Learning and Translating Temporal Abstractions of Behaviour across Humans and Robots

Tanmay Shankar
August 29th, 2024
CMU-RI-TR-24-63

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:

Jean Oh, *chair*	Carnegie Mellon University
David Held	Carnegie Mellon University
Shubham Tulsiani	Carnegie Mellon University
Amy Zhang	University of Texas, Austin

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

Abstract

Humans are remarkably adept at learning to perform tasks by imitating other people demonstrating these tasks. Key to this is our ability to reason abstractly about the high-level strategy of the task at hand (such as the recipe of cooking a dish) and the behaviours needed to solve this task (such as the behaviour of pouring liquid into a pan), while ignoring irrelevant details (such as the precise angle at which to pour).

In this thesis, we describe steps towards imbibing robots with these abilities; i.e. to learn and translate temporal abstractions of behaviour across humans and robots. In Part II, we first explore the question "How can we learn and represent temporal abstractions of agent behaviours and their effects on their environment?", We present two methods to do so in Chapter 2 and Chapter 3, adopting an unsupervised representation learning perspective to skill learning.

In Part III, we then address the question "How can we understand demonstrator task strategies in terms of these abstractions, and translate these to corresponding abstractions for a robot to execute?". Specifically, In Chapter 4, we explore how agents can learn correspondences between their own skills, and those of a demonstrator, or how we can *translate* skills from demonstrators to agents, inspired by unsupervised machine translation in natural language.

In Part IV, we begin to consider the effects of agent behaviors on their environments. In Chapter 5, we consider how agents can jointly reason about the skills they execute and the effects these skills have on the objects and environments that these agents interact with. We do this by extending our prior skill learning work to learn temporally abstract representations of agent-environment interactions.

In Part V, we revisit these questions; addressing translating temporal abstractions of behaviour from humans to robots from a perspective of imitating desired environmental change. In Chapter 6, we explore how we can translate environment-aware task strategies across humans and robots. We introduce TransAct, a framework to address this. TransAct empowers robots to consume in-domain human task demonstrations, retrieve and compose corresponding robot-environment interactions with similar environmental effects to perform similar tasks themselves in a zero shot manner, without access to paired demonstrations or dense annotations.

In Part VI, we explore other application domains that our work enables. In particular, in Chapter 7, we explore the domain of encouraging older adults to exercise more with an interactive robot artist. We employ the idea of mapping abstract representations from human exercise routines to robot paint strokes.

In Part VII, we offer a broader perspective on the paradigm of unsupervised translation of temporal abstractions of behaviour across humans and robots. We finally discuss future research directions enabled by our work.

Acknowledgements

I would not have been able to write this thesis without several people who have been a part of my journey. First and foremost, my advisor, Jean. When I first started my Ph.D., Jean saw a vision for my Ph.D. that perhaps I myself did not see. She gave me all of the freedom I needed and more to go about my research, and supported me expressing my own research ideas in every way. She has been an incredibly kind, supportive, and empathetic advisor even during challenging personal times, for which I am incredibly grateful.

My committee, of Dave Held, Shubham Tulsiani and Amy Zhang - for whose time and effort I am thankful for. Dave has consistently given me actionable feedback not only through my Ph.D., but also during my MS at CMU. I'm thankful for Shubham for having been a role model for me professionally; I have tried to imbibe his research process not only during my Ph.D., but also during my time working with him at Meta. I'm grateful for Amy, who has spent much of her time brainstorming ideas about abstractions with me, and for having given me research areas to look at, research advice, and her thoughts on my career directions. I'm also thankful for my other advisors along the way; Abhinav Gupta and Stuart Anderson at Meta, and Kris Kitani and Katharina Muelling during my MS at CMU.

I've had other wonderful collaborators throughout my research career as well; Lerrel Pinto, Nick Rhinehart, Yixin Lin, Aravind Rajeswaran, and Vikash Kumar. I'm also incredibly grateful for the administrative support I've received from the Robotics Institute, particularly from Suzanne Muth during my Ph.D., and B.J. Fecich during my MS. I'm also thankful for the support of the AI Caring grant [1], and for the collaborators its facilitated me with, including Roshni Kaushik and Rayna Hata.

I'm also grateful for having been able to go through my Ph.D. with the support and help of the BIG Lab at CMU. In particular, my collaborators and partners throughout my AI Caring work, Peter Schaldenbrand and Vihaan Misra, who've put in so much work into realizing our collective project vision; Eliot Xing, for having helped me set up much of my robot infrastructure, and for the lab at large for giving me feedback, cheering me on, and having supported me in my journey. I'm also thankful for the mentees I've had along the way; particularly at the BIG Lab, Chaitanya Chawla, Lawrence Chen, and Almutwakel Hassan, whom I've learnt a lot about mentoring from.

Beyond my academic collaborators in my career, I am so so thankful for my broader support system and community who've made this journey possible for me. My Pittsburgh community - Abhijat Biswas, who has been a partner in crime right from my undergrad days, and together with Sudharshan Suresh, Raunaq Bhirangi, and Ben Eisner, have been there for me during the toughest times of my Ph.D. I could not have done my Ph.D. without their support in research and careers strategies, immigration issues and more. My friends in Pittsburgh, Anirudh Vemula, Pragna Mannam, Dhruv

[1] I acknowledge funding support from NSF IIS-2112633, *i.e.* AI Caring

Saxena, Cara Craig, Bolutito Babatunde, I thank you for the innumerable board game nights and hang outs. My soccer community, both in real life and virtually, have helped me maintain my sanity during this journey, and have made my time in Pittsburgh fun, competitive and enjoyable. I also have a larger community outside of Pittsburgh who've helped me in my life, my undergrad friends, my friends from Chennai, and more friends in other parts of the world whom I've picked up along the way.

To Ally Bartoszewicz, who has been an endless supply of support, optimism, and celebration, for whom I am so grateful. She has helped me through my Ph.D. by cheering me on no matter how it was going, taking care of me when I needed it, and teaching me so much about living a purposeful and happy life. For Ally's family, who have made me feel at home here in Pittsburgh. To my extended family, who've always cheered me on and expressed how proud they are of me.

To my sister, Anusha Shankar, who has been the most incredible emotionally supportive sister I could ask for, whose experience I have been able to learn from, and whom I've been able to lean on throughout my life.

To my parents, who have given me everything I have and could ever want in my life. I owe my Dad all of my nature to live a disciplined life, my knack for public speaking and my work ethic; he was incredibly proud of me when I first came to CMU in 2014, and would have been just as proud to see me graduate my MS and Ph.D.. My Mom, who has instilled in me all of my curiosity, my desire to experiment and play around with science, and if not for whom I would likely have ended up in a very different career path. I cannot express enough gratitude for my parents, who have given me everything.

Contents

Contents **vii**

List of Tables **xi**

List of Figures **xiii**

I Introduction **1**

1 Introduction **2**

1.1 Background: Acquiring and Representing Agent Skills 3
1.2 Translating Skills from Humans to Robots 4
1.3 Learning and Representing Effects of Skills 4
1.4 Translating Environment-Aware Task Strategies 4
1.5 Translating Abstractions in Art Domains 5
1.6 List of Papers 5

II Background: Acquiring and Representing Agent Skills **6**

2 Learning Robot Skills with Temporal Variational Inference **7**

2.1 Introduction 7
2.2 Related Work 8
2.3 Approach 10
2.3.1 Preliminaries 10
2.3.2 Decomposition of trajectory likelihood 11
2.3.3 Temporal Variational Inference 11
2.3.4 Learning Skills with Temporal VI 13
2.4 Experiments 16
2.4.1 Qualitative Evaluation of Learned Space 18
2.4.2 Reconstruction of Demonstrations 20
2.4.3 Downstream Tasks 21
2.5 Conclusion 22

3 Discovering Motor Programs by Recomposing Demonstrations 23
3.1 Introduction . . . 23
3.2 Related Work . . . 25
3.3 Approach . . . 27
3.3.1 Primitive Discovery via Recomposition . . . 28
3.3.2 Enforcing Simplicity and Parsimony . . . 29
3.4 Experiments . . . 30
3.4.1 Visualizing the Space of Primitives . . . 31
3.4.2 Semantic Segmentation Transfer using Learned Abstractions . 32
3.4.3 Composing Primitives for Hierarchical RL . . . 33
3.5 Conclusion . . . 35

III Translating Skills from Humans to Robots 36

4 Learning Unsupervised Skill Correspondences across Domains 37
4.1 Introduction . . . 37
4.2 Related Work . . . 39
4.3 Approach . . . 41
4.3.1 Pre-requisites . . . 41
4.3.2 Problem Setting . . . 41
4.3.3 Using sequential information of skills . . . 42
4.3.4 Distributional perspective on learning translations . . . 42
4.4 Experiments . . . 45
4.4.1 Learning meaningful skill correspondences . . . 46
4.4.2 Transferring task strategies across domains . . . 50
4.4.3 Additional Comparisons and Discussion . . . 51
4.5 Conclusion . . . 52

IV Learning and Representing Effects of Skills 53

5 Learning Abstract Representations of Agent-Environment Interactions 54
5.1 Introduction . . . 54
5.2 Introduction . . . 55
5.3 Related Work . . . 56
5.4 Approach . . . 56
5.4.1 Background – Learning Behavioral Abstractions . . . 56
5.4.2 Building Temporal Abstractions over Environment State . . . 57
5.4.3 Building Interaction Abstractions from Behavioral and Environmental Abstractions . . . 58
5.5 Experiments . . . 59

5.5.1 Datasets and Experimental Setup 59
5.5.2 Preliminary Qualitative Results 62
5.5.3 Preliminary Quantitative Results 65
5.5.4 Suitability for Transfer 65
5.6 Conclusion 66

V Translating Environment-Aware Task Strategies 67

6 Translating Agent-Environment Interactions across Humans and Robots 68
6.1 Introduction 68
6.2 Related Work 71
6.3 Approach 73
6.3.1 Preliminaries – Learning Behavioral Abstractions 73
6.3.2 Learning Interaction Abstractions 73
6.3.3 Learning Representations of Interaction Abstractions 74
6.3.4 Transferring Interactions from Humans to Robots 76
6.4 Datasets and Experimental Setup 78
6.5 Experimental Results 79
6.5.1 Modelling Interactions 79
6.5.2 Facilitating Human to Robot Task Transfer 81
6.5.3 Ablation of various loss components 82
6.6 Limitations 82
6.7 Conclusions and Future Work 83

VI Translating Abstractions in Art Domains 84

7 Coach FRIDA: Rewarding Exercise in Older Adults with Interactive Robot Artists 85
7.1 Motivation 85
7.2 Introduction 85
7.3 Approach 86
7.4 Synergies and Collaborations 88

VII Conclusions and Future Directions 89

8 Conclusions and Future Directions 90
8.1 Conclusions 90
8.2 Future Directions 91
8.2.1 Short Horizon - Exploring Alternate Application Domains . . 91

8.2.2 Medium Horizon - Introducing New Machinery 92
8.2.3 Long Horizon - Unsupervised Translation of Abstractions as a Paradigm 96

Bibliography **98**

List of Tables

2.1 Trajectory Reconstruction Error of our approach and baselines across various datasets. The baselines were adapted from (1) Kingma and Welling [41], (2) Ijspeert et al. [24], (3) Niekum et al. [23], (4) Shankar et al. [22]. 19

2.2 Average Rewards of our approach and baselines on various RL environments, across 100 episodes. Baselines are adapted from (1) Lillicrap et al. [46], (2) Esmaili et al. [25], (3) Kulkarni et al. [9]. 19

4.1 Evaluating Skill-Tuple Distribution Matching via our approach against various baselines, across half of the considered domain pairs. We present results on the remaining domain pairs including the human-robot pairs in the supplementary material. Baselines are adapted from (1): Ganin et al. [77], (2): Zhu et al. [75], (3): Liu et al. [86]. Lower is better for all metrics except the label accuracy. 47

4.2 Evaluating Task-Strategy Transfer: Evaluating average rewards obtained by translating a source domain demonstration to the target domain via our approach, with and without finetuning, against a hierarchical RL baseline, adapted from Kulkarni et al. [9], across 3 tasks. Rewards for the HRL baseline, and our approach with finetuning are averaged across 10 episodes. Results on additional domain pairs are shared in supplementary material. * - While we evaluate rewards of target domain demonstrations here for comparison, our approach is intended for situations where such demonstrations are unavailable on the target domain. 48

5.1 Comparison of reconstruction error of environmental and interaction abstractions against baseline approaches for reconstructing environmental and joint agent environmental state respectively. Lower is better. 64

6.1 State Reconstruction Error. Lower is better. * represents state errors computed over environment state alone. The full *TransAct* approach outperforms baseline approaches on reconstructing interactions across all datasets. *TransAct* also facilitates accurate reconstruction of interactions transferred from humans to robots comparable to approaches trained on the target domain data themselves. 77

List of Figures

2.1 Depiction of the key distributions q, π, and η, and the probabilistic graphical model underlying our approach. Note the dependence between trajectories and options. We also depict the variables that each of the three networks reason about, and the information they make use of to do so. 10

2.2 Latent space of skills for MIME dataset. Note the clustering of skills into left and right handed reaching, returning and sliding skills, along with additional hybrid skills. Note the emergence of clusters of skills based on semantic notions of skills in each case. Further, note the diverse range of skills present in each case. 15

2.3 Latent space of skills for Roboturk dataset. Note the clustering of skills into single armed pushing, grasping, and placing in different locations and to different extents. Note the emergence of clusters of skills based on semantic notions of skills in each case. Further, note the diverse range of skills present in each case. 16

2.4 Latent space of skills for Mocap dataset. While less structured than (a) and (b), the space consists of diverse skills ranging such as running, front flips and punching. Note the emergence of clusters of skills based on semantic notions of skills in each case. Further, note the diverse range of skills present in each case. 17

3.1 Sample motor programs that emerge by discovering the space of motor programs from a diverse set of robot demonstration data in an unsupervised manner. These motor programs facilitate understanding the commonalities across various demonstrations, and accelerate learning for downstream tasks. 23

3.2 An overview of our approach. Our abstraction network takes in an observed demonstration τ_{obs} and predict a sequence of latent variables $\{z\}$. These $\{z\}$ are each decoded into their corresponding motor programs via the motor program network. We finally recompose these motor programs into the recomposed trajectory. 27

3.3 Visualization of the embedding of the latent representation of motor programs learned by our model (depicted on the left are the start configurations of each motor program, displayed at the corresponding position in the embedded space), and a set of sample primitives unrolled in time (depicted on the right). Each row corresponds to one primitive, annotated with semantic labels of what this motor primitive resembles. 30
3.4 Depiction of execution of the learned primitives on real world Baxter robot. Each row is a single primitive, while columns show progress of the primitive over time. Row 1 depicts a left handed reaching primitive, while row 2 shows a right handed returning primitive. More visualizations provided in supplementary material, and videos are provided in the webpage. 32
3.5 Visualization of consistent segmentations. Each row represents a different instance of a "Drop Objects" task from the MIME Dataset, while each column represents a time-step in the demonstration. White frames represent predicted segmentation points, while colored boxes represent ground truth semantic annotations. Red boxes are reaching primitives, blue boxes are grasping, orange boxes are placing, and green boxes are returning primitives. We see our model predicts 4 motor programs - reaching, grasping, placing the object a small distance away, and returning. This is consistent with the true semantic annotations of these demonstrations, and the overall sequence of primitives expected of the "Drop Box" task. 33
3.6 RL training curves with and without motor programs. Solid lines denote mean success rate, while shaded region denotes ± 1 standard deviation across 10 random seeds. 34

4.1 Sample skill correspondences learnt by our unsupervised approach, across the 4 different morphological robots and a human demonstrator. We visualize a "placing skill" as translated by our approach, used to place objects to the left of each of the agents. Note the semantic correspondence between these skills, despite our approach being completely unsupervised. 38
4.2 Overview of our approach. We translate the original learnt skill space to the target domain. We then construct explicit density estimates over the original target and translated source skill-tuple spaces. We train our translation model to maximize the likelihood of randomly sampled translated source and original target skill-tuples under these densities respectively, affording meaningful skill-correspondences across both robots. 43

4.3 Sample skill correspondences learnt by our unsupervised approach, across the 4 different morphological robots and a human demonstrator. We visualize a "reaching skill" as translated by our approach, used to reach towards objects in the workspace of each of the agents. Note the semantic correspondence between these skills, despite our approach being completely unsupervised. 45

5.1 Architecture of one proposed approach. We learn a robot skill encoder q^r and an environmental abstraction encoder q^e, that learn representations of behavioral and environmental abstractions z^r and z^e respectively. We then combine z^r and z^e into a joint interaction abstraction z^j. We can then condition robot policy π on interaction abstraction z^j, in addition to state inputs s^r and s^e. 57

5.2 Architecture of second proposed approach. We learn a joint encoder for robot skills and environmental abstractions that learn joint representations of behavioral and environmental abstractions z^j. We can then condition robot policy π on interaction abstraction z^j, in addition to state inputs s^r and s^e. 58

5.3 Depiction of preliminary results on both the environmental abstractions (left column), and the interaction abstractions (right column), across Roboturk dataset (top row) and Robomimic dataset (bottom row). These embedding spaces depict abstract representation spaces that model both agent behaviors and the their effects on the environment. In all four cases, note the clustering of similar motions into similar parts of their respective latent spaces. For dynamic versions of these spaces, view https://sites.google.com/view/interaction-abstractions. 60

5.4 Qualitative results on environmental abstractions. We show two object state demonstrations for each dataset, (labelled ground truth), and their corresponding reconstructions via our learnt abstractions (labelled reconstructions). Note the accuracy of the reconstructions in capturing the object trajectory despite the low dimensional nature of the latent encoding. 61

5.5 Qualitative results on interaction abstractions. We show two demonstrations for each dataset of the respective robot interacting with objects (labelled ground truth), and their corresponding reconstructions via our learnt abstractions (labelled reconstructions). We visualize the same tasks as in Figure 5.4, for reasons in Section 5.5.4. Note how the robot skills executed *lead* to the environmental abstractions depicted in Figure 5.4. The reconstructed space is able to capture this, and therefore the overall pattern of motions of agent and environment. . . 63

6.1 Overview of interaction abstractions learnt by *TransAct*. We depict actual samples of the learnt representation space, visualized with corresponding trajectories from both human and real robot demonstrations. We depict 4 different interactions here, opening a box, opening a drawer, pouring from a cup, and transporting a cup. 69
6.2 An overview of *TransAct*. (a) our interaction abstraction learning approach (top) and (b) zero shot transfer approach (bottom). (a) A demonstrated robot trajectory τ^r and environment trajectory τ^e are encoded into behavioral and environmental abstractions z^r and z^e respectively, via their respective encoders q^r and q^e. z^r and z^e are combined into a joint interaction abstraction z^j. Robot policy π is then conditioned on z^j, in addition to state inputs s^r and s^e. (b) To transfer a given human query demonstration τ^e to the robot, we encode its environment abstraction, then retrieve the nearest neighbor robot interaction observed in the training dataset, and rollout the robot policy to retrieve the transferred trajectory. 70
6.3 Depiction of the learnt latent representation space of interactions from the real world robot and human datasets. Note the variety of interactions captured in the space, the clustering of similar interactions from similar tasks into similar parts of each space, and the local continuity of each space. For dynamic version of these spaces (and similar spaces for the other datasets), view https://sites.google.com/view/interaction-abstractions. 72
6.4 Depiction of our transferred embedding space from humans $\mathcal{D}_{\text{HComp}}$ to robots $\mathcal{D}_{\text{R}}$ (left) and the robot latent space from $\mathcal{D}_{\text{R}}$ (right). In both figures, each symbol $\diamond, \square, \times, \circ$ represents an interaction; symbols $\diamond, \square, \times$ on the left represent the compositional query task a human demonstration is taken from. For both figures, the color the symbol represents the task (from $\mathcal{D}_{\text{R}}$) this interaction is from. 74
6.5 Depiction of human query demonstrations for 3 different compositional tasks, as well as 2 transferred trajectories each executed on the robot. At the top, our approach translates the human sequence of reaching and opening a box to a sufficient angle to pour into, releasing the box, grasping and transporting a cup above the box, pouring beads from the cup into the box, placing the cup back, and returning. In the middle, we observe it is able to translate reaching to and opening the drawer, reaching and grasping the cup, transporting and placing the cup before finally returning. In the bottom, it similarly reaches for a cup, transports it above another container, pours beads, and then subsequently grasps a stirrer and stirs the container. Our approach is capable of doing this despite significant variation in the relative configuration of the objects. 80

7.1 Our approach to Coach-FRIDA. The pipeline takes in as input a user exercise; processes this exercise, and then creates a drawing as a reward for the user! The drawing gets progressively more engaging as the user progresses with their exercises. 86
7.2 Learnt Representation Space of Expert Exercises adapted from Chapter 2, for use to evaluate downstream exercise performance. 87

8.1 Overview of translation of dextrous skills from humans to robots. . . 92
8.2 Overview of translation of temporal abstractions of EMG signals to Prosthetic Hand skills. 93
8.3 Overview of translation of temporal abstractions on demonstration data, to abstractions learnt from interaction data on the target domain. 94
8.4 Overview of framework of learning multi-domain multi-modal abstractions from multiple domains. 95
8.5 Overview of paradigm of translation of temporal abstractions from abstract source domains to target domains. 96

I

Introduction

1 Introduction

Humans have a remarkable ability to efficiently learn to perform tasks from a variety of sources of information-from demonstration, instruction, recipes, *etc* . Consider the ability to learn to solve tasks by watching others demonstrate similar tasks; the ability to learn by imitation. For example, children quickly learn the skills needed to play a new sport by watching their parents perform skills such as kicking a ball; amateur cooks can quickly learn to cook new dishes by watching chefs demonstrate how to make similar dishes on YouTube. Indeed, humans are able to learn from such demonstrations despite significant differences between themselves and the demonstrator, including visual perspectives, variations in the objects they interact with, the specifics of the tasks they solve and environments they interact in. For example, we can easily follow a chef's demonstrations on YouTube despite operating in a differently kitchen, with different equipment, often without the same ingredients.

This remarkable aptitude for 'learning by imitation' may be attributed to two factors. First, humans can acquire and abstractly reason about behaviors that they-and other humans-execute. Humans can quickly acquire new skills; for example, humans can quickly learn to cut a new vegetable needed for a new recipe with a knife. People can also reason abstractly about these skills; for example, we reason abstractly about cutting strokes while chopping vegetables, rather than considering the precise position and direction of the knife, or the exact length of the stroke. Second, humans can reason abstractly both about the task at hand and the strategy adopted by a demonstrator to solve it. For example, consider the task of cooking a dish, given a demonstration of a chef cooking a similar dish. People ignore irrelevancies (*ex.* differences in kitchens), focus on patterns environmental change needed (*ex.* changes to the state of vegetables), and reason about the high-level sequence of skills needed to effect these environmental changes, *i.e.* the task strategy (*ex.* steps of a recipe). Having recognized the task strategy adopted by the demonstrator, humans can then find a corresponding set of skills to execute for ourselves [1]. This allows them to adopt the demonstrator's strategy for themselves, therefore quickly and efficiently solving the task at hand.

This ability to reason abstractly about behaviors and the task at hand allows humans to learn from other sources of information just as easily - from instruction, recipes, *etc* . This is natural; abstract reasoning allows humans to focus on the underlying behaviors needed for a task, regardless of the modality of task specification (such as demonstration or natural language instruction). This remarkably powerful ability is one of the components responsible for human proficiency at learning and solving tasks.

We explore how we can enable robots with these abilities. The core idea our thesis explores is *how we can learn and translate temporal abstractions of behaviour across humans and robots*, so as to facilitate their ability to 'learn by imitation'. This is an enticing prospect; it would enable robots to acquire and abstractly represent both their own skills, and those of a demonstrator. In building such abstract reasoning, we seek to answer the following two research questions. *Q1) How can robots learn to acquire and represent behaviors-or skills-that may be composed to solve a variety of tasks?* and *Q2) How do robots learn correspondences between their own skills, and those of a demonstrator, in order to reason about these skills and their effects to solve complex tasks?*

We take steps towards answering these questions in our thesis. Part II focuses on answering the first question, *i.e.* enabling agents with behavioral abstractions. In particular, Part II focuses on building abstract reasoning about the behaviors that other agents and they themselves execute. Part III, Part IV, & Part V shift our attention the second question. Part III focuses on learning correspondences across the skills that various agents learn, so as to transfer sequences of skills across agents. Part IV then focuses on how we can also abstractly model the *effects* of these behaviors on the agents' environments and the objects they interact with. Part V then focuses on how we can build abstract representations of the task strategies that demonstrators follow to solve a task. Building abstract representations of how humans and robot solve tasks enables us to view human and robot tasks strategies from a unified perspective, by focusing on the commonalities that occur across them, and ignoring irrelevant differences and details. This unified perspective in turn allows us to transfer or *translate* such strategies across human and robot agents, a powerful development towards learning by imitation.

1.1 Background: Acquiring and Representing Agent Skills

Part II of this thesis addresses how we can enable agents to acquire and abstractly represent about behaviors or skills. We do so in the context of an agent learning behaviors or skills for themselves, typically from demonstrations from the same agent performing a variety of different tasks. We present two ways of doing this. In Chapter 2, we present an approach unsupervised skill-learning we present an unsupervised method to learn robot skills from demonstrations. We do so by constructing a latent representation of skills, that is learnt by a temporal variational inference. This approach serves as a building block for much of our subsequent work. In Chapter 3, we present an alternate approach for unsupervised skill-learning from demonstrations; using a temporal alignment loss to re-compose skills to form demonstrations. In particular, Chapter 2 serves as a building block for all of our subsequent work.

1.2 Translating Skills from Humans to Robots

Part III of this thesis addresses how we can enable agents to abstractly reason about behaviors or skills. We explore this in the context of agents reasoning about both their own and other agents' skills, and how these skills correspond with one another. Chapter 4 focuses on the problem of learning correspondences of an agents' own skills to those of a differently embodied demonstrator, a key problem to transferring task strategies from demonstrators to learners. We approach this problem by translating skills from one agent to another, using machinery from unsupervised machine translation to learn these cross domain skill correspondences.

1.3 Learning and Representing Effects of Skills

The results in Part II & Part III focus on behavioral abstractions, and how we may represent agent behaviors in isolation. Part IV serves to extend these behavioral abstractions to consider their effects on objects or their environments, and how the agents interact with their environments. In Chapter 5, we apply the skill learning framework of Chapter 2 to learn temporal abstractions of environmental state. We then combine these environmental abstractions with the behavioral abstractions from Chapter 2 to build temporally abstract representations of agent-environment interactions. We then demonstrate the ability of these interaction abstractions to successfully model a diverse set of agent-environment interactions, including manipulating free and articulated across various phases of manipulation.

1.4 Translating Environment-Aware Task Strategies

Our work in Chapter 4, like many other such approaches, addresses the learning by imitation problem by copying the demonstrators' actions to the learner agent. However, it is important to also consider and imitate the effects of these actions on solving the task, rather than imitating the demonstrators' actions alone. In Part V, we explore how we can transfer such environmental effect-aware task strategies from humans to robots, rather than transferring skills alone. Chapter 5 provides us the framework of interaction abstractions, which serves as a way to represent environment-aware task strategies. In Chapter 6, we explore how we can *translate* these environment-aware task strategies from humans to robots. Specifically, we translate the interaction abstractions defined in Chapter 5 across humans and robots, by introducing a framework TransAct. TransAct extends Chapter 4 beyond translating agent skills alone, to also translating their effects on the environments. This contribution in this thesis pieces together our behavioral abstractions and task abstractions to form a framework for abstractions, taking a step towards abstract reasoning for learning by imitation.

1.5 Translating Abstractions in Art Domains

In Part VI, we explore how the idea of mapping abstract representations of human motion to the paint and drawing strokes of interactive robot artists. In particular, in Chapter 7, we explore this for the application domain of art therapy for encouraging older adults to exercise more.

1.6 List of Papers

The work described in this thesis corresponds to the following papers.

1. Chapter 2 corresponds to the ICML paper, Learning Robot Skills with Temporal Variational Inference.

2. Chapter 3 corresponds to the ICLR paper, Discovering Motor Programs by Recomposing Demonstrations.

3. Chapter 4 corresponds to the ICML paper, Translating Robot Skills: Learning Unsupervised Skill Correspondences across Domains.

4. Chapter 5 corresponds to the CoRL workshop paper on aligning human and robot representations, Learning Abstract Representations of Human & Robot Task Strategies.

5. Chapter 6 corresponds to the IROS paper, Translating Agent-Environment Interactions across Humans and Robots.

6. Chapter 7 corresponds to an HRI submission in progress.

II

Background: Acquiring and Representing Agent Skills

2 Learning Robot Skills with Temporal Variational Inference

2.1 Introduction

The robotics community has long since sought to acquire general purpose and reusable robotic skills, to enable robots to execute a wide range of tasks. The idea of such skills is attractive; by abstracting away the details of low-level control and reasoning over high-level skills instead, a robot can address more complex and longer term tasks. Further, the ability to *compose* skills leads to a combinatorial increase in the robot's capabilities [2], ideally spanning the abilities required for the robot to complete the desired tasks. For example, a robot equipped with reaching, grasping, and pouring skills could compose them to make a cup of tea as well as pour cereal into a bowl.

Indeed, the promise of skills has been explored in several contexts, be it options in reinforcement learning (RL) [3], operators in the planning [4], primitives and behaviours in robotics [5], or abstractions [6]. Nomenclature aside, these ideas share the notion of eliciting a certain pattern or template of action in response to a particular situation, differing in their exact implementations and how these skills are obtained. For example, while previous works [5, 7, 8] manually defined these skills, more recent approaches have sought to *learn* these skills, either from interaction [9] or from demonstrations [10, 11, 12, 13, 14].

Learning skills from demonstrations is appealing, since it allows non-robotics-expert humans to demonstrate solving the target tasks, bypassing the tedium of manually specifying these skills or carefully engineering solutions to the tasks [15]. But even using demonstrations, approaches such as Xu et al. [10], Huang et al. [11] require heavy supervision such as segmentation of demonstration data. Instead, learning skills in an unsupervised, data-driven manner not only avoids having to annotate demonstrations; it also enables the use of large scale and diverse demonstration data in robotics [16, 17]. This in turn enables learning a correpsondingly diverse set of skills, as well as how to use them to achieve a variety of tasks.

Aside from the issue of learning and representing individual skills, is the notion of *composing* them. The efficacy of these skills is rather limited when used in isolation; by selecting and sequencing the appropriate skills however, a robot can achieve a variety of complex tasks, as illustrated by the tea and cereal example. A rich body of literature addresses composing skills, including sequencing skills [18, 19], hierarchical approaches [8, 10, 11, 20, 21, 22], options [3], etc. Some works among these [12, 13, 14] address jointly learning skills and how to compose them. Jointly learning both levels of this hierarchy not only addresses how to use these skills, but also allows for adapting the skills based on how useful they are to the task at hand.

Unfortunately, these works have their own limitations. While Konidaris and Barto [8], Neumann et al. [18], Niekum et al. [23] do learn to sequence skills, they assume restrictive primitive representations - such as DMPs [24]. While Shankar et al. [22] learn continuous representations of primitives, it requires an additional post hoc policy learning step. Fox et al. [12], Krishnan et al. [13] do afford directly usable policies, but critically are restricted to a fixed number of discrete options.

In this paper, we propose a framework to jointly learn options and how to compose and use them from demonstrations in an unsupervised manner. At the heart of our framework is a *temporal variational inference* (TVI) based on a corresponding factorization of trajectory likelihoods. We adopt a latent variable representation of options; this allows us to treat the problem of inferring options as inferring latent variables, which we may then accomplish via our temporal variational inference. Specifically, optimizing the objective afforded by our temporal variational inference with respect to the distributions involved naturally gives us low and high-level policies of options.

We evaluate our approach's ability to learn options across three datasets, and demonstrate that our approach can learn a meaningful space of options that correspond with traditional skills in manipulation, visualized at https://sites.google.com/view/learning-causal-skills/home. We quantify the effectiveness of our policies in solving downstream tasks, evaluated on a suite of tasks.

2.2 Related Work

Learning from Demonstrations: Learning from Demonstrations (LfD) addresses solving tasks by learning from demonstrations of the task being solved by an expert. This may be accomplished by simply cloning the original demonstration [25], or fitting the demonstration to a trajectory representation [19, 26] or a policy [27]. More recent efforts have sought to segment demonstrations into smaller behaviors, and fitting a model to the resulting segments [23, 28, 29, 30]. Argall et al. [15] presents a thorough review of these techniques. Our work falls into the broad paradigm of LfD, but seeks to learn hierarchical policies from demonstrations.

Sequencing Primitives: The concept of learning movement primitives using predefined representations of primitives [19, 26] has been a popular technique to capturing robotic behaviors. A natural next step is to sequence such primitives to perform longer horizon downstream tasks, [5, 18, 23]. Konidaris and Barto [8], Konidaris et al. [21] address merging these skills into skill-trees. Lioutikov et al. [20, 31] address sequencing primitives using probabilistic segmentation and attribute grammars respectively. Our work differs from a majority of these works in that we jointly learn both the representation of primitives and how these primitives must be sequenced.

