulation. Similarly, Wolfe et al. [128] propose a vertically integrated hierarchical task network for combined task and motion planning for mobile manipulators which has built-in structure for suggesting locations the mobile base should be in to execute a pick and place action. Rather than assume the given state abstraction is sound for a particular task, we formalize a data structure that captures this regularity in task and motion planning for mobile manipulators and prove under what conditions it is sufficient for planning with a given set of manipulation skills. Most similar to our work in the TAMP literature are approaches that attempt to leverage semantic maps for improving task and motion planning. Galindo et al. [39] investigate how semantic maps can act as a hybrid knowledge base for task and motion planning in the context of navigating around an environment. This work is similar to ours in that Galindo et al. [39] use a semantic map to improve task planning, but differs in that they only extract additional information from a semantic map, while we identify a new data structure (an AOSM) that is built-on top of a semantic map and is provably sufficient for supporting planning with a set of manipulation skills. Kuipers [73] proposed the spatial semantic hierarchy, which is a model of knowledge for large-scale spaces that organizes qualitative and quantitative information in a hierarchical fashion on top of geometric and semantic map information. Within the context of a spatial semantic hierarchy, an AOSM can be constructed to aid with providing target locations to the navigation control laws for supporting manipulation skills.

4.2 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 4.2, 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 [2, 80], 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

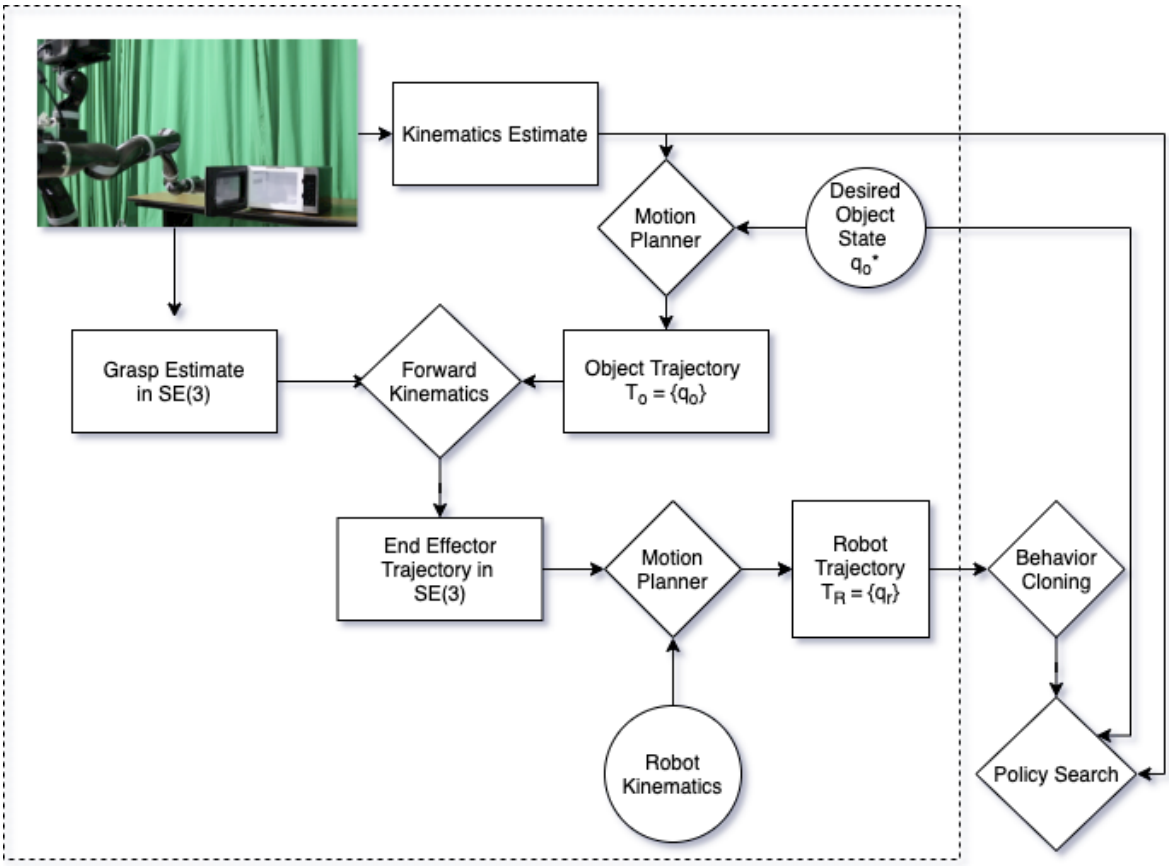


FIGURE 4.2: **System overview** illustrating our proposed framework for generating demonstrations with a motion planner and subsequently performing policy search. The dashed box contains the steps from Algorithm 2.

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.

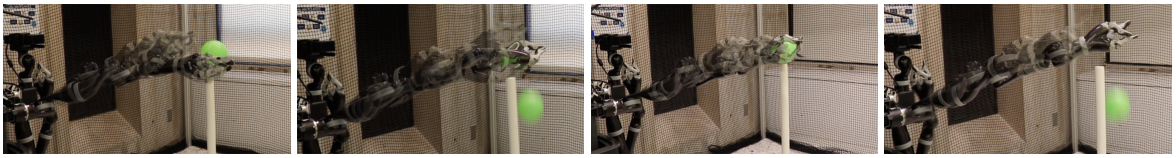
Our approach is outlined in Algorithm 1, and can broken down into five main steps: 1) collect initial trajectories from a motion planner using estimated object kinematics, 2) fit a policy with these initial trajectories, 3) gather rollouts to sample rewards for the current policy based on the kinematic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



(A) Human Demo (B) Bootstrapped from (a) (C) Motion Plan Demo (D) Bootstrapped from (c)

FIGURE 4.3: **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. Videos can be found in our supplemental video. (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.

4.2.1 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 [110] for DMPs, and maximize the likelihood of the demonstration actions under the policy for neural networks.

4.2.2 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.

4.3 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.

4.3.1 Simulation Experiments

We used PyBullet [30] 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:

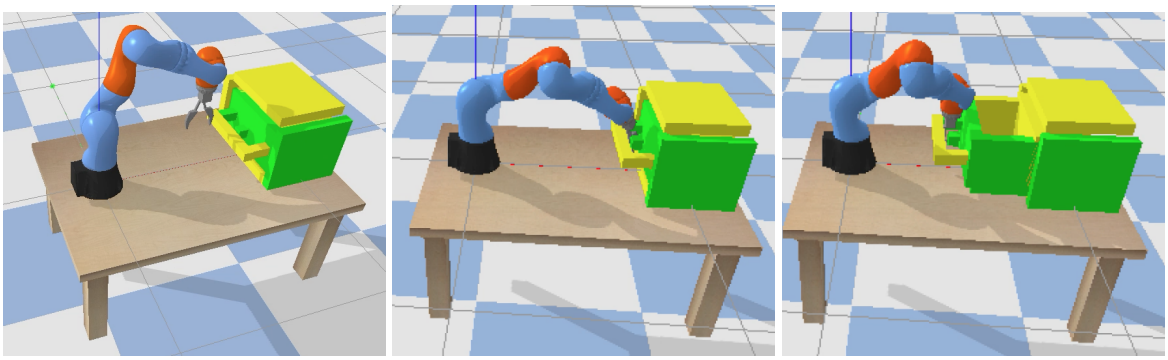
$$r_t = -c \, ||q_O^* - q_O||_2^2 - a_t^T R a_t, \tag{4.1}$$

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 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 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 [59]. 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 [100], which essentially adds the behavior cloning loss to the natural policy gradient loss, 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 4.5a.

The second simulated task was to open a drawer (Figure 4.4). 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 [115]. We used 32 basis functions for each of



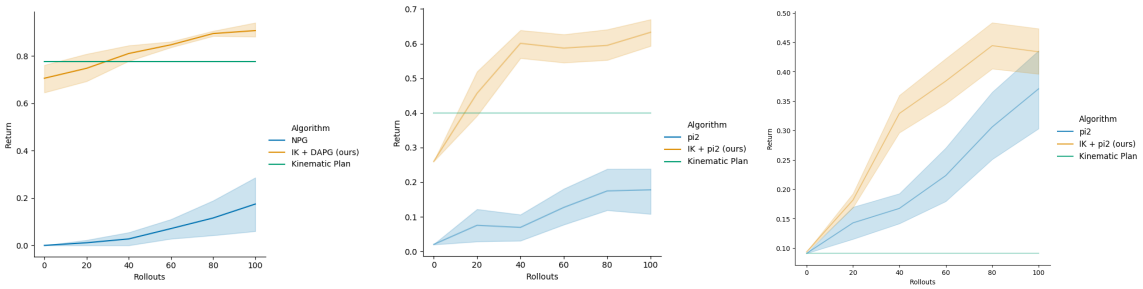
(A) (B) (C)

FIGURE 4.4: **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.

the DMPs. The pretrained policy was initialized using Locally Weighted Regression (LWR) [110] with a single demonstration. The results of this experiment are shown in Figure 4.5b.

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 4.5c.

The results of our simulated tasks can be found in Figure 4.5. 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.



(A) Microwave closing (MLP) (B) Drawer opening (DMP) (C) T-ball (DMP)

FIGURE 4.5: **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.

4.3.2 Real-world Experiments

For all our real-world experiments, we used a 7DoF Jaco arm [22] to manipulate objects (Figure 4.1). We used ROS and MoveIt! [28] as the interface between the motion planner (RRT* [60] 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 space, action space, and reward were defined in the same way as in our simulated results (Section 4.3.1). Both experiments used DMP policies initialized with LWR [110] and optimized with PI^2-CMA [115] 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 4.3.1). 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 4.6.

[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.