Hierarchical Policy Learning: Fox et al. [12], Krishnan et al. [13], Kipf et al. [14], Sharma et al. [32] also address the problem of learning options from demonstrations without supervision. However these works are restricted to a discrete set of options, which must be pre-specified. Further, Fox et al. [12], Krishnan et al. [13] employ a forward-backward algorithm for inference that requires an often intractable marginalization over latents. The CompILE framework [14] makes use of continuous latent variables to parameterize options, but requires carefully designed attention mechanisms, and is evaluated in relatively low-dimensional domains. Smith et al. [33] and Bacon et al. [34] derive policy gradients to address hierarchical policy learning in the RL setting.

Learning Trajectory Representations: Shankar et al. [22] learning representations of primitives, but need to adopt an additional phase of training to produce usable policies. Co-Reyes et al. [35] also approach hierarchical RL from a trajectory representation perspective. Our work shares this notion of implicitly learning a representation of primitives, but differs in that we do not require an additional high-policy learning step to perform hierarchical control.

Learning Temporal Abstractions: Kim et al. [6] and Gregor et al. [36] both address learning temporal abstractions in the form of 'jumpy' transitions. Kim et al. [6] seeks to learn a generic partitioning of the input sequence into temporal abstractions, while Gregor et al. [36] learns temporal abstractions with beliefs over state to capture uncertainity about the world. While both these works adopt variational bounds of a similar form to ours, our bound objective is derived in terms of usable option *policies* rather than abstract or belief states.

Compositional Policies: Both Xu et al. [10] and Huang et al. [11] address unseen manipulation tasks by learning compositional policies, but require heavy supervision to do so. Andreas et al. [37] and Shiarlis et al. [38] compose modular policies in the RL and LfD settings respectively, using policy sketches to select which policies to execute. While our work approaches doesn't explicitly address compositionality, we too seek to benefit from the benefits of such compositionality.

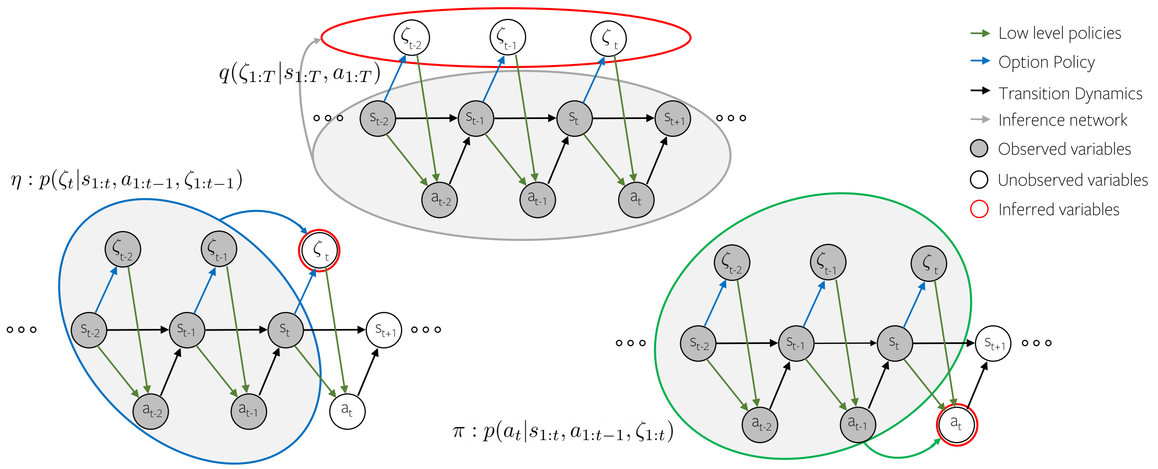

$q(\zeta_{1:T}|s_{1:T}, a_{1:T})$
$\eta : p(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1})$
$\pi : p(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t})$
Low level policies
Option Policy
Transition Dynamics
Inference network
Observed variables
Unobserved variables
Inferred variables

Figure 2.1: Depiction of the key distributions q, π, and η, and the probabilistic graphical model underlying our approach. Note the dependence between trajectories and options. We also depict the variables that each of the three networks reason about, and the information they make use of to do so.

2.3 Approach

2.3.1 Preliminaries

Throughout our paper, we adopt an undiscounted Markov Decision Process without rewards (denoted as MDP\R). An MDP\R is a tuple $\mathcal{M} : \langle S, A, P\rangle$, that consists of states s in state space S, actions a in action space A, and a transition function between successive states $P(s_{t+1}|s_t, a_t)$.

Options: An option $\omega \in \Omega$ [3] formally consist of three components - an initiation set $\mathcal{I}$, a policy $\pi : S \to A$, and a termination function $\beta : S \to [0, 1]$. When an option is invoked in a state in $\mathcal{I}$, policy π is executed until the termination function dictates the option should be terminated (i.e., $\beta(s) = 1$). As in Fox et al. [12], Smith et al. [33], we assume options may be initiated in the entire state space.

Options as Latent Variables: We assume that the identity of an option being executed is specified by a latent variable z, that may be either continuous or discrete. We also consider that at every timestep, a high-level policy $\eta : S \to \Omega \times [0, 1]$ selects the identity z_t of the option to be invoked, as well as a binary variable b_t of whether to terminate the option or not. This option informs the low-level policy π's choice of action, constructing a trajectory as per the generative process in **??**. We denote the sequence of options executed during a trajectory as a sequence of these latent variables, $\zeta = \{z_t, b_t\}_{t=1}^T$.

2.3.2 Decomposition of trajectory likelihood

Consider a trajectory $\tau = \{s_t, a_t\}_{t=1}^{T}$, and the sequence option sequence that generated it $\zeta = \{z_t, b_t\}_{t=1}^{T}$. The joint likelihood $p(\tau, \zeta)$ of the trajectory and these options under the generative process described in **??** may be expressed as follows:

$$\begin{aligned} p(\tau, \zeta) = p(s_1) \prod_{t=1}^{T} & \eta(\zeta_t | s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) \\ & \pi(a_t | s_{1:t}, a_{1:t-1}, \zeta_{1:t}) p(s_{t+1} | s_t, a_t) \end{aligned} \tag{2.1}$$

The distributions $\eta(\zeta_t | s_{1:t}, a_{1:t-1}, \zeta_{1:t-1})$ and $\pi(a_t | s_{1:t}, a_{1:t-1}, \zeta_{1:t})$ implicitly capture the causal restriction that future variables (ex. $s_{t+1:T}$) do not influence earlier variables (ex. $a_{1:t}$). Further, both π and η may be queried at any arbitrary time t with the information available till that time. Ziebart et al. [39] formalized the notion of *causally conditioned distributions* to represent such distributions, a notion that plays a part in the formulation of our approach. We refer the reader to Ziebart et al. [39], Kramer [40] for a more thorough treatment of this concept.

2.3.3 Temporal Variational Inference

To reiterate, our goal is to learn options and how to use them from demonstrations; this formally corresponds to learning low and high level policies π and η, from a dataset of N demonstrations, $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$. This is equivalent to inferring latent variables ζ from a trajectory, since policies π and η essentially reason about the choice of ζ and its effect on the choice of actions. The representation of options as latent variables ζ we adopt hence allows us to view the problem of inferring options from a perspective of inference of latent variables.

This allows us to employ unsupervised latent variable inference techniques; we employ a variant of variational inference (VI) [41] to infer latent options ζ. Our choice of VI over the forward-backward style algorithm employed in Fox et al. [12], Krishnan et al. [13] is because VI is amenable to both continuous and discrete latent variables z. Further, VI bypasses the often intractable marginalization over latents in favor of sampling based approach.

In standard VI, a variational distribution $q(z|x)$ is used to infer latents z from observed data x, approximating the unknown conditional $p(z|x)$. One then optimizes the likelihood of observations given the predicted latents under a learned decoder $p(x|z)$. In our case, we seek to infer the *sequence* of options $\zeta = \{z_t, b_t\}_{t=1}^{T}$ from a trajectory $\tau = \{s_t, a_t\}_{t=1}^{T}$; we estimate the conditional $p(\zeta|\tau)$ with a variational approximation $q(\zeta|\tau)$.

To retrieve usable policies that can be queried at inference or "test" time, we require policies that can reason about the current choices of option ζ_t and action a_t given the observations available *so far*, i.e. $s_{1:t}$, $a_{1:t-1}$, and $\zeta_{1:t-1}$. This precludes optimizing the conditional $p(\tau|\zeta)$ as would be done in standard VI. Instead, we optimize the joint likelihood $p(\tau, \zeta)$ of trajectories and latents with respect to the causally conditioned π and η. Not only does this afford us directly usable policies as desired, but this objective naturally arises from the variational bound constructed below.

We now formally present temporal variational inference, a variant of VI suitable for our sequential latent variables, that accounts for the causal restriction of these latent variables based on the decomposition of trajectory likelihood in Section 2.3.2. We begin with the standard objective of maximizing the log-likelihood of trajectories across the dataset, $\mathcal{L} = \mathbb{E}_{\tau\sim\mathcal{D}}\Big[\log p(\tau)\Big]$. $\mathcal{L}$ is lower bounded by J, where:

$$\begin{aligned}
J &= \mathbb{E}_{\tau\sim\mathcal{D}}\Big[\log p(\tau)\Big] - D_{KL}\Big[q(\zeta|\tau)||p(\zeta|\tau)\Big] \\
&= \mathbb{E}_{\tau\sim\mathcal{D}}\Big[\log p(\tau)\Big] - \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\log \frac{q(\zeta|\tau)}{p(\zeta|\tau)}\Big] \\
&= \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\log p(\tau) + \log p(\zeta|\tau) - \log q(\zeta|\tau)\Big] \\
&= \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\log p(\tau,\zeta) - \log q(\zeta|\tau)\Big]
\end{aligned}$$

Substituting the joint likelihood decomposition from Equation (2.1) above yields the following objective:

$$\begin{aligned}
J = \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_t &\{\log \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) \\
&+ \log \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) + \log p(s_{t+1}|s_t, a_t)\} \\
&+ \log p(s_1) - \log q(\zeta|\tau)\Big]
\end{aligned} \tag{2.2}$$

Assuming distributions π, η, and q are parameterized by θ, ϕ, and ω respectively, we may optimize these using standard gradient based optimization of J:

$$\begin{aligned}
\nabla J = \nabla_{\theta,\phi,\omega}\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_t &\{\log \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) \\
&+ \log \eta(b_t, z_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1})\} - \log q(\zeta|\tau)\Big]
\end{aligned} \tag{2.3}$$

Note that the dynamics $p(s_{t+1}|s_t, a_t)$ and initial state distribution $p(s_1)$ factor out of this gradient, as derived in the supplementary material.

Parsing the objective J: We provide a brief analysis of how objective J and its implied gradient update Equation (2.3) jointly optimizes π, η, and q to be consistent with each other. Three interacting terms optimize q. The first two terms encourage q to predict options ζ that result in high likelihood of actions a_t under π, and that are likely under the current estimate of the high level policy η. The final $-\log q(\zeta|\tau)$ term encourages maximum entropy of q, discouraging q from committing to an option unless it results in high likelihood under π and η. This entropy term also prevents q from trivially encoding all trajectories into a single option.

Low-level policy π is trained to increase the likelihood of selecting actions a_t given the current option being executed ζ_t. High-level policy η is trained to mimic the choices of options made by the variational network (i.e. options that result in high likelihood of demonstrated actions), only using the available information at time t to do so.

Reparameterization: While Equation (2.3) can be implemented via REINFORCE [42], we do so only for inferring discrete variables $\{b_t\}_{t=1}^T$. As in standard VI, we exploit the continuous nature of z's and employ the reparameterization trick [41] to enable efficient learning of $\{z_t\}_{t=1}^T$. Rather than sample latents $\{z_t\}_{t=1}^T$ from a stochastic variational distribution $q(\zeta|\tau)$, $\{z_t\}_{t=1}^T$ is parameterized as a differentiable and deterministic function of the inputs τ and a noise vector ϵ, drawn from an appropriately scaled normal distribution. In practice, we parameterize q as an LSTM that takes τ as input, and predicts mean μ_t and variance σ_t of the distribution $q(\{z_t\}_{t=1}^T|\tau)$. We then retrieve $\{z_t\}_{t=1}^T$ as $\{z_t = \mu_t + \sigma_t\epsilon_t\}_{t=1}^T$.

In our case, gradients flow from our objective J through *both* the low and high-level policies π and η to the variational network q. This in contrast with standard VI, where gradients pass through the decoder $p(x|z)$. This reparameterization enables the efficient gradient based learning of q based on signal from both the low and high-level policies.

Features of objective: The temporal variational inference we present has several desirable traits that we describe below.
(1) First and foremost, J provides us with causally conditioned low and high-level policies π and η. These policies are directly usable at inference time towards solving downstream tasks, since they are only trained with information that is also available at inference time.
(2) With TVI, we can adopt a continuous parameterization of options, $z \in \mathbb{R}^n$. This eliminates the need to pre-specify the number of options required; instead we may learn as many options as are required to capture the behaviors observed in the data, in a data driven manner. The continuous space of options also allows us to reason about how similar the various learned options are to one another (allowing substitution of options for one another).
(3) The joint training of π, η, and q implied by J not only allows training the high-level policy η to based on the available options, but also allows the adaptation of the low-level options π based on how useful they are to reconstructing a demonstration.
(4) The objective J is also amenable to gradient based optimization, allowing us to learn options efficiently.

2.3.4 Learning Skills with Temporal VI

Equipped with this understanding of our temporal variational inference (TVI), we may retrieve low and high-level policies π and η by gradient based optimization of the

Algorithm 1 Temporal Variational Inference for Learning Skills

```
Input: $\mathcal{D}$                                   ▷ Require a demonstration dataset
Output: $\pi, \eta$                                    ▷ Output low and high-level policies
1: Initialize $\pi_\theta, \eta_\phi, q_\omega$                       ▷ Initialize networks
2: Pretrain $\pi$ as VAE                               ▷ Pretrain latent representation
3: for $i \in [1, 2, ..., N_{\text{iterations}}]$ do
4:     $\tau_i \leftarrow \mathcal{D}$                                  ▷ Retrieve trajectory from dataset
5:     $\zeta \sim q(\zeta|\tau_i)$                  ▷ Sample latent sequence from variational network
6:     $J \leftarrow \sum_t \log \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) + \sum_t \log \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) - \log q(\zeta|\tau)$ 3Evaluate likelihood objective under current policy estimates
7:     Update $\pi_\theta, \eta_\phi, q_\omega$ via $\nabla_{\theta,\phi,\omega} J$
```

objective J presented. We first make note of some practical considerations required to learn skills with TVI, then present a complete algorithm to learn skills using TVI.

Policy Parameterization: We parameterize each of the policies π and η as LSTMs [43], with 8 layers and 128 hidden units per layer. The recurrent nature of the LSTM naturally captures the causal nature of π and η. In contrast, q is parameterized as a bi-directional LSTM, since q reasons about the sequence of latents ζ given the entire trajectory τ. q, π and η all take in a concatenation of trajectory states and actions as input. π and η also take in the sequence of latents until the current timestep as input. Since η reasons about the choice of latents to solve the task occurring in the demonstration, η also takes in additional information about the task, such as task ID and object-state information. π predicts the mean μ_a and variance σ_a of a Gaussian distribution from which actions $a \in \mathcal{A}$ are drawn. η and q both predict mean μ_z and variance σ_z of a Gaussian distribution from which latent variables z are drawn. η and q also predict the probability of terminating a particular option $p(b)$, from which binary termination variables b are drawn. While q predicts the entire sequence of ζ's, latents are retrieved from η during a rollout via the generative process in **??**.

Pretraining the Low-level Policy: Optimizing our joint objective J with randomly initialized low-level policies results in our training procedure diverging. During initial phases of training, the random likelihoods of actions and options under the random initial policies provide uninformative gradients. To counteract this, we initialize the low-level policy to capture some meaningful skills, by pretraining it to reconstruct demonstration segments in a VAE setting. Specifically, we draw trajectory segments from the dataset and encode them as a single latent z (i.e. a single option). We train the low-level policy (i.e. as a decoder) to maximize the likelihood of the actions observed in this trajectory segment given the latent z predicted by the encoder.

Algorithm: We present the full algorithm for learning skills via temporal variational inference in Algorithm 1. After the pre-training step described above, there

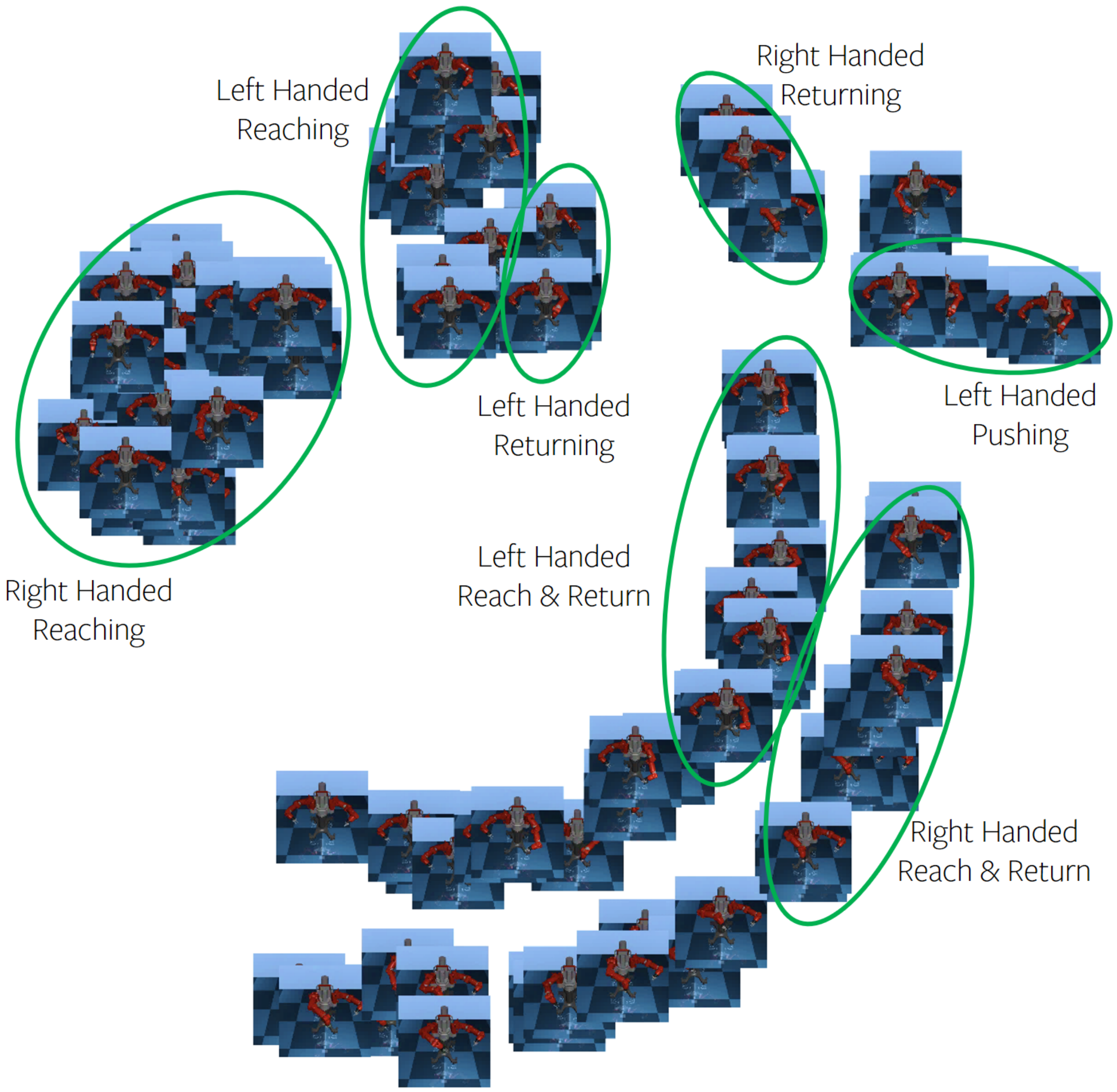


Figure 2.2: Latent space of skills for MIME dataset. Note the clustering of skills into left and right handed reaching, returning and sliding skills, along with additional hybrid skills. Note the emergence of clusters of skills based on semantic notions of skills in each case. Further, note the diverse range of skills present in each case.

are three steps that occur every iteration in training. (1) For every trajectory, a corresponding sequence of latent variables ζ is sampled from the varitional network q. (2) This likelihood of this estimated sequence of latents ζ is evaluated under the current policy estimates, giving us objective J. (3) The gradients of J are then used to update the three networks π, η, and q.

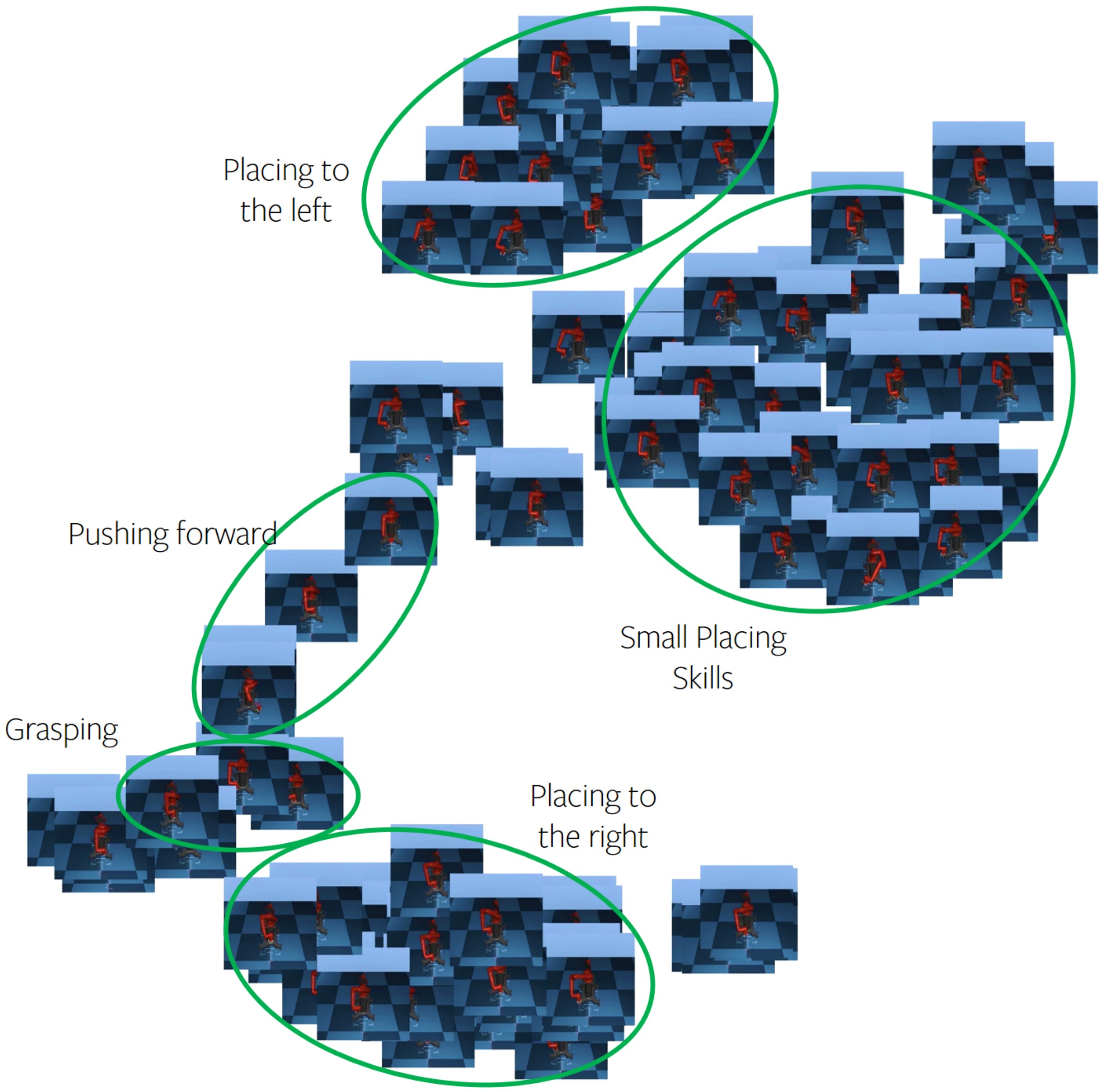


Figure 2.3: Latent space of skills for Roboturk dataset. Note the clustering of skills into single armed pushing, grasping, and placing in different locations and to different extents. Note the emergence of clusters of skills based on semantic notions of skills in each case. Further, note the diverse range of skills present in each case.

2.4 Experiments

We would like to understand how well our approach can discover options from a set of demonstrations in an unsupervised manner, and quantify how useful the learnt policies are for solving a set of target tasks. To this end, we evaluate our approach across the

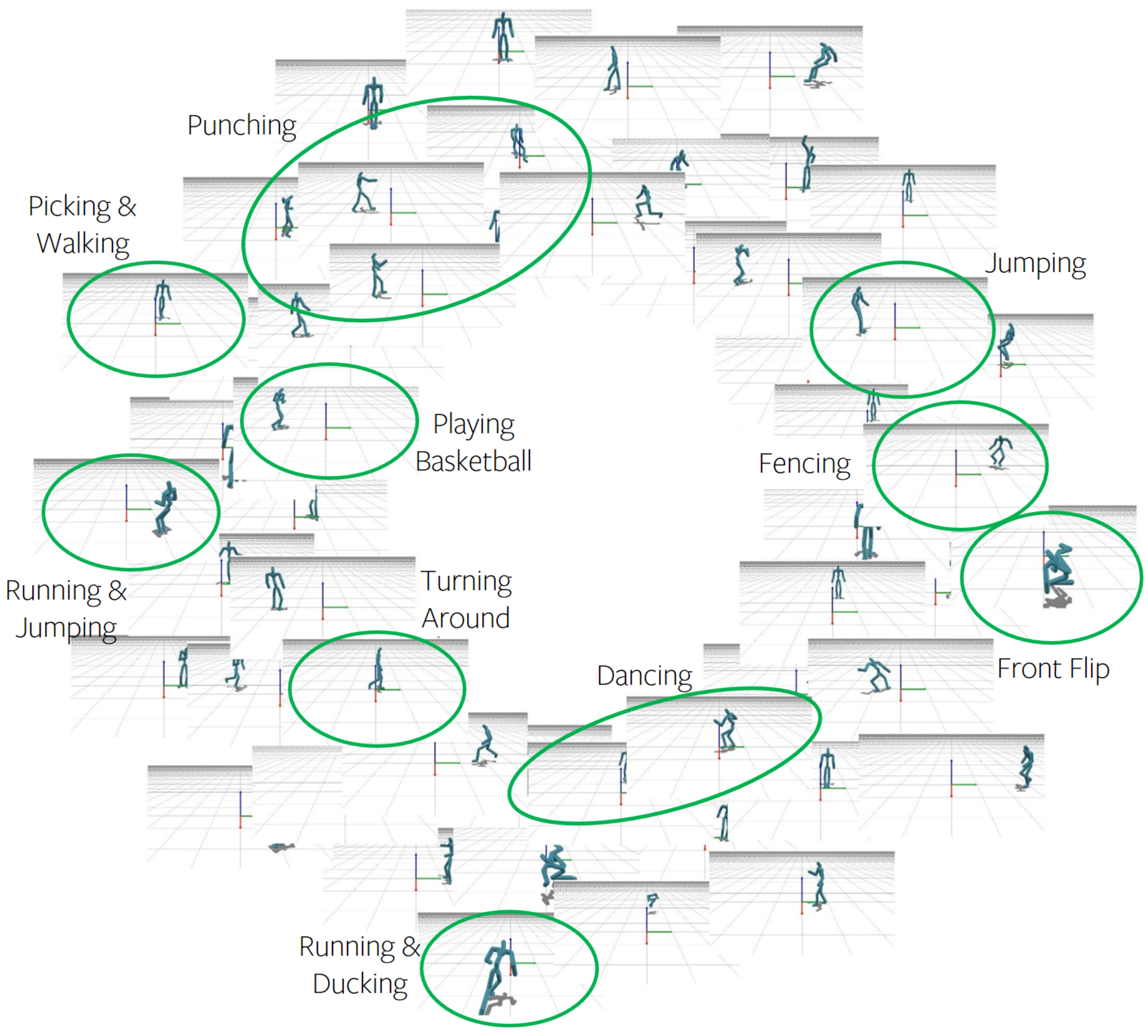


Figure 2.4: Latent space of skills for Mocap dataset. While less structured than (a) and (b), the space consists of diverse skills ranging such as running, front flips and punching. Note the emergence of clusters of skills based on semantic notions of skills in each case. Further, note the diverse range of skills present in each case.

three datasets described below, as well as a suite of simulated robotic tasks. We present visualizations of the results of our model at https://sites.google.com/view/learning-causal-skills. We first describe the datasets used below.

MIME Dataset: The MIME Dataset [16] consists of 8000+ kinesthetic demonstrations across 20 tasks (such as pushing, bottle-opening, stacking, etc.) collected on a Baxter robot. We use the 16 dimensional joint-angles of the Baxter (7 joints for each of the 2 arms, and a gripper for each arm) as the input and prediction space for our model.

Roboturk Dataset: The Roboturk Dataset [17] consists of 2000+ demonstrations

collected on a suite of tasks (such as bin picking, nut-and-peg assembly, etc.) by teleoperating a simulated Sawyer robot. These teleoperation demonstrations are hence more noisy than the kinesthetic MIME dataset. We use the 8 dimensional joint angles of the Sawyer (7 arm joints, and a single gripper), as well as the robot-state and object-state vectors provided in Mandlekar et al. [17] to train our model.
CMU Mocap Dataset: The CMU Mocap Dataset [44] consists of 1953 motions collected by tracking visual markers placed on humans, while performing a variety of different actions. These actions include punching, jumping, performing flips, running, etc. We use the local (i.e., relative to a root node on the agent itself) 3-D positions of each of the 22 joints recorded in the dataset, for a total of 66 dimensions.
Preprocessing and Train-Test Split: In both the MIME and Roboturk datasets, the gripper values are normalized to a range of $\{-1, 1\}$, while joint angles are unnormalized. The trajectories across all datasets are downsampled (in time) by a factor of 20. For each dataset, we set aside 500 randomly sampled trajectories that serve as our test set for our experiments in Section 2.4.2. The remaining trajectories serve as the respective training sets.

2.4.1 Qualitative Evaluation of Learned Space

The first question we would like to answer is - “Is our approach able to learn a diverse space of options?”. We answer this by presenting a qualitative analysis of the space of options learned by our model.

For a set of 100 demonstrations, we retrieve the sequence of latent z’s and their corresponding trajectory segments executed over each demonstration from our model. We embed these latent z’s in a 2-D space using T-SNE [45], and visualize the trajectory segments at their corresponding position in this 2-D embedded space. We visualize the learned embedding spaces for the MIME dataset in Figure 2.2, for the Roboturk dataset in Figure 2.3, and for the Mocap dataset in Figure 2.4. Dynamic visualizations of these figures are available at https://sites.google.com/view/learning-causal-skills.

Note the emergence of *clusters* of skills on the basis of types of motions being executed across these datasets.

In the case of the MIME dataset Figure 2.2, we note that the emergent skills are separated on the basis of the nature of motion being executed, as well as the *arm* of the robot being used. This is expected in a bimanual robot, as skills with the left and right hands are to be treated differently, as are skills using both hands. The skills that our approach captures correspond directly to traditional notions of skills in the manipulation community, such as reaching, returning, sliding / pushing, etc. Our approach further distinguishes between left handed reaching and right handed reaching, etc., a useful ability in addressing downstream tasks.

For the Roboturk dataset Figure 2.3, the space is separated on the basis of the nature, shape and direction of motion being executed. For example, placing motions to the left and right of the robot appear separately in the space. Additional placing

Table 2.1: Trajectory Reconstruction Error of our approach and baselines across various datasets. The baselines were adapted from (1) Kingma and Welling [41], (2) Ijspeert et al. [24], (3) Niekum et al. [23], (4) Shankar et al. [22].

Method	MIME Data	Roboturk Data	Mocap Data
Flat VAE [1]	0.14	0.23	0.08
Flat DMP [2]	0.36	0.54	3.45
H-DMP [3]	0.02	0.06	0.01
Disco-MP [4]	0.02	0.03	0.03
Ours	0.02	0.04	0.04

Table 2.2: Average Rewards of our approach and baselines on various RL environments, across 100 episodes. Baselines are adapted from (1) Lillicrap et al. [46], (2) Esmaili et al. [25], (3) Kulkarni et al. [9].

Method	SawyerPick-PlaceBread	SawyerPick-PlaceCan	SawyerPick-PlaceCereal	SawyerPick-PlaceMilk	SawyerNut-AssemblyRound
Flat RL [1]	0.11	0.34	0.18	0.14	0.44
Flat IL [2]	0.60	0.65	0.29	0.67	0.49
Hierarchical RL (No Init) [3]	0.13	0.11	0.04	0.15	0.36
Hierarchical RL (W/ Init) [3]	0.41	0.37	0.22	0.34	0.54
Ours	1.54	1.22	1.54	0.48	1.88

motions that move the arm to a lesser extent also appear separately. The space also captures finer motions such as closing the grippers down (typically in a position above the workspace).

For the Mocap dataset Figure 2.4, the learned space is not as clearly structured as in the case of MIME and Roboturk datasets. We believe this is due to the much larger range of motions executed in the dataset, coupled with the high-dimensional nature of the humanoid agent. Despite this lack of structure, a diverse set of skills is still learned. For example, the space consists of running, walking, and jumping skills, which constitute a majority of the dataset. More interesting skills such as fencing, performing flips, and boxing skills were also present and captured well by our model.

In both the MIME and Roboturk datasets, the correspondence of many emergent skills with traditional notions of skills in the manipulation community is indicative that our approach can indeed discover diverse robotic skills from demonstrations without supervision.

2.4.2 Reconstruction of Demonstrations

We evaluate how well our approach can use the learned skills to reconstruct the demonstrated trajectories; i.e. whether it is able to capture the overall structure of the demonstrations in terms of the skills being executed. We do so quantitatively and qualitatively. Quantitatively, we measure the average state distance (measured by the mean squared error) between the reconstructed trajectory and the original demonstration, across the 500 randomly sampled unseen demonstrations in each dataset. We compare our approach's performance on this metric against a set of baselines:

- **Flat-VAE:** We train an LSTM VAE [41] to reconstruct demonstrations. This represents a flat, non-hierarchical baseline with a learned representation. The architecture used is an 8 layer LSTM with 128 hidden units, like our policies.
- **Flat-DMP:** We fit a single Dynamic Movement Primitive [24] to each trajectory in the dataset. This represents a non-hierarchical baseline with a predefined trajectory representation.
- **H-DMP:** We evaluate a hierarchical DMP based approach similar to Niekum et al. [23], that first segments a trajectory by detecting changepoints in acceleration, and then fits individual DMPs to *each* segment of the trajectory. This represents a hierarchical baseline with a predefined trajectory representation.
- **Disco-MP:** We also evaluate the method of Shankar et al. [22], which is directly optimized for trajectory reconstruction. This approach represents a hierarchical baseline with a learned trajectory representation.
- **Ours:** We obtain predicted trajectories from our model obtaining the latent z's that occur in a demonstration from our q network, and rolling out the low-level policy with these z's as input.

We present the average state distances obtained by our approach against these baselines in Table 2.1. The flat VAE is able to achieve a reasonably low reconstruction error, indicating the benefits of a learnt trajectory representation over a predefined representation such as DMPs. Combinining a learnt trajectory representation with the ability to compose primitives naturally leads to a further decrease in the trajectory reconstruction error, as observed in the Disco-MP baseline and our approach. Note that our approach is able to achieve a similar reconstruction error to that of Disco-MP, which is explicitly optimized to minimize (aligned) state distances, as well as the H-DMP baseline, which heavily overfits to a single trajectory (thus promising low reconstruction error). This demonstrates that our approach indeed captures the overall structure of demonstrations and is able to represent them faithfully. These trends are consistently observed across all three datasets we evaluate on.

To qualitatively analyse how well our approach captures the structure of demonstration, we visualize the reconstructed trajectories predicted by our approach against the corresponding ground truth trajectory. These results are presented in our website: https://sites.google.com/view/learning-causal-skills/home. We observe that our approach is indeed able to capture the rough sequence of skills that occur in the demonstrated trajectories. In the case of the MIME and Roboturk datasets, our model predicts similar reaching, returning, sliding etc. primitives when the ground truth trajectory executes a corresponding primitives. Further, the rough shape of the arms during the predicted skills correlate strongly with the shapes observed in the ground truth trajectories; this is consistent with the quantitative results presented above. Our approach also notably captures fine motions (such as opening and closing of the gripper) in trajectories well. In case of the Mocap dataset, the overall shape and trend of rolled out trajectories align very closely with the original demonstration, showing the use of our approach in learning skills across widely differently structured data and morphology of agents.

2.4.3 Downstream Tasks

We would also like to evaluate how useful our learned policies are for solving a set of target tasks. This is central to our motivation of jointly learning options *and* how to use them. We test our learned policies on a suite of 6 simulated robotic tasks from the Robosuite [17] environment on the Sawyer robot. Since the MIME dataset lacks any object information, we disregard the Baxter tasks in Robosuite [17], and instead use tasks from Robosuite for which the Roboturk dataset has demonstrations. In the pick-place tasks, a reward of 1 is given to objects successfully placed in the bin. In the nut-and-peg assembly tasks, a reward of 1 is given for successfully placing a nut. Both tasks also have additional rewards based on conditions of how the task was executed. We point the reader to [17] for a full description of these tasks and their reward structure. We compare the performance of our method on these tasks against the following baselines:

- **Flat RL:** We train flat policies on each of the 6 tasks in the reinforcement learning setting, using DDPG [46].
- **Flat IL:** We train flat policies to mimic the actions observed in the demonstrations of each of the 6 tasks, and subsequently finetune these policies in the RL setting.
- **Hierarchical RL:** We train hierarchical policies as in [9] in the pure RL setting, with (W/ Init) and without (No Init) any prior initialization.
- **Ours:** We fine-tune the low and high-level policies obtained by our model in the RL setting.

All baseline policies are implemented as 8 layer LSTMs with 128 hidden units, for direct comparison with our policies. The RL based approaches are trained with DDPG with the same exploration processes and hyperparameters (such as initializations of the networks, learning rates used, etc.), as noted in the supplementary. We evaluate each of these baselines along with our approach by testing out the learned policies over 100 episodes in the 6 environments, and reporting the average rewards obtained in Table 2.2.

We observe that the Flat RL and Hierarchical RL baselines are unable to solve these tasks, and achieve low rewards across all tasks. This is unsurprising given the difficulty of exploration problem underlying these tasks [17]. Pretraining policies with imitation learning somewhat allieviates this problem, as observed in the slightly higher rewards of the Flat IL baseline. This is likely because the policies are biased towards actions similar to those seen in the demonstrations. Training hierarchical policies initialized with our approach is able to achieve significantly higher rewards than both the IL and RL baselines consistently across most environments. By providing these policies with suitable notions of skills that extend over several timesteps, we are able to bypass reasoning over low-level actions for 100's of timesteps, thus guiding exploration in these tasks more efficiently.

2.5 Conclusion

In this paper, we presented a framework for jointly learning robotic skills and how to use them from demonstrations in an unsupervised manner. Our temporal variational inference allows us to construct an objective that directly affords us usable policies on optimization. We are able to learn semantically meaningful skills that correspond closely with the traditional notions of skills observed in manipulation. Further, our approach is able to capture the overall structure of demonstrations in terms of the learned skills. We hope that these factors contribute towards accelerating research in robot learning for manipulation.

3 Discovering Motor Programs by Recomposing Demonstrations

3.1 Introduction

We have seen impressive progress over the recent years in learning based approaches to perform a plethora of manipulation tasks [47, 48, 49, 50, 51]. However, these systems are typically task-centric *savants* – able to only execute a single task that they were trained for. This is because these systems, whether leveraging demonstrations or environmental rewards, attempt to learn each task *tabula rasa*, where low to high level motor behaviours, are all acquired from scratch in context of the specified task. In contrast, we humans are adept at a variety of basic manipulation skills *e.g.* picking, pushing, grasping *etc* , and can effortlessly perform these diverse tasks via a unified manipulation system.

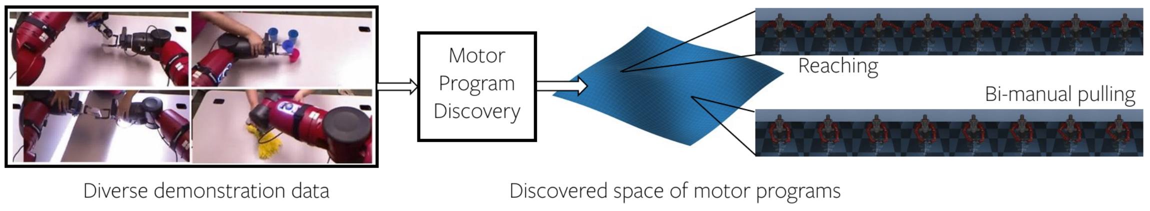

Motor Program Discovery
Reaching
Bi-manual pulling
Diverse demonstration data
Discovered space of motor programs

Figure 3.1: Sample motor programs that emerge by discovering the space of motor programs from a diverse set of robot demonstration data in an unsupervised manner. These motor programs facilitate understanding the commonalities across various demonstrations, and accelerate learning for downstream tasks.

How can we step-away from the paradigm of learning task-centric savants, and move towards building similar unified manipulation systems? We can begin by not treating these tasks independently, but via instead exploiting the commonalities across them. One such commonality relates to the primitive actions executed to accomplish the tasks – while the high-level semantics of tasks may differ significantly, the low and mid-level *motor programs* across them are often shared *e.g.* to either pick or push an object, one must move the hand towards it. This concept of *motor programs* can be traced back to the work of Lashley [52], who noted that human motor movements consist of 'orderly sequences' that are not simply sequences of stimulus-response patterns. The term 'motor programs' is however better attributed to Keele [53] as being representative of 'muscle commands that execute a movement sequence uninfluenced by peripheral feedback', though later works shifted the focus from muscle commands to the movement itself, while allowing for some feedback [54]. More directly relevant to our motivation is Schmidt's notion of 'generalized' motor programs [55] that can allow abstracting a class of movement patterns instead of a singular one. In this work, we present an approach to discover the shared space of (generalized) motor programs underlying a variety of tasks, and show that elements from this space can be composed to accomplish diverse tasks. Not only does this allow understanding the commonalities and shared structure across diverse skills, the discovered space of motor programs can provide a high-level abstraction using which new skills can be acquired quickly by simply learning the set of desired motor programs to compose.

We are not the first to advocate the use of such mid-level primitives for efficient learning or generalization, and there have been several reincarnations of this idea over the decades, from 'operators' in the classical STRIPS algorithm [4], to 'options' [3] or 'primitives' [56] in modern usage. These previous approaches however assume a set of manually defined/programmed primitives and therefore bypass the difficulty of discovering them. While some attempts have been made to simultaneously learn the desired skill and the underlying primitives, learning both from scratch is difficult, and are therefore restricted to narrow tasks. Towards overcoming this difficulty, we observe that instead of learning the primitives from scratch in the context of a specific task, we can instead discover them using demonstrations of a diverse set of tasks. Concretely, by leveraging demonstrations for different skills *e.g.* pouring, grasping, opening *etc* , we discover the motor programs (or movement primitives) that occur across these.

We present an approach to discover movement primitives from a set of *unstructured* robot demonstration *i.e.* demonstrations without additional parsing or segmentation labels available. This is a challenging task as each demonstration is composed of a varying number of unknown primitives, and therefore the process of learning entails both, learning the space of primitives as well as understanding the available demonstrations in context of these. Our approach is based on the insight that an abstraction of a demonstrations into a sequence of motor programs or primitives, each of which correspond to an implied movement sequence, and must yield back the demonstration when the inferred primitives are 'recomposed'. We build on this and formulate an unsupervised approach to jointly learn the space of movement primitives, as well as a parsing of the available demonstrations into a high-level sequence of these primitives.

We demonstrate that our method allows us to learn a primitive space that captures the shared motions required across diverse skills, and that these motor programs can be adapted and composed to further perform specific tasks. Furthermore, we show that these motor programs are semantically meaningful, and can be recombined to solved robotic tasks using reinforcement learning. Specifically, solving *reaching* and *pushing* tasks with reinforcement learning over the space of primitives achieves 2 orders of magnitude faster training than reinforcement learning in the low-level control space.

3.2 Related Work

Our work is broadly related to several different lines of work which either learn task policies from demonstrations, or leverage known primitives for various applications, or learn primitives in context of known segments. While we discuss these relations in more detail below, we note that previous primitive based approaches either require: a) a known and fixed primitive space, or b) annotated segments each corresponding to a primitive. In contrast, we *learn* the space of primitives *without requiring this segmentation annotation*, and would like to emphasize that ours is the first work to do so for a diverse set of demonstrations spanning multiple tasks.

Learning from Demonstration: The field of learning from demonstrations (LfD) [57] has sought to learn to perform tasks from a set of demonstrated behaviors. A number of techniques exist to do so, including cloning the demonstrated behavior [25], fitting a parametric model to the demonstrations [19, 26], or first segmenting the demonstrations and fitting a model to each of the resultant segments [23, 28, 29, 30]. We point the reader to Argall et al. [15] for a comprehensive overview of the field. Rather than directly learn to perform tasks, we use demonstrations to learn a diverse set of composable and reusable primitives, or motor-programs that may be used to perform a variety of downstream tasks.

Learning and Sequencing Motion Primitives: Several works [19, 26] learn motion primitives from demonstrations using predetermined representations of skills such as Dynamic Movement Primitives (DMPs) [56]. A few other works have approached the problem from an optimization perspective [58]. Given these primitives, a question that arises is how to then sequence learned skills to perform downstream tasks. Several works have attempted to answer this question - [18] builds a layered approach to adapt, select, and sequence DMPs. Konidaris and Barto [8], Konidaris et al. [21] segments demonstrations into sequences of skills, and also merge these skills into skill-trees. However, the predetermined representations of primitives adopted in these works can prove restrictive. In particular, it prevents learning arbitrarily expressive motions and adapting these motions to a generic downstream task. We seek to move away from these fixed representations of primitives, and instead learn representations of primitives along with the primitives themselves.

Latent Variable Models: The family of latent variable models (LVMs) provides us with fitting machinery to do so. Indeed, the success of LVMs in learning representations in deep learning has inspired several recent works [14, 35, 59, 60] to learn latent representations of trajectories. SeCTAR [35] builds a latent variable conditioned policy and model that are constrained to be consistent with one another, and uses the learned policies and model for hierarchical reinforcement learning. The CompILE framework [14] seeks to learn variable length trajectory segments from demonstrations instead, and uses latent variables to represent these trajectory segments, but is evaluated in relatively low-dimensional domains. We adopt a similar perspective to these works and learn continuous latent variable representations (or abstractions) of trajectory *segments*.

Hierarchical RL and the Options Framework: The related field of hierarchical reinforcement learning (HRL) learns a layering of policies that each abstract away details of control of the policies of a lower level. The Options framework [3] also learns similar temporal abstractions over sequences of atomic actions. While promising, its application has traditionally been restricted to simple domains due to the difficulty of jointly learning internal option policies along with a policy over those options. Recent works have managed to do so with only a reward function as feedback. The Option-Critic framework [34] employs a policy-gradient formulation of options to do so, while the Option-Gradient [33] learns options in an off-policy manner. In contrast with most prior work in the options framework, [12, 13, 61] learn options from a set of demonstrations, rather than in the RL setting. In similar spirit to these works, we too seek to learn abstractions from a given set of demonstrations, however unlike DDO [12], DDCO [13], and CompILE [14], we can learn primitives beyond a discrete set of options in a relatively high dimensional domain.

Hierarchical representations of demonstrations: The idea of hierarchical task representations has permeated into LfD as well. In contrast to reasoning about demonstrations in a flat manner, one may also infer the hierarchical structure of tasks performed in demonstrations. A few recent works have striven to do so, by representing these tasks as programs [10, 62], or as task graphs [11]. Both Xu et al. [10] and Huang et al. [11] address generalizing to new instances of manipulation tasks in the low-shot regime by abstracting away low-level controls.

The idea of policy sketches, i.e. a sketch of the sub-tasks to be accomplished in a particular task, has become popular [37, 38]. Andreas et al. [37] learn modular policies in the RL setting provided with such policy sketches. Shiarlis et al. [38] provides a modular LfD framework based on this idea of policy sketches. While all of these works address learning policies at various levels from demonstrations, unlike our approach, they each assume access to heavy supervision over demonstrations to do so.

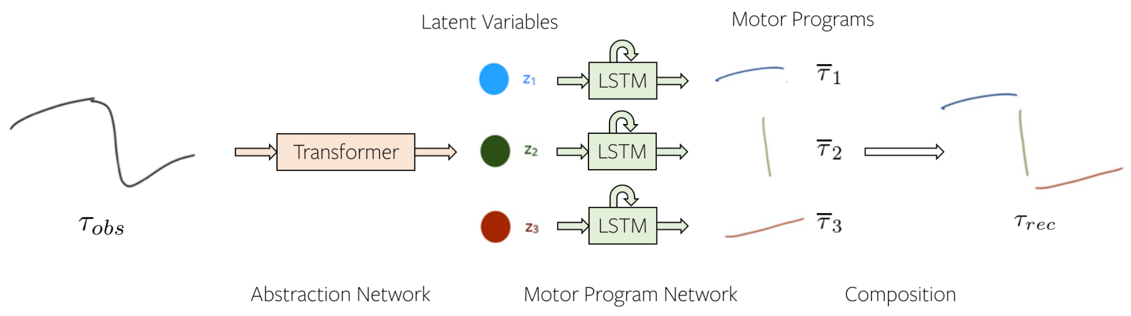


Figure 3.2: An overview of our approach. Our abstraction network takes in an observed demonstration τ_{obs} and predict a sequence of latent variables $\{z\}$. These $\{z\}$ are each decoded into their corresponding motor programs via the motor program network. We finally recompose these motor programs into the recomposed trajectory.

3.3 Approach

We seek to discover the space of motor programs directly from unstructured demonstrations in an unsupervised manner, and show that these can help understanding similarities across tasks as well as quickly quickly adapting to and solving new tasks. Building on ideas of Keele [53] and Schmidt [55], we define a motor program M as a movement pattern that may be executed in and of itself, without access to sensory feedback. Concretely, a 'movement sequence' or a motor program M is a sequence of robot joint configurations. Our goal is to learn the space of such movement patterns that are present across a diverse set of tasks. We do so via learning a 'motor program network' that maps elements $z \in R^n$ to corresponding movement sequences *i.e.* $\mathcal{M} : z \longrightarrow M$.

Given a set of N unlabelled demonstrations $\{\tau_i\}_{i=1,2,...,N}$ that consist of sequences of robot states $\{s_1, s_2, ..., s_T\} \in S$, our aim is to learn the shared space of motor programs via the motor program network $\mathcal{M}$. However, as each demonstration τ_i is unannotated, we do not apriori know what subsequences of the demonstration correspond to a distinct motor program. Therefore, to learn motor programs from these demonstrations, we need to simultaneously learn to understand the demonstrated trajectories in terms of composition of motor programs. Thus in addition to learning the network $\mathcal{M}$, we also learn a mapping $\mathcal{A}$ from each demonstrated trajectory τ_i to the underlying sequence of motor programs $\{M_1, M_2, ..., M_K\}$ (and associated latent variables $\{z_1, z_2, ..., z_K\}$) executed during the span of the trajectory. We call this mapping $\mathcal{A} : \tau_i \longrightarrow \{M_1, M_2, ..., M_K\}$ the abstraction network, as it abstracts away the details of the trajectory into the set of motor programs (i.e., abstractions). Note that both the abstraction and motor program networks are learned using only a set of demonstrations from across diverse tasks.

3.3.1 Primitive Discovery via Recomposition

Our central insight is that we can jointly learn the space of motor programs and the abstraction of the demonstration by enforcing that the implied 'recomposition' is faithful to the original demonstration. Concretely, an abstraction of a demonstration into a sequence of motor programs, each of which corresponds to an implied motion sequence, must yield back the original demonstration when the inferred motor programs are decoded and 'recomposed'. We operationalize this insight to jointly train the motor program and abstraction networks from demonstrations.

Learning Overview and Objective: Our approach is outlined in Fig 3.2, where given an input demonstration trajectory τ_{obs}, we use the abstraction network to predict a sequence of (a variable number of) latent codes $\{z_k\}$. These are each decoded into corresponding sub-trajectories via the learned motor program network $\mathcal{M}$.

$$\{z_k\} = \mathcal{A}(\tau_{obs}); \quad \bar{\tau}_k = \mathcal{M}(z_k) \tag{3.1}$$

Given the decoding of the predicted motor programs, we can recompose these sub-trajectories to obtain a *recomposed* demonstration trajectory τ_{rec}, and penalize the discrepancy between the observed and the recomposed demonstrations. Denoting by $\oplus$ the concatenation operator, our loss for a given demonstration τ_{obs} is therefore characterized as:

$$\tau_{rec} = \bar{\tau}_1 \oplus \bar{\tau}_2 \cdots \oplus \bar{\tau}_K; \quad L(\tau_{obs}; \mathcal{M}, \mathcal{A}) = \Delta(\tau_{obs}, \tau_{rec}) \tag{3.2}$$

As the trajectories τ_{obs}, τ_{rec} are possibly of different lengths, we use a pairwise matching cost between the two trajectories, where the optimal alignment is computed via dynamic time warping [63]. This provides us with a more robust cost measure that handles different prediction lengths, is invariant to minor velocity perturbations, and enables the model to discard regions of inactivity in the demonstrations. Given two trajectories $\tau_a \equiv (s_1 \cdots s_M)$, $\tau_b \equiv (s_1 \cdots s_N)$, and a distance metric δ over the state space, the discrepancy measure between trajectories can be defined as the matching cost for the optimal matching path P among all possible valid matching paths $\mathcal{P}$ (*i.e.* paths satisfying monotonicity, continuity, and boundary conditions [63]):

$$\Delta(\tau_a, \tau_b) = \min_{P \in \mathcal{P}} \sum_{(m,n) \in P} \delta(\tau_a[m], \tau_b[n]) \tag{3.3}$$

As the recomposed trajectory comprises of distinct primitives, each of which implies a sequence of states, we sometimes observe discontinuities *i.e.* large state changes between the boundaries of these primitives. To prevent this, we additionally incorporate a smoothness loss $L_{sm}(\tau_{rec})$ that penalizes the state change across consecutive time-steps if they are larger than a certain margin. Our overall objective, comprising of the reconstruction objective and the smoothness prior, can allow us to jointly learn the space of motor programs and the abstraction of trajectories in an unsupervised manner.

Network Architecture and Implementation Details. We parameterize our motor program network $\mathcal{M}$ and our abstraction network $\mathcal{A}$ as neural networks. In particular, the motor program network is a 4 layer LSTM [64] that takes a single 64 dimensional latent variable z as input, and predicts a sequence of 16 dimensional states. For our abstraction network, we adopt the Transformer [65] architecture to take in a varying length 16 dimensional continuous joint angle trajectory τ as input, and predict a variable number of latent variables $\{z\}$, that correspond to the sequence of motor programs $\{M\}$ executed during trajectory τ. We find the transformer architecture to be superior to LSTMs for processing long trajectories, due to its capacity to attend to parts of the trajectory as required.

Our abstraction network $\mathcal{A}$ predicts a varying number of primitives by additionally predicting a 'continuation probability' p_k after each motor program variable z_k. We then predict an additional primitive only if the sampled discrete variable from p_k is 1, and therefore also need to learn the prediction of these probabilities. While the loss function above can directly yield gradients to the predicted motor program encoding z_k via $\mathcal{M}$, we use gradients using REINFORCE [42] (with $\Delta(\tau_{obs}, \tau_{rec}) + L_{sm}(\tau_{rec})$ as negative reward) to learn prediction of p_k.

3.3.2 Enforcing Simplicity and Parsimony

While the objective described so far can in principle allow us to jointly learn the space of motor programs and understand the demonstration trajectories as a composition of these, there are additional properties we would wish to enforce to the bias the learning towards more desirable solutions. As an example, our framework presented so far can allow a solution where each demonstration is a motor program by itself *i.e.* the abstraction network can learn to map each demonstration to a unique z, and the primitive decoder can then decode this back. However, this is not a suitable solution as the learned motor programs are not 'simple'. On the other extreme, a very simple notion of a motor program is one that models each transition independently. However, this is again undesirable as this does not 'abstract' away the details of control or represent the demonstration as a smaller number of motor programs. Therefore, in addition to enforcing that the learned motor programs recompose the demonstrations, we also need to enforce simplicity and parsimony of these motor programs.

We incorporate these additional biases by adding priors in the objective or model space. To encourage the abstraction model to learn parsimonious abstractions of the input demonstrations, we penalize the number of motor primitives used to recompose the trajectory, by adding a small constant to the negative reward used to train the continuation probability p_k if the corresponding sample yielded an additional primitive. To enforce simplicity of motor primitives, we observe that the trajectories yielded by a classical planner (in our case, RRT-Connect) *e.g.* to go from an initial to final state are 'simple' and we therefore initialize the motor primitive network using the decoder of a pretrained autoencoder on random planner trajectories for (start, goal) state

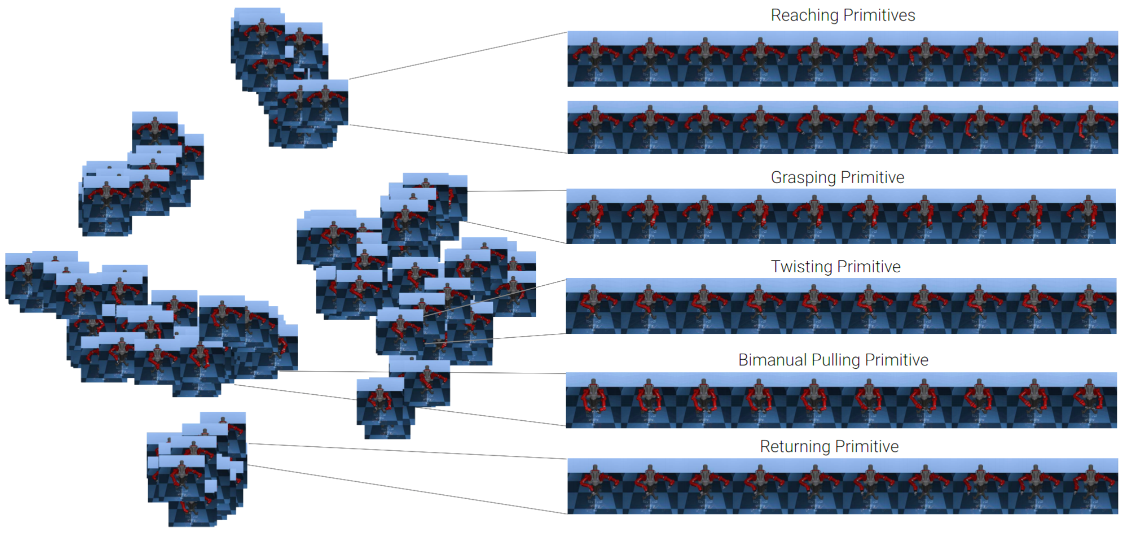


Figure 3.3: Visualization of the embedding of the latent representation of motor programs learned by our model (depicted on the left are the start configurations of each motor program, displayed at the corresponding position in the embedded space), and a set of sample primitives unrolled in time (depicted on the right). Each row corresponds to one primitive, annotated with semantic labels of what this motor primitive resembles.

pairs. Note that this notion of 'plannability' as 'simplicity' is merely one plausible alternative, and alternate ones can be explored *e.g.* 'linearity' [66] or 'predictability of motion' [67].

3.4 Experiments

We would like to ascertain how well our approach is capable of achieving our objective of successfully discovering and learning a good representation of the space of motor programs. Further, we seek to verify whether despite being learned in an unsupervised manner without semantic grounding, the learned primitive space is semantically meaningful. We would also like to evaluate how well they can be used to solve downstream RL tasks. We first provide a description of the data we wish to learn primitives from, followed by describing our quantitative and qualitative experiments towards verifying these three axes.

Dataset: We use the MIME dataset [16] to train and evaluate our model. The dataset consists of over 8000 kinesthetic demonstrations of 20 tasks (such as pouring, pushing, bottle opening, stacking objects, etc.) collected on a real-world Baxter Robot. While the dataset has head and hand-mounted RGBD data, we use the Baxter joint angle trajectories to train our model. We consider a 16 dimensional space as our input and prediction space, consisting of 7 joints for each of the 2 arms, along with a scalar value for each gripper (we ignore torso and head joints). We emphasize this is a higher dimensional domain than most other related works consider. The gripper values are re-scaled to a $0 - 1$ range, while the joint angles are unnormalized. We temporally down-sample all joint angle data by a constant factor of 20 from the original data frequency of 100 Hz.

We randomly sample a train set of 5900 demonstrations from all 20 tasks, with a validation set of 1600 trajectories, and a held-out test set of 850 trajectories. To help evaluate the learned motor programs, we manually annotate a set of 60 test trajectories (3 trajectories from each task) with temporal segmentation annotations, as well as semantic labels of 10 primitives (such as reaching, twisting, pushing, etc.) that occur in these 60 trajectories. Note that these labels are purely for evaluation, our model does not have access to these annotations during training.

3.4.1 Visualizing the Space of Primitives

We would first like to evaluate the quality of the learned abstractions in and of themselves, i.e. *Is our approach able to discover the space of motor programs, and learn a good representation of this space?* We qualitatively answer this question by visualizing the latent representation space of motor primitives learned by our model. We first randomly sample a set of 500 trajectories unseen during training, then pass these trajectories through our model, and retrieve the predicted latent variables $\{z\}$ for each of these trajectories and their corresponding movement sequences $\{\overline{\tau}\}$. We then embed the latent variables in a 2-dimensional space using T-SNE [45], and visualize the corresponding movement sequences at their corresponding position in this 2-D embedded space, as in Figure 3.3. We provide a GIF version of Figure 3.3 (and other visualizations) at https://sites.google.com/view/discovering-motor-programs/home.

We observe that clusters of movement sequences emerge in this embedded space based on the relative motions executed during the course of these trajectory segments (similar latent variables correspond to similar movement sequences and vice versa). While these clusters are not explicitly semantically labelled by our approach, the motions in these clusters correlate highly with traditional notions of skills in robot manipulation, i.e. reaching motions (top and top-left clusters), returning motions (bottom cluster), bi-manual motions (visible to the bottom right of the left most cluster and the bottom of the right most cluster), etc. We visualize a few such primitives (reaching, twisting, grasping, bi-manual pulling, etc.) among these to the right of Figure 3.3. This shows our model learns smooth mappings $\mathcal{M}$ and $\mathcal{A}$, and is capable of *discovering* such primitives in an unsupervised manner without explicit temporal segmentation or semantic labels, which we believe is an encouraging result.

Interestingly, the model learns abstractions that pick up on the *trend* of the motion, rather than distinguishing between whether the left or right hand is used for the motion. This is particularly notable in the case of reaching and returning motions, where both left and right-handed reaching and returning motions appear alongside each other in their respective clusters in the embedded space.

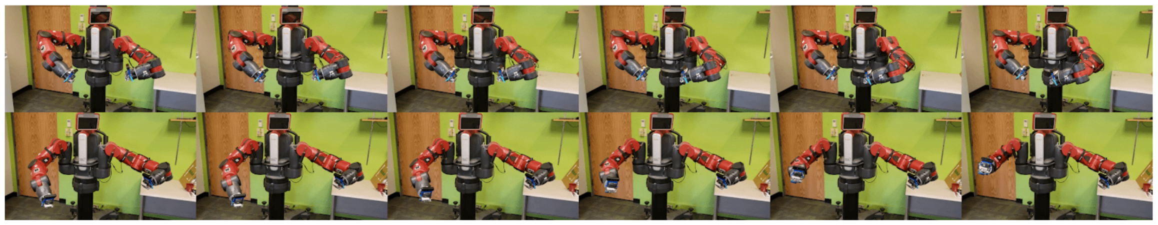


Figure 3.4: Depiction of execution of the learned primitives on real world Baxter robot. Each row is a single primitive, while columns show progress of the primitive over time. Row 1 depicts a left handed reaching primitive, while row 2 shows a right handed returning primitive. More visualizations provided in supplementary material, and videos are provided in the webpage.

Executing Primitives on a real Robot

We would ideally like the learned primitives from our model to be useful on a real Baxter robot platform, and be suitably smooth, feasible, and correspond to the motions executed in simulation (i.e. be largely unaffected by the noise of execution on a real robot). To verify whether our model is indeed able to learn such primitives, we execute a small set of learned primitives on a real Baxter robot, by feeding in the trajectory predicted by the model into a simple position controller. We visualize the results of this execution in Figure 3.4, (see project webpage for videos). Despite not explicitly optimizing for feasibility or transfer to a real robot, the use of real-world Baxter data to train our model heavily biases the model towards primitives that are inherently feasible, relatively smooth and can be executed on a real robot without any subsequent modifications.

3.4.2 Semantic Segmentation Transfer using Learned Abstractions

As the 'recomposed' trajectory can be aligned to the original demonstration via sequence alignment, our predicted abstractions induce a partitioning of the demonstrated trajectory (corresponding to the aligned boundaries of the predicted primitives). We test whether the predicted abstraction and induced partitions are consistent across different demonstrations from the same task. To this end, we select 3 instances of the "Drop Object" task [16] from the annotated test set, and retrieve the induced segmentations of the demonstration. We then visualize these segmentations and motor programs predicted for each of these 3 demonstrations as depicted in Figure 3.5, along with the ground truth semantic labels of primitives for each of the demonstrations.