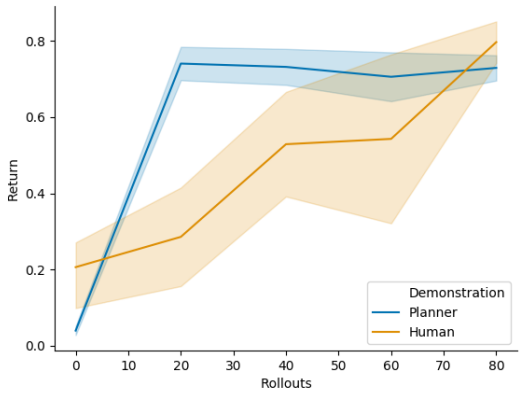


FIGURE 4.6: **Hardware experiment** comparing our initialization scheme with human demonstration. Results are shown as mean and standard error, aggregated across three random seeds.

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 4.3). 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 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.

4.4 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.

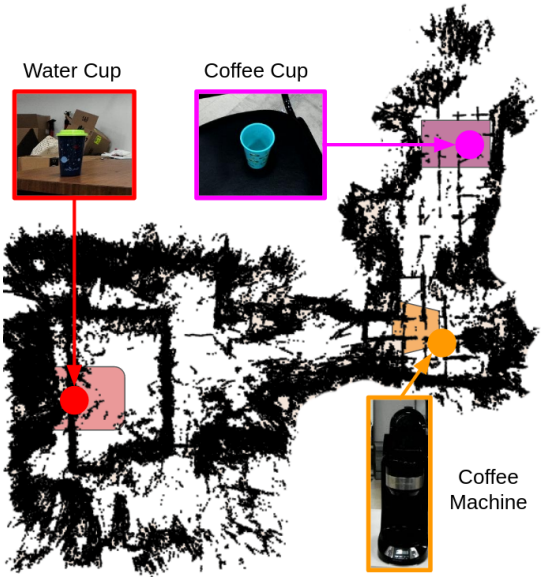
Our methodology is agnostic to the motion planner, motor policy class, and policy search algorithm, making it a widely applicable paradigm for learning robot motor policies. We demonstrate the power of our methodology by bootstrapping different policy classes with demonstrations from humans and a motion 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. This enables robots to effectively learn action abstractions via autonomous interaction with the environment.

Chapter 5

Learning Navigation Abstractions for Planning with Manipulation Skills

5.1 Introduction

Planning for mobile manipulation is difficult because of its long-horizon nature. There are two approaches to addressing this difficulty: subtask decomposition and structural decomposition. The former approach decomposes the problem into smaller subtasks (e.g: hierarchical planning [16, 96]), and leverages abstractions in two forms: action abstractions, also called *skills*, which package motor behaviors into a single invokable action, and perceptual abstractions, typically represented as grounded *symbols*, which compactly represent the relevant aspects of task state. Learned abstractions can address complex planning problems [66], but existing approaches are sample inefficient because they do not exploit structure present in the robot and the world. The second approach—structural decomposition—aims to design algorithms that do just that. Navigation stacks typically focus on building maps and localizing a robot in a map [34, 12], and using those maps to navigate to a goal via path planning [84]. Research in robotic manipulation structures the task of effectively interacting with objects [86] into component algorithms such as object recognition [17], interactive perception [19], grasp synthesis [18], kinematic motion planning [76], and learning for manipulation [70]. This approach can produce algorithms that generate useful behavior while avoiding learning entirely.



(A) An Action-Oriented Semantic Map for a coffee preparation task.

(B) Spot executing portable manipulation skills in coffee preparation task. Given a new environment with these objects, our approach efficiently constructs the navigation abstractions—both action and state—to support planning using these skills.

FIGURE 5.1: An AOSM for a coffee preparation task. (a) The underlying semantic map consists of a 3D point cloud of the scene (black points) along with the detected pose and attributes of objects. (b) Given a set of portable manipulation skills (start top left clockwise: pouring water, picking up a cup, placing a cup, and pushing a brewing button), an AOSM also includes a distribution over poses where the robot can execute each skill (visualized by colored areas in map (a)).

We propose to combine these two complementary approaches by exploiting structural assumptions to efficiently learn high-level abstractions. We begin by splitting abstractions to do with manipulation from those to do with navigation. Manipulation abstractions are expensive to learn but are typically object-centric and therefore portable, while navigation abstractions are *not* portable: how the robot should abstract its map pose and navigate between locations depends on the specifics of a single scene. Efficiently learning the navigation components of the abstraction, which must be re-learned for each task, is thus critical. We therefore assume a given (pre-learned or hand-constructed) set of portable manipulation abstractions (both skills and symbols), and consider how to efficiently generate the navigation abstractions that support planning with them in a novel environment.

Our key insight is that spatial and non-spatial state variables typically contribute independently to whether a motor skill can be executed; and that under those conditions, a unique data structure—an Action-Oriented Semantic Map (AOSM) [104] (Figure 5.1a), which encodes the spatial locations from which manipulation skills can be executed—is necessary and sufficient to generate all the navigation abstractions required to support manipulation planning. We provide an algorithm to autonomously and efficiently construct an AOSM from a given set of manipulation skills using well-established mapping and path planning algorithms; a robot can thereby complete its abstract representation of a new task by constructing its navigation components in just a few minutes of robot time. We evaluate our approach in both simulation (using AI2Thor [64]) and on real robot hardware (a Boston Dynamics Spot). In simulation, our approach decreases the number of interactions required to learn navigation abstractions by an order of magnitude compared to the state of the art, and enables the robot to transfer learned symbols to new environments. On real robot hardware, our system generates a representation of a coffee-making task for two different kitchen environments in a few minutes.

5.2 Related Work

Our work focuses on learning state abstractions that enable long-horizon task planning by leveraging manipulation skills and semantic maps, similar to Task and Motion Planning (TAMP) frameworks. However, our work differs from TAMP based on the assumptions we make:

Rather than use motion planning to generate manipulation behaviors, we treat manipulation skills as black-box skills that can be implemented with or without motion planning (e.g: learned motor policies [3]), and only require a model of the environment to support path planning for locomotion, which is readily accessible using off-the-shelf SLAM.

TAMP solutions integrate high-level task planning with low-level continuous motion planning to exploit a planning hierarchy where different specialized planning and learning algorithms can exploit the structure present at each level [57] and across modes [40]. However, whereas standard TAMP approaches assume access a given state abstraction is sound for a particular task [128, 57], we formalize an independence property between spatial and non-spatial state variables to more efficiently learn a sufficient representation for planning with given manipulation skills. Most similar to our work are TAMP approaches that leverage semantic maps for improving task and motion planning. Galindo et al. [39] investigate how semantic maps can act as a hybrid knowledge base for TAMP in the context of navigation. This work also uses a semantic map to improve task planning, but only extracts additional information from a semantic map, whereas we identify a specific augmentation to a semantic map that is provably sufficient for supporting manipulation planning. Our work is also related to approaches that leverage Large Language Models (LLMs) for task planning. These approaches [4, 47] generally assume the existence of a preprocessed map that enables navigation to support manipulation. Our work here formalizes this data structure and lays the theoretical foundations for how this specific data structure can not just be used in task planning with LLMs, but also for learning symbols for task planning.

5.3 Exploiting Spatial Independence for Learning Abstractions

Problem Definition We are interested in the problem of a robot that must navigate an environment and manipulate objects to achieve a goal. To this end, we represent the decision problem as an MDP, and factor the state $s \in S$ into the state of the robot S_r and the state of the environment S_e: $S = S_r \times S_e$. Furthermore, the state of the environment can be factored into a discrete set of q objects (or entities) the robot may manipulate, $S_\Omega = \Omega_1 \times ... \times \Omega_q$, and a map of the environment $m \in M$, $S_e = M \times S_\Omega$. This structured representation of the

environment is often called a *semantic map* [64]. Since the robot and all of the objects exist in a physical space, they each have a pose in the map. Therefore, we factor the state of the robot S_r into some pose S_b in the map and any other information describing the state of the robot S'_r: $S_r = S_b \times S'_r$, and similarly for each object $\Omega_i \in \Omega$: $\Omega_i = \Omega_i^b \times \Omega'_i$. The task-specific semantic map defines a constraint function on the feasible poses of the robot, and can be used in conjunction with a path planner $N(s_b, s'_b)$ to generate trajectories through the space of robot poses S_b from a start state s_b to a set of goal states s'_b (i.e: locomote the robot around the scene).

Given the above setting, our problem is formalized follows. For a given set of portable manipulation options O and a semantic map S_e, we must take plans that consist only of manipulation actions (called a manipulation-only plan $p_O = \{o_1, ..., o_{p_o}\}, \forall i \in \{1, ..., p_o\}, o_i \in O$, where p_o is the length of the plan p_O), and learn a portable abstract representation that supports generating task-specific navigation behaviors based on S_e that can be interleaved into the manipulation-only plan to make the probability of success non-zero. Note that even though the state space is fully observable, it crucially does not include information about what configurations in space afford manipulation, which is what our approach learns.

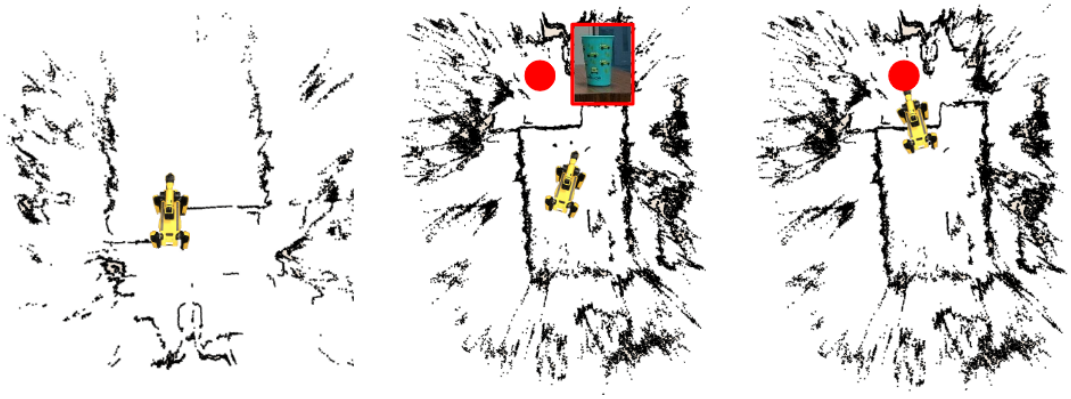


FIGURE 5.2: An example figure of a robot iteratively constructing an AOSM in a novel environment. (Left): The robot has a partial map of the environment and has not seen any objects. (Middle): The robot moves around to construct more of the map, and the vision model identifies a cup (position visualized as red circle). (Right): The robot uses a learned navigation symbol to sample a pose to pick the cup, and then navigates to that pose in order to execute the manipulation skill.