The alignment between a recomposed trajectory and the original demonstration also allows us to transfer semantic annotations from a demonstration to the predicted primitives, by simply copying labels from the demonstration to their aligned timepoints

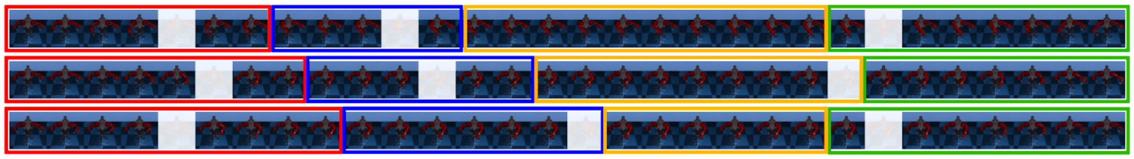


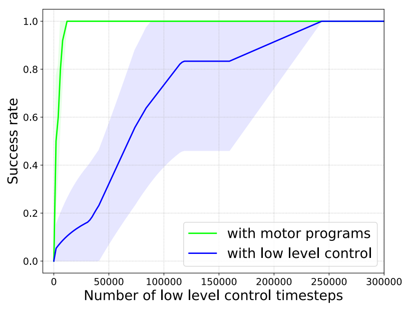
Figure 3.5: Visualization of consistent segmentations. Each row represents a different instance of a "Drop Objects" task from the MIME Dataset, while each column represents a time-step in the demonstration. White frames represent predicted segmentation points, while colored boxes represent ground truth semantic annotations. Red boxes are reaching primitives, blue boxes are grasping, orange boxes are placing, and green boxes are returning primitives. We see our model predicts 4 motor programs - reaching, grasping, placing the object a small distance away, and returning. This is consistent with the true semantic annotations of these demonstrations, and the overall sequence of primitives expected of the "Drop Box" task.

in the primitives. Therefore using our small *set* of annotated demonstrations, we can construct a small library of semantically annotated primitives. Given a novel, unseen demonstration, we can compute its predicted primitives and assign each a semantic label by copying the label of the nearest primitive from the library. This allows us to transfer semantic segmentations from our small annotated test set to unseen demonstrations. Our model's transfer of semantic segmentation achieves label accuracy on the set of 30 held out trajectories (across all 20 tasks) of 58%, while a *supervised* LSTM baseline that predicts semantic labels from trajectories (trained on 30 annotated test trajectories) achieves 54% accuracy.

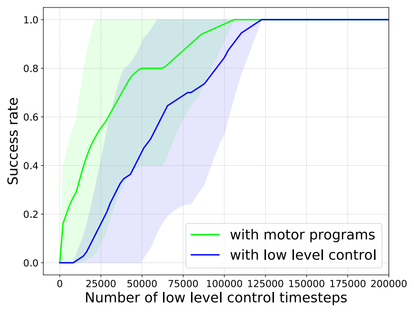
The consistency of our abstractions coupled with the ability to transfer semantic segmentations across demonstrations shows our model is capable of understanding commonalities across demonstrations, and is able to reason about various demonstrations in terms of motor programs shared across them, despite being trained in an unsupervised manner.

3.4.3 Composing Primitives for Hierarchical RL

One of our primary motivations for learning motor programs is that they can be composed together to solve downstream robotic tasks. To evaluate whether the motor programs learned by our model are indeed useful for such downstream tasks, we adopt a hierarchical reinforcement learning setup (as described in detail in the supplementary). For a given task, we train a policy to predict the sequence of motor programs to execute. Given the predicted latent representations, each motor program is decoded into its corresponding motion sequence using the previously learned motor program network. We retrieve a sequence of desired joint velocities from this motion sequence and use a joint velocity controller to execute these "low-level" actions on the



(a) Sparse Baxter Reacher Task



(b) Sparse Baxter Pushing Task

Figure 3.6: RL training curves with and without motor programs. Solid lines denote mean success rate, while shaded region denotes ± 1 standard deviation across 10 random seeds.

robot. The motor program network and the joint velocity controller together serve as an "abstraction" of the low-level control that is executed on the robot. Hence, the policy must learn to predict motor programs that correspond to motion sequences useful for solving the task at hand.

As demonstrated in Fig. 3.6, training a policy using motor programs is several orders of magnitude more efficient than training with direct low-level actions. For the sparse reaching task, the motor program policy learns within 50 motor program queries, 2 orders of magnitude speedup in low-level control time-steps. For the sparse pushing task, the motor program policy learns within 1000 motor program queries, or a 2X speedup with respect to low-level control time-steps. We note that executing a motor program corresponds to 50 low level control steps (see appendix for details). However, as these motor programs are executed without environment feedback, the improvement in efficiency in terms of environment interactions is all the more significant.

3.5 Conclusion

We have presented an unsupervised approach to discover motor programs from a set of *unstructured* robot demonstrations. Through the insight that learned motor programs should recompose into the original demonstration while being simplistic, we discover a coherent and diverse latent space of primitives on the MIME [16] dataset. We also observed that the learned primitives were semantically meaningful, and useful for efficiently learning downstream tasks in simulation. We hope that the contributions from our work enable learning and executing primitives in a plethora of real-world robotic tasks. It would also be interesting to leverage the learned motor programs in context of continual learning, to investigate how the discovered space can be adapted and expanded in context of novel robotic tasks.

III

Translating Skills from Humans to Robots

4 Learning Unsupervised Skill Correspondences across Domains

4.1 Introduction

Humans have a remarkable ability to efficiently learn to perform tasks by watching others demonstrate similar tasks. For example, children quickly learn the skills needed to play a new sport by watching their parents perform skills such as kicking a ball. Notably, they are able to learn from these visual demonstrations despite significant differences between themselves and the demonstrator, including visual perspectives, environments, kinematic and dynamic properties, and morphologies. This ability may be attributed to two factors; firstly, humans have well-developed basic motor skills that we can execute with little effort. Second, we can recognize the sequence of skills (or the high-level strategy) the demonstrator uses, and understand a corresponding set of skills that we can execute ourselves [1].

In this paper, we explore how we can endow robots with this ability - i.e., to adopt task strategies from agents with different embodiments (such as morphologically different robots, or even human demonstrators), and then execute corresponding skills to solve similar tasks. Key to solving this problem is for the robot to identify how its owns skills correspond to those of the demonstrator, which is tremendously powerful. First, it allows the robot to adopt the demonstrator's strategies for solving various tasks. Second, by adopting and *adapting* these strategies for itself, the robot can efficiently learn to solve a variety of tasks previously outside its repertoire. Finally, it allows us to understand the task strategies used by various agents in a unified manner, enabling robots to learn from data collected from a heterogeneous collection of tasks and agent embodiments. Skills provide a natural framework to facilitate such concise knowledge transfer across agents compared to low-level controls; skills inherently abstract away low-level details that may differ across the demonstrator and the learner, and instead focus on the commonalities between them, such as the task strategy. For example, skills such as reaching and placing on robot manipulators abstract away differences in morphologies or configuration, and are thus well grounded across robots,

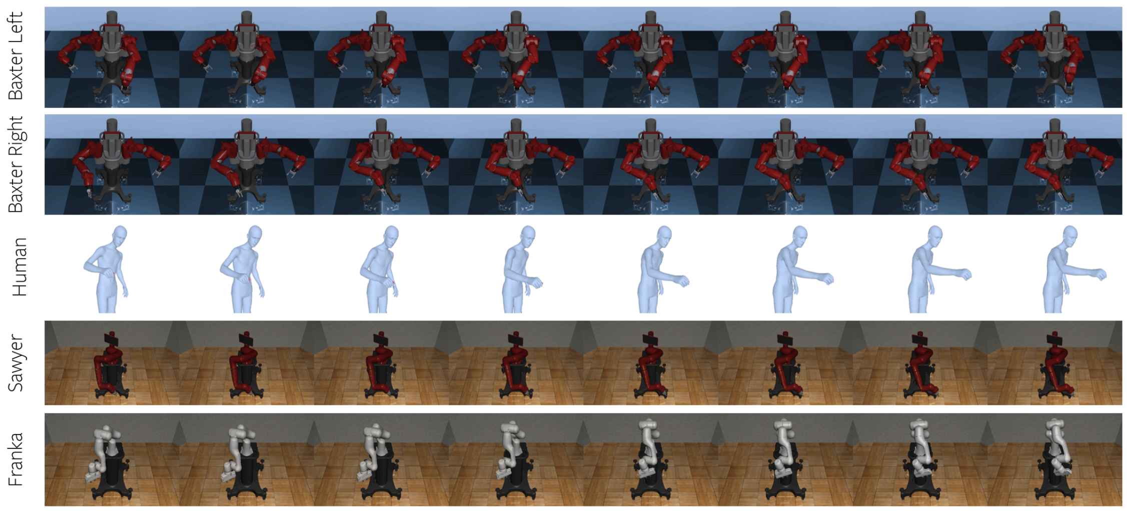


Figure 4.1: Sample skill correspondences learnt by our unsupervised approach, across the 4 different morphological robots and a human demonstrator. We visualize a "placing skill" as translated by our approach, used to place objects to the left of each of the agents. Note the semantic correspondence between these skills, despite our approach being completely unsupervised.

while there may not be obvious correspondence in their low-level actions.

How can one acquire correspondences between skills? Stated more formally, given agents with different embodiments, a set of unlabelled demonstrations of each agent solving a variety of tasks, and a method for extracting skills from those demonstrations, how can one learn correspondences between the skills of these different agents? Learning such correspondences is straightforward when supervised pairs of skills or trajectories are available. However, collecting and annotating such data is time-consuming and tedious, and requires a high degree of human expertise, particularly at the scale necessary to successfully learn correspondences across a diverse set of skills. In contrast, we explore whether we can learn such skill correspondences in a fully *unsupervised* manner, i.e., without access to supervised skill or trajectory pairs. Learning *unsupervised* skill correspondences can enable learning from a diverse and heterogeneous dataset spanning multiple tasks and robots (e.g. Sharma et al. [16], Mandlekar et al. [17]). While unsupervised learning of correspondences is more scalable, it is also difficult at the same time due to the lack of a grounded learning signals that can enable end-to-end learning.

To overcome these challenges with unsupervised learning of correspondences, we propose leveraging the following insights. Our first insight and contribution is that differently embodied agents use similar task strategies (in terms of sequences of skills) to solve similar tasks; in other words, the sequences of skills executed by different agents to solve similar tasks ought to belong to similar *distributions*. For example, to make a cup of tea, both a robot and a human would likely first reach for the kettle, grasp it, move it appropriately over the cup, and then begin to pour the tea, irrespective of their exact morphology. Our second insight and contribution is that learning skill correspondences without access to supervised data closely mirrors unsupervised machine translation (UMT), where the objective is to learn a translation between representations in different languages (such as between word embeddings across languages), without access to parallel data [68, 69]. Inspired by these two insights, we present our third contribution - deriving an unsupervised objective to guide our learning towards meaningful skill correspondences.

We approach the problem of learning unsupervised skill correspondences by learning a translation model to map from the skill space on one "source" agent to the skill space on another "target" agent. We construct an unsupervised objective that encourages the translation model to respect our first insight - i.e., to preserve the sequences of skills observed across both the (translated) source and target agents. To do this, we take inspiration from our second insight, and specifically from Zhou et al. [70], and construct explicit probability density models over the skill sequences observed in the source and target domains, and then train the translation model to match these distributions. We demonstrate that our approach is able to learn semantically meaningful correspondences across four different robots (the Franka Panda, Sawyer, Baxter Left and Right hands) *and* a human agent, despite being completely unsupervised, as depicted in Figures 4.1 and 4.3. The learnt correspondences facilitate transferring task strategies across across domains, as we demonstrate on a set of downstream tasks. Our results are visualized at `https://sites.google.com/view/translatingrobotskills/home`

4.2 Related Work

Skill Learning: Eysenbach et al. [71], Sharma et al. [72] both address unsuperivsed skill learning from interaction data, by constructing information theoretic approaches. Fox et al. [12], Krishnan et al. [13], Kipf et al. [14], Sharma et al. [16], Shankar et al. [22], Gregor et al. [36], Shankar and Gupta [73], Kim et al. [74] instead learn skills or abstractions from unlabelled demonstration data by performing latent variable inference. These frameworks all learn skills in the context of a single domain, while we seek to learn correspondences of skills *across* domains.

Unsupervised Correspondence Learning: Our problem bears a close resemblance with unsupervised machine translation [68, 70], and unpaired image translation [75, 76]. Specifically, they all share the notion of learning correspondences across representations learnt from unpaired data:

- In Machine Translation: Conneau et al. [68] leveraged domain-adversarial training [77] to align word embeddings of two languages. Lample et al. [69], Sennrich et al. [78] use the idea of back-translation to constraint the learnt translation models across languages. Zhou et al. [70] learn bilingual word embeddings by matching explicit density functions over the word embedding spaces.

- In Image Translation: The vision community has similarly explored the unpaired *image-to-image* translation setting [75, 76], using cycle-consistency [75] or contrastive losses [76].
- In Video: Bansal et al. [79], Wang et al. [80] extend Cycle-GAN [75] to the video domain, by incorporating temporal consistency losses.

Domain Transfer in Robotics and Graphics: The robotics and graphics communities have taken interest in cross-domain transfer of policies in recent years:

- Policy Transfer with Paired Data: Gupta et al. [81], Sermanet et al. [82] learn morphology and viewpoint invariant feature spaces for policy transfer respectively, but require paired data to do so.
- Policy Transfer via Modularity: Hejna et al. [83], Devin et al. [84], Sharma et al. [85] address morphological transfer by modularity in their policies. Hejna et al. [83], Sharma et al. [85] adopt modularity in a hierarchical sense, but leverage common grounding of subgoals across morphologies to perform transfer. We do not assume access to such common grounding.
- State based Policy Transfer: Liu et al. [86], Schroecker and Isbell [87], Ammar et al. [88] address cross-morphology transfer by state based imitation learning.
- Motion Retargetting: The graphics community has addressed transferring behaviors across morphologically different characters [89, 90, 91, 92], using carefully handcrafted kinematic models. These works transfer behaviors by imitating joint positions, and performing inverse kinematics to retrieve the character state, which is infeasible for widely different morphological characters.
- Unsupervised Action Correspondence: Zhang et al. [93], Kim et al. [94], Smith et al. [95] address learning *low-level* state and action correspondences from unpaired, unaligned interaction and demonstration data respectively. While similar in spirit, our work argues that learning high-level skill correspondence instead is a more natural choice, as mentioned in the introduction.

Applicability to learning skill correspondences: While successful in their respective problem domains, existing approaches are not trivially applicable to our problem. The adversarial training used in most of these approaches is notoriously difficult to train, and is prone to the mode dropping problem [96]. They require strong constraints such as pixel-wise or joint-wise identity losses [75, 93], or inherent similarity of the spaces to be aligned, such as word embeddings across languages [68, 70]. Learnt skill representations do not possess this property in general. Shortcomings notwithstanding, these approaches provide insight into how we may pursue learning unsupervised skill correspondences.

4.3 Approach

4.3.1 Pre-requisites

We begin by first reviewing an important prerequisite - the skill learning pipeline of Shankar and Gupta [73], which provides us a learnt representation of robotic skills given an unlabelled robot demonstration dataset. Their method first represents robotic skills as continuous latent variables z, and introduce a Temporal Variational Inference (TVI) to infer these skills or latent variables. TVI trains a variational encoder $q(z|\tau)$ that takes as input a robot trajectory $\tau = \{s_1, a_1, ...s_{n-1}, a_{n-1}, s_n\}$, and outputs a sequence of skill encodings $z = \{z_1, z_2, ...z_n\}$. Note that these skill encodings repeat over time for the duration of a given skill. TVI also trains a latent conditioned policy $\pi(a|s, z)$ that takes as input robot state s, and the chosen skill encoding z, and predicts the low-level action a that the robot should execute. We direct the reader to Shankar and Gupta [73] for a more thorough description of their skill learning approach.

4.3.2 Problem Setting

Consider two robots with potentially differing morphologies and environments (collectively called "domain"), represented as source and target domains, or M_{S} and M_{T} respectively. Associated with each domain is an unlabelled and unsegmented dataset of demonstrations of the robot performing various tasks represented as D_{S} and D_{T} respectively. D_{S} and D_{T} are collected independently, on an intersecting (but not identical) set of tasks. This is equivalent to the setting in machine translation without *parallel* data, with access only to *monolingual* corpora.

We also assume access to a skill-learning pipeline, that can take in such an unlabelled demonstration dataset D of a robot M performing various tasks, and learns a representation $\mathbb{Z}$ of skills on a given robot. We specifically use TVI [73], but we believe our approach is compatible with any unsupervised skill-learning approach that affords a continuous representation space. We train the skill-learning pipeline independently on each of the domain datasets D_{S} and D_{T}. This affords us skill encoders q_{S} and q_{T}, and latent conditioned policies π_{S} and π_{T} for each domain. We may then encode trajectories in both domains $\tau_{\text{S}} \in D_{\text{S}}$ and $\tau_{\text{T}} \in D_{\text{T}}$ into their constituent skills $\{z^t_{\text{S}}\}_{t=1}^{|\tau_{\text{S}}|}$ and $\{z^t_{\text{T}}\}_{t=1}^{|\tau_{\text{T}}|}$, via q_{S} and q_{T} respectively. We represent the space of skills learnt on the entire dataset in each of the two domains as $\mathbb{Z}_{\text{S}}$ and $\mathbb{Z}_{\text{T}}$ respectively.

Our goal is to learn semantically meaningful correspondences between skills in the source domain, $z_{\text{S}} \in \mathbb{Z}_{\text{S}}$, and those in the target domain, $z_{\text{T}} \in \mathbb{Z}_{\text{T}}$, that helps solve tasks in the target domain M_{T}. We specify these correspondences by learning a "translation" function $T_{\text{S}\to\text{T}} : \mathbb{Z}_{\text{S}} \to \mathbb{Z}_{\text{T}}$, that maps skills in the source domain z_{S}, to translated skills in the target domain $z_{\text{S}\to\text{T}}$. This mirrors the word translation models present in Conneau et al. [68], Zhou et al. [70]. Learning meaningful skill-correspondences thus amounts to learning a good translation model T. We can also translate the entire source space $\mathbb{Z}_{\text{S}}$ to the target domain, to retrieve the translated source space, $\mathbb{Z}_{\text{S}\to\text{T}}$.

4.3.3 Using sequential information of skills

Our first insight is that differently embodied agents follow similar strategies to address similar tasks - in other words, sequences of skills executed by two different agents ought to belong to similar distributions. It is important to consider *sequences* of skills here rather than skills in isolation since the ordering of skills can guide our learning towards the right correspondences. Constraining correspondences based on individual skills alone could lead to incorrect results. For example, "reaching" skills in one domain corresponding to "grasping" skills in another domain is incorrect, but could be prevented by sequential information, since grasping skills are systematically executed *after* reaching skills.

However, processing entire sequences of skills is computationally infeasible for arbitrarily long trajectories. Instead, we draw inspiration from the simple yet powerful bigram models in the NLP community, and consider consecutive pairs, or *tuples* of skills. While less expressive than the full skill-sequences, these tuples retain enough information of the ordering of skills to guide the correspondence learning towards the right solution, and are far more computationally tractable. We thus construct a skill-tuple space, which represents the various transitions between skills observed in a given domain. Each consecutive pair of skills (z^t, z^{t+1}) in the original space $\mathbb{Z}$ of a given domain is represented as a single point x^t in the skill-tuple space $\mathbb{X}$ of that domain. For initial and terminal skills $z^{t=0}$ and $z^{t=|\tau|}$, we pad these tuples, and treat the preceding and succeeding skill encodings as appropriately dimensioned 0's respectively.

The skill-tuple spaces for source and target domains are represented as $\mathbb{X}_{\text{S}}$ and $\mathbb{X}_{\text{T}}$ respectively, and are implemented as a set of skill-tuples from each domain, i.e. $\mathbb{X}_{\text{S}} = \{x_{\text{S},i}\}_{i=1}^{N_{\text{S}}}, \mathbb{X}_{\text{T}} = \{x_{\text{T},i}\}_{i=1}^{N_{\text{T}}}$. The procedure to construct these spaces is presented in the sub-routine of **??**. We also construct a translated skill-tuple space, $\mathbb{X}_{\text{S}\to\text{T}}$, that is simply the translation of all skills present in $\mathbb{X}_{S}$, i.e. $\mathbb{X}_{\text{S}\to\text{T}} = \{x_{S\to T,i}\}_{i=1}^{N_S}$. Translating a skill-tuple $x_{\text{S}}^t = (z_{\text{S}}^t, z_{\text{S}}^{t+1})$ from source to target is done by translating the individual skills z_{S}^t and z_{S}^{t+1} to the target domain,$z_{\text{S}\to\text{T}}^t$ and $z_{\text{S}\to\text{T}}^{t+1}$, and constructing the translated skill-tuple as $x_{\text{S}\to\text{T}}^t = (z_{\text{S}\to\text{T}}^t, z_{\text{S}\to\text{T}}^{t+1})$.

4.3.4 Distributional perspective on learning translations

We would like our translation model to exhibit two properties - first, to translate source skills z_{S} such that they belong to the distribution of target skills, and second, to capture all modes of skills that exist in the target domain. Dropping modes of skills limits the set of skills the target robot can use, reducing its capabilities. GANs [97] and domain-adversarial approaches [77] satisfy the first property since they optimize

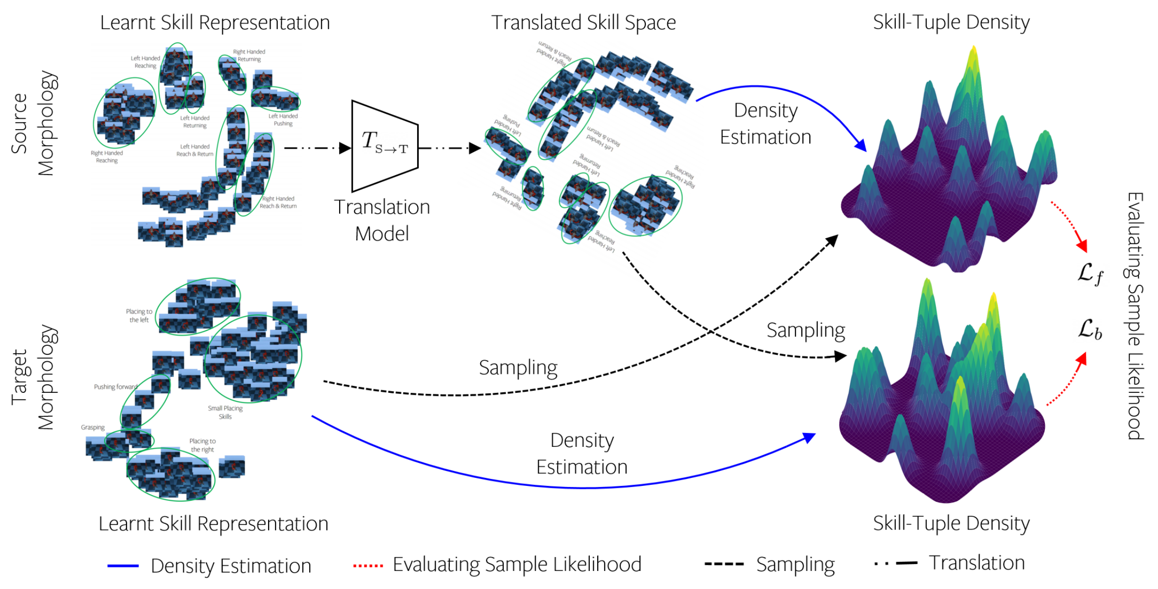


Figure 4.2: Overview of our approach. We translate the original learnt skill space to the target domain. We then construct explicit density estimates over the original target and translated source skill-tuple spaces. We train our translation model to maximize the likelihood of randomly sampled translated source and original target skill-tuples under these densities respectively, affording meaningful skill-correspondences across both robots.

for how realistic the generated samples are (or how indistinguishable source and target domain features are [77]), but often drop modes of the "real" data [96].

Instead, we approach this problem from an *explicit* distribution perspective, and maintain explicit probability density estimates over the translated source and target skill-tuple spaces, $\mathbb{X}_{S\to T}$ and $\mathbb{X}_T$, $p(x_{S\to T})$ and $p(x_T)$ respectively. Following our insight that sequences of skills on different robots ought to belong to similar distributions, we would like these distributions $p(x_{S\to T})$ and $p(x_T)$ to match. We can optimize our translation model to enforce this, by maximizing the likelihood of translated skill-tuples $x_{S\to T}$ under the target skill-tuple distribution $p(x_T)$, and the likelihood of target skill-tuples x_T under the translated skill-tuple distribution $p(x_{S\to T})$. This is similar in spirit to optimizing *both* the forward and reverse KL between two distributions. We formalize this objective and provide intuition for it below.

We can train a translation model $T_{S\to T}$ to translate skills in such a way that the translated skill tuples $x_{S\to T}$ belong to the distribution of target skill-tuples $p_T(x)$, or that they are realistic with respect to the target distribution. We do this by maximizing the "forward" likelihood $\mathcal{L}_f$ of the translated skill-tuples $x_{S\to T}$, under the target density $p_{\text{T}}(x)$:

$$\mathcal{L}_f = \mathbb{E}_{x_{\text{S}} \sim p(x_{\text{S}}), x_{\text{S}\to\text{T}} \,\sim\, T_{\text{S}\to\text{T}}(.|x_{\text{S}})} \Big[\log p_{\text{T}}(x_{\text{S}\to\text{T}}) \Big] \tag{4.1}$$

To encourage the second mode-covering property in the translation model, we can also optimize the "backward" likelihood $\mathcal{L}_b$, of the *target* skill-tuples x_{T} under the *translated source* skill-tuple space $p_{\text{S}\to\text{T}}(x)$:

$$\mathcal{L}_b = \mathbb{E}_{x_{\text{T}} \sim p(x_{\text{T}})} \Big[\log p_{\text{S}\to\text{T}}(x_{\text{T}}) \Big] \tag{4.2}$$

By combining these two objectives we may construct our final objective $\mathcal{L}$ for training the translation model, by simply combining these two objectives, $\mathcal{L} = \mathcal{L}_f + \mathcal{L}_b$:

$$\begin{aligned} \mathcal{L} = \mathbb{E}_{x_{\text{S}} \sim p(x_{\text{S}}), x_{\text{S}\to\text{T}} \,\sim\, T_{\text{S}\to\text{T}}(.|x_{\text{S}})} &\Big[\log p_{\text{T}}(x_{\text{S}\to\text{T}}) \Big] \\ + \mathbb{E}_{x_{\text{T}} \sim p(x_{\text{T}})} &\Big[\log p_{\text{S}\to\text{T}}(x_{\text{T}}) \Big] \end{aligned} \tag{4.3}$$

We can operationalize our full objective $\mathcal{L}$, by instantiating the target and translated source skill-tuple distributions $p_{\text{T}}(x_{\text{S}\to\text{T}})$, and $p_{\text{S}\to\text{T}}(x_{\text{T}})$ respectively. To do so, we construct explicit estimates of these distributions. We follow Zhou et al. [70]'s choice of representing these distributions using Gaussian Mixture Models (GMM), owing to their expressive power and efficiency in low-dimensional spaces, but note that a wide variety of explicit density estimators may be used here.

Here, the target space skill-tuple distribution, $p_{\text{T}}(x)$, is a GMM with N_{T} Gaussian kernels, each centered at a target skill-tuple $x'_{\text{T},i} \sim \mathbb{X}_{\text{T}}$:

$$p_{\text{T}}(x) = \sum_{i=1}^{N_{\text{T}}} p(x'_{\text{T},i})\ p(x|x'_{\text{T},i}) = \sum_{i=1}^{N_{\text{T}}}\ p(x|x'_{\text{T},i}) \tag{4.4}$$

Zhou et al. [70] use kernel weights $p(x'_{\text{T},i})$ proportional to the frequency of the word the kernel represents. In our setting, each kernel is simply one of the skill-tuples in the *continuous* target skill space $\mathbb{X}_{\text{T}}$, motivating our choice of equal mixture weights over all kernels (i.e. skill-tuples), i.e. $p(x'_{\text{T},i}) = 1/N_{\text{T}}$. The mixture components $p(x|x'_{\text{T},i})$ are specified by a fixed variance Gaussian centered at $x'_{\text{T},i}$, i.e. $p(x|x'_{\text{T},i}) = \mathcal{N}(x|x'_{\text{T},i}, \sigma^2)$, where σ specifies the standard deviation of the Gaussian kernel. Similarly, the translated source skill-tuple distribution $p_{\text{S}\to\text{T}}(x)$ is also represented as a GMM, with $N_{\text{S}\to\text{T}}$ equally weighted Gaussian kernels, each centered at a *translated* skill-tuple $x'_{\text{S}\to\text{T},i} \sim \mathbb{X}_{\text{S}\to\text{T}}$:

$$p_{\text{S}\to\text{T}}(x) = \sum_{i=1}^{N_{\text{S}\to\text{T}}}\ p(x|x'_{\text{S}\to\text{T},i}) \tag{4.5}$$

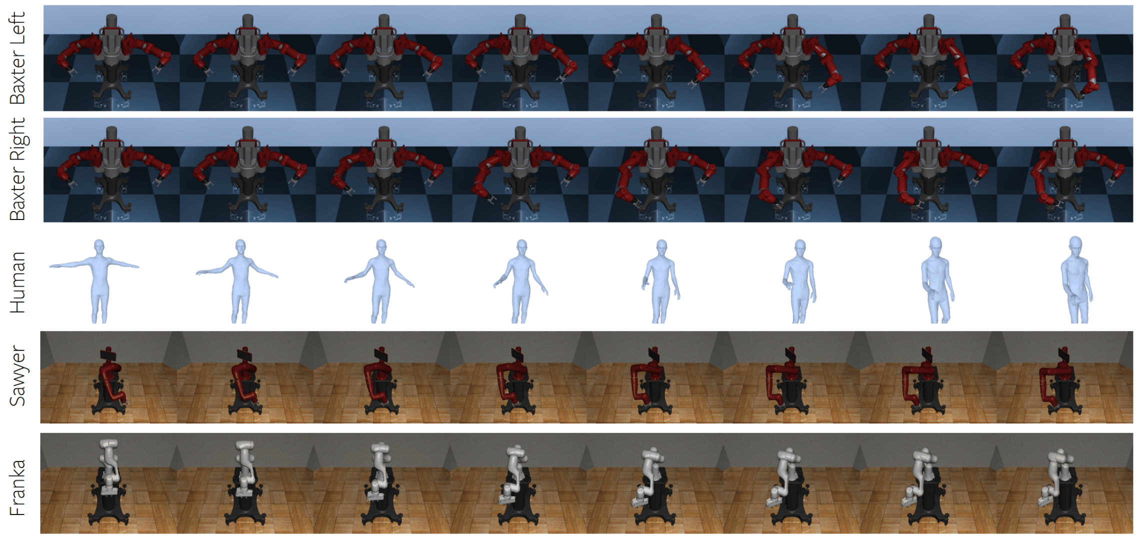


Figure 4.3: Sample skill correspondences learnt by our unsupervised approach, across the 4 different morphological robots and a human demonstrator. We visualize a "reaching skill" as translated by our approach, used to reach towards objects in the workspace of each of the agents. Note the semantic correspondence between these skills, despite our approach being completely unsupervised.

Using these parametrizations of distributions in our overall objective Equation (4.3), we have our unsupervised objective for learning skill correspondences.

Parsing the objective: In contrast with adversarial training methods, which require alternating training phases and stability tricks such as gradient penalties, our objective amounts to simple maximum likelihood (albeit in two directions). It is hence simpler, more stable, and quicker to converge.Intuitively, $\mathcal{L}_f$ captures how "realistic" each translated skill-tuple looks with respect to the target skill-tuple distribution, while $\mathcal{L}_b$ captures how well the translated skills cover the modes of the target skill-tuple space. We present a pictorial representation of our overall approach in Figure 6.2, and the full algorithm in the supplementary material.

4.4 Experiments

Agents: We evaluate our approach's ability to learn skills correspondences across the following robots - the Sawyer robot (Sawyer), the Franka Panda robot (Franka), the Baxter left hand (Bax-L), and the Baxter right hand (Bax-R). We consider domain pairs between each of these robots (Franka to Sawyer, Bax-R to Franka, Bax-L to Sawyer, Bax-R to Sawyer). In addition, we also consider translating from *human* demonstrators to each of the above robots. This results in 4 additional domain pairs.

Together, these domain-pairs span a wide variety of differences in morphology and embodiment of agents, such as number, type and configuration of joints, link lengths, joint limits, and dynamic properties, etc. The human agents also have different degrees of freedom (DoF) (23, compared to the robots' 8), and consist of complex joints with multiple DoFs (such as the shoulder), compared to the robots' single DoF joints. We believe our results across these diverse set of domains shows the efficacy of our approach.