Approach Our approach is based on autonomously constructing an Action-Oriented Semantic Map (AOSM) [104] and using it for task planning. Formally, an AOSM $(O, S_e, (V, E))$

is a data structure where O is a set of k portable manipulation options, S_e is a semantic map, and (V, E) is a topological graph. The topological graph (V, E) is an undirected graph that contains k nodes $V = \{v_1, ..., v_k\}$, where each node v_j represents a region of configuration space for the base of the mobile manipulator (i.e: each node v_j represents a set of poses in the semantic map). Node v_j corresponds to the set of poses in the semantic map that have a non-zero probability of being in the initiation set of option o_j. So, $v_j = \{p \in I_{o_j} | p \in m\}$. The node v_j is also referred to as a navigation symbol σ^{o_j} for the option o_j, since a symbol is a probabilistic binary classifier for testing membership of a set, and this symbol only depends on whether the robot's configuration is within a specific region of space that is relevant for navigation (discussed in more detail below). An edge $e = (v_a, v_b) \in E$ represents that a motion planner $N(v_a, v_b)$ can be used to successfully navigate from the set of poses in v_a to the set of poses represented by v_b. AOSMs were introduced in Rosen et al. [104], where they were hand-crafted by a user. Here, we assume access to a set of portable manipulation skills O and the semantic map S_e, and we provide a novel algorithm for learning the topological graph (V, E) that consists of the navigation symbols and edge connectivity between them, which together define an AOSM.

When a robot has access to an AOSM, it can sample poses in the map that enable the robot to execute its manipulation skills (Figure 5.2). When the navigation symbols are learned in an object-centric spatial frame (i.e: the regions of space are in an object-centric frame instead of a map frame), they can be ported to new environments by grounding to global poses based on the known poses of the objects in the semantic map S_e. Once an AOSM has been constructed, given a manipulation-only plan $p_O = \{o_1, ..., o_{p_o}\}, \forall i \in \{1, ..., p_o\}, o_i \in O$, a starting base pose S_b^0, and a path planner $N(s_b, s_b^{'})$, we can use the AOSM to sample poses from the navigation preconditions of each manipulation option $\{S_b^1, ..., S_b^{p_o}\}, \forall S_b^i \sim \sigma^{o_i}$, and leverage the the path planner to synthesize a sequence of locomotion path plans $p_N = \{n_1, ..., n_{p_o - 1}\}, n_i \sim N(S_b^{i-1}, S_b^i)$ that can be interleaved into the manipulation plan p_O, $p_{O'} = \{o_1, n_1, o_2, n_2, ..., , o_{p_o - 1}, n_{p_o - 1}, o_{p_o}\}$. This augmented plan has the requisite additional actions required to make the manipulation-only plan feasible in the specific map the robot finds itself in. An AOSM can only can be used when it is possible to decompose initiation

sets into navigation and manipulation preconditions and represent them separately. In this work, we prove this assumes a crucial independence property of the factors of the initiation set, which we formally describe in the rest of this section.

First, note that we can define navigation symbol as a symbol σ whose factors (the set of state variables the grounding classifier depends on) are the robot's mobile base state variables S_b, $\text{Factors}(\sigma) = S_b$ (we call this type of factor a spatial factor). To determine whether a state variable is in the factor associated with the initiation set of a manipulation option (i.e: the state variable is a defining state variable for that set of states), we can use the notion of *projection*. The projection of a list of state variables v out of a set of states X is defined as $\text{Proj}(X, v) = \{s | \exists x \in X, s[i] = x[i], \forall i \notin v\}$, which removes any restrictions on the values of the state variables v for the states in X. If we project out a state variable from a set of states and it changes the set of states, we say that the state variable is a defining state variable for that set of states (since deciding whether a state is a member of X depends on a restricted value for v). If that set of states is the initiation set I_o of an option o, then that collection of state variables is by definition the factors of I_o, $\text{Factors}(I_o)$. In this case, the set of states describing the initiation set can be described by the intersection of independent state sets [66]. Formally, we say a factor f_s is independent in the initiation set I_o when: $I_o = \text{Proj}(I_o, \text{Factors}(I_o)/f_s) \cap \text{Proj}(I_o, f_s)$. With this definition, we now define the spatial independence property:

Definition 5.3.1 (Spatial Independence)**.** The initiation set I_o for an option o's has the spatial independence property if:

$$I_o = \text{Proj}(I_o, \text{Factors}(I_o)/S_b) \cap \text{Proj}(I_o, S_b). \tag{5.1}$$

Note that when learning a probabilistic symbolic representation, the sets are replaced with distributions and the intersection is replaced with multiplication, and therefore the independence property is defined exactly as conditional independence. When an option's initiation set has the spatial independence property, we can construct an independent symbol to represent $\text{Proj}(I_o, \text{Factors}(I_o)/S_b)$ which by definition is a navigation symbol since it it only depends on S_b. Intuitively, this projection represents the set of base locations the robot must be in order

to successfully execute the option o without regards to the state of the rest of the world. [1] Since an AOSM captures the navigation symbols, then when the spatial independence property holds for an option, an AOSM is a necessary and sufficient characterization of the spatial components of the initiation set.

We now formally describe under what conditions we will resolve a manipulation option o for some set of starting states Z. Consider an option o that has an associated navigation symbol σ^o to characterize part of its initiation set I_o: $I_o = \text{Proj}(Z, S_b) \cap \sigma^o$. Then this implies that if the agent is in a state that is an element of Z, and only changes the robot's mobile base pose to be an element of the navigation symbol without changing anything else, then the resulting state would be an element of the initiation set of the option. We prove that if our assumptions regarding the initiation set of a manipulation option are satisfied, then we can synthesize a locomotive behavior from our navigation stack using our learned navigation symbol, which means we can generate the navigation stack to support a specific option. Since the state is Markovian, proving for the more general case where we aim to generate a navigation stack to support a manipulation plan follows from repeated applications of Theorem 1, and so we omit it.

If a manipulation option's o_i initiation set can be written using the definition of spatial independence (Equation 1) from the current set of states Z, then sampling a location l from σ^{o_i} and synthesizing and executing a path plan from the navigation stack to l from a start state in Z is sufficient for enabling the robot to execute the manipulation option o_i.

Theorem 1. *If, for a starting set of states Z, the initiation set I_{o_i} for a manipulation option $o_i \in O$ can be characterized as in Equation 1, then a location l sampled from the associated navigation symbol $l \in \sigma^{o_i}$ can be used in conjunction with a path planner to locomote the robot to a state s that is within I_{o_i} the initiation set of o_i as long as there is a collision-free path.*

Proof. By our assumptions, we know that the initiation set for the manipulation option can be decomposed into $I_o = \text{Proj}(Z, S_b) \cap \sigma^o_s$. We also assume that the agent starts in a state z element of Z ($z \in Z$). We can then use the pose l that is sampled from the navigation symbol

[1] We note that this assumption may be violated in realistic domains (for example, the location of objects may constrain what locations the robot can execute a manipulation option from), but we later discuss how we can still use an AOSM to synthesize effective navigation abstractions even when this assumption is not met.

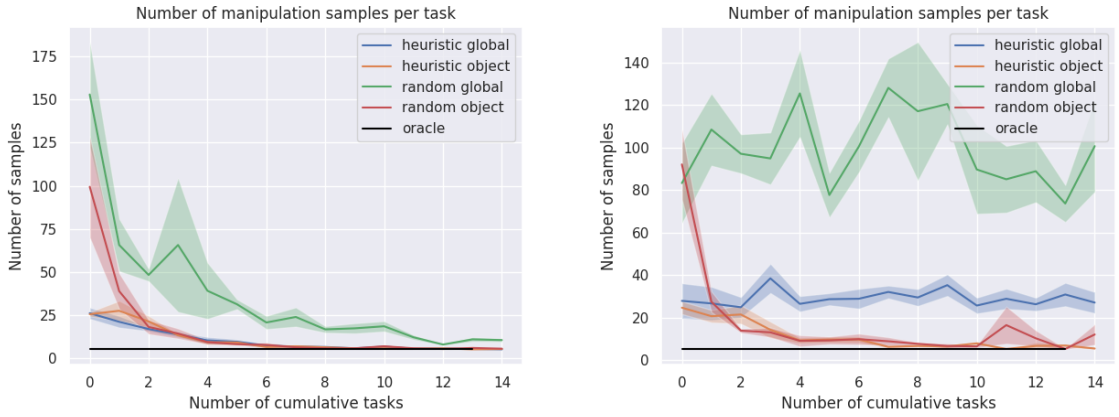


FIGURE 5.3: Results for our experiments on transferability of learning abstractions (left and right are single-scene setting/multi-scene setting respectively). We report the cumulative number of sampled locations that manipulation actions are attempted from against the average cumulative number of times the agent has successfully completed the plan (bars are standard error across 5 seeds.)

σ^{o_i} to synthesize a navigation action n_i that starts from z and ends at location l, $n_i \in N(z, l)$ as long as there is a collision free path through the environment. The effect of executing n_i from z by definition only affects spatial state variables S_b, and so the resulting state is an element of $Proj(Z, S_b)$ and also an element of navigation symbol σ_s^o. Therefore it the resulting state is an element of the intersection of $Proj(Z, S_b)$ and σ_s^o, which is by definition the initiation set of o_i based on the Equation 1.

With an AOSM, given a manipulation-only plan, we can synthesize the requisite navigation actions to interleave into the plan and support execution. To evaluate the probability of the entire plan, we first learn a portable symbolic vocabulary similar to James et al. [51] (described in Section 2.1.3) but do not include spatial information about the objects or robot in the observations, and then separately learn navigation symbols using the spatial data in an object-centric frame. With the portable symbolic vocabulary, manipulation-plans can be generated, and with the addition of the navigation symbols grounded for a specific environment, we can evaluate the probability of a manipulation-only plan with navigation actions interleaved in.