Datasets: We make use of the following datasets for each agent. For the Sawyer robot, we use the Roboturk dataset [17], which consists of roughly 2000 demonstrations across 8 different tasks. For the Franka robot, we use the RoboMimic dataset [98], which has 800 demonstrations across 4 tasks. For the Baxter robot, we utilize the MIME dataset [16]. For human demonstrations, we consider the GRAB dataset [99], which consists of 10 different people manipulating various objects. Additional details, including preprocessing steps, are provided in our supplement.

Skill Representation: We learn skill-representations for each of the agents independently from their respective datasets, using TVI [73]. We use a 16-dimensional skill-representation space for each agent, that is learnt from joint-states and joint velocities. We then freeze these learnt skill representations for all of our experiments. We follow the preprocessing steps and training parameters specified by [73]. We also manually annotate 50 skills in each domain with one of 6 semantic labels of what type of skills they were. We emphasize that our approach is not trained with these labels, but rather is only evaluated against them.

4.4.1 Learning meaningful skill correspondences

The first question we would like to answer is - *"Can our unsupervised method learn meaningful skill-correspondences between skill spaces?"* We first present qualitative results towards answering this question.

Translating Individual Skills and Entire Trajectories: We present two sets of visualizations; individual skills and their translations, as well as entire trajectories (or sequences of skills) and their translations. We first visualize a set of source domain trajectories τ_{S}, obtained by rolling out their skill encodings $\{z^t_{\text{S}}\}_{t=1}^{|\tau|}$ via the source domain policy π_{S}. We then translate these sequences of skill encodings to the target domain, i.e. $\{z^t_{\text{S}\to\text{T}}\}_{t=1}^{|\tau|}$, and visualize rollouts of the *target* policy π_{T} conditioned on these translated skills. We present visualizations of individual skills and their translations, as well as visualizations of entire trajectories and their translations in Figures 4.1 and 4.3 and at `https://sites.google.com/view/translatingrobotskills/home`.

Analysing Qualitative Skill Translations: Despite being learned in a purely unsupervised manner, we observe our translation model is able to learn good semanti-

Table 4.1: Evaluating Skill-Tuple Distribution Matching via our approach against various baselines, across half of the considered domain pairs. We present results on the remaining domain pairs including the human-robot pairs in the supplementary material. Baselines are adapted from (1): Ganin et al. [77], (2): Zhu et al. [75], (3): Liu et al. [86]. Lower is better for all metrics except the label accuracy.

Domain Pair	Approach:	$\mathcal{L}_f$	$\mathcal{L}_b$	Chamfer Distance	Cycle Error	Label Accuracy	Supervised Z Error
Franka to Sawyer	DomAd [1]	56.4	70.5	20.1	17.3	8.2%	23.8
	Cycle GAN [2]	39.5	48.3	12.8	**8.4**	12.5%	19.7
	State Im [3]	66.8	81.4	28.3	32.8	24.2 %	29.2
	Ours	**21.2**	**35.9**	**8.7**	8.9	**70.7 %**	**10.2**
Baxter-L to Saw	DomAd [1]	50.9	60.1	24.2	21.3	9.8%	14.9
	Cycle GAN [2]	38.5	53.7	17.1	**8.8**	12.0%	15.4
	State Im [3]	66.5	73.1	38.9	22.4	19.3 %	36.8
	Ours	**16.8**	**33.9**	**7.8**	9.0	**68.7%**	**9.0**
Baxter-R to Franka	DomAd [1]	62.5	80.1	25.8	27.4	10.2%	26.2
	Cycle GAN [2]	49.9	62.6	19.8	14.5	17.2 %	23.9
	State Im [3]	79.3	92.3	29.3	37.1	24.3 %	25.1
	Ours	**21.0**	**16.6**	**9.4**	**12.7**	**65.8%**	**11.8**

Table 4.2: Evaluating Task-Strategy Transfer: Evaluating average rewards obtained by translating a source domain demonstration to the target domain via our approach, with and without finetuning, against a hierarchical RL baseline, adapted from Kulkarni et al. [9], across 3 tasks. Rewards for the HRL baseline, and our approach with finetuning are averaged across 10 episodes. Results on additional domain pairs are shared in supplementary material. * - While we evaluate rewards of target domain demonstrations here for comparison, our approach is intended for situations where such demonstrations are unavailable on the target domain.

Domain Pair	Approach:	Downstream Task					
		Reach (Source)	Reach (Target)	Push (Source)	Push (Target)	Slide (Source)	Slide (Target)
Franka to Sawyer	Demonstration	32.6	34.8	41.9	**42.1**	39.6	37.8
	HRL [1]	11.2	10.5	14.3	16.3	12.3	11.5
	Ours (Zero Shot)		33.7		39.5		39.7
	Ours (Fine-tune)		**35.1**		40.8		**43.1**
Baxter-L to Sawyer	Demonstration	36.4	34.8	45.7	42.1	43.4	37.8
	HRL [1]	9.7	10.5	13.3	16.3	13.6	11.5
	Ours (Zero Shot)		37.1		40.9		39.1
	Ours (Fine-tune)		**38.3**		**42.3**		**40.6**
Baxter-R to Franka	Demonstration	35.9	32.6	45.9	41.9	44.5	**39.6**
	HRL [1]	10.1	11.2	12.9	14.3	12.4	12.3
	Ours (Zero Shot)		36.1		42.0		37.4
	Ours (Fine-tune)		**37.4**		**43.6**		39.1

cally meaningful *coarse* skill correspondences between semantic clusters of skills that emerge in the original spaces, across all domain pairs. For example, from Figures 4.1 and 4.3, we see source domain placing skills correspond to target domain placing skills across all domain pairs, as do reaching, placing in other directions and pushing / sliding skills to the left or to the right, and returning skills to their rest configuration. However, our model is unable to guarantee learning perfect *finer* correspondences (such as between twisting or grasping skills), or between variations of skills that belong to the same semantic cluster in the original skill spaces (such as placing and reaching with different arm shapes, as in Figures 4.1 and 4.3). We emphasize that our method is unsupervised, and instead exploits similar contexts of skills across domains to learn correspondences; these results are expected as a result. Finer skills and intra-cluster variations of such skills are often executed in similar contexts, such as after reaching or before placing skills, making them difficult to disambiguate. Despite this, our model often does translate finer skills (such as twisting and grasping) correctly, suggesting its potential for improvement via techniques such as iterative refinement [68], or via additional inductive biases.

Using the learnt correspondences, we are able to transfer the source domain trajectory's overall structure, and sequence of skills executed, remarkably well to the target domain. Our model does so despite variations in the precise shape of the robot arms, or start and end positions of these respective skills. This provides further evidence that the learnt correspondences are indeed semantically meaningful, and additionally suggests these learnt correspondences would be useful in transferring *task strategies* across domains. We believe this is a powerful result, especially since our approach does so in a completely unsupervised manner.

Quantitative Evaluation of Skill-Tuple Distribution Matching: To quantitatively measure how well our approach can match the distributions of skill-tuples across robots, we evaluate the following unsupervised metrics. We first evaluate the forward and backward GMM densities $\mathcal{L}_f$ and $\mathcal{L}_b$. We also compute the *Chamfer distance* [100] between the skill-tuple spaces, which represents the nearest neighbor distances across two point sets. Together, these three metrics specify how close the skill-tuple distributions across domains are. We evaluate the following *supervised* metrics, using the manually annotated semantic labels associated with 50 skills in each domain. We reiterate these labels are unseen at train time, and used only for evaluation. We evaluate how accurately the learnt correspondences preserve semantics across domains, by measuring how well the semantic labels associated with a set of skills in the source domain match with those in the target domain, reported as the *label accuracy* in Table 4.1. We also evaluate a *supervised Z error* - i.e. the average distance in latent space between the translated source domain skills, and the nearest target domain skills with the same semantic label.

We compare our approach against the following alignment baselines in Table 4.1. Our first baseline is *Domain-adversarial training* [77], where we match skill-tuple distributions by training the translation model to fool a discriminator network trained to identify domains given skill-tuples. We compare against *Cycle-GAN* [75], where we train two translation models between skills from source and target domains to be cycle-consistent with one another, while also optimizing a domain-adversarial objective [77]. Finally, we also compare against a *State-based Imitation* approach [86], where we transfer trajectories across domains by copying end-effector states across robots, using inverse kinematics to retrieve the closest feasible joint state.

Analysis of Qualitative Results of Skill-Tuple Distribution Matching: From Table 4.1, we observe that our approach is notable able to learn correspondences that achieves a much higher semantic label transfer accuracy, consistent with our approach learning good *coarse* correspondences. The baseline approaches in contrast, are only able to achieve random-level label transfer accuracy, indicating their inability to learn correct correspondences. For example, the domain adversarial baseline ends up mapping several different types of skills in the target domain to single modes of skills in the target domain, across all domain pairs. The Cycle-GAN baseline somewhat mitigates this approach, but takes a shortcut in the learning and simply places a single source skill nearby every mode of target skill. In contrast, our approach explicitly optimizes for distribution matching, and achieves significantly better unsupervised distribution matching metrics than the implicit baseline approaches.

Further, the state-based imitation baseline relies on Inverse Kinematics (IK) on the target agent. Since the agents have differing workspaces from one another, IK often fails to recover feasible joint states. Even when successful, for redundant 7-DoF robots, there are multiple possible IK solutions to choose from, often leading to wildly different trajectories (and strategies) than those demonstrated. Bypassing these issues with an IK based approach require constructing involved engineering pipelines for each *pair* of robots or agents, which is infeasible. Not only is our approach able to bypass this agent-pair specific engineering effort, but is also able to outperform this IK based approach on all metrics presented in Table 1, 3, and 4.

4.4.2 Transferring task strategies across domains

As mentioned in the introduction, these learnt correspondences allow a robot to adopt a demonstrator's task strategy for itself, by translating a demonstrated task strategy (specified as a sequence of skills) from the source domain to the target robot. We evaluate how well the learnt correspondences help transfer task strategies across robots as follows.

Task Strategy Transfer Setup: We consider a set of tasks adapted from Mandlekar et al. [17], described in the appendix. We then select a demonstration for this task present in the source dataset, and encode this instance of a "task strategy" as a sequence of skills via its respective domain skill-encoder q_{S}. We then evaluate how well the *translation* of this task strategy performs on the same tasks in the *target domain*, both without fine-tuning (i.e., in a zero-shot manner), and after fine-tuning for 50 episodes. We evaluate this transfer against a hierarchical RL baseline, i.e. a high-level policy trained in the target domain over 500 episodes, and report our results in Table 4.2, as well on additional domain pairs in **??** in the supplement. While we report the performance of a demonstrated trajectory for the same task in the *target* domain for comparison purposes, we note that our approach is intended for the setting where such target domain demonstrations are unavailable or difficult to procure. We also provide visualizations of the rollouts from the original and translated strategies in https://sites.google.com/view/translatingrobotskills/home.

Analysing results on task strategy transfer: The demonstrations (Row 1 of Tables 2 and 5) in both domains successfully solve each task. The reward values of these demonstrations serve as benchmarks of when the task is solved. The Hierarchical RL baseline rarely solves tasks, and instead achieves moderate rewards from moving towards solving the task, such as reaching halfway to the object.

We observe that the task strategies translated by our method are able to achieve appreciable task performance across all domain pairs and tasks, solving these tasks even without fine-tuning these strategies. Without fine-tuning, we see that the translated task strategies follow a semantically reasonable sequence of skills given the target task, achieving an appreciable task reward, but often reach slightly differing goal states than desired. Given our translation model only observes skill encodings, and no additional state information, this is expected. Upon fine-tuning these translated task strategies in the target domain, we observe a consistent increase in task performance, since fine-tuning allows for picking slightly different (but semantically similar) skills that reach more appropriate goal locations, etc.

Despite these marginal differences, our translated task strategies are often able to achieve performance on par with demonstrations in the target domain, and outperform hierarchical RL on the target domain. Our approach is significantly faster to converge than hierarchical RL, needing 10 times less training episodes to achieve superior performance, because our approach bypasses the inefficient random exploration needed to learn appropriate skill sequences for a given task. These results suggest that our approach does indeed learn correspondences that facilitate transferring task strategies across robots, and could serve as a viable alternative to learning policies from scratch when target domain demonstrations are unavailable.

4.4.3 Additional Comparisons and Discussion

We provide several additional points of discussion and comparisons against the baselines in the appendix. Specifically, we discuss the relative training time of our approach versus the baseline approaches, assumptions of learning, validity of the premise, ablations, and limitations. We direct the reader to the appendix for these details.

4.5 Conclusion

We introduce a purely unsupervised approach to learn skill correspondences across differently embodied agents from different domains. Our approach learns semantically meaningful correspondences across multiple robot-robot and human-robot domain pairs, and helps transfer task-strategies across domains, without the need for any fine-tuning, despite being completely unsupervised. We believe that our approach could enable learning of correspondences across more general temporal abstractions, such as between skills and language instructions, or between skills and human video demonstrations in the wild. This is useful when the learner robot cannot currently solve the demonstrated task, or would benefit from learning a new strategy to do so. One practical scenario for this is robots could learn to solve new tasks by watching large collections of unlabelled datasets of humans performing tasks. Our work serves as a stepping stone to such future research. More broadly, our work takes small steps towards — (1) greater technical equity in robotics, by sharing large-scale training data across multiple different robots; and (2) democratizing robot programming, by enabling non-expert users specify task strategies to robots.

IV

Learning and Representing Effects of Skills

5 Learning Abstract Representations of Agent-Environment Interactions

5.1 Introduction

Consider an amateur cook learning a new dish by watching a chef demonstrate how to make a similar dish on YouTube; even amateurs can achieve excellent results doing so. Humans owe this aptitude for 'learning by imitation' to reasoning abstractly about human behaviours–including their own–and the task at hand. People ignore irrelevancies (ex. differences in kitchens), focus on patterns environmental change needed (ex. steps of recipes), and skill-sequences that effect these environmental changes (ex. techniques of chefs). In doing so, people abstractly represent task strategies; *i.e.* , the sequence of skills and patterns of environmental changes needed to accomplish the task. The prospect of equipping *robots* with these abilities is enticing. Such abstract representations of task strategies would allow us to view human and robot task strategies from a unified perspective. This in turn would enable robots to learn to solve various tasks by watching human demonstrators solve similar tasks in different environments from their own.

Despite making significant advances in building high-level understanding of agent behavior, work in this area has suffered from significant pitfalls. These works either learn abstractions over agent behaviors [12, 13, 22, 71, 72, 73], or over changes in environment state [101], but do not consider both together. While Sutton et al. [3], Sharma et al. [72] maintain notions of pre-conditions and effects, these notions are often hand-crafted into the abstractions, or are simplistic conditions on state. Further, approaches that *learn* temporal abstractions of behaviors are often unaware of the patterns of environmental and object state change they induce (i.e., their effects). Conversely, most environmental state abstractions are unaware of the behaviors that caused them. As a result, traditional approaches that use such abstractions to solve tasks typically need to perform a search for an appropriate sequence of abstractions to solve the task - a difficult problem that requires highly engineered heuristics to solve.

In this chapter, we argue that it is therefore important to understand interactions between agents (humans and robots alike) *and* their environments *together*. To do so, we build temporal abstractions of interactions. These interaction abstractions differ from their behavioral and environmental counterparts in that they maintain *explicit* notions of an agent and environment (or objects in it). By doing so, we hope to equip agents with a *high-level* understanding of the effects of their behaviors, and therefore an understanding of which of their behaviors are needed to affect desired environmental changes. By then *aligning* understandings of effects of human and robot behaviors, we hope to transfer strategies of solving tasks from human demonstrators to robots. We do so by first applying a robot skill learning framework [73] to learn temporal abstractions of environmental state. We then combine this with the original robot skills, to build a framework for interaction abstractions. We show promising results that these interaction abstractions would aid transfer between human and robot task strategies.

5.2 Introduction

In Part II and Part III of the thesis, we explored ways to first learn temporal abstractions of agent behaviour, and then translate such temporal abstractions of agent behaviour across humans and robots in an unsupervised manner. Common to all of these approaches is the notion of modelling agent behaviour alone. This is a limitation of these works; they neglect to model the environment state, and the effects of agent behaviour on the environment state. In particular, modeling the patterns of changes of environment state can be informative in translating behaviour from humans to robots - in some cases, modeling such patterns inform *how* the behaviour should be translated.

In this chapter, we argue that it is therefore important to understand interactions between agents (humans and robots alike) *and* their environments *together*. To do so, we build temporal abstractions of interactions. These interaction abstractions differ from their behavioral and environmental counterparts in that they maintain *explicit* notions of an agent and environment (or objects in it). By doing so, we hope to equip agents with a *high-level* understanding of the effects of their behaviors, and therefore an understanding of which of their behaviors are needed to affect desired environmental changes. By then *aligning* understandings of effects of human and robot behaviors, we hope to transfer strategies of solving tasks from human demonstrators to robots. We do so by first applying a robot skill learning framework [73] to learn temporal abstractions of environmental state. We then combine this with the original robot skills, to build a framework for interaction abstractions. We show promising results that these interaction abstractions would aid transfer between human and robot task strategies.

5.3 Related Work

The robot learning community has made efforts towards realizing various parts of this vision. Sivakumar et al. [102], Arunachalam et al. [103], Ye et al. [104], Qin et al. [105], Wu et al. [106] engineer mappings between human and robot state to facilitate imitation of human videos. Shankar et al. [107], Smith et al. [108] have sought to learn such correspondences between humans and robots without manual specification, resorting to representation alignment machinery such as Zhu et al. [75], or unsupervised domain adaptation machinery [70, 107]. While successful in their own right, these approaches seek to transfer individual states or actions across domains, and lack a higher level understanding of the behaviors at hand. The community has attempted to introduce such higher level understanding in the form of abstractions. Zhang et al. [93], Hansen-Estruch et al. [101], Gelada et al. [109], Li et al. [110] works on learning *state* abstractions that facilitate ignoring irrelevant components of environments, and making analogies across various environment and task instances [101]. Sutton et al. [3], Fox et al. [12], Krishnan et al. [13], Shankar et al. [22], Eysenbach et al. [71], Sharma et al. [72], Shankar and Gupta [73] learn *temporal* abstractions of agent behavior, that facilitate reasoning over longer term behaviors.

5.4 Approach

5.4.1 Background – Learning Behavioral Abstractions

We first describe an important building block of our work–a behavioral abstraction framework. We consider behavioral abstractions, or skills, are a representation of an agent acting consistently for a temporally extended period. Examples of such skills include a person stirring (a pot), or a person flipping an object such as a pancake, etc. In this paper, we specifically consider the behavioral abstraction framework of Shankar and Gupta [73]; though any such framework could be used [13, 14, 22, 28, 71]. Shankar and Gupta [73] learn behavioral abstractions, or skills, of agents from demonstrations in an unsupervised manner. Their method first represents robot skills as continuous latent variables z^r (subscript r depicts of the robot), and introduce a Temporal Variational Inference (TVI) to infer these skills or latent variables. Consider an agent trajectory $\tau^r = \{s^r_1, a^r_1, ...s^r_{n-1}, a^r_{n-1}, s^r_n\}$, where s^r_t is the state of the agent, a^r_t is the agent's action at time t, and n is the length of the trajectory. TVI trains a variational encoder $q^r(z|\tau^r)$ that takes as input a agent trajectory τ^r and outputs a sequence of skill encodings $z^r = \{z^r_1, z^r_2, ...z^r_k\}$, where $k < n$ is a learnt number of skills executed. TVI also trains a latent conditioned policy $\pi(a|s, z^r)$ that takes as input robot state s, and the chosen skill encoding z, and predicts the low-level action a that the agent should execute. TVI trains q^r and π to reconstruct the actions observed in the trajectory τ^r. We direct the reader to [73] for a thorough description of their skill

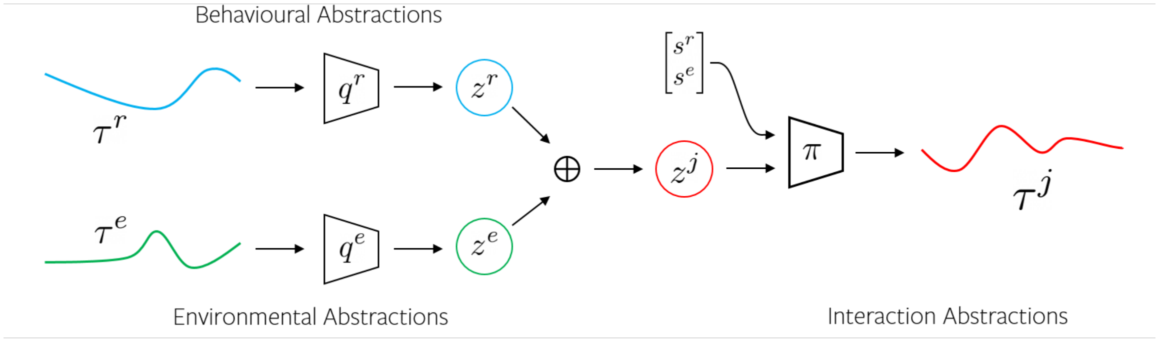


Figure 5.1: Architecture of one proposed approach. We learn a robot skill encoder q^r and an environmental abstraction encoder q^e, that learn representations of behavioral and environmental abstractions z^r and z^e respectively. We then combine z^r and z^e into a joint interaction abstraction z^j. We can then condition robot policy π on interaction abstraction z^j, in addition to state inputs s^r and s^e.

learning approach.

5.4.2 Building Temporal Abstractions over Environment State

In order to build temporal abstractions of interactions between agents and environment state, one also needs temporal abstractions of environment state. These abstractions exemplify patterns of motion that objects often exhibit, *e.g.* , a bottle cap rotating and moving up away from the bottle, or a kettle being tilted downwards to pour from it. One can imagine how such abstractions are useful – they could be used to specify steps that need to happen in a kitchen recipe, or more generally, describe changes in environmental state that need to occur. In this section, we describe how we adapt the behavioral abstractions of [73] to learn temporal abstractions of environment state. Consider a corresponding trajectory $\tau^e = \{s^e_1, a^e_1, ...s^e_{n-1}, a^e_{n-1}, s^e_n\}$ of *environment* state s^e over time. Here, a^e_t represents the change in environmental state at a given timestep t, rather than a notion of agent action. We can construct an equivalent *environmental* variational encoder $q^e(z|\tau^e)$, that predicts an equivalent sequence of latent encodings $z^e = \{z^e_1, z^e_2, ..., z^e_k\}$, that represent temporally abstract changes in environmental state. We can train an equivalent environment state “policy” π^e, conditioned on the desired environmental abstraction z^e. As in TVI, we train q^e and π^e to reconstruct the change in environmental state a^e_t at every timestep. We present preliminary results showing the efficacy of these environmental state abstractions below.

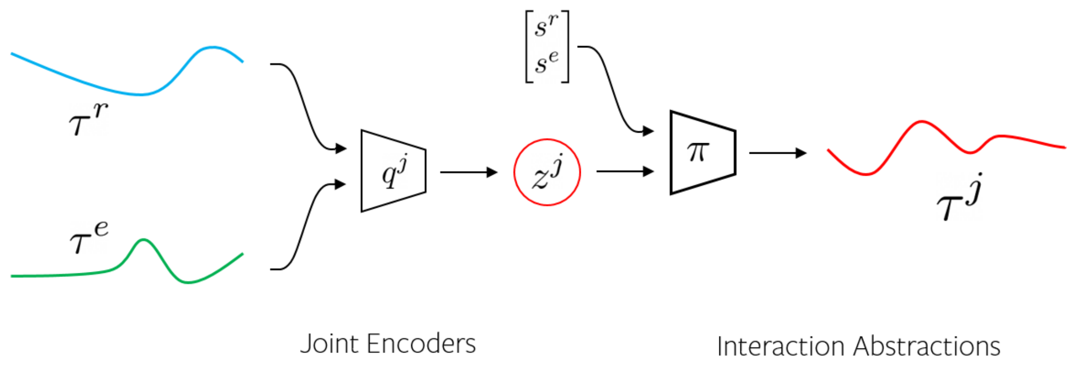


Figure 5.2: Architecture of second proposed approach. We learn a joint encoder for robot skills and environmental abstractions that learn joint representations of behavioral and environmental abstractions z^j. We can then condition robot policy π on interaction abstraction z^j, in addition to state inputs s^r and s^e.

5.4.3 Building Interaction Abstractions from Behavioral and Environmental Abstractions

In order to develop a high-level understanding of how agents interact with their environments, we would also like to build temporal abstractions of *interactions*. Examples of such abstractions would include using a cutting skill to chop an onion, or using a flipping skill to turn over an omelette. To reiterate, the proposed interaction abstractions differ from behavioral and environmental counterpart in that interaction abstractions maintain *explicit* notions of an agent and environment (or objects in it), *e.g.* stirring a pot, or flipping a pancake. In contrast, prior behavioral abstractions [13, 73] only maintain *implicit* notions of objects, *e.g.* , stirring or flipping skills. As a result, we view the interaction abstractions from an active perspective; *i.e.* , that the explicit agent effects changes on its environment. In contrast, we view the environmental abstractions above from a passive perspective, in that the state of objects in an environment changes, agnostic of what agent effects these changes.

Given behavioral abstractions z^r and environmental abstractions z^e, we combine these abstractions to form interaction abstractions z^j. We explore several options of combining these abstractions; the simplest is to simply concatenate them, *i.e.* , $z^j = \left[z^r z^e\right]$. This is depicted in Figure 5.1. We also explore the joint interaction abstraction encoder architecture depicted in Figure 5.2, where a single encoder predicts the interaction abstraction encoding provided with trajectories of both robot and environment. We propose to continue to explore further approaches to learning such interaction abstractions. These interaction abstractions thus contain information of both the agent skill being executed, and its (desired) effect on the environmental state. We can inform the policy of this desired agent skill and desired pattern of environmental change by conditioning the policy π on z^j rather than on z^r alone, *i.e.* , $\pi = \pi(a|s, z^j)$.

We now describe the pipeline of how we construct interaction abstractions z^j. We first sample agent and environmental state trajectories τ^r & τ^e from the appropriate dataset. We then pass these trajectories through their corresponding encoders q^r and q^e, to retrieve latent encodings of these abstractions, $\{z^r_1, z^r_2, ..., z^r_k\}$ & $\{z^e_1, z^e_2, ..., z^e_k\}$ respectively. We concatenate these encodings to form latent encodings of the interaction, $\{z^j_1, z^j_2, ..., z^j_k\}$. We then condition the agent policy π on this desired interaction abstraction $\{z^j\}^k_{t=1}$. During inference, given a pattern of desired interactions, $\{z^j_1, z^j_2, ..., z^j_k\}$, we can condition the agent policy on these desired interactions $\pi(a|s, z^j)$, and query it at the current state s^r, s^e, for the next agent action to execute a^r. We depict this pictorially in Figure 5.1. During training, we can update encoders q^r & q^e, and policy π, to optimize a reconstruction loss on the state changes observed in trajectories τ^r & τ^e. We specifically optimize the likelihood of observed state changes, as in TVI.

5.5 Experiments

We would like to answer the two following questions with respect to our proposed abstractions. Firstly, can the learnt environmental and interaction abstractions accurately model interactions between agents and their environments? Second, do the learnt abstractions spaces show promise in facilitating downstream task strategy transfer from humans to robots? In this section, we present preliminary results towards answering these questions.

5.5.1 Datasets and Experimental Setup

We present results on the Roboturk dataset [17] and the RoboMimic dataset [98], datasets of tele-operated demonstrations collected on the Sawyer robot and the Franka Panda respectively. In addition we also present results on the GRAB dataset [99], which consists of *humans* manipulating various objects themselves. These datasets consist of their respective agents interacting with a variety of different objects, such as milk cartons, cereal, bread, cans, differently shaped pegs, tools, *etc* Collectively, these datasets span a variety of interactions, including lifting, moving, releasing, pushing, rotating, reorienting, *etc* We present both qualitative and quantitative results for both the environmental and interaction abstractions.

In each case, we have access to the joint-state of the agents, 8 joint angles for the robots, and a 24 dimensional joint state for the humans. We also make use of the object state in these trajectories, consisting of 6 object pose. In each case we assume the “actions” of the agents are joint state velocities.

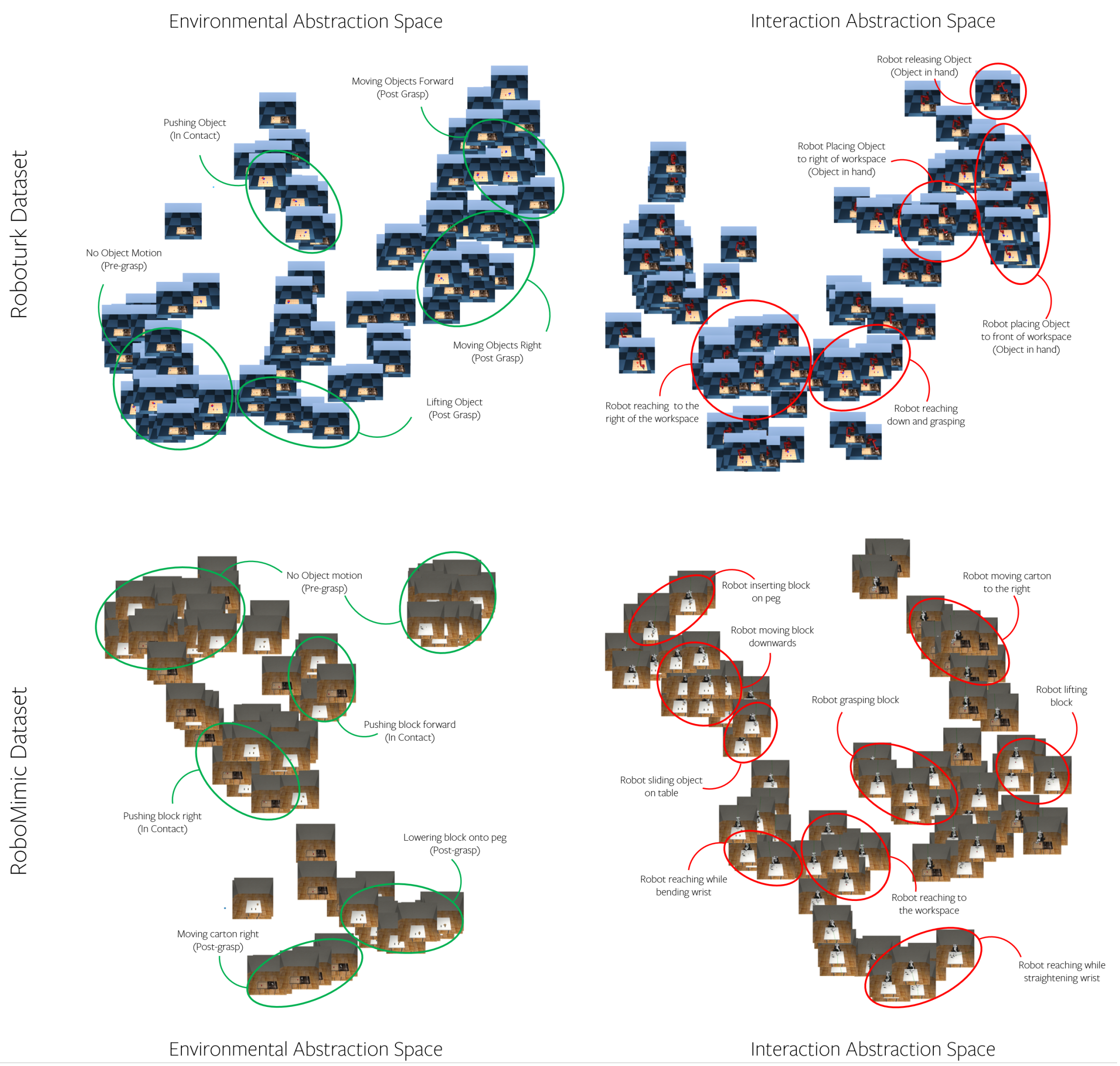


Figure 5.3: Depiction of preliminary results on both the environmental abstractions (left column), and the interaction abstractions (right column), across Roboturk dataset (top row) and Robomimic dataset (bottom row). These embedding spaces depict abstract representation spaces that model both agent behaviors and the their effects on the environment. In all four cases, note the clustering of similar motions into similar parts of their respective latent spaces. For dynamic versions of these spaces, view https://sites.google.com/view/interaction-abstractions.

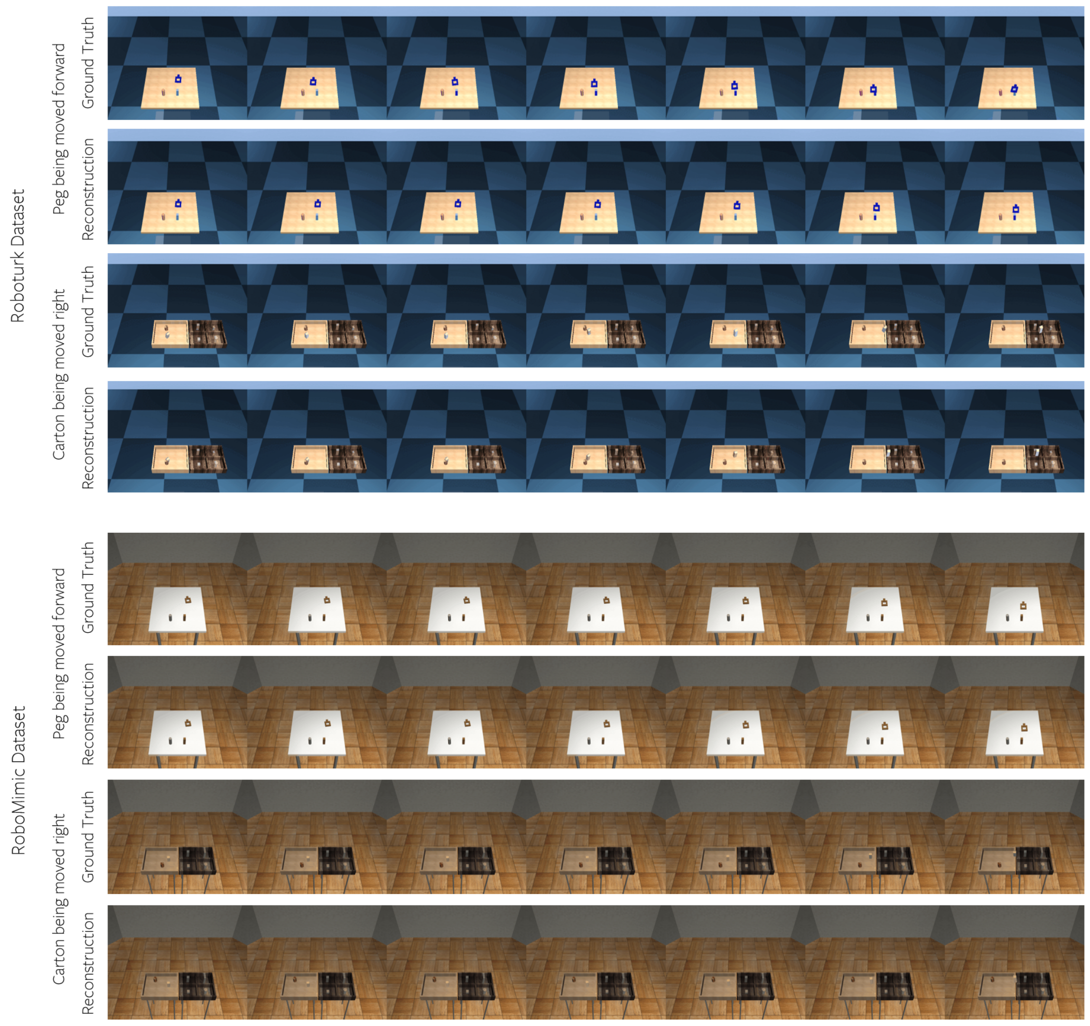


Figure 5.4: Qualitative results on environmental abstractions. We show two object state demonstrations for each dataset, (labelled ground truth), and their corresponding reconstructions via our learnt abstractions (labelled reconstructions). Note the accuracy of the reconstructions in capturing the object trajectory despite the low dimensional nature of the latent encoding.

5.5.2 Preliminary Qualitative Results

We first present qualitative results; in the form of visualizations of both individual abstractions and the representation spaces $\mathbb{Z}^e$ & $\mathbb{Z}^j$ of abstractions. We provide static versions of these visualizations in our main paper, and include dynamic visualizations on our project webpage, https://sites.google.com/view/interaction-abstractions.

Environmental Abstractions

We first present preliminary results on applying the behavioral abstraction framework of Shankar and Gupta [73] to learning environmental abstractions. To do so, we consider object state trajectories present in the various datasets, consisting of 6-D pose (position and orientation) of the object. We provide these object state trajectories as input to TVI [73]. TVI then provides us with a latent space $\mathbb{Z}^e$, that represents temporal abstractions of object state. Each z^e in this space $\mathbb{Z}^e$ represents a different pattern of environmental state. In this case, since we represent environment state as object state, each z^e may be thought of as a different pattern of object state.

We present a 2D visualization of this space on the left in Figure 6.3, produced via T-SNE [45]. Note the clustering of similar patterns of motion of objects into similar parts of the latent space. We manually annotate the various clusters of environmental abstractions present. Our environmental abstraction space is able to capture a variety of different object motions, including pushing, lifting, and moving objects.

In addition, we also visualize individual samples of abstractions present in the space $\mathbb{Z}^e$, in Figure 5.4 and https://sites.google.com/view/interaction-abstractions. We visualize ground truth trajectories (of a peg being moved forward, and a milk carton being moved to the right), and their respective reconstructions via their learnt abstraction encodings z^e. Notice the high fidelity reconstructions of the trajectories, and the similarity in trend of object motions in the ground truth and reconstructed trajectories. We note that while these results are to be expected from TVI [73], this validates the soundness of the learnt *environmental* state abstractions.

Interaction Abstractions

Having verified the soundness of environmental abstractions, we may now analyze how well they combine with prior behavioral abstractions (*i.e.* , robot skills) to form our proposed interaction abstractions. To do so, we feed in trajectories of agent and object state to their respective encoders q^r & q^e, retrieving their predicted latent encodings z^r & z^e respectively, concatenate these encodings to form a interaction abstraction encoding z^j, and finally reconstruct the joint agent-object trajectory τ^j from this encoding. As above, each z^j in the interaction abstraction space $\mathbb{Z}^j$ represents a different pattern of interaction between agent and object state.

As above, we present a 2D visualization of this latent interaction abstraction space

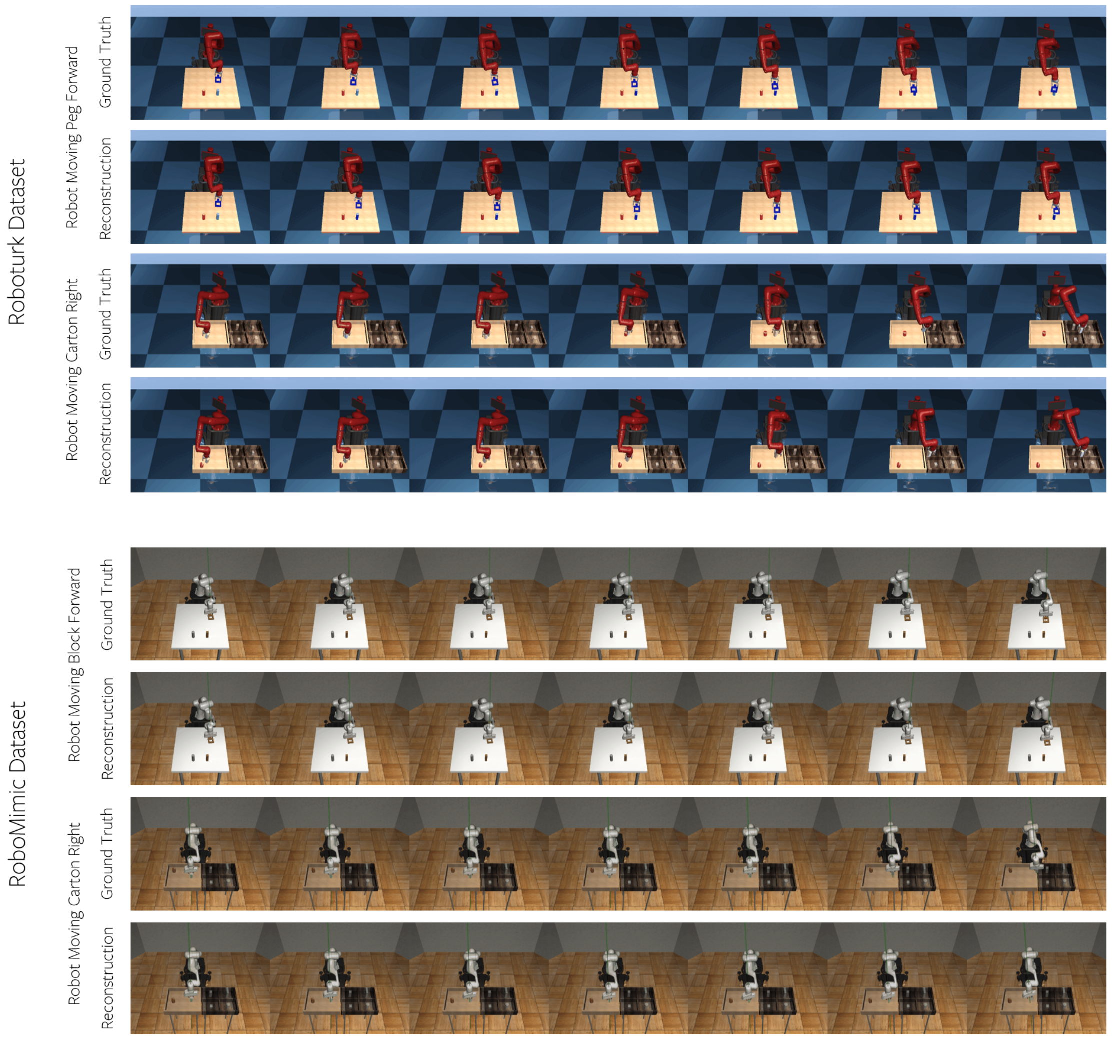


Figure 5.5: Qualitative results on interaction abstractions. We show two demonstrations for each dataset of the respective robot interacting with objects (labelled ground truth), and their corresponding reconstructions via our learnt abstractions (labelled reconstructions). We visualize the same tasks as in Figure 5.4, for reasons in Section 5.5.4. Note how the robot skills executed *lead* to the environmental abstractions depicted in Figure 5.4. The reconstructed space is able to capture this, and therefore the overall pattern of motions of agent and environment.

on the right of Figure 6.3, also produced by T-SNE. Dynamic visualizations of this are available at https://sites.google.com/view/interaction-abstractions. As in the case of environmental abstractions, the latent space is clustered based on the nature of interactions taking place. In each cluster, the robot and object undergo common

Table 5.1: Comparison of reconstruction error of environmental and interaction abstractions against baseline approaches for reconstructing environmental and joint agent environmental state respectively. Lower is better.

	Environmental State			Agent & Environment State			
Dataset	LSTM	VAE	EnvAbs (Ours)	LSTM	VAE	IntAbs Joint (Ours)	IntAbs Factored (Ours)
RoboTurk	1.40	1.83	**0.46**	1.78	1.93	0.72	**0.62**
RoboMimic	1.33	1.47	**0.38**	1.58	1.75	0.84	**0.58**
GRAB	1.16	1.38	**0.57**	2.30	2.24	0.93	**0.80**

patterns of change of state. There is a single cluster that models no interactions between the agent and the robot (*i.e.* , the robot is reaching towards the object and has not made contact with it yet). There are other clusters modeling interactions with the object in-hand, such as moving or lifting the objects. There are additionally clusters where the agent is making or breaking contact with the object, such as the robot releasing the object cluster in the top right. We emphasize that our method is able to capture such clustering of interactions despite being trained without any supervision over the types of interactions, robot skills, or object motions.

In addition to this, we also present visualizations of individual interactions present in the space $\mathbb{Z}^j$ in Figure 5.5, and in https://sites.google.com/view/interaction-abstractions. For ease of comparison, we visualize the same trajectories as visualized in Figure 5.4, *i.e.* the *robot* moving a peg forward, and the robot placing the milk carton to the right. Note that in contrast with the above environmental abstractions, where we refer to these object moiton patterns from a passive perspective, here we think of the interaction abstractions as active abstractions, where the robot effects desired change in its environment. We observe in Figure 5.5 that the interaction abstractions in each case capture *both* the robot skill (or behavioral abstraction) that is executed by the robot, *as well as* the effect it has on environmental state. Particularly, we observe the robot executing skills that result in the environmental state changes observed in Figure 5.4. Together with our other qualitative results, this suggests our proposed interaction abstractions are capable of abstractly modelling both agent behavior, and their effects on environmental state.

5.5.3 Preliminary Quantitative Results

In addition to the above qualitative results, we also present preliminary *quantitative* results to further verify the ability of our approach to abstractly model interactions. We do so by measuring the reconstruction error of both the environmental and interaction abstractions, and comparing this against other baseline approaches of predicting trajectories of environmental and joint agent-environment state respectively. We present these results in Table 6.1. Note that we do only compare approaches reconstructing environmental state amongst one another, and not against approaches reconstructing joint agent-environmental state, and vice versa.

We consider two variants of our interaction abstraction approaches. The first is a straightforward application of TVI [73] to reconstructing joint agent-environmental state, represented by joint latent encodings (rather than the factored encodings presented in Section 6.3.2). We term this approach as IntAbs-Joint. The second is the factored version of interaction abstractions present in Section 6.3.2, termed as IntAbs-Factored. We compare these approaches against two baselines; an auto-regressive LSTM trained to reconstruct trajectories, and a VAE approach that uses a *single* latent variable (across all timesteps) to represent the trajectory.

In the case of environmental state, we observe that our environmental abstractions are more accurate predictors of trajectories than the flat baselines, across all 3 datasets. This trend is mirrored in the case of reconstructing joint agent-environment state, where both interaction abstractions also perform better than the baselines. Further, we observe that our factored interaction abstractions are also able to outperform the joint interaction abstractions, suggesting that our choice of modelling the agent abstractions and environmental abstractions with separate latent variables is appropriate.

5.5.4 Suitability for Transfer

We now seek to verify whether these abstractions also exhibit characteristics suitable to transfer human to robot strategies.

Factored Encodings for Transfer

We first consider suitability for transfer along the architectural axis. By choosing to adopt a *factored* encoding of the agent and environmental abstractions Figure 5.1 rather than a joint encoding, our approach possesses a compositional architecture. Consider transferring from a human to robot task strategies. After training interaction abstractions for both human and robot agents $\mathbb{Z}^{j,human}$ & $\mathbb{Z}^{j,robot}$ respectively, we believe we can then transfer between these human and robot interaction abstractions, by swapping out the particular agent encoder q^r for that of another agent $q^{r'}$. While this also requires the alignment of the environmental abstractions associated with both agents, this is a fairly straightforward problem to solve, using machinery such as Cycle-GAN Zhu et al. [111] *etc* .

Unified Perspectives of Skills and Task Strategies

Our approach can represent similar interactions across different agents well, as exemplified by the similarity of representations of similar tasks across datasets in Figure 5.4 & Figure 5.5. By aligning representations of environmental abstractions across these agents (a relatively straightforward problem given machinery like Zhu et al. [75]), we could view such interactions from a unified perspective.

Consider two agents interacting in two domains D_1 & D_2. Given an alignment (or mapping) $f(z^e_{D_1}) \rightarrow z^e_{D_2}$ of environmental abstractions across domains D_1 and D_2, we may align the interaction abstractions $z^j_{D_1}$ & $z^j_{D_2}$ associated with $z^e_{D_1}$ & $z^e_{D_2}$ respectively using the same mapping; thus facilitating easy transfer of task strategies across these domains.

5.6 Conclusion

In this work, we present a framework for learning abstract representations of interactions between agents and their respective environments. We use these interaction abstractions to represent task strategies adopted by agents. These interaction abstractions could also model forward and inverse dynamics between agents' skills and their effects on the objects in their environments, and transfer task strategies from human demonstrators to robots. We believe this could enable robots to adopt the task strategies of human demonstrators, therefore broadening their repertoire of tasks.

V

Translating Environment-Aware Task Strategies

6 Translating Agent-Environment Interactions across Humans and Robots

6.1 Introduction

Consider an amateur cook learning a new dish by watching a chef demonstrate how to make a similar dish on YouTube; even amateurs can achieve excellent results doing so. This aptitude for "learning by imitation" is owed to the abstraction of human behaviours–including their own–and of the task at hand. People ignore irrelevancies (*e.g.* differences in kitchens), and focus on patterns in the environmental changes needed (*e.g.* steps of the cooking process), and skill-sequences that effect these environmental changes (*e.g.* techniques of chefs, such as chopping or stirring skills). For example, when an amateur cook pours liquid into a pan from a cup, they adeptly ensure the cup progressively tilts at an increasing angle as liquid falls into the pan, by bending their wrists above the pan. The precise angle of the bottle and wrist matter less than the *pattern* of motion that *both* the bottle and wrist undergo; indeed, humans performing similar pouring motions across a variety of bottles and liquids.

This exemplifies how well people abstractly represent task strategies. Such task strategies capture the *how* the task is performed in addition to *what* task is performed, *i.e.* they capture patterns of environmental change that occur during a task, and the sequence of skills they need to execute to accomplish the task at hand. The prospect of equipping *robots* with these abilities is enticing. Being able to transfer such abstract representations of task strategies would enable robots to consume human demonstrations, then execute their own corresponding skills that result in similar environmental changes, and even compose interactions they have only encountered individually to accomplish novel tasks.

In this chapter, we take a step towards realizing this vision. Our primary contribution is *TransAct*, a framework that first learns temporally abstract representations

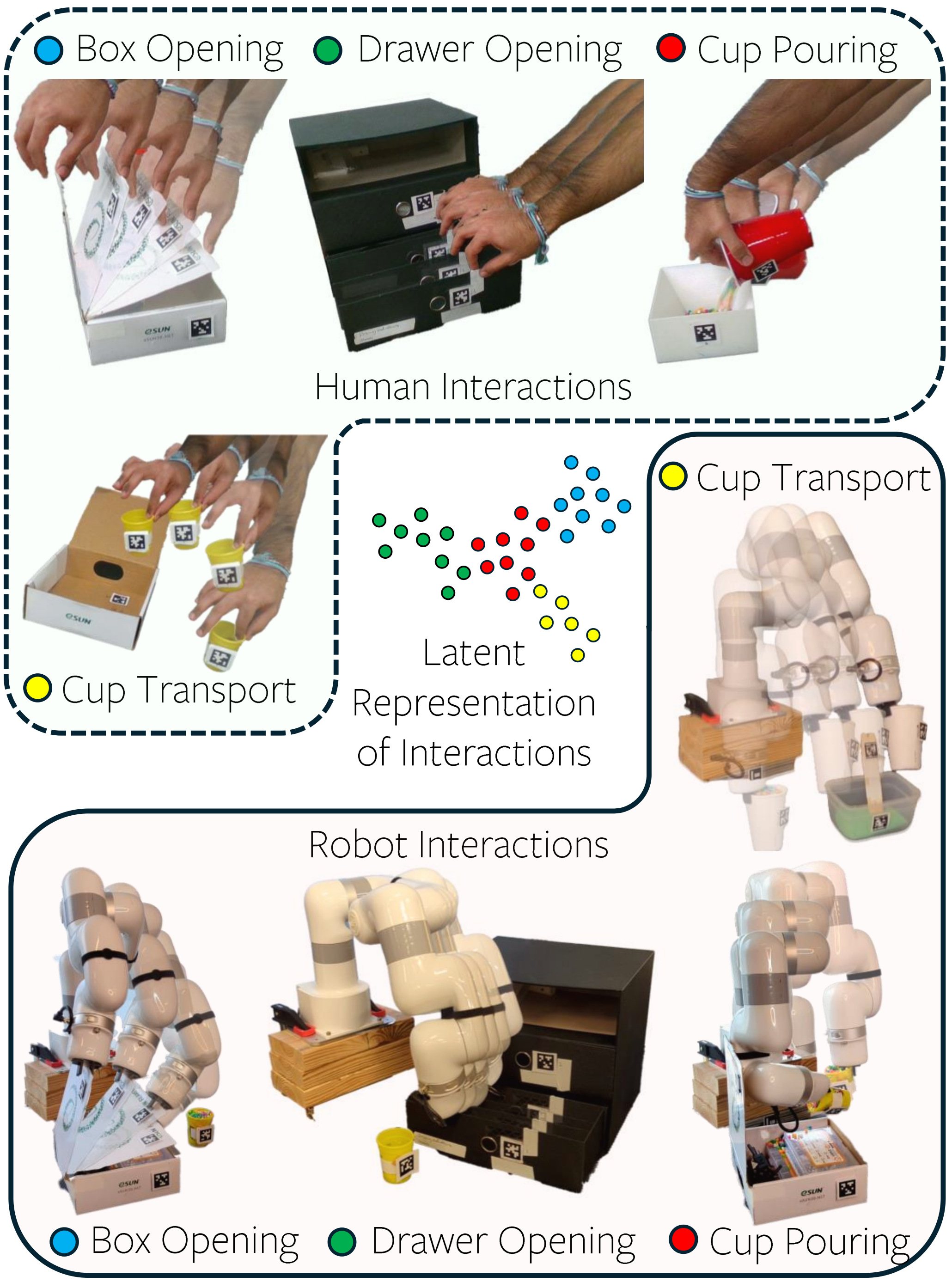


Figure 6.1: Overview of interaction abstractions learnt by *TransAct*. We depict actual samples of the learnt representation space, visualized with corresponding trajectories from both human and real robot demonstrations. We depict 4 different interactions here, opening a box, opening a drawer, pouring from a cup, and transporting a cup.

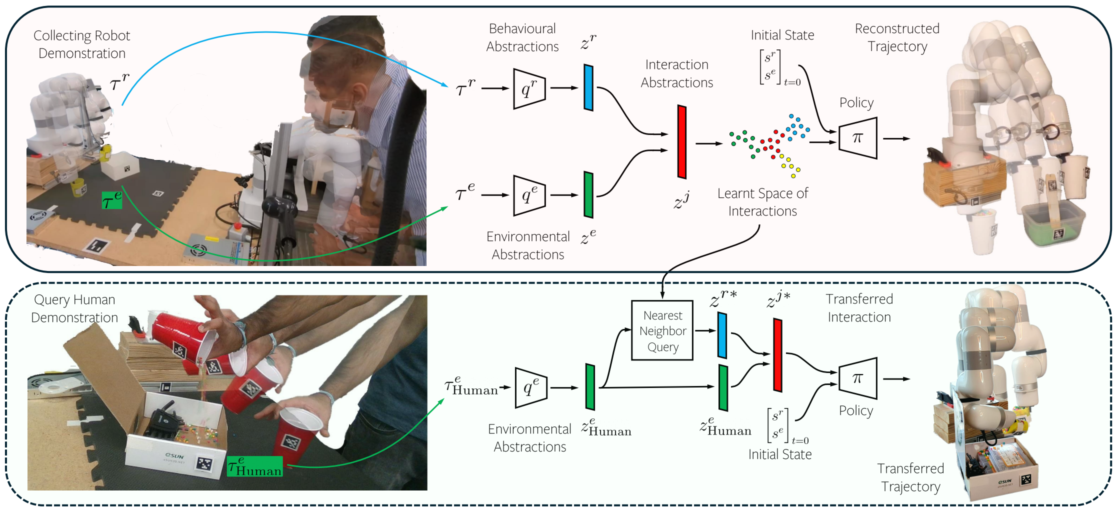


Figure 6.2: An overview of *TransAct*. (a) our interaction abstraction learning approach (top) and (b) zero shot transfer approach (bottom). (a) A demonstrated robot trajectory τ^r and environment trajectory τ^e are encoded into behavioral and environmental abstractions z^r and z^e respectively, via their respective encoders q^r and q^e. z^r and z^e are combined into a joint interaction abstraction z^j. Robot policy π is then conditioned on z^j, in addition to state inputs s^r and s^e. (b) To transfer a given human query demonstration τ^e to the robot, we encode its environment abstraction, then retrieve the nearest neighbor robot interaction observed in the training dataset, and rollout the robot policy to retrieve the transferred trajectory.

of agent-environment interactions, then translates such interactions from human demonstrators to robot learners. *TransAct* has three important facets that enable it to do so; first, *TransAct* builds on prior robot skill learning work [73], and learns temporal abstractions of *interactions*, rather than modelling lower-level states or actions. We hope to provide a higher-level understanding of agent behaviors *and* their environmental effects, and therefore an understanding of which of their behaviors are needed to affect desired environmental changes. Second, *TransAct* models agent *and* environment trajectories for the *entire* duration of any given interaction (rather than just the start and goal), making it suitable for tasks where the full object trajectory is important, *e.g.* pouring from a cup or stirring liquid in a cup. Third, *TransAct* facilitates transfer of these interactions across human demonstrators and robot learners. We consider in-domain transfer, *i.e.* from human demonstrations collected in the same environment as the robot learners. Despite being restrictive, such in-domain transfer is still valuable. By transferring human interactions to robot interactions that lead to similar environmental effects as their human demonstrators, *TransAct* enables our learner robot to compose interactions it has only encountered individually to accomplish novel tasks specified by the human demonstrator even without access to paired demos, semantic or temporal segmentation annotations.

Our subsequent contributions are as follows. Second, we introduce a new training setting with several auxiliary objectives that imbibe *TransAct* with these traits. Third, we collect real-world human and robotic interaction datasets to evaluate our approach on. Finally, we demonstrate that *TransAct* accurately represents various diverse agent-environment interactions across both our real-world human and robot datasets, and existing simulated robot datasets; *TransAct* also translates real-world human demonstrations to our real-world robot, enabling it to perform these of complex novel tasks by composing interactions with similar environmental effects. We present visualizations of our results at https://sites.google.com/view/interaction-abstractions.

6.2 Related Work

Human to Robot Imitation Learning

Many works have addressed the human to robot imitation learning problem. [102, 103, 104, 105, 106, 112, 113], engineer mappings between demonstrators and robot state to facilitate imitation of human demonstrators. [107, 108, 114] have sought to learn such correspondences between human demonstrators and robots without manual specification, resorting to representation alignment machinery such as [75], or unsupervised domain adaptation machinery [70, 107]. These works have achieved remarkable success, by transferring individual states or actions across domains. However, with the exception of [107], these works lack a higher level understanding of the behaviors at hand [112, 113]. We argue that such higher-level abstractions are more transferrable across domains, as noted in [107]. [107] itself only transfers agent motion across domains. In contrast, our work aims to transfer higher-level abstractions of interactions across humans and robots.

Spatial and Temporal Abstractions of Behavior

The community has attempted to introduce higher level understanding in the form of abstractions. [93, 101, 109, 110] learn *state* abstractions that facilitate ignoring irrelevant components of environments, and making analogies across various environment and task instances [101]. [3, 12, 13, 22, 71, 72, 73] learn *temporal* abstractions of agent behavior, that facilitate reasoning over longer-term behaviors.

Despite making significant advances in building a high-level understanding of agent behavior, these works have significant pitfalls. Works that learn abstraction over agent behaviors [12, 13, 22, 71, 73], are often unaware of the patterns of environmental and object state change they induce (i.e., their effects). Conversely, most environmental state abstractions are unaware of the behaviors that caused them [36, 74, 93, 101, 109, 110]. These approaches therefore typically need to perform a search for an appropriate sequence of abstractions to solve the task - a difficult problem that often requires highly engineered heuristics (to plan with), or rewards (to learn from) to solve. In

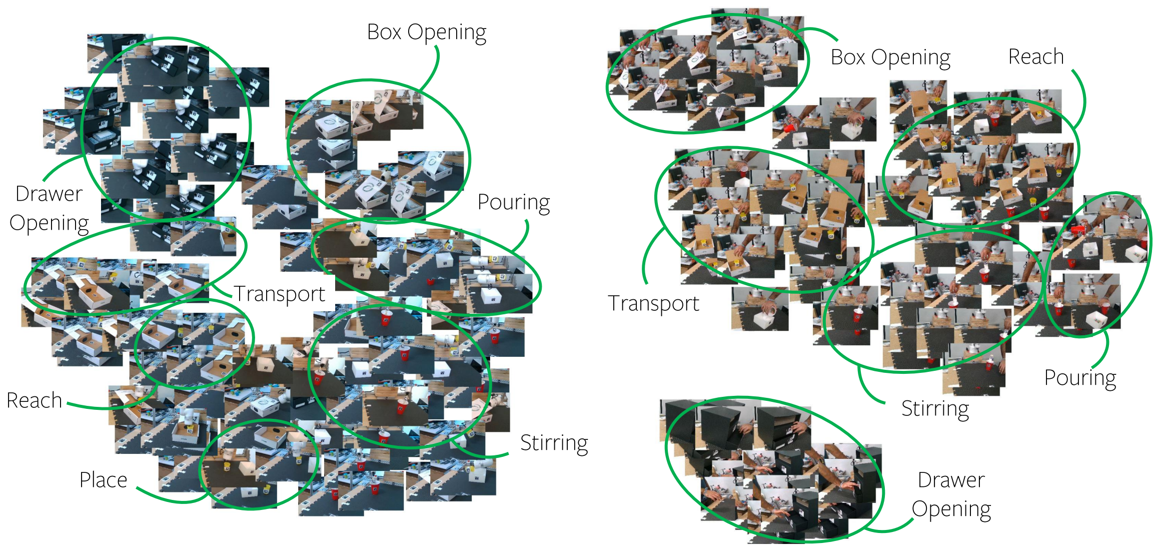


Figure 6.3: Depiction of the learnt latent representation space of interactions from the real world robot and human datasets. Note the variety of interactions captured in the space, the clustering of similar interactions from similar tasks into similar parts of each space, and the local continuity of each space. For dynamic version of these spaces (and similar spaces for the other datasets), view https://sites.google.com/view/interaction-abstractions.

contrast, our work attempts to model the interaction between agent and environment; thereby enabling efficient retrieval of behaviors to cause a desired environmental effect; thus facilitating easy transfer of interactions across humans and robots.

Dynamics Aware Skill Learning

[3, 72, 115, 116] maintain notions of pre-conditions and effects alongside agent skills. These works typically only model the initial and final goal environment state. In contrast, our approach models the environment trajectory for the entire duration of the interaction, making them unsuitable for tasks where the object textittrajectory is important *e.g.* pouring from a cup, opening a box or drawer, *etc* Further, [3, 72, 115, 116] use the agent's own experience in a reinforcement learning context, and are therefore unsuitable for facilitating an agent imitating a human demonstrator, compared to our approach, which is directly trained on demonstration data.

6.3 Approach

6.3.1 Preliminaries – Learning Behavioral Abstractions

We first describe an important building block of our work–a behavioral abstraction framework. A behavioral abstraction, or skill, is a representation of an agent acting consistently for a temporally extended period, *e.g.* a person stirring (a pot), or a person flipping an object such as a pancake, etc. We consider the behavioral abstraction framework of [73]; though any such framework could be used [13, 14, 22, 28, 71]. [73] learns behavioral abstractions, or skills, of agents from demonstrations in an unsupervised manner. Their method represents robot skills as continuous latent variables z^r (subscript r depicts the agent) and introduces a Temporal Variational Inference (TVI) to infer these skills or latent variables. Consider an agent state-action trajectory $\tau^r = \{s^r_1, a^r_1, ...s^r_{n-1}, a^r_{n-1}, s^r_n\}$, where s^r_t is agent state, a^r_t is the agent's action at time t, and n is trajectory length. TVI trains a variational encoder $q^r(z|\tau^r)$ that takes as input a agent trajectory τ^r and outputs a sequence of $k < n$ skill encodings $z^r = \{z^r_1, z^r_2, ...z^r_k\}$. TVI also trains a latent conditioned policy $\pi(a|s, z^r)$ that predicts the agent action a given the chosen skill encoding z, and. TVI optimizes q^r and π to maximize the likelihood of the *actions* observed in the trajectory τ^r. We direct the reader to [73] for a thorough description of their framework.

6.3.2 Learning Interaction Abstractions

Building Temporal Abstractions over Environment State

We can adapt the behavioral abstractions of [73] to learn equivalent temporal abstractions of environment state. Such abstractions could be used to specify patterns of object motion (more generally, changes in environmental state) that need to occur during a task, *e.g.* , a bottle cap rotating and moving up away from the bottle, or a kettle being tilted downwards to pour from it. Consider a corresponding trajectory $\tau^e = \{s^e_1, a^e_1, ...s^e_{n-1}, a^e_{n-1}, s^e_n\}$ of *environment* state s^e over time. a^e_t represents the change in environmental state at a given timestep t, rather than a notion of agent action. We construct an equivalent *environmental* variational encoder $q^e(z|\tau^e)$, that predicts an equivalent sequence of latent encodings $z^e = \{z^e_1, z^e_2, ..., z^e_k\}$, that represent temporally abstract changes in environmental state.