5.4 Simulation and Hardware Experiments

We test the hypothesis that exploiting the spatial independence property of manipulation options increases sample efficiency and transferability of learned abstractions. First, we investigate the effect of leveraging the spatial independence assumption on the number of samples required to learn a useful set of abstractions for planning. Secondly, we evaluate the effectiveness of transferring abstractions from a training environment to a novel environment. Together, these experiments highlight how AOSMs can be used to efficiently learn and transfer abstractions with only a few number of interactions with the environment.

Coffee Preparation Task We conduct both of our experiments in a simulated mobile manipulation domain, AI2Thor [64], using a coffee preparation task in 15 virtual kitchens. In this task the robot must navigate through a large simulated kitchen and manipulate objects; to successfully make coffee, it must pick up a cup, bring it to a coffee machine, turn on the coffee machine to make the beverage, and then pick up the prepared coffee mug. We assume the robot has access to a set of portable manipulation skills (**PickUp(Mug)**, **ToggleOn(CoffeeMachine)**, **PutIn(Mug,CoffeeMachine)**, **MakeCoffee(Mug,CoffeeMachine)**) that can be reused across different kitchen scenes, but that the agent must construct navigation abstractions for each different scene. AI2Thor provides semantic maps of each scene, which include a 2D occupancy grid of the environment, the number of objects in the environment, their object type and attributes, and their pose. We use 77 different objects, each characterized by a vector of length 108. We also include the 3D position and 1D yaw of the robot's base (4 additional state variables), resulting in a low-level observation vector of 8320 elements.

Simulation Experiment: Spatial Independence for Learning Symbols In the first experiment, our goal is to evaluate how leveraging the spatial independence assumption affects the samples required to construct a symbolic vocabulary that supports planning. We therefore evaluate a state-of-the-art baseline [53] for learning symbols that does not incorporate the spatial independence assumption against an augmentation of the approach that does leverage

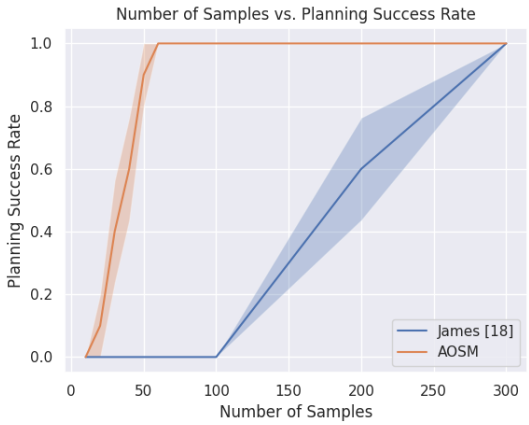


FIGURE 5.4: Learning symbols for the coffee preparation task, without the spatial independence assumption (James et al. [53]) and with the spatial independence assumption (AOSM). We report the number of sampled interactions with the environment against the planning success rate across 10 seeds.

the spatial independence assumption. We report performance as a function of the number of samples from the environment.

Part of the model learning process requires identifying which factors are independence since there is no a priori assumption about the structure of the initiation and effect sets of the skills. Partitioning is done via DBSCAN clustering [35], and the precondition classifiers are learned using a SVM [29] with an RBF kernel (hyperparameters are optimized using grid search. The effect density estimation is performed with a kernel density estimators [105, 94] with a Gaussian kernel, with a grid search over the bandwidth.

Approaches We use a codebase for learning symbols [53] that is state-of-the-art but does not leverage any spatial independence assumptions as our baseline. More details on the algorithm can be found in [53], but in summary: the robot collects transition data in an environment by either randomly navigating to a pose or choosing manipulation skills to execute, and then uses this data to learn a model describing the preconditions and effects of the skills via a partitioning and clustering process. Part of the model learning process requires identifying which factors are independent since there is no a priori assumption about the structure of the initiation and effect sets of the skills.

For these set of experiments, all of the approaches perform a similar procedure. For a

given scene and current step of the plan o, the robot 1) uses rejection sampling to sample a pose l from the associated navigation symbol σ^o 2) uses the path planner to move to location l, and 3) attempts to run the manipulation option o. If the agent fails to successfully execute the manipulation option, the location l is added as a negative sample to the dataset used to train σ^o; the robot repeats these steps until successful execution. When the robot is successful in executing the manipulation option, location l is added as a positive sample to the dataset used to train σ^o, and the robot proceeds to the next plan step. These navigation symbols are trained using Gaussian Process classifiers with an RBF kernel.

There are two important design choices when learning navigation symbols that can be chosen indepedently of each other: 1) which spatial frame are the navigation symbols learned in, and 2) what proposal distribution is used for rejection sampling. In [66], the global map frame is used as the spatial frame and a random distribution for sampling, and we call this baseline **random global**. Learning symbols in the map frame enables the robot to leverage a path planner to generate navigation behaviors, but it means that the robot must relearn the symbols when the scene changes. To exploit the structure of object-centric skills, an object-centric spatial frame can be used to learn the symbols, which the agent can transform into a map frame given a semantic map that includes object pose. This enables the agent to effectively transfer learned information from one map to another. Using an object-centric frame with a random sampling distribution is akin to the approach in James et al. [51], which we term **random object**. However, using a uniform distribution as the proposal distribution is extremely inefficient since the robot will try manipulating objects from locations extremely far from the object. Kaelbling and Lozano-Pérez [57] proposed exploiting the nature of space using a geometric heuristic that samples poses near the object, and so we call the baseline that uses the geometric heuristic for sampling poses and learning in a global map frame **heuristic global**. The final approach learns in an object-centric spatial frame and uses the geometric heuristic to sample poses, which to our knowledge has not been used in conjunction to learn symbols. We call this baseline **heuristic object**, and it corresponds to our assumption. To give an upper-bound on performance, we also evaluate an oracle, which always samples feasible manipulation locations.

To determine how effectively each approach enables learned abstractions to be transferred to different environments, we use investigate two experimental settings: when the agent successfully finishes executing the plan, 1) the scene is reset to the initial configuration and the agent retries executing the plan (the single-scene setting), and 2) a new scene is chosen and the agent retries executing the plan (the multi-scene setting). In the single-scene setting there is no need for transfer and the choice of spatial frame does not matter. This lets us evaluate how important the chosen proposal distribution is for learning navigation symbols. In the multi-scene setting, the agent must also transfer the learned symbols to different scenes, which lets us evaluate how useful the choice of frame is for transfer.

Metrics To evaluate the usefulness of the resulting abstractions, we use Fast Downward [44], an off-the-shelf symbolic planner, to plan using the resulting symbolic vocabulary. We then use a binary metric to determine how useful the representation is for planning: if the resulting plan accomplishes the goal, then the symbolic vocabulary is deemed successful. Otherwise, the symbolic vocabulary is deemed a failure. Our goal is to minimize the interactions required to learn a successful symbolic vocabulary for planning. We collect 1000 transitions with 10 different random seeds.

Results The results of our experiment are in Figure 5.4. As the number of environmental samples increases, the success rate of planning with the symbols improves for both approaches, as expected. Learning with the spatial independence assumption, however, is able to learn a successful symbolic vocabulary with a nearly 100% planning success rate with about 50 samples, where as the baseline approach that does not leverage the spatial independence requires about 300 samples. This is due in part to the fact that, without leveraging the spatial independence assumption, the baseline requires more samples to learn to disentangle spatial information from non-spatial information, which is challenging since the spatial data is continuous. Our approach builds in the disentanglement between the spatial and non-spatial data, easing learning. These results demonstrate that our approach—which structures in the independence assumption—is more sample efficient than state-of-the-art approaches to learning abstractions. Examples of the learned symbolic vocabulary are in Figure 5.5.

```
(:action pick-up-mug-from-table
        :parameters (?r - robot ?m - mug)
        :precondition (and (on-table ?m) (at-table ?r))
        :effect (and (not (on-table ?m)) (on-robot-hand ?m))
)
```

```
(:action PutMugInMachine-partition-0-8
    :parameters (?a - CounterTop ?b - CoffeeMachine ?c - Mug)
    :precondition (and (notfailed) (CounterEmpty2 ?a)
            (MachineOnEmpty ?b) (MugFilledHeld ?c) (AtMachine ?b))
    :effect (and (MugFilledInMachine ?c) (MachineOnHoldingMug ?b)
            (CounterHoldingMug ?a) (not (CounterEmpty2 ?a))
            (not (MugFilledHeld ?c)) (not (MachineOnEmpty ?b)))
```

FIGURE 5.5: Example operators for two manipulation skills with the navigation symbols injected into the preconditions (red highlight). (Left): A learned operator for the **PickUp(Mug)** skill in AI2Thor. Symbols are renamed manually to provide human interpretability (Right): A hand-specified operator for the **PutIn(Mug,CoffeeMachine)** skill in the Spot experiment.

Simulation Experiment: Transfer of Learned Abstractions In the second set of experiments, our goal is to evaluate how AOSMs help transfer learned abstractions to novel environments. For these experiments, we provided a manipulation-only plan that prepares coffee. The robot must construct the navigation symbols that enable it to generate navigation behaviors that enable those actions to be executed. There are two important design choices when learning navigation symbols that can be chosen independently of each other: 1) which spatial frame are the navigation symbols learned in, and 2) what proposal distribution is used for rejection sampling. We evaluate different choices of these design choices in two settings: one where the robot learns symbols in a single scene, and one where it must learn symbols across different scenes (i.e: transfer is necessary). For each task execution in a scene, we report the cumulative total number of manipulation skills the robot executed, until the plan succeeded.

Our results can be see in Figure 5.3. The main takeaway is that learning symbols in an object-centric frame is important for transferability.

FIGURE 5.6: An example demonstration of the Spot building an AOSM and using it to prepare coffee. (Left): Spot navigates around the space, identifies objects, and constructs an AOSM. (Right): With the AOSM and a manipulation-only plan, the Spot can synthesize the navigation abstractions to locomote around the environment to successfully execute the manipulation skills.