Combining Behavioral and Environmental Abstractions into Interaction Abstractions

We now learn temporal abstractions of *interactions*, to build a high-level understanding of how agents interact with their environments. Given agent and environmental state trajectories τ^r & τ^e from a dataset, we encode these trajectories into their corresponding abstractions $\{z^r_1, z^r_2, ..., z^r_k\}$ & $\{z^e_1, z^e_2, ..., z^e_k\}$ via their respective encoders q^r

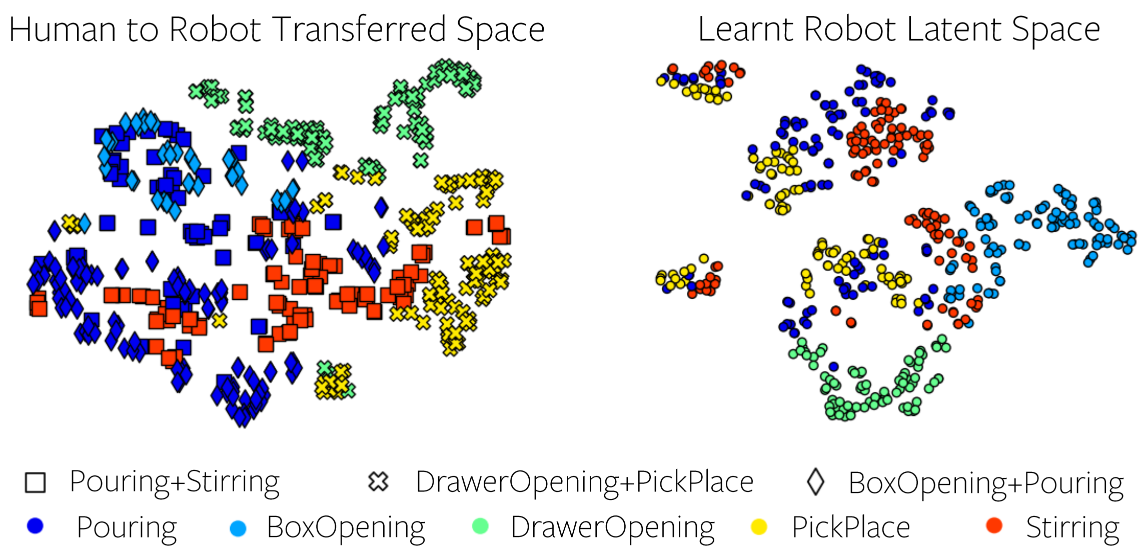


Figure 6.4: Depiction of our transferred embedding space from humans $\mathcal{D}_{\text{HComp}}$ to robots $\mathcal{D}_{\text{R}}$ (left) and the robot latent space from $\mathcal{D}_{\text{R}}$ (right). In both figures, each symbol $\diamond, \square, \times, \circ$ represents an interaction; symbols $\diamond, \square, \times$ on the left represent the compositional query task a human demonstration is taken from. For both figures, the color the symbol represents the task (from $\mathcal{D}_{\text{R}}$) this interaction is from.

and q^e. We then combine these behavioral and environmental abstractions by concatenating their encodings to form latent encodings of the interaction, $\{z_1^j, z_2^j, ..., z_k^j\}$, where *i.e.* $z_k^j = \left[z_k^r \; z_k^e\right]$. The interaction abstractions $\{z^j\}$ specifies the sequence of environment state changes necessary for the task to be solved, as well as the sequence of skills the agent needs to execute to effect these changes. We inform the agent policy of this desired skill and desired pattern of environmental change by conditioning the policy π on z^j rather than on z^r alone as in [73]. Here, we can retrieve the agent's action by querying $\pi = \pi(a|s, \{z^j\}_{t=1}^k)$. We depict this pictorially in Figure 6.2.

Such interaction abstractions (as opposed to behavioral abstractions alone) therefore provide a unified view of patterns of interactions across different agents in their environments. This in turn facilitates transfer across humans and robots, since we can retrieve skills that result in similar environmental changes across different agents.

6.3.3 Learning Representations of Interaction Abstractions

To learn representations of interactions z^j that facilitate downstream task transfer, there are 4 properties we desire of our representation, that TVI [73] does not afford.

Fidelity of underlying interactions

We would like the learnt representation to capture a given interaction with high fidelity, *i.e.* accurately reconstruct the trajectories of both the agent and the environment from the representation. TVI optimizes for reconstruction of the agent's actions, but suffers when agent and environment *states* deviate from the demonstrations, due to the issue of compounding errors.

To mitigate this, we introduce an additional cumulative state reconstruction loss $\mathcal{L}_{\text{state}}$, encouraging the model to recover to the true states, from previous (potentially erroneous) state predictions made by the model itself. Given the model's predicted actions $\hat{a_t}$, we can integrate these actions over time to retrieve state predictions: $\hat{s_t} = s_{t=1} + \sum_{t'=1}^{t} \hat{a_{t'}}$. We then minimize the distance between the integrated state predictions $\hat{s_t}$ and the observed states s_t, for all timesteps t, *i.e.* $\mathcal{L}_{\text{state}} = \sum_{t=1}^{T} \|\hat{s_t} - s_t\|_2^2$.

Architectural transfer from source to target

We facilitate transferring interactions from humans to robots via our architecture; by adopting a factored representation $z^j = \begin{bmatrix} z^r & z^e \end{bmatrix}$, where abstraction encoders q^e and q^r are independent of one another. Consider encoders q^r and q^e trained on a robot dataset $\mathcal{D}_{\text{R}}$. We reuse the learnt environment abstraction encoder q^e on arbitrarily complex query demonstrations from any other agent collected within the same environment in a zero-shot fashion, since the underlying environment trajectories themselves are from similar distributions.

Robustness and Smoothness of Representation

We facilitate easy retrieval and transfer of interactions across agents, by learning a locally continuous representation, *i.e.* one that is robust to small perturbations in the input trajectories, such as small differences in environment trajectories across agents. We encourage this property by considering the Jacobian of the encoder $J_q = \frac{\partial q(z|\tau)}{\partial \tau}$. Past work has shown the benefit of regularizing the Jacobian of such encoder networks on the robustness of their representations [117] and local continuity in the representation [118]. We follow [117], and construct a regularization loss $\mathcal{L}_{\text{jac}}$ to regularize the Frobenius norm of the Jacobian, *i.e.* $\mathcal{L}_{\text{jac}} = \|J_q\|_F^2$.

Contrastive Representations of Interactions

Intuitively, interactions from the same tasks, or interactions that result in similar object motions (or lack thereof) are easily retrieved if encoded into similar parts of the latent space, and further away from interactions from different tasks or inducing different object motions. We implement this by employing contrastive style losses [119], [120], [121] to place similar interactions close together *i.e.* being *contractive* [118] w.r.t. similar interactions, and distinct skills farther apart, *i.e.* being *discriminative* of distinct skills.

Consider a pair of interaction trajectory segments τ_m, τ_n. We introduce two contrastive losses between the representations of these interactions, z_m, z_n. The first loss, $\mathcal{L}^{\text{task}}_{\text{cont}}$, is based on whether these trajectories belong to the same task or not. For this loss, we define positive pairs via a membership function $\Delta^{\text{task}}_{m,n} = 1$ when τ_m, τ_n belong to the same task; negative pairs simply have $\Delta^{\text{task}}_{m,n} = 0$.

The second loss, $\mathcal{L}^{\text{env}}_{\text{cont}}$, is based on whether their environmental trajectories τ^e_m, τ^e_n are similar to each other or not. We use the the distance between representations of environment trajectories $\|z^e_m - z^e_n\|^2_2$ as a proxy of the distance between these trajectories; this choice is made reasonable because the Jacobian regularizer $\mathcal{L}_{\text{jac}}$ encourages local continuity in the representation. In principle, distances such as dynamic time warping distance between trajectories (as in [22]) could also be used. For this environment-trajectory based contrastive loss $\mathcal{L}^{\text{env}}_{\text{cont}}$, we define positive pairs via a membership function $\Delta^{\text{env}}_{m,n} = 1$ when the distance $\|z^e_m - z^e_n\|^2_2$ is below a threshold δ. Negative pairs have $\Delta^{\text{env}}_{m,n} = 0$.

For both $\mathcal{L}^{\text{task}}_{m,n}$ and $\mathcal{L}^{\text{env}}_{m,n}$, for a positive pair of interactions (*i.e.* belonging to the same task, or having similar environmental trajectories), we pull their representations z_m, z_n closer together to within a threshold ϵ, by minimizing the positive loss $\mathcal{L}^{+}_{m,n} = \max(\epsilon, \|z_m - z_n\|^2_2)$. Similarly, for a negative pair of interactions, (*i.e.* from different tasks, or having different environmental trajectories), we push their representations apart until they are at least ϵ apart, by minimizing the negative loss $\mathcal{L}^{-}_{m,n} = \max(0, \epsilon - \|z_m - z_n\|^2_2)$. For both $\mathcal{L}^{\text{task}}_{m,n}$ and $\mathcal{L}^{\text{env}}_{m,n}$, we compute the contrastive loss for every pair of trajectories in a batch $\mathcal{B}$,

$$\mathcal{L}^{\left\{\text{env,task}\right\}}_{\text{cont}} = \sum_{m,n \in \mathcal{B}} \left\{\Delta_{m,n}\mathcal{L}^{+}_{m,n} + (1 - \Delta_{m,n})\mathcal{L}^{-}_{m,n}\right\} \tag{6.1}$$

where $\Delta_{m,n}$ is the appropriate membership function from $\Delta^{\text{task}}_{m,n}, \Delta^{\text{env}}_{m,n}$. We can combine these losses into a single contrastive loss: $\mathcal{L}_{\text{cont}} = \mathcal{L}^{\text{env}}_{\text{cont}} + \mathcal{L}^{\text{task}}_{\text{cont}}$.

Final Objective

Our full objective supplements the TVI objective with our proposed auxiliary objectives: $\mathcal{L}_{\text{LIA}} = \mathcal{L}_{\text{TVI}} + \lambda_{\text{state}}.\mathcal{L}_{\text{state}} + \lambda_{\text{jac}}.\mathcal{L}_{\text{jac}} + \lambda_{\text{cont}}.\mathcal{L}_{\text{cont}}$, where λ's indicates the weights of our auxiliary losses.

6.3.4 Transferring Interactions from Humans to Robots

Given a human query demonstration τ_{Human} collected in the same environment as $\mathcal{D}_{\text{R}}$, we would like to transfer the interactions present in this demonstration to a robot, such that the environmental effects of the translated interactions are similar to those of the demonstration itself.

Table 6.1: State Reconstruction Error. Lower is better. * represents state errors computed over environment state alone. The full *TransAct* approach outperforms baseline approaches on reconstructing interactions across all datasets. *TransAct* also facilitates accurate reconstruction of interactions transferred from humans to robots comparable to approaches trained on the target domain data themselves.

Dataset	Baseline Approaches			*TransAct* + Ablations				
	VAE [122]	H-DMP [23]	TVI [73]	*TransAct* (Ours) Full	*TransAct* (Ours) No $\mathcal{L}_{\text{state}}$	*TransAct* (Ours) No $\mathcal{L}_{\text{jac}}$	*TransAct* (Ours) No $\mathcal{L}_{\text{cont}}$	*TransAct* (Ours) Un-Factored
Real Robot Data: $\mathcal{D}_{\text{R}}$ (Ours)	0.94	1.12	0.16	0.07	0.19	0.06	**0.04**	0.12
Human Data: $\mathcal{D}_{\text{H}}$ (Ours)	1.91	3.64	0.76	0.36	0.59	0.31	**0.28**	0.46
Human Data: $\mathcal{D}_{\text{HComp}}$ (Ours)	2.65	4.85	1.38	0.81	1.26	0.78	**0.57**	0.99
Human Data: $\mathcal{D}_{\text{H}}$ (Ours) *	0.83	1.74	0.26	0.12	0.34	0.11	**0.09**	0.17
H2R: $\mathcal{D}_{\text{H}} \rightarrow \mathcal{D}_{\text{R}}$ (Ours) *	3.13	3.75	2.98	**0.28**	0.59	1.04	1.19	2.83
Human Data: $\mathcal{D}_{\text{HComp}}$ (Ours) *	1.32	2.84	0.57	0.26	0.49	0.24	**0.21**	0.34
H2R: $\mathcal{D}_{\text{HComp}} \rightarrow \mathcal{D}_{\text{R}}$ (Ours) *	3.94	4.15	2.78	**0.40**	0.92	1.66	1.79	3.58
RoboTurk [17]	1.93	3.48	0.72	0.36	0.57	0.35	**0.31**	0.39
RoboMimic [98]	1.75	2.09	0.84	0.40	0.52	0.38	**0.32**	0.49
FrankaKitchen [60]	0.84	1.85	0.56	**0.22**	0.48	0.37	0.27	0.26
DAPG [123]	2.46	3.85	0.97	**0.46**	0.78	0.49	0.47	0.59
DexMV [124]	2.61	3.58	1.05	0.51	0.63	0.49	**0.46**	0.57

We operationalize this by first encoding the various interactions observed in the robot dataset $\mathcal{D}_{\mathrm{R}}$ into a pre-computed set of N interactions $\mathbb{Z} = \{z_1^j, z_2^j, ..., z_N^j\}$, using the encoders q^r and q^e trained on $\mathcal{D}_{\mathrm{R}}$. We then encode the human query trajectory τ_{Human} into a sequence of query environment abstractions $\{z_k^e\}_{k=1}^K$, via the same q^e as above. We then retrieve the nearest neighbors of $\{z_k^e\}_{k=1}^K$, from the environmental component of the pre-computed latent space $\mathbb{Z}$, $\{z_k^{e*}\}_{k=1}^K$, and the corresponding *robot* behavioral abstractions $\{z_k^{r*}\}_{k=1}^K$ that would likely cause such environmental change. We then rollout our robot policy π conditioned on $\{z_k^{r*}, z_k^e\}_{k=1}^K$, to retrieve a robot trajectory that has the same environmental effect as the original query trajectory τ_{Human}. We depict this process pictorially in the bottom half of Figure 6.2.

6.4 Datasets and Experimental Setup

We use several demonstration datasets to evaluate our approach on. Together, these datasets span human and robotic domains, real world and simulation, a variety of different objects (*e.g.* boxes, cups, drawers, differently shaped pegs, etc.), and interactions (*e.g.* including lifting, moving, releasing, pushing, rotating etc.).

Real-World Setup and Datasets

We collect a two small real-world datasets. The first is a robotic dataset $\mathcal{D}_{\mathrm{R}}$ of a X-ARM Lite6 robot performing 5 different tasks, *Pouring from a cup*, *Stirring a cup*, *Box Opening*, *Drawer Opening*, and *Pick and Place*. We collect a parallel dataset, $\mathcal{D}_{\mathrm{H}}$ of a human performing the same 5 individual tasks. We collect another human dataset, $\mathcal{D}_{\mathrm{HComp}}$, performing 3 additional tasks composed from the individual tasks in $\mathcal{D}_{\mathrm{H}}$. These tasks are *BoxOpening+Pouring*, *i.e.* opening a box, then pouring beads from a cup into the box, *DrawerOpening+PickPlace*, *i.e.* opening a drawer, then placing a cup into the open drawer, and finally *Pouring+Stirring*, *i.e.* pouring beads into a cup, then stirring the poured beads.

We collect 10 robot and human demonstrations for each task, and 6 human demonstrations for each composed task in $\mathcal{D}_{\mathrm{HComp}}$. We collect data on a "puppet" robot, that is tele-operated by kinesthetically controlling a "master" bot as in Figure 6.2. We record the 6 DoF joint state, and gripper state as the robot state, and treat joint-velocities and gripper opening and closing as the robot action. For each human demonstration, we employ the hand tracking from [125] to record a 25 dimensional joint state of the human hand (consisting of the base of the palm, index, thumb, and middle fingers). For the human and robot datasets, we place Apriltags [126] [127] [128] on each of the 2 objects involved a task for object-state estimation, and collect 6-D pose (position and orientation) of each object as the object state (also pictured in Figure 6.2).

Simulated Datasets

We also present results on the following publicly available simulated robot datasets — Roboturk [17] (Sawyer robot), RoboMimic [98] and FrankaKitchen [129], (Franka Panda), and DAPG [123] and DexMV [124] (Adroit hand). For each dataset, we use the robot joint states and gripper state as robot state, joint-velocities as robot actions, and the 6-D object pose of the primary object in the respective task as the object state. Note that we assume a fixed number of objects in each dataset, and concatenate object poses for each object as input to our model.

6.5 Experimental Results

We evaluate the ability of our proposed abstractions to *learn* and *transfer* interactions across humans and robots, and design a set of experiments to answer two questions. Firstly, how well can the learnt interaction abstractions accurately model interactions between agents and their environments? Secondly, how well do the learnt abstractions facilitate transferring task-strategies from humans to robots?

6.5.1 Modelling Interactions

Reconstructing Sequences of Interactions

We first evaluate how accurately our model can reconstruct sequences of agent-environment interactions. Not only does this implicitly require modelling individual interactions well, but can involve reconstructing a sequence of $15-20$ agent-environment interactions in some tasks.

Quantitative Reconstruction Measures We encode and reconstruct each trajectory in each dataset, and compute the average mean-squared reconstruction error of both the agent and environment state, presented in Table 6.1. We compare our approach against several baselines based on a non-hierarchical VAE [122], a hierarchical primitive based approach that learns DMPs for each interaction segment (as opposed to our learnt representation) [23], and TVI [73], the skill learning approach [73]. As presented in Table 6.1, our approach achieves significantly lower reconstruction errors than the other baseline approaches, despite needing to satisfy additional constraints imposed by the auxiliary losses.

Visualizing Interaction Reconstructions We visualize reconstructions of individual interactions in Figure 6.1 for the human and robot datasets. As seen in Figure 6.1, the real robot is able to accurately reconstruct various interesting interactions observed in the demonstrations, such as pulling open a drawer, pushing the lid of a box open,

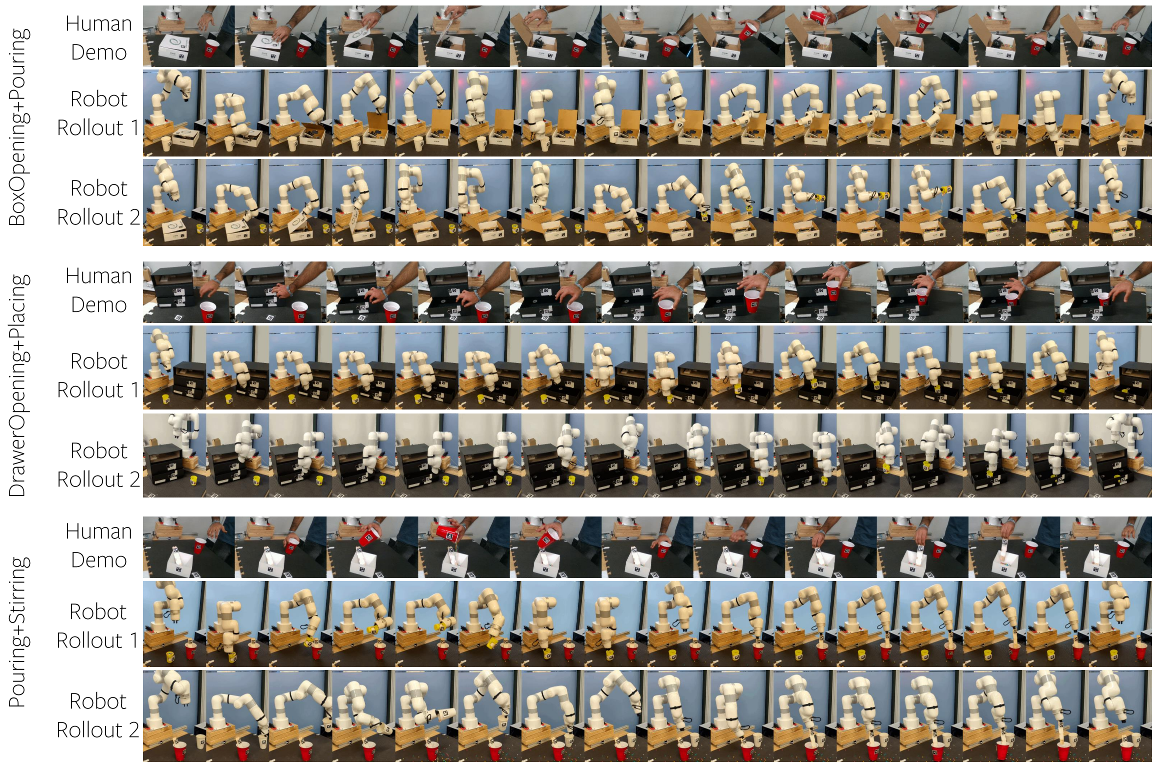


Figure 6.5: Depiction of human query demonstrations for 3 different compositional tasks, as well as 2 transferred trajectories each executed on the robot. At the top, our approach translates the human sequence of reaching and opening a box to a sufficient angle to pour into, releasing the box, grasping and transporting a cup above the box, pouring beads from the cup into the box, placing the cup back, and returning. In the middle, we observe it is able to translate reaching to and opening the drawer, reaching and grasping the cup, transporting and placing the cup before finally returning. In the bottom, it similarly reaches for a cup, transports it above another container, pours beads, and then subsequently grasps a stirrer and stirs the container. Our approach is capable of doing this despite significant variation in the relative configuration of the objects.

and tilting an object to pour from it. We defer visualizing entire trajectories to our website and supplemental video due to space constraints.

Visualizing Space of Interactions

We also present 2D visualizations of the *space* of interactions for the real world robot and human datasets in Figure 6.3, produced by feeding a set of interactions $\{z_k^j\}_{k=1}^N$ to T-SNE [45]. We provide higher resolution dynamic visualizations of these spaces and other datasets on our website https://sites.google.com/view/interaction-abstractions. Qualitatively, we note the clustering of similar patterns of motion of objects from the same tasks into similar parts of the latent space, manually annotated for clarity. This clustering is afforded by our contrastive losses and jacobian regularizer. Our interaction abstraction space is able to capture a variety of different interactions, including picking objects, tilting an object (for pouring), pushing a box lid open, *etc*

6.5.2 Facilitating Human to Robot Task Transfer

We now assess our framework's capability in transferring interactions between humans and robots.

Analysis of Human-to-Robot Transfer Reconstruction

We transfer query demonstrations from our human datasets $\mathcal{D}_{\mathrm{H}}$ and $\mathcal{D}_{\mathrm{HComp}}$ via the approach described in Section 6.3.4. We reiterate that the model is trained only on $\mathcal{D}_{\mathrm{R}}$, but evaluated on $\mathcal{D}_{\mathrm{H}}$ and $\mathcal{D}_{\mathrm{HComp}}$. Retrieving query trajectories from $\mathcal{D}_{\mathrm{HComp}}$ having trained our encoders on $\mathcal{D}_{\mathrm{R}}$ evaluates how well the model can compose seen interactions into novel and more complex sequences previously unseen by the robot.

We compute the reconstruction error between the *environmental* state of the rollouts and the query demonstrations (the agent states are incomparable in general); these results are presented in Table 6.1 in rows H2R $\mathcal{D}_{\mathrm{R}} \rightarrow \mathcal{D}_{\mathrm{H}}$ and H2R $\mathcal{D}_{\mathrm{R}} \rightarrow \mathcal{D}_{\mathrm{HComp}}$, with the reconstruction error of a model trained on the human datasets $\mathcal{D}_{\mathrm{H}}$ and $\mathcal{D}_{\mathrm{HComp}}$ as baselines in rows $\mathcal{D}_{\mathrm{H}}$ * and $\mathcal{D}_{\mathrm{HComp}}$ *. In the transfer rows, H2R $\mathcal{D}_{\mathrm{R}} \rightarrow \mathcal{D}_{\mathrm{H}}$ and H2R $\mathcal{D}_{\mathrm{R}} \rightarrow \mathcal{D}_{\mathrm{HComp}}$, the transferred robot trajectories are able to achieve environment state reconstruction errors in similar ranges to those from the approach trained on $\mathcal{D}_{\mathrm{H}}$ and $\mathcal{D}_{\mathrm{HComp}}$ respectively. The baseline approaches and ablations of our approach without $\mathcal{L}_{\mathrm{cont}}$ and $\mathcal{L}_{\mathrm{jac}}$ all learn latent spaces where nearest neighbor retrieval from human to robot trajectories results in unrelated and inaccurate trajectories, leading to poor transfer reconstruction.

We visualize these transferred trajectories in Figure 6.5. Our approach composes sequences of robot-environment interactions to complete these novel compositional tasks specified by their human demonstrations. Note the ability of our model to follow the high level sequence of skills observed in the demonstrations. This is particularly notable despite our approach having access to only task-level labels, but not temporal segmentation labels, skill-level semantic labels, or other forms of supervision. Our approach is able to do this despite variation in the relative configuration of the objects from the human queries - indicating our approach goes beyond stitching together demonstrated trajectories. Our approach learns a spatially and temporally abstract representation of these interactions that offers a unified perspective of interactions across domains.

Analysis of Transferred Latent Spaces

We also present a visualization of the transferred latent space in Figure 6.4, akin to the visualization of the latent space in Figure 6.3. The transferred space shows that our approach can compose individual interactions to transfer human query demonstrations that consist of novel interaction sequences. The transferred space is also clustered similarly to the original robot latent space, despite being queried in a zero-shot fashion on *human* trajectories, further exemplifying the viability of our transfer approach. These results together indicate the ability of our approach to transfer human interactions to robot interactions that have a similar effect on their environments.

6.5.3 Ablation of various loss components

To determine the efficacy of our proposed auxiliary losses, we present an ablation study in the right half of Table 6.1, where our approach is trained without each one of the auxillary losses in different columns. As expected, removing the state-reconstruction loss $\mathcal{L}_{\text{state}}$ hurts the reconstruction performance for all datasets, as well as the human to robot transfer. Interestingly, removing the Jacobian and contrastive losses $\mathcal{L}_{\text{jac}}$ and $\mathcal{L}_{\text{cont}}$ *improve* state reconstruction on most individual datasets, but drastically hurts the downstream transfer performance. Here, the model is able to place more emphasis on reconstructing trajectories accurately, but fails to create a latent space conducive to transfer. Using a non-factored encoder model achieves similar reconstruction performance as our factored approach, but fails to aid transfer as ours does; this may be attributed to nearest neighbor retrieval of joint agent-environment abstractions z^j being less meaningful for different agents than retrieving pure environment abstractions.

6.6 Limitations

In its current form, *TransAct* requires human demonstrations be collected in the same environment as the robot demonstrations. Relaxing these conditions would improve the applicability of our approach to human demonstrations in the wild, such as YouTube videos *etc* We believe this could be achieved by constructing additional objectives to ground environmental states between human and robot domains, as in

[75], [93]. *TransAct* is also currently unable to detect and recover from task failures from the stochastic nature of contact; this may be remedied by running a *closed* loop robot policy conditioned on learnt or transferred interactions.

6.7 Conclusions and Future Work

Our primary contribution is our *TransAct* framework, which first learns abstract representations of agent-environment interactions, then translates such interactions from human demonstrators to robot learners. Despite the limitations listed above, *TransAct* serves as a valuable tool to enable learner robots to consume human task demonstrations, and compose interactions with similar environmental effects encountered individually to accomplish these novel tasks specified by a human demonstrator. *TransAct*'s ability to model both the agent *and* environment for the entire duration of their interaction is powerful. This is particularly valuable when a learner robot has some primitive skills, but learning how to compose such skills to solve more complex tasks is expensive, a scenario that is increasingly prevalent in the robotics community. We hope that our work on enabling the transfer of task strategies (the *how* and the *what* of tasks) from humans to robots facilitates further research in this domain.

VI

Translating Abstractions in Art Domains

7 Coach FRIDA: Rewarding Exercise in Older Adults with Interactive Robot Artists

7.1 Motivation

In Part VI, we explore how the idea of mapping abstract representations of human motion to the paint and drawing strokes of interactive robot artists.

7.2 Introduction

Prior studies have shown (a) art therapy positively impacting care-recipients receiving treatment [130, 131], and (b) physical exercise via interactive games positively impacting older adults with cognitive decline [132]. Motivated by these insights, this chapter aims to create a robot artist system that rewards older adults for exercising with drawings sketched by an interactive robot artist. The robot's drawings become more engaging as the user improves, motivating them to exercise effectively. Our interactive robot artist system is implemented by first estimating the quality of a user's exercise routine by comparing it against expert exercise demonstrations; and then using this quality to modulate the drawings that the robot draws. We develop a robot artist system, that first consumes user exercise routines, and maps it to a plan of drawing or paint strokes to execute. Our technical contributions include the development of a representation learning based method to evaluate user exercise routines, an unsupervised method to map from various user exercise routines to paint strokes the robot uses, and the integration of our own work and work from other projects within our larger AI Caring grant towards our interactive robot artist system.

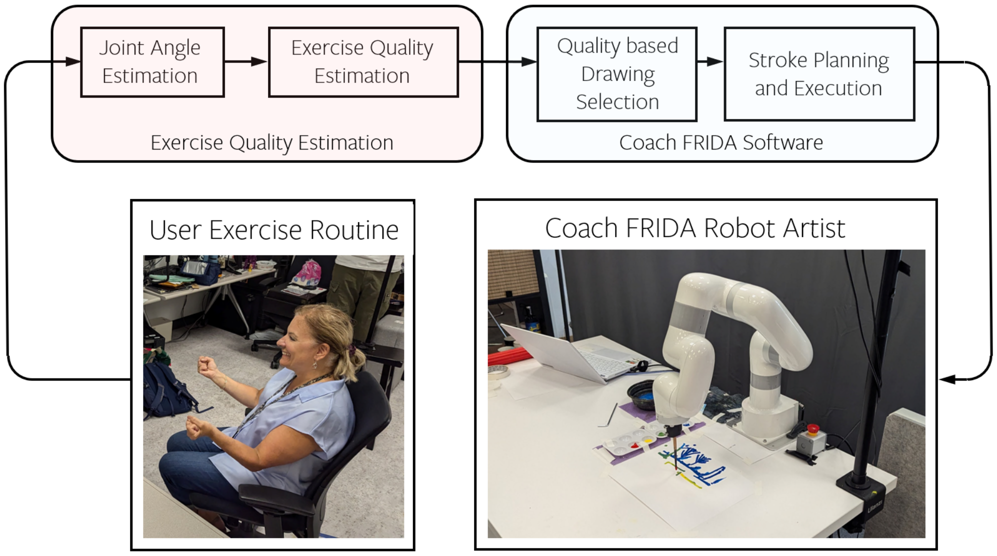

Joint Angle Estimation
Exercise Quality Estimation
Exercise Quality Estimation
Quality based Drawing Selection
Stroke Planning and Execution
Coach FRIDA Software
User Exercise Routine
Coach FRIDA Robot Artist

Figure 7.1: Our approach to Coach-FRIDA. The pipeline takes in as input a user exercise; processes this exercise, and then creates a drawing as a reward for the user! The drawing gets progressively more engaging as the user progresses with their exercises.

7.3 Approach

Our system, as depicted in Figure 7.1, first uses the pipeline from [133, 134] to receive joint angles of the user exercise routine. We then explore two different ways of evaluating the user exercise routine, and how well the user performed their exercises. The first method, used by [133, 134], selects a small number of expert demonstrations to compare against, and labels the user's performance based on the single closest demonstration. The other method that we propose uses a learned representation of all expert exercises via our prior work [73, 135], Part II, Part V. Specifically, the learned representation space consumes expert exercise demonstrations and semantic labels of the quality of these demonstrations. We developed a contrastive learning objective that clusters demonstrations performing at the same level together, and far away from demonstrations that are distinct from each other. This method enables the system to compare a user's exercise against all expert exercise demonstrations in the expert exercise dataset, analyzing user exercises with a more expressive model.

The next component of our interactive robot artist system is the mapping from the performance of the user exercise to generating a drawing that the robot artist draws. For this component, we currently employ a decision-tree like approach that maps each exercise to a set of initial drawings that the interactive robot artist samples from to draw to begin the exercise session. Based on how well the user then performs their first set of exercises, we then construct a second mapping from each level of

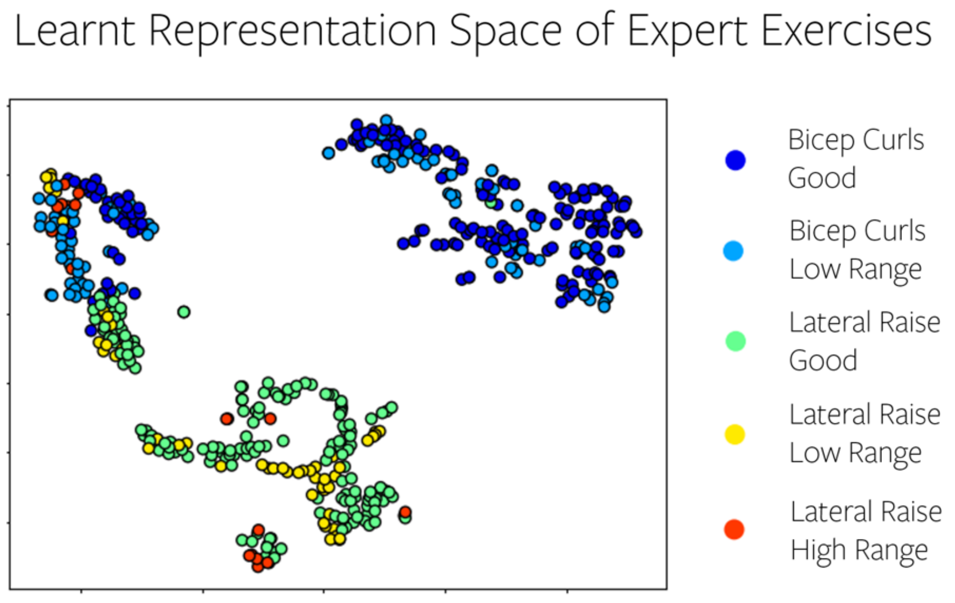


Fig. 3: Learnt Representation Space of Expert Exercises adapted from [3], for use to evaluate downstream exercise performance.