Robot Hardware Demonstration We demonstrate the effectiveness of AOSMs for enabling mobile manipulators to plan long-horizon tasks by executing a coffee preparation task on a

Boston Dynamic Spot platform (Figure 5.1b). In this coffee preparation task, the robot must gather coffee grinds and water, pour them both into a coffee maker, close the lid of the coffee maker, and push a button to turn it on. We supply the robot with a set of portable manipulation skills **PickUp(CoffeeGrinds), PickUp(WaterCup), Place(CoffeeGrinds), Place(WaterCup), Pour(WaterCup), Pour(CoffeeGrinds), CloseLid(CoffeeMachine)** and **PushButton(CoffeeMachine)**, whose implementation on the robot can be seen in Figure 5.1b. The objects are scattered around the room, and so the robot must navigate the environment correctly to successfully execute the manipulation skills.

Our demonstration of using an AOSM on a real robot can be seen in full detail in Figure 5.6. We first manually drive the robot around and use an off-the-shelf SLAM implementation to generate a 3D geometric map of the environment which the robot can use to navigate to 3D poses. The robot then constructs a semantic map that captures the spatial pose and semantic attributes of each of the relevant objects in the scene. Once the robot is equipped with a set of manipulation skills, it generates an AOSM of the scene using hand-crafted navigation symbols, which enables it to sample navigation poses that support successfully executing each of its manipulation skills. The robot then uses a hand-specified PDDL of the coffee preparation task to generate the manipulation-only plan using Fast Downward, which results in: **PickUp(WaterCup)**,**Pour(WaterCup)**, **Place(WaterCup)**,**PickUp(CoffeeGrinds)**,**Pour(CoffeeGrinds)**, **Place(CoffeeGrinds)**, and then **CloseLid(CoffeeMachine)**, **PushButton(CoffeeMachine)**. With the AOSM, the robot can synthesize a navigation stack to support plan execution (Figure 5.6).

We time how long it takes the robot to construct an AOSM in 2 different environments. Navigating the environment to observe the objects and then constructing the AOSM takes an average of 82.5 seconds. Executing the plan for the coffee preparation task takes on average 140 seconds. These timings demonstrate the efficacy of AOSMs to enable a robot to rapidly generate the navigation abstractions for supporting task execution.

5.5 Conclusion

We have proven that an Action-Oriented Semantic Map is a sufficient spatial data structure that can be constructed on top of a semantic map for synthesizing a navigation stack to support task planning with manipulation skills. Once a robot has built an AOSM, it can produce plans using its manipulation skills and synthesize a sufficient navigation stack to execute the task plan. This enables robots to autonomously learn perceptual abstractions, given a set of action abstractions (such as those provided in Chapters 4 and 5).

Chapter 6

Constructing Abstractions for Robotic Planning with Mixed Reality

6.1 Introduction

A long-term goal of robotics is designing robots intelligent enough to enter a person's home and perform daily chores for them. This requires the robot to learn specific action and perceptual abstractions that can only be acquired after entering the home and interacting with the humans living there. For example, there may be a trinket that the robot has never encountered before, and the owner might want to instruct the robot on how to handle the item (i.e., object manipulation information), as well as directly specify where the item should be kept (i.e., navigation information). To approach this problem, one must consider two sub-problems: a) the agent's representation of **object manipulation actions** and **semantic information about the environment**, and b) the method with which an agent can **learn this knowledge from a teacher**.

Semantic maps provide a representation sufficient for navigating an environment [126], but map information alone is insufficient for enabling object manipulation. Conversely, there are knowledge bases that store requisite object manipulation information [129, 71, 15, 101, 97], but do not help with navigation or grasping in novel orientations. Previous studies [38, 103, 46] have shown that Mixed Reality (MR) interfaces are effective for specifying navigation commands and programming egocentric robot behaviors. However, none of these works have demonstrated the use of MR interfaces for teaching high-level object manipulation actions,

and semantic information of the environment.

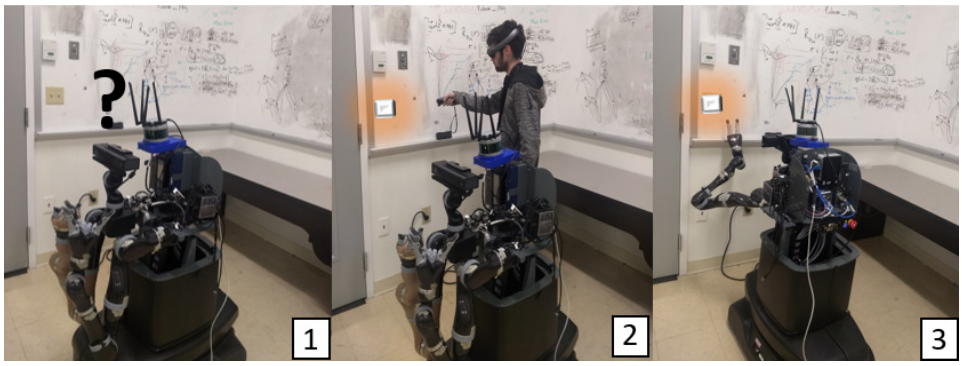


FIGURE 6.1: Our MR system being used to generate an AOSM so the robot can flip a light switch. **(1)**: Initially, the robot does not know the location of the light switch, how to grasp it, nor how to turn it off. **(2)**: A human using MR teaches the robot the object's global pose Γ, the "grasp" attribute, and the initiation and termination pose for the "turn off" action (highlighted in orange).**(3)**: The robot is now able to autonomously plan to navigate to the lights, motion plan to grasp the light switch, and execute the policy to flip it off.

Our contribution is a system that enables humans to teach robots both object manipulation actions—in a local object frame of reference—and b) semantic information about objects in a global map. We use a Mixed Reality Head Mounted Display (MR-HMD) to enable humans to teach a robot a plannable representation of their environment. By *plannable*, we mean structured representations that are searchable with AI planning tools [75]. By *teach*, we mean having the human explicitly provide information necessary for instantiating our representation. Our representation, the Action-Oriented Semantic Map (AOSM), enables robots to perform complex object manipulation tasks that require navigation around an environment. To test our system for building AOSMs, three novice humans used our MR interface to teach a robot an AOSM, allowing the robot to autonomously plan to navigate to a bottle, pick it up, and throw it out. In addition, we report the quantitative results of two expert users who demonstrated the power of learning AOSMs via MR by also teaching a robot to autonomously plan to flip a light switch off (Figure 6.1) and manipulate a sink faucet to the closed position. To the best of our knowledge, this is the first work that presents a learnable representation for planning manipulation and navigation tasks on a robot via an MR interface.

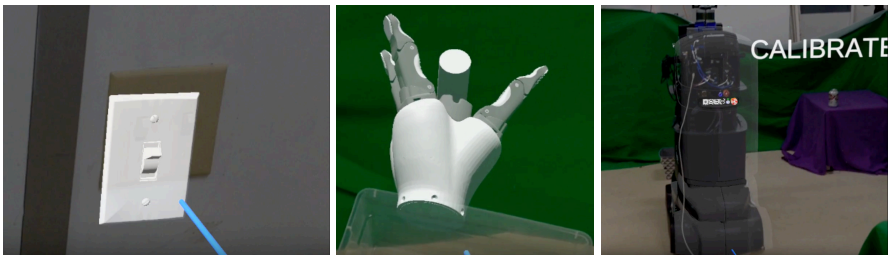


FIGURE 6.2: The perspective of a user with our MR interface (all visualizations here are from the actual interface). **Left:** A closeup image of a user grounding the global pose Γ of the light switch using our MR interface. **Middle:** A user specifying the terminating pose for the "throw away" action of the bottle object, with the annotated grasp pose visualized as a white robot gripper. **Right:** An image of the virtual robot overlaid on top of the real robot while calibrating the MR-HMD's map with the robot's map M. The text "CALIBRATE" indicates to the user what information they are specifying.

6.2 Action-Oriented Semantic Maps

We first formalize AOSMs by describing the object classes, object instances, and object manipulation actions it contains, which are all defined with a local frame. Next, we illustrate the grounding of an object's semantic information to a global frame of reference. Lastly, we describe our method of using MR to build an AOSM from a human trainer.

6.2.1 Defining Action-Oriented Semantic Maps

An Action-Oriented Semantic Map is a tuple $AOSM = \langle C, O, M, A \rangle$, where C is a set of object classes; O is a set of objects instantiated from C, where we define instantiation as assigning all attributes of a class to values representing a real-world object; M is a 2D occupancy grid of the environment; and A is a set of high-level object-specific actions parameterized over objects in O.

Each high-level action $a \in A$ is akin to an option [117]. Given an object from $o \in O$ and a high-level action a, a "policy", an "initiation set", and a "termination set" for the option is specified. In the next subsection we will describe using MR to acquire each of these components in detail.

Each class within C is constructed with a set of attributes α, a 6D local frame Λ, a global pose Γ, and a kinematic mesh model τ. The 6D (position and orientation) local frame Λ is

necessary to define spatial attributes for high-level actions with respect to an object, regardless of the global pose Γ. The model τ is defined in the local frame with respect to Λ.

The 3D kinematic mesh model of each object class, τ, is specified with respect to Λ. The purpose of the 3D mesh is twofold. Firstly, it provides an interface for the teachers to manipulate and specify the local coordinate frame of an object so that the skill specification is intuitive. Secondly, it allows a manipulable interface to the object in MR allowing the trainer to visualize and manipulate a virtual object, which is the typical mode of interaction with an object in MR.

Each class has a set of attributes α, akin to class attributes in Object-Oriented Markov Decision Processes [33]. Attributes are used to represent information required for planning object manipulation behaviors. Spatial attributes, like "grasp", which defines how the object should be grasped with respect to τ, are specified with respect to Λ. In our experiments, the only attribute we have for our classes is "grasp", but other complex domains require more attributes for planning.

Once objects are instantiated, they have a global pose Γ in the map, and the agent knows where the object is and can navigate to it. Moreover, a high-level action a is defined with respect to the local frame Λ of the object class c. Specifically, the policy, initiation set and termination set of a high-level action a are all defined in the object class's local frame. This allows transfer of learned high-level actions to different objects within the same class and to different poses in the global frame, enabling the robot to reproduce and generalize the learned skill later when executing a plan.

Whenever an object is instantiated, Γ is grounded to M, and τ is rendered by the MR-HMD based on Γ. Γ is the pose of the local frame's origin with respect to M's origin. The purpose of the global pose Γ is so that information defined with the local frame Λ for an object is now grounded within the map M, enabling the robot to know where in the environment it should navigate to in order to perform object manipulation behaviors. The purpose of rendering τ in MR is so that the teacher can specify Γ by dragging the virtual object model, and directly view whether Γ is correctly specified (i.e: if the virtual τ is overlaid on top of the real object).

6.2.2 Instantiating AOSMs with Mixed Reality

In order to ground the poses of the virtual items to our semantic map, the map maintained by the MR-HMD must be linked to the robot map M. MR-HMDs already have a built-in capability to make a 3D mesh model of the environment for mapping, which is used for localization. However, there is no inherent link between the MR-HMD's map and the robot map M, which is required to use MR to specify an object's global pose Γ. To resolve this issue, a static transform that defines how to convert global poses in the virtual environment maintained by the MR-HMD to poses in the robot's map must first be defined (Figure 6.2). Our method of performing this calibration using MR is explained in Section 6.3.

After calibrating the MR-HMD and the robot map, the user can teach the robot object manipulation and semantic information of the environment (as described in Section 6.3.2). The user is presented with a list of object classes C from the AOSM. When the user selects a class, a virtual representation of the object's mesh τ is visualized in front of the user as a 3D mesh (Figure 6.2), and an interaction process is initiated, where the user supplies each of the necessary attribute values within the object's local frame Λ. For example, in case of the "grasp" attribute, the user is presented with a visualization of the object's mesh τ along with a virtual model of the robot's end effector (Figure 6.2). The user is then able to pose the virtual end effector to grasp the virtual object mesh τ. Users are able to supply manipulation information using the high-level actions for an object by filling in the parameters. The users first select an object to add an high-level action to, and then manipulate a virtual representation of the object's mesh τ into the desired initiation and termination poses. Because τ is an articulated 3D mesh model, users can specify the initiation and termination poses by selecting a link with the controller, and then manipulate it with their controller to the desired pose. For the purposes of our MR interface implementation, these initiation and termination poses were in terms of the mobile-manipulator's end effector so that our system could check when these poses were reached. This process allows users to not have to specify any low-level manipulation control such as environment-specific grasp operations.

Because there are several design choices for the MR interface that can be made based on the desired task, we selected several household tasks and conducted an iterative design study

to understand what factors were important for enabling novice humans to teach a robot an AOSM. This design process allowed us to include features that were not initially considered by the expert designers, but were desired by the novice users.

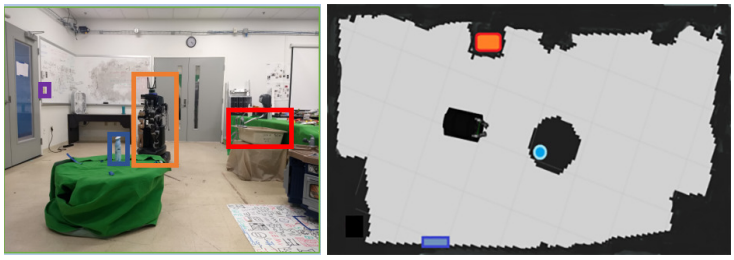


FIGURE 6.3: Images from our Household AOSM. **Left:** One perspective image with our three classes C being instantiated with objects O (light switch (purple), bottle (blue), sink (red)) and robot (orange). **Right:** The 2D occupancy grid M in our Household AOSM (Colored shapes and robot added to the map for visualization purposes).

6.3 Iterative Design Study

We conducted an iterative design study with two expert users (two of the project researchers) and three novice users in order to design and improve our MR interface, as well as demonstrate the capabilities of AOSMs.[1]

6.3.1 Study Task

To demonstrate that an AOSM can be built by a human using MR, we selected several household tasks for a mobile manipulator to perform, which we represent within what we term the "Household AOSM". We chose three different chores: throwing away bottles, turning off light switches, and closing sink faucets. Our test environment is shown in Figure 6.3.

Each element of our Household AOSM ($AOSM = \langle C, O, M, A \rangle$) is defined as follows:

- C: a list of three object classes: bottle (a drinking container with no kinematic articulation), faucet (a sink faucet, which has a revolute joint connected to a sink base, which could be closed), and light switch (which has a revolute joint connected to the wall). Each of the classes have a 6D local frame Λ, a global pose Γ, and a kinematic mesh

[1] A video can be found at https://youtu.be/-09b250TTe8

model τ. In order to keep the AOSM as simple as possible, we only encoded one attribute: "grasp", which represents a 6D pose in the class frame Λ that indicated how to grasp the object for manipulation.

- O: a list of the instantiated objects from the list of classes. In our experimental space, we had one bottle, one light switch, and one sink faucet. Rather than requiring users to build τ from scratch, we supplied various primitive shapes and predefined object models for the user to choose from to represent the objects, which is reasonable considering there are many existing object models freely available to be downloaded [1]. Therefore, when instantiating objects, users were responsible for defining the "grasp" attribute needed for the high-level action manipulation actions, as well as the global pose Γ of the object within the map M which is needed for navigation (Figure 6.2).

- M: a 2D occupancy grid M that represents the experimental space (Figure 6.3). The map is updated with new semantic information when an object o is instantiated and its global pose Γ is grounded in the map. It is this underlying map that enables the robot to autonomously plan navigation around the environment.

- A: For our demonstration, we paired one high-level action with each object to represent the three chores. However, it should be be noted our framework is flexible enough to allow an arbitrary number of high-level actions to be defined throughout the interaction by the user. Our actions are as follows:

    1. For the bottle class objects, the high-level action "throw away" was meant to pick up a bottle and move it to a trash can in a fixed spot (Figure 6.2).
    2. For the light switch class, a "turn off" high-level action was meant to flip the switch to the off position from the on position.
    3. For the sink faucet class, a "close faucet" high-level action was meant to close the faucet.

  Users were responsible for using our MR interface to define the initiation and termination poses of these actions, while the policy attached π was implemented using an

existing motion planner to move the robot's end effector to the grasp pose with respect to the initiation pose, and then compute and execute a motion to manipulate the object to the termination pose. The policy was first planned within the local frame Λ, and then transformed into the global map frame based on Γ, enabling the robot to move its end effector to the necessary locations in the map to manipulate the object. We can also use Dynamical Movement Primitives [50] as a policy within the local frame Λ.

Our study was implemented on a Kinova Movo with a single 7 DoF arm. Movo is equipped with the capability to make a 2D occupancy map of its environment using a LIDAR sensor, as well as localize and navigate to specified poses. When planning any of the object manipulation actions, the robot would autonomously move its base between 0.8 and 1.25 meters behind the object's global pose Γ, depending on what the "grasp" attribute and global pose Γ of the object was, enabling the agent to execute the local policy and manipulate the object into its termination pose from the initiation pose (Figure 6.1). While this range of approach distances was chosen by hand for the purposes of completing our specified chores, they could in practice be specified by the user via the MR interface. For all of these motion behaviors, we use the motion and path planning stack that is included with the Movo robot. By supplying our metric map M to the path planning stack, we are able to autonomously navigate the robot to specific points while avoiding occupied space.

6.3.2 Mixed Reality Interface

The two most commercially-available MR-HMDs are the Microsoft HoloLens and the MagicLeap. We have previously used the Microsoft HoloLens for facilitating human-robot interactions [103], but chose to use the Magic Leap for this work because it provides higher precision head-pose estimation. However, the following work can be applied to any MR-HMD system. Our codebase for the MR interface is publicly available.[2]

We used Unity, a 3D game engine, to develop the virtual environment for the MR interface, by developing a scene that maintains virtual objects, and deploy it to the Magic Leap. By connecting the Magic Leap to a ROS network, we are able to share information between

[2] https://github.com/h2r/ActionOrientedSemanticMaps

the MR interface and a ROS-enabled robot. (A more detailed description of how this can be done may be found in Whitney et al. [127]). Crucially, our system is developed such that no objects in the Unity scene need to be pre-instantiated; the user is able to construct the scene completely at runtime via our MR interface.

The Unity-ROS interface allows the Unity scene to output information on the ROS network to communicate to the robot, or listen to information from the robot to update the virtual scene. In general, MR interfaces enable users to see visualizations of 3D meshes overlaid on top of the physical workspace, as well as interact with these visualizations using controllers or hand gestures. We leverage MR to enable users to instantiate virtual representations of the objects from a set of classes using an MR menu, supply attribute and high-level action information needed for object manipulation by interacting with the model τ of objects in order to specify initiation and termination poses, and ground objects to the map (i.e: specify Γ) for the purpose of navigation by dragging the virtual objects over their real-world counterparts (Figure 6.1).

To define the static transform that is needed to convert Unity poses to ROS poses, we enable users to drag a virtual model of the robot over the real one to align them together (Fig 6.2), similarly to how they would ground the global pose Γ of an object. After the user drags the virtual robot over the real one, we save the transformation from the virtual robot pose to the real robot pose as the static transform from Unity to ROS poses. With this transform, we now have a way to use a MR-HMD to ground poses of objects in the robot's map. More information on pose transformation between MR-HMDs and robot maps can be found in Whitney et al. [127].

6.3.3 Rapid Iterative Testing and Evaluation

We took a Rapid Iterative Testing and Evaluation (RITE) [87] approach to quickly identify and fix issues with the system.

For the purpose of the iterative design study, we limited the Household AOSM by removing the light switch and sink faucet from the Household AOSM, and only focused on the bottle object. Users in our study were instructed to specify the "grasp" attribute for the bottle

class, the initiation and termination pose of the "throw away" action, and the global pose Γ of the object in the map. The expert users then completed the full Household AOSM by also handling the sink faucet and light switch.

We built an initial interface for the system, and tested and iterated on the design of the interface. We tested the initial system with two expert users (two of the project researchers), iterated on the design, and then tested and iterated with three novice users, who used our interface until they successfully performed the task. We then tested the final system with the expert users.

System V0

We started with an initial design for the MR interface, system V0, that was derived from previous MR interfaces we have used with robotic systems [103]. The interface allows users to drag virtual representations over objects in the real world that they want to interact with, as described in the Section 6.3.2. However, we noticed that users have slight calibration issues with hand gestures (i.e., it is hard to accurately capture hand gestures), such that we decided to use a hand controller instead to circumvent this calibration issue. We drew inspiration from the MagicLeap's toy app which uses the hand controller to orient objects in front of the controller. Thus, our initial design improved on our previous interfaces by introducing a hand controller to replace gesture in order to attempt to address user issues with positioning virtual representations.

We then tested system V0 with the expert users. We quickly found that the expert users would sometimes unknowingly misalign the virtual representations over the real-world objects. For example, after specifying the global pose for a specific bottle, the user would walk around the room to specify other attribute and action information; however, after physically walking in the room, and thus changing perspective, the user would notice that global pose of the object appeared misaligned with the real-world object. We implemented an intervention (i.e., edit) function for the sequence of human actions for an item, such that the MR interface would permit users to respecify and edit information in the AOSM. We also noticed

that scenes would sometimes become cluttered with specifications for multiple items; consequently, we implemented a color scheme for objects to make differentiation between virtual objects easier.

System V1

We tested system V1 with the first novice user. The major observation from the user concerned the sensitivity of the hand controller, which the user found to be overly sensitive to touch and thus difficult to use to precisely position the virtual representations. We reduced the sensitivity of the controller for the subsequent version.

System V2

Feedback from the second novice user centered on a desire to know what action they were specifying for the robot at any given time, as they sometimes lost their place in the sequence while adding states. We addressed this issue for the subsequent version by implementing a text display in the virtual workspace that identifies whether they were specifying action information, object pose/attribute information or calibration information (Figure 6.2).

System V3

The third novice user tested system V3, and did not have any major issues with using the system.

We therefore proceeded to test system V3 with the original expert users. The experienced users were able to use this final version of the system to complete more complex cleanup tasks, such as turning off sinks and turning off lights.

6.3.4 Overall Impressions of System

The interviews with the three novice users revealed that, overall, they liked the system and found the system intuitive when they used it.

One notable consideration revealed during user testing concerns sensitivity of the hand controller; users varied in how sensitive they wanted the hand controller to be in response

to their input. The first novice user found the hand controller too sensitive (prompting a reduction in sensitivity); the second novice user did not report any issues with sensitivity; the third novice user found it not sensitive enough.

Ultimately, the insight from novice user expectations of the system helped guide the design of the final system. The two expert users tested the final system with complex tasks of flipping light switches and turning off faucets.

As predicted, a major problem of the MR interface was the decalibration due to drift. Over time, users would see the virtual objects drift away from their calibrated poses because the MR-HMD was not able to accurately localize itself within a large space with a constantly moving user, making their groundings inaccurate for the robot. To resolve this, multiple interventions to edit specified information was required. Although allowing users to readjust the transform between Unity and ROS made this issue less pressing, users reported that it was cumbersome to do this repeatably. Therefore, high-precision pose tracking is crucial for using MR to specify semantic information about the robot's environment. Another option is to incorporate autonomous perception modules beyond SLAM into the MR interface, such as object detection and pose estimation, which can leverage the user-specified information to enable the object's pose estimate to be robust to decalibration due to drift between the robot and the MR-HMD. The human-specified information can be used in conjunction with iterative computer vision algorithms, like ICP for pose registration [27], which are are sensitive to initial starting points and would benefit from human input.

6.4 Results

In order to evaluate our system, we demonstrated that our final MR interface enables an expert user to build the full Household AOSM to sufficiently perform all three high-level behaviors: navigating to a bottle and throwing it away, navigating to a light switch and turn it off, and navigating to a sink faucet to close it. For each object, users were tasked with specifying an object's Γ, "grasp" attribute, and the initiation and termination pose for the associated action (as discussed in Section 6.3). Once the users trained the robot with this information, the robot was able to plan with the Household AOSM. For planning, the agent

autonomously performs a multi-step plan of a) moving to a position near the object's Γ (as described in Section 6.3), b) grasping the object based on the "grasp" attribute and initiation pose, and c) manipulating the object into its termination pose (Figure 6.1). Whenever the agent fails to execute the plan, we enable the user to intervene (i.e., edit) any specified information.

To quantify our evaluation, we recorded both the total time it took to teach the high-level action, specify the global pose of the instantiated object, and have the robot autonomously plan to execute the behavior. In addition to the total time, we also record the number of interventions required until a successful plan is executed.

There is no fair baseline comparison to our method because we are the first work to present a representation that has both semantic and planning information that is learnable via MR. Comparing against direct teleoperation or kinesthetic teaching in the real workspace is not a valid baseline because there is no way to specify the position and orientation of all the links in an object by controlling the robot's arm, which is needed for specifying the initiation and termination pose of an action. A 2D visual interface that uses our metric map M is also not a valid baseline because it does not provide any geometric information about the location of the objects, only geometric information of obstacles slightly above floor height, and therefore can not be used to label object pose information. Making a 3D static map of the environment and visualizing it on a 2D monitor is also not a valid baseline because user intervention requires a dynamically updated model of the room to respecify information, which a static map does not provide. Continuously mapping a large 3D dynamical scene with on-board robot sensor data is not a fair comparison because it requires the user to move the robot to acquire desired view points, introducing a conflating factor of robot control that is not encountered with the MR interface. A projector-based augmented reality interface is also not a feasible comparison because it does not provide a method for manipulating or visualizing 3D kinematic mesh models, which is necessary for defining our high-level actions.

For the bottle task, our expert user took 31 seconds, and had 0 interventions. For the light switch task, the expert user took 91 seconds, and had 4 interventions. For the sink faucet task, the expert user took 45 seconds and 3 interventions. Note that the total times include all of the interventions (i.e: the timer was not stopped between each intervention). Therefore, the light

switch and sink faucet task have longer reported times due to the number of interventions needed to complete the task, but the average intervention time for the light switch task was 22.75 seconds, and 15 seconds for the sink faucet. It took less than 2 minutes to complete each of our tasks.

6.5 Conclusion

We present a solution to enable users to teach robots both high-level actions for object manipulation and semantic map representations for navigation via an MR interface. We introduced Action-Oriented Semantic Maps (AOSMs), a plannable representation which can enable a human to teach a robot information needed for object manipulation and navigation through MR. To demonstrate that humans can build AOSMs to plan for complex object manipulation tasks, we showed that novice and expert users can program a mobile manipulator to perform three tasks: picking up a bottle and throwing it in the trash, closing a sink faucet, and flipping a light switch. Ultimately, our contributed methods and interfaces enable humans to intuitively and effectively teach mobile manipulators both perceptual and action abstractions that are sufficient for planning in long-horizon tasks.

Chapter 7

Conclusion

The thesis of this dissertation is that algorithms and interfaces that exploit spatial structure can enable mobile manipulators to effectively acquire perceptual and action abstractions via autonomous interaction with the environment, or from a human teacher. As we have discussed, action abstractions are crucial for enabling robots to overcome the inherent long-horizon nature present in tasks that humans face every day. And in order to leverage action abstractions, robots must also acquire perceptual abstractions that can provide the agent to plan for a wide range of tasks, and in novel environments. While acquiring both of these abstractions through autonomous interaction with the environment offers avenues to scalable robot learning, robots still ultimately also need to learn user-specific abstractions directly from humans to adapt to their preferences and lifestyles. This dissertation has demonstrated that by leveraging algorithms and interfaces (like MR devices) that exploit spatial structure, both types of abstractions (perceptual and action) can be acquired effectively from both sources of information (autonomous interactions and from human teachers).

While this dissertation has offered solutions to key problems in enabling robots to enter society and help humans at large, there are still critical assumptions made across the works that need to be relaxed in order to be more practical. Most critically, we assume our state is fully observable, which is practical in cases where the robot can view the object it is manipulating entirely, or when the robot has premapped and environment it intends to interact with. But in dynamic scenes where other agents are present, it is unreasonable to assume the robot will maintain accurate information about the state of the world. Future work will address how to learn perceptual and action abstractions for mobile manipulators in the face of partially

observable domains. In addition, we have proposed methods for robots to autonomously acquire action abstractions (Chapters 3 and 4) and perceptual abstractions (Chapter 5), and separately proposed interfaces and methods for humans to teach action and perceptual abstractions (Chapter 6). Future work will combine all of these methods together, and investigate the novel problems that arise when needing to resolve conflicts in human-specified abstractions, and robot-specific learned abstractions.

Bibliography

[1] Unity 3D. Unity Asset Store. URL https://assetstore.unity.com/.

[2] Ben Abbatematteo, Stefanie Tellex, and George Konidaris. Learning to generalize kinematic models to novel objects. *Proceedings of the 3rd Conference on Robot Learning*, 2019.

[3] Ben Abbatematteo, Eric Rosen, Stefanie Tellex, and George 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.

[4] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[5] M. Tuluhan Akbulut, Utku Bozdogan, Ahmet Tekden, and Emre 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, 2021.

[6] Mete Akbulut, Erhan Oztop, Muhammet Yunus Seker, Hh X, Ahmet Tekden, and Emre Ugur. Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing. In Jens Kober, Fabio Ramos, and Claire 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 2021.

[7] Barrett Ames, Allison Thackston, and George Konidaris. Learning symbolic representations for planning with parameterized skills. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 526–533. IEEE, 2018.

[8] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[9] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 2009.

[10] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[11] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20. Citeseer, 1997.

[12] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. *Artificial Intelligence Research and Development*, pages 363–371, 2008.

[13] Emanuele Bastianelli, Domenico Bloisi, Roberto Capobianco, Guglielmo Gemignani, Luca Iocchi, and Daniele Nardi. Knowledge representation for robots through human-robot interaction. *arXiv preprint arXiv:1307.7351*, 2013.

[14] Patrick Beeson and Barrett Ames. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 928–935. IEEE, 2015.

[15] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoğlu, and Georg Bartels. Know rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2018.

[16] Pascal Bercher, Ron Alford, and Daniel Höller. A survey on hierarchical planning-one abstract idea, many concrete realizations. In *IJCAI*, pages 6267–6275, 2019.

[17] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446), 2019.

[18] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[19] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[20] Gabriele Bolano, Christian Juelg, Arne Roennau, and Ruediger Dillmann. Transparent robot behavior using augmented reality in close human-robot interaction. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–7. IEEE, 2019.

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

[22] Alexandre Campeau-Lecours, Hugo Lamontagne, Simon Latour, Philippe Fauteux, Véronique Maheu, François Boucher, Charles Deguire, and Louis-Joseph Caron L'Ecuyer. Kinova modular robot arms for service robotics applications. In *Rapid Automation: Concepts, Methodologies, Tools, and Applications*, pages 693–719. IGI Global, 2019.

[23] Tathagata Chakraborti, Sarath Sreedharan, Anagha Kulkarni, and Subbarao Kambhampati. Projection-aware task planning and execution for human-in-the-loop operation of robots in a mixed-reality workspace. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4476–4482. IEEE, 2018.

[24] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3381–3388. IEEE, 2017.

[25] 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.

[26] Ching-An Cheng, Xinyan Yan, Nolan Wagener, and Byron Boots. Fast Policy Learning through Imitation and Reinforcement. *arXiv e-prints*, art. arXiv:1805.10413, May 2018.

[27] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Object recognition supported by user interaction for service robots*, volume 3, pages 545–548. IEEE, 2002.

[28] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.

[29] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

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

[31] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

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

[33] Carlos Diuk, Andre Cohen, and Michael 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.

[34] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[35] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[36] 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.

[37] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[38] Samir Yitzhak Gadre, Eric Rosen, Gary Chien, Elizabeth Phillips, Stefanie Tellex, and George Konidaris. End-user robot programming using mixed reality. In *Proceedings of the IEEE International Conference on Robotics and Automation (in press)*. IEEE, 2019.

[39] Cipriano Galindo, Juan-Antonio Fernández-Madrigal, Javier González, and Alessandro Saffiotti. Robot task planning using semantic maps. *Robotics and autonomous systems*, 56 (11):955–966, 2008.

[40] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[41] Nakul Gopalan, Eric Rosen, GD Konidaris, and Stefanie Tellex. Simultaneously learning transferable symbols and language groundings from perceptual data for instruction following. *Robotics: Science and Systems XVI*, 2020.

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

[43] Marc Hanheide, Moritz Göbelbecker, Graham S Horn, Andrzej Pronobis, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, et al. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*, 247:119–150, 2017.

[44] Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[45] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[46] Baichuan Huang, Deniz Bayazit, Daniel Ullman, Nakul Gopalan, and Stefanie Tellex. Flight, camera action! Using natural language and mixed reality to control a drone. *ICRA*, 2019.

[47] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[48] A.J. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1547–1554, 2002.

[49] Auke J Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in neural information processing systems*, pages 1547–1554, 2003.

[50] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[51] Steven James, Benjamin Rosman, and George Konidaris. Learning portable representations for high-level planning. In *International Conference on Machine Learning*, pages 4682–4691. PMLR, 2020.

[52] Steven James, Benjamin Rosman, and George Konidaris. Autonomous learning of object-centric abstractions for high-level planning. In *International Conference on Learning Representations*, 2021.

[53] Steven James, Benjamin Rosman, and George Konidaris. Autonomous learning of object-centric abstractions for high-level planning. In *International Conference on Learning Representations*, 2022.

[54] Yuqian Jiang, Fangkai Yang, Shiqi Zhang, and Peter Stone. Task-motion planning with reinforcement learning for adaptable mobile service robots. In *IROS*, pages 7529–7534, 2019.

[55] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[56] Tom Jurgenson and Aviv Tamar. Harnessing reinforcement learning for neural motion planning. *arXiv preprint arXiv:1906.00214*, 2019.

[57] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[58] Gregory Kahn, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. Plato: Policy learning using adaptive trajectory optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3342–3349. IEEE, 2017.

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

[60] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

[61] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Robotics Autom.*, 3:43–53, 1987.

[62] Alexander Khazatsky, Ashvin Nair, Daniel Jing, and Sergey 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.

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

[64] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

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

[66] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

[67] George Dimitri Konidaris and Andrew G Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

[68] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.

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

[70] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *arXiv preprint arXiv:1907.03146*, 2019.

[71] Norbert Krügera, Christopher Geibb, Justus Piaterc, Ronald Petrickb, Mark Steedmanb, Florentin Wörgötterd, Aleš Udee, Tamim Asfourf, Dirk Krafta, Damir Omrcene, et al. Object-action complexes: Grounded abstractions of sensorimotor processes.

[72] Dennis Krupke, Frank Steinicke, Paul Lubos, Yannick Jonetzko, Michael Görner, and Jianwei Zhang. Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[73] Benjamin Kuipers. The spatial semantic hierarchy. *Artificial intelligence*, 119(1-2):191–233, 2000.

[74] Andrey Kurenkov, Ajay Mandlekar, Roberto Martin-Martin, Silvio Savarese, and Animesh Garg. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. *arXiv preprint arXiv:1909.04121*, 2019.

[75] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems*, pages 8733–8744, 2018.

[76] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

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

[78] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[79] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

[80] Xiaolong Li, He Wang, Li Yi, Leonidas Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. *arXiv preprint arXiv:1912.11913*, 2019.

[81] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

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

[84] Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, and Robin De Keyser. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86:13–28, 2016.

[85] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh 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.

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

[87] Michael C Medlock, Dennis Wixon, Mark Terrano, Ramon Romero, and Bill Fulton. Using the rite method to improve products: A definition and a case study. *Usability Professionals Association*, 51, 2002.

[88] K. Mülling, 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.

[89] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[90] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[91] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

[92] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017.

[93] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[94] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065, 1962.

[95] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan 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.

[96] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54 (5):1–35, 2021.

[97] David Paulius and Yu Sun. A survey of knowledge representation and retrieval for learning in service robotics. *arXiv preprint arXiv:1807.02192*, 2018.

[98] Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[99] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *2012 IEEE International Conference on Robotics and Automation*, pages 3515–3522. IEEE, 2012.

[100] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[101] Karinne Ramirez-Amaro, Michael Beetz, and Gordon Cheng. Transferring skills to humanoid robots by extracting semantic representations from observations of human activities. *Artificial Intelligence*, 247:95–118, 2017.

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

[103] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. Communicating robot arm motion intent through mixed reality head-mounted displays. In *International Symposium on Robotics Research*, 2017.

[104] Eric Rosen, Nishanth Kumar, Nakul Gopalan, Daniel Ullman, George Konidaris, and Stefanie Tellex. Building plannable representations with mixed reality. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11146–11153. IEEE, 2020.

[105] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832, 1956.

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

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

[108] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[109] Stefan 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.

[110] Stefan Schaal and Christopher G Atkeson. Constructive incremental learning from only local information. *Neural computation*, 10(8):2047–2084, 1998.

[111] Muhammet Yunus Seker, Mert Imre, Justus Piater, and Emre Ugur. Conditional neural movement primitives. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[112] Seiji Shaw, Ben Abbatematteo, and George Konidaris. Rmps for safe impedance control in contact-rich manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2707–2713, 2022.

[113] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[114] Thorsten Spexard, Shuyin Li, Britta Wrede, Jannik Fritsch, Gerhard Sagerer, Olaf Booij, Zoran Zivkovic, Bas Terwijn, and Ben Krose. Biron, where are you? Enabling a robot to learn new places in a real home environment by integrating spoken dialog and visual localization. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 934–940. IEEE, 2006.

[115] Freek Stulp and Olivier Sigaud. Path integral policy improvement with covariance matrix adaptation. *arXiv preprint arXiv:1206.4621*, 2012.

[116] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[117] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999.

[118] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

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

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

[121] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

[122] Tarik Tosun, Eric Mitchell, Ben Eisner, Jinwook Huh, Bhoram Lee, Daewon Lee, Volkan Isler, H Sebastian Seung, and Daniel Lee. Pixels to plans: Learning non-prehensile manipulation by imitating a planner. *arXiv preprint arXiv:1904.03260*, 2019.

[123] Ming-June Tsai. *WORKSPACE GEOMETRIC CHARACTERIZATION AND MANIPULABILITY OF INDUSTRIAL ROBOTS (KINEMATICS)*. The Ohio State University, 1986.

[124] Maximilian Ulmer, Elie Aljalbout, Sascha Schwarz, and Sami Haddadin. Learning robotic manipulation skills using an adaptive force-impedance action space. *arXiv preprint arXiv:2110.09904*, 2021.

[125] Michael Walker, Hooman Hedayati, Jennifer Lee, and Daniel Szafir. Communicating robot motion intent with augmented reality. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 316–324. ACM, 2018.

[126] Matthew R Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. Learning semantic maps from natural language descriptions. In *Robotics: Science and Systems*, 2013.

[127] David Whitney, Eric Rosen, Daniel Ullman, Elizabeth Phillips, and Stefanie Tellex. ROS reality: A virtual reality framework using consumer-grade hardware for ROS-enabled robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[128] Jason Wolfe, Bhaskara Marthi, and Stuart Russell. Combined task and motion planning for mobile manipulation. In *Twentieth International Conference on Automated Planning and Scheduling*, 2010.

[129] Florentin Wörgötter, Alejandro Agostini, Norbert Krüger, Natalya Shylo, and Bernd Porr. Cognitive agents—a procedural perspective relying on the predictability of object-action-complexes (oacs). *Robotics and Autonomous Systems*, 57(4):420–432, 2009.

[130] Mandy Xie, Karl Van Wyk, Ankur Handa, Stephen Tyree, Dieter Fox, Harish Ravichandar, and Nathan D Ratliff. Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration. In *6th Annual Conference on Robot Learning*.

[131] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The international journal of Robotics Research*, 4(2):3–9, 1985.

[132] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.