Figure 7.2: Learnt Representation Space of Expert Exercises adapted from Chapter 2, for use to evaluate downstream exercise performance.

performance for each exercise, to a set of intermediate prompts that the robot artist then uses to make the drawing more engaging. For example, consider the figure below.

On the left, we have an initial image, selected by the system for the "Lateral Raises" exercise, of an unremarkable grassy plain. Our system will begin to draw this and then, if the user performs their first set of exercises well, on par with a "good" expert demonstration, our system uses the prompt on the right to update the drawing to an aesthetic hilly landscape with forests in the foreground. Alternately, if the user's exercise performance can be improved upon, the system continues drawing the initial grassy plain. We use the Co-FRIDA system [136, 137] to have the robot draw these images.

7.4 Synergies and Collaborations

Our project leverages infrastructure from other AI Caring projects at CMU in its system. In particular, results from Y3.CR.3.1 [133, 134] affords us both a state estimation system, as well as the exercise evaluation pipeline that we can utilize to receive the user exercise routine as input to our system. We are able to bootstrap our work with these joint state estimation and evaluation pipelines. Our work also makes use of the FRIDA and Co-FRIDA systems [136, 137], developed through the AI-CARING international supplement, for the optimization process of paint strokes to match the desired image chosen by our system. Our work works in a complementary fashion to both these projects: we receive state input from [133, 134], process it via our approach, and use that to inform the drawings that are eventually drawn by the FRIDA system [136, 137]. Our project therefore aims to build infrastructure to interface between and integrate components from these two AI-CARING projects to build an interactive robot artist.

VII

Conclusions and Future Directions

8 Conclusions and Future Directions

8.1 Conclusions

My thesis takes steps towards answering the question of *"How can we learn and translate temporal abstractions of behaviour across humans and robots?"*. There are several important takeaways to be gleaned from our thesis along the way, I highlight them below.

The first of these takeaways is towards answering the question - *"How can we learn and represent temporal abstractions of agent behaviours and their effects on their environment?"*. We demonstrate that we can approach this question from a representation learning perspective, highlighting the idea that skill learning can be framed as a representation learning problem that various unsupervised machine approaches can be employed to solve. This idea in turn allows us to learn a diverse set of skills across a variety of different robots, domains, and datasets in an unsupervised manner, that can be composed to solve a variety of downstream tasks.

We then shift focus to answering the question - *"How can we understand demonstrator task strategies in terms of these abstractions, and translate these to corresponding abstractions for a robot to execute?"*. Our second key takeaway is the idea of approaching this question from a perspective of unsupervised translation of temporal abstractions of behavior. In particular, we make the insight that the problem of learning cross-domain skill correspondences is equivalent to the unsupervised machine translation problem. Employing machinery from the unsupervised machine translation community towards our problem in turn enables us to equip target domain robots with task strategies by translating over the sequence of skills executed by a human demonstrator. We believe that our approach could enable learning of correspondences across more general temporal abstractions, such as between skills and language instructions, or between skills and human video demonstrations in the wild.

Our third key takeaway is that we can then revisit both of these questions together, from a perspective of imitating desired environmental change. We introduce the *TransAct* framework, which first learns abstract representations of agent-environment interactions, then translates such interactions from human demonstrators to robot learners. TransAct empowers robots to consume in-domain human task demonstrations, retrieve and compose corresponding robot-environment interactions with similar environmental effects to perform similar tasks themselves in a zero shot manner, without access to paired demonstrations or dense annotations.

Further, our work takes small steps towards — (1) greater technical equity in robotics, by enabling diverse skill learning across multiple robots and domains; and (2) democratizing robot programming, by enabling non-expert users specify task strategies to robots via unsupervised translation.

Perhaps the most important takeaway of this thesis is viewing the idea of unsupervised translation of temporal abstractions across domains as a potent general paradigm to view a broad set of problems. We discuss potential directions that fit within this paradigm below.

8.2 Future Directions

During my time working on these directions, there have been several research directions that have struck me as interesting and important follow up directions to my prior work. I describe these directions in 3 distinct horizons below, along with initial thoughts on how to approach these directions.

8.2.1 Short Horizon - Exploring Alternate Application Domains

In the short horizon, I believe the most feasible and prudent future directions are exploring alternate application domains where the ideas of translating temporal abstractions of behaviours from humans to robots is fruitful.

Translating Dexterous Human Behaviour to Dexterous Robot Hands

A natural extension to our work in this thesis is to translate dexterous behaviours from human hands to dexterous robot hands. There are several interesting challenges to this problem - the inherent morphology gap between human hands and their robot counterparts, the differences in degrees of freedom available to these respective agents, and the inability of robot hands to receive tactile feedback at the same fidelity as humans. These challenges nonwithstanding, it would be interesting to apply ideas from Part III and Part V to translating skills (and their effects) across dexterous human and robot hands.

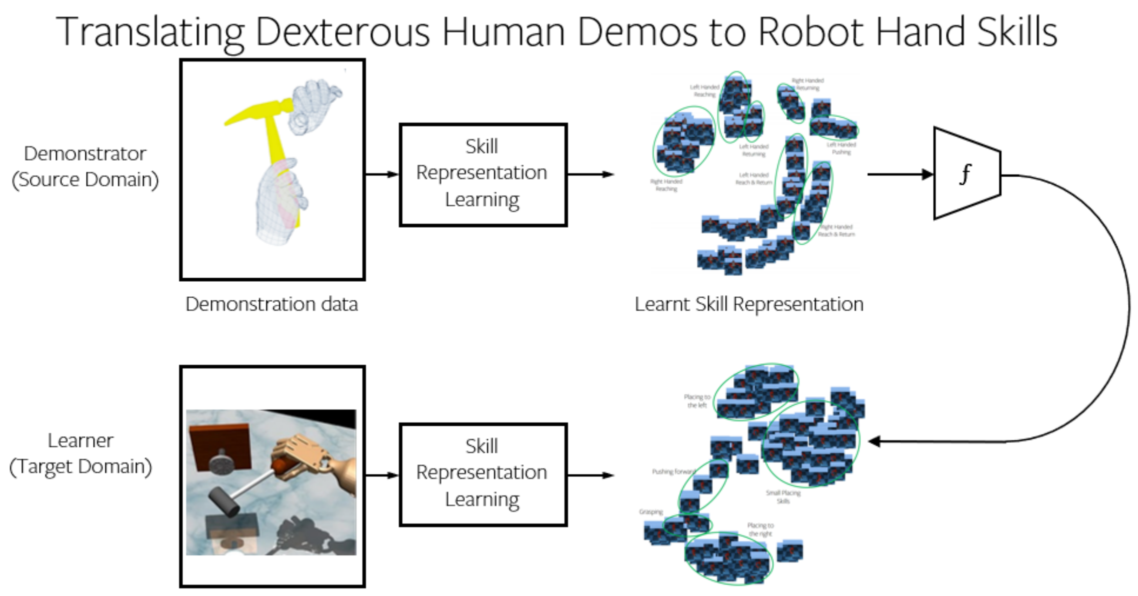

Translating Dexterous Human Demos to Robot Hand Skills
Demonstrator (Source Domain)
Skill Representation Learning
Demonstration data
Learnt Skill Representation
f
Learner (Target Domain)
Skill Representation Learning

Figure 8.1: Overview of translation of dextrous skills from humans to robots.

Translating EMG Signals to Prosthetic and Bionic Hands

Another area I am personally passionate about is moving beyond using robot hands, towards developing motor skills and control strategies for prosthetic and bionic hands. I believe exploring applying Part III and Part V to translating signals such as EMG, to such prosthetic and bionic hand skills could be a powerful development. For example, given a dataset of people performing household manipulation tasks equipped with EMG sensors, we could learn temporal abstractions of EMG signals across multiple different users / demonstrators, and translate these abstractions to corresponding motor skills on prosthetic and bionic hands. Such work could improve the accessibility of the technology developed in this thesis. More importantly, it would also provide insight into how we can design this technology with accessibility in mind.

8.2.2 Medium Horizon - Introducing New Machinery

In the medium horizon, I believe the most prudent directions to explore are introducing new machinery that allows us to solve problems previously restricted by our choices of approach.

Translating with Interaction Data

One of the biggest limitations of the breadth of work introduced in this thesis is its requirement upon demonstration data in the source and target domains. When the source domain is human teachers, using demonstrations is not an unreasonable

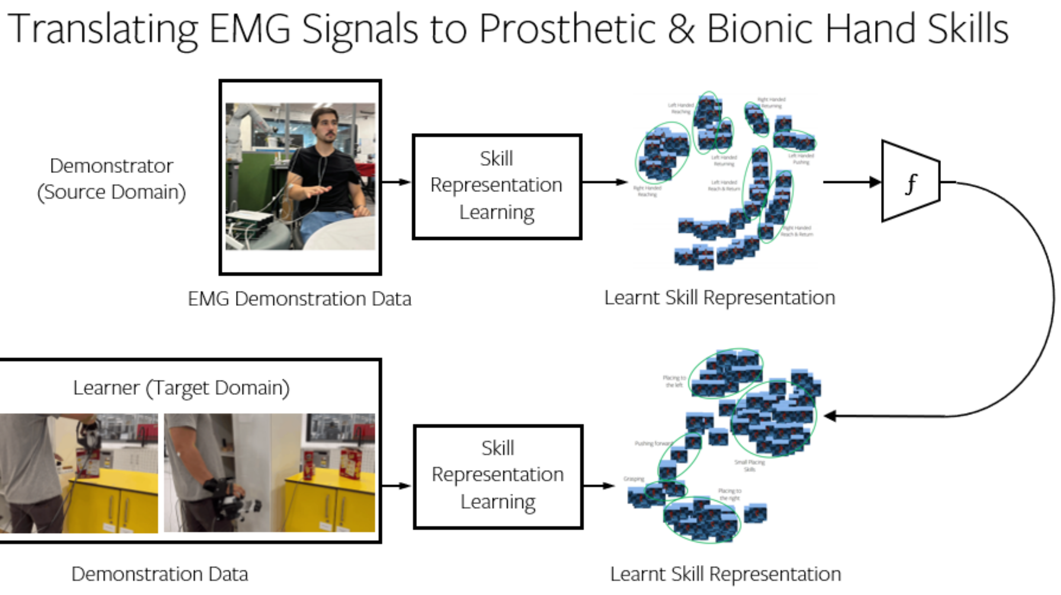

Translating EMG Signals to Prosthetic & Bionic Hand Skills
Demonstrator (Source Domain)
EMG Demonstration Data
Skill Representation Learning
Left Handed Reaching
Right Handed Returning
Left Handed Returning
Left Handed Pushing
Left Handed Reach & Return
Right Handed Reaching
Right Handed Reach & Return
Learnt Skill Representation
f
Learner (Target Domain)
Demonstration Data
Skill Representation Learning
Placing to the left
Pushing forward
Small Placing Skills
Grasping
Placing to the right
Learnt Skill Representation

Figure 8.2: Overview of translation of temporal abstractions of EMG signals to Prosthetic Hand skills.

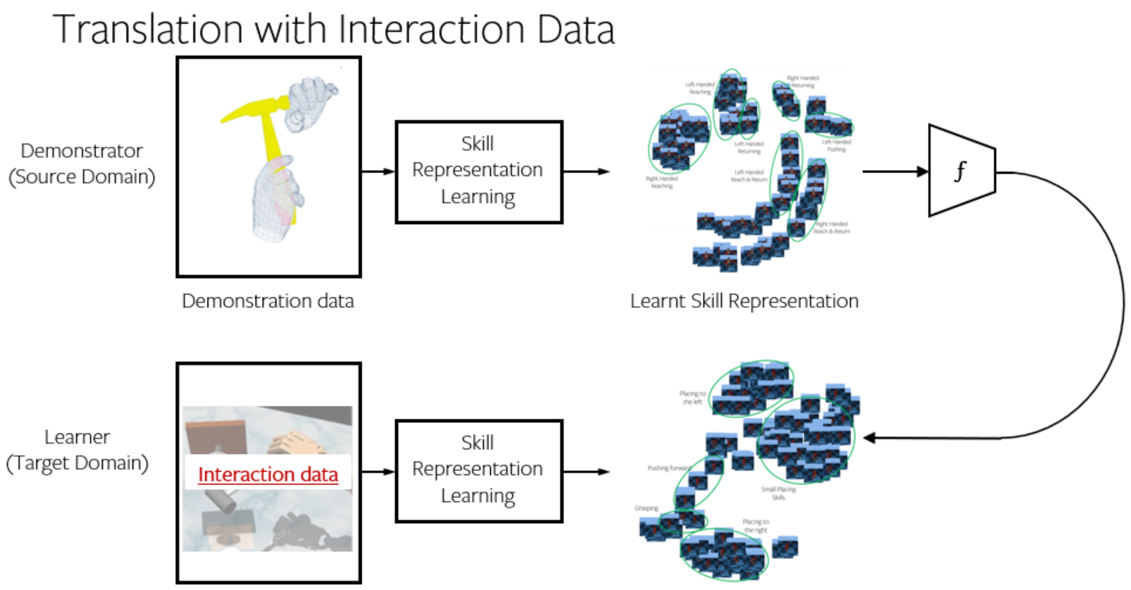

Translation with Interaction Data
Demonstrator (Source Domain)
Demonstration data
Skill Representation Learning
Learnt Skill Representation
f
Learner (Target Domain)
Interaction data
Skill Representation Learning

Figure 8.3: Overview of translation of temporal abstractions on demonstration data, to abstractions learnt from interaction data on the target domain.

proposition, considering the availability of large scale video data of people performing various tasks.

However, in the target domain of a robot, requiring demonstration data on the target robot is indeed a significant limitation. Collecting demonstration data on a new target robot is relative expensive data collection process, and requires expertise of operating the robot, and sensorization of the environment. Moreover, this scales poorly with the number of new target domain robots in consideration.

As a result, I believe introducing machinery to enable translation of abstractions with *interaction* data instead of demonstration data would be incredibly powerful. It would enable us to apply the broader idea of unsupervised translation of abstractions from human demonstrators to target robots that only have play data available in their respective domains.

Translating by Matching Environmental Effects

In order to achieve such translation to a target domain without expert demonstration data, we require some other grounding signal, *i.e.* other information that can ground data across the source and target domains without requiring heavy expensive labelling.

I believe the right signal to explore here is that of matching environmental effects, the idea we introduced in Part V. This signal is available to us irrespective of whether or not there is expert demonstration data in the source and target domains. Instead, it relies on correspondence between environmental states in the source and the target

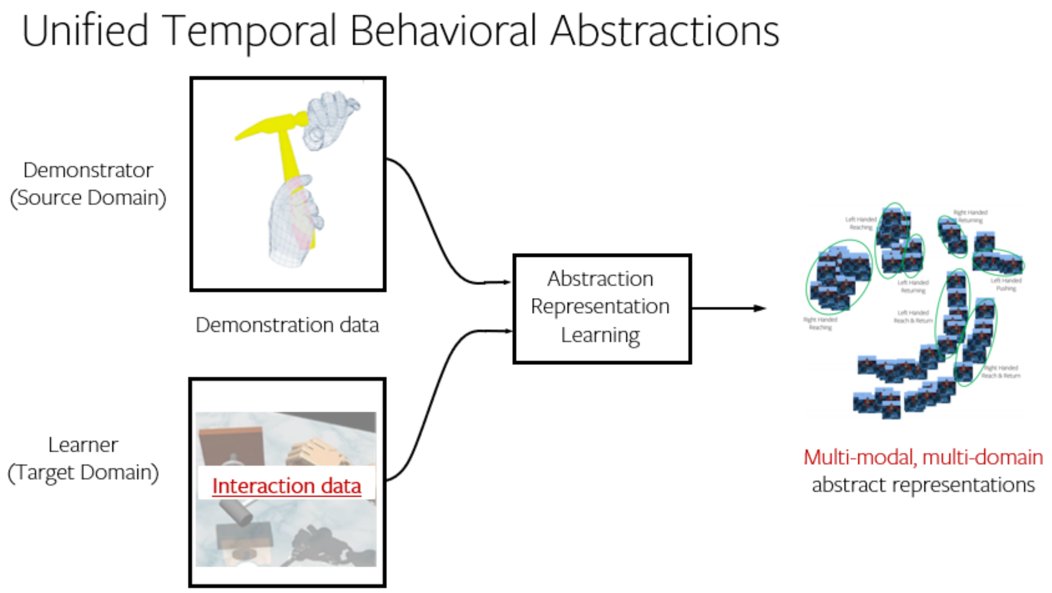


Figure 8.4: Overview of framework of learning multi-domain multi-modal abstractions from multiple domains.

domain. This correspondence itself is easier to learn than the full skill translation problem.

There are a few ways I believe we can approach translating abstractions by matching environmental effects. Consider -

- Matching distributions of object states across domains.
- Matching distributions of contact (via tactile feedback) across domains.
- Using Bisimulation metrics as a measure of when potential effects of skills in states are similar.

Learning Unified Temporal Abstractions

There is a parallel paradigm of approaches I think we can explore in addition to unsupervised translation of abstractions across domains. This paradigm is to learn unified temporal abstractions across multiple domains and multiple modalities; rather than learning abstractions individually in each domain then learning translations across these domains.

This direction is particularly well motivated in the context of recent developments in foundation models, where approaches that have been successfully trained on large amounts of data across various domains.

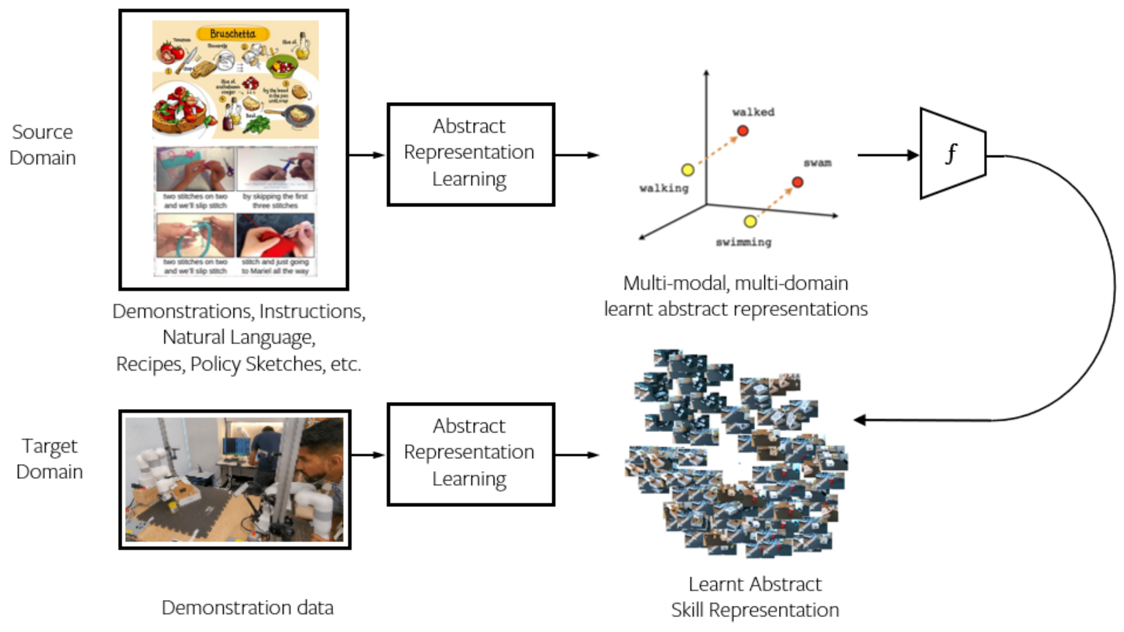

Source Domain
Demonstrations, Instructions, Natural Language, Recipes, Policy Sketches, etc.
Abstract Representation Learning
walked
walking
swam
swimming
Multi-modal, multi-domain learnt abstract representations
f
Target Domain
Demonstration data
Abstract Representation Learning
Learnt Abstract Skill Representation

Figure 8.5: Overview of paradigm of translation of temporal abstractions from abstract source domains to target domains.

8.2.3 Long Horizon - Unsupervised Translation of Abstractions as a Paradigm

To reiterate, the most important takeaway of this thesis is viewing the idea of unsupervised translation of temporal abstractions across domains as a potent general paradigm to view a broad set of problems.

While we address translating human demonstrations in this thesis, consider when humans are incapable of demonstrating the desired tasks themselves. In domains such as remote tele-operation, instructing virtual characters, or address tasks dangerous for humans; instead of providing demonstrations, people can provide instructions, or descriptions of their strategies. I believe we can similarly translate human task strategies specified as natural language descriptions, schemas, policy sketches, or recipes etc., to agent task strategies, using the machinery introduced in this thesis.

In each case, I believe we can learn some temporal abstraction of the task strategy in the source domain. For example, for natural language descriptions, we can learn phrase representations, and translate them to corresponding robot skills. Similarly, we could learn representations of gestures, to translate them to robot skills needed for the task. Similar forms of temporal abstractions are conceivable for other forms of specfying human task strategies - such as from EMG signals, steps in a recipe, etc.

I believe this idea can help us take steps towards enabling robots to match humans' ability to learn from diverse sources of information, such as demonstrations, natural

language instructions, recipes, and so on.

Bibliography

[1] Andrew N. Meltzoff and M. Keith Moore. Imitation of facial and manual gestures by human neonates. Science, 198(4312):75–78, 1977. doi: 10.1126/science.198.4312.75.

[2] Tomas Lozano-Perez Leslie Pack Kaelbling. Learning composable models of parameterized skills. In IEEE Conference on Robotics and Automation (ICRA), 2017. URL http://lis.csail.mit.edu/pubs/lpk/ICRA17.pdf.

[3] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence, 1999.

[4] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. Artificial intelligence, 1971.

[5] K. Muelling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. In 2010 10th IEEE-RAS International Conference on Humanoid Robots, pages 411–416, Dec 2010. doi: 10.1109/ICHR.2010.5686298.

[6] Taesup Kim, Sungjin Ahn, and Yoshua Bengio. Variational temporal abstraction. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 11566–11575. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9332-variational-temporal-abstraction.pdf.

[7] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. The International Journal of Robotics Research, 32(3):263–279, 2013. doi: 10.1177/0278364912472380. URL https://doi.org/10.1177/0278364912472380.

[8] George Konidaris and Andrew Barto. Skill chaining: Skill discovery in continuous domains. In the Multidisciplinary Symposium on Reinforcement Learning, Montreal, Canada, 2009.

[9] Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, page 3682–3690, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[10] Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In ICRA, 2018.

[11] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In CVPR, 2019.

[12] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. arXiv preprint arXiv:1703.08294, 2017.

[13] Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. arXiv preprint arXiv:1710.05421, 2017.

[14] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In ICML, 2019.

[15] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. Robotics and autonomous systems, 2009.

[16] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In CoRL, 2018.

[17] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In Conference on Robot Learning, 2018.

[18] Gerhard Neumann, Christian Daniel, Alexandros Paraschos, Andras Kupcsik, and Jan Peters. Learning modular policies for robotics. Frontiers in computational neuroscience, 8:62, 2014.

[19] Jan Peters, Jens Kober, Katharina Mülling, Oliver Krämer, and Gerhard Neumann. Towards robot skill learning: From simple skills to table tennis. In

Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 627–631. Springer, 2013.

[20] Rudolf Lioutikov, Gerhard Neumann, Guilherme Maeda, and Jan Peters. Learning movement primitive libraries through probabilistic segmentation. The International Journal of Robotics Research, 36(8):879–894, 2017. doi: 10.1177/0278364917713116. URL https://doi.org/10.1177/0278364917713116.

[21] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. IJRR, 2012.

[22] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=rkgHY0NYwr.

[23] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. Learning and generalization of complex tasks from unstructured demonstrations. IEEE, 2012.

[24] 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.

[25] Nasser Esmaili, Claude Sammut, and GM Shirazi. Behavioural cloning in control of a dynamic system. IEEE, 1995.

[26] Jens Kober and Jan Peters. Learning motor primitives for robotics. In ICRA, 2009.

[27] S. Schaal. Learning from demonstration. In Advances in Neural Information Processing Systems 9, pages 1040–1046, Cambridge, MA, 1997. MIT Press. URL http://www-clmc.usc.edu/publications/S/schaal-NIPS1997.pdf. clmc.

[28] Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory Hager, Pieter Abbeel, and Ken Goldberg. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. In RR. 2018.

[29] Adithyavairavan Murali, Animesh Garg, Sanjay Krishnan, Florian T Pokorny, Pieter Abbeel, Trevor Darrell, and Ken Goldberg. Tsc-dl: Unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning. In ICRA, 2016.

[30] Franziska Meier, Evangelos Theodorou, Freek Stulp, and Stefan Schaal. Movement segmentation using a primitive library. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.

[31] Rudolf Lioutikov, Guilherme Maeda, Filipe Veiga, Kristian Kersting, and Jan Peters. Learning attribute grammars for movement primitive sequencing. The International Journal of Robotics Research, 39(1):21–38, 2020. doi: 10.1177/0278364919868279. URL https://doi.org/10.1177/0278364919868279.

[32] Mohit Sharma, Arjun Sharma, Nicholas Rhinehart, and Kris M. Kitani. Directed-info GAIL: learning hierarchical policies from unsegmented demonstrations using directed information. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019. URL https://openreview.net/forum?id=BJeWUs05KQ.

[33] Matthew Smith, Herke Hoof, and Joelle Pineau. An inference-based policy gradient method for learning options. In ICML, 2018.

[34] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In AAAI, 2017.

[35] John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. arXiv preprint arXiv:1806.02813, 2018.

[36] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=S1x4ghC9tQ.

[37] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In ICML, 2017.

[38] Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. arXiv preprint arXiv:1803.01840, 2018.

[39] B. D. Ziebart, J. A. Bagnell, and A. K. Dey. The principle of maximum causal entropy for estimating interacting processes. IEEE Transactions on Information Theory, 59(4):1966–1980, April 2013. ISSN 1557-9654. doi: 10.1109/TIT.2012.2234824.

[40] Gerhard Kramer. Directed information for channels with feedback. 1998.

[41] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL http://arxiv.org/abs/1312.6114. cite arxiv:1312.6114.

[42] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 1992.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[44] CMU. Cmu graphics lab motion capture database. 2002. URL http://mocap.cs.cmu.edu.

[45] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. 2008.

[46] 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.

[47] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. JMLR, 2016.

[48] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Josh Tobin Szymon Sidor, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. arXiv preprint arXiv:1808.00177, 2018.

[49] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. IJRR, 2018.

[50] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In ICRA, 2016.

[51] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In NIPS, 2016.

[52] Karl Spencer Lashley. The problem of serial order in behavior, volume 21.

[53] Steven W Keele. Movement control in skilled motor performance. Psychological bulletin, 70(6p1):387, 1968.

[54] Jack A Adams. A closed-loop theory of motor learning. Journal of motor behavior, 3(2):111–150, 1971.

[55] Richard A Schmidt. A schema theory of discrete motor skill learning. Psychological review, 82(4):225, 1975.

[56] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Ijspeert. Learning movement primitives. In RR, 2005.

[57] Monica N Nicolescu and Maja J Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In AAMAS, 2003.

[58] Anca D Dragan, Katharina Muelling, J Andrew Bagnell, and Siddhartha S Srinivasa. Movement primitives via optimization. In ICRA, 2015.

[59] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. arXiv preprint arXiv:1804.02808, 2018.

[60] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. arXiv preprint arXiv:1903.01973, 2019.

[61] Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. Machine Learning, 2016.

[62] Shao-Hua Sun, Hyeonwoo Noh, Sriram Somasundaram, and Joseph Lim. Neural program synthesis from diverse demonstration videos. In ICML, 2018.

[63] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In KDD workshop, 1994.

[64] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. IEEE, 2013.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

[66] Oliver Kroemer, Christian Daniel, Gerhard Neumann, Herke Van Hoof, and Jan Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In ICRA, 2015.

[67] Daniel M Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. Neural networks, 1998.

[68] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. CoRR, abs/1710.04087, 2017. URL http://arxiv.org/abs/1710.04087.

[69] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. 2018. URL https://openreview.net/forum?id=rkYTTf-AZ.

[70] Chunting Zhou, Xuezhe Ma, Di Wang, and Graham Neubig. Density matching for bilingual word embedding. In Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL), Minneapolis, USA, June 2019. URL https://arxiv.org/abs/1904.02343.

[71] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

[72] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

[73] Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 8624–8633. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/shankar20b.html.

[74] Taesup Kim, Sungjin Ahn, and Yoshua Bengio. Variational temporal abstraction. In Advances in Neural Information Processing Systems 32, pages 11566–11575. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9332-variational-temporal-abstraction.pdf.

[75] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Computer Vision (ICCV), 2017 IEEE International Conference on, 2017.

[76] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for conditional image synthesis. 2020.

[77] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. Journal of Machine Learning Research, 17(59):1–35, 2016. URL http://jmlr.org/papers/v17/15-239.html.

[78] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. pages 86–96, August 2016. doi: 10.18653/v1/P16-1009. URL https://aclanthology.org/P16-1009.

[79] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. 2018.

[80] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. pages 2566–2576, 2019. doi:

10.1109/CVPR.2019.00267. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Wang_Learning_Correspondence_From_the_Cycle-Consistency_of_Time_CVPR_2019_paper.html.

[81] Abhishek Gupta, Coline Devin, Yuxuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. 2017. URL https://openreview.net/forum?id=Hyq4yhile.

[82] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. Proceedings of International Conference in Robotics and Automation (ICRA).

[83] Donald Hejna, Lerrel Pinto, and Pieter Abbeel. Hierarchically decoupled imitation for morphological transfer. pages 4159–4171, 2020.

[84] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. pages 2169–2176, 2017. doi: 10.1109/ICRA.2017.7989250. URL https://doi.org/10.1109/ICRA.2017.7989250.

[85] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. pages 2593–2603, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/8a146f1a3da4700cbf03cdc55e2daae6-Abstract.html.

[86] Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. 2020. URL https://openreview.net/forum?id=rylrdxHFDr.

[87] Yannick Schroecker and Charles L Isbell. State aware imitation learning. 30, 2017. URL https://proceedings.neurips.cc/paper/2017/file/08e6bea8e90ba87af3c9554d94db6579-Paper.pdf.

[88] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. page 2504–2510, 2015.

[89] Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. 2008. doi: 10.1145/1399504.1360626. URL https://doi.org/10.1145/1399504.1360626.

[90] Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. Skeleton-aware networks for deep motion retargeting. ACM Trans. Graph., 39(4), July 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392462. URL https://doi.org/10.1145/3386569.3392462.

[91] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 8639–8648. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00901. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Villegas_Neural_Kinematic_Networks_CVPR_2018_paper.html.

[92] M. Abdul-Massih, I. Yoo, and B. Benes. Motion style retargeting to characters with different morphologies. Computer Graphics Forum, 36(6):86–99, 2017. doi: https://doi.org/10.1111/cgf.12860. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12860.

[93] Qiang Zhang, Tete Xiao, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Learning cross-domain correspondence for control with dynamics cycle-consistency. 2021. URL https://openreview.net/forum?id=QIRlze3I6hX.

[94] Kuno Kim, Yihong Gu, Jiaming Song, Shengjia Zhao, and Stefano Ermon. Domain adaptive imitation learning. arXiv preprint arXiv:1910.00105, 2020.

[95] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. AVID: learning multi-stage tasks via pixel-level translation of human videos. 2020. doi: 10.15607/RSS.2020.XVI.024. URL https://doi.org/10.15607/RSS.2020.XVI.024.

[96] Ke Li and Jitendra Malik. Implicit maximum likelihood estimation, 2019. URL https://openreview.net/forum?id=rygunsAqYQ.

[97] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. page 2672–2680, 2014.

[98] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In arXiv preprint arXiv:2108.03298, 2021.

[99] Omid Taheri, Nima Ghorbani, Michael J. Black, and Dimitrios Tzionas. GRAB: A dataset of whole-body human grasping of objects. In European Conference on Computer Vision (ECCV), 2020. URL https://grab.is.tue.mpg.de.

[100] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. July 2017.

[101] Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 8407–8426. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/hansen-estruch22a.html.

[102] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. RSS, 2022.

[103] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. arXiv preprint arXiv:2203.13251, 2022.

[104] Jianglong Ye, Jiashun Wang, Binghao Huang, Yuzhe Qin, and Xiaolong Wang. Learning continuous grasping function with a dexterous hand from human demonstrations, 2022.

[105] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos, 2022.

[106] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance, 2022.

[107] Tanmay Shankar, Yixin Lin, Aravind Rajeswaran, Vikash Kumar, Stuart Anderson, and Jean Oh. Translating robot skills: Learning unsupervised skill correspondences across robots. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 19626–19644. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/shankar22a.html.

[108] Laura M. Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. AVID: learning multi-stage tasks via pixel-level translation of human videos. CoRR, abs/1912.04443, 2019. URL http://arxiv.org/abs/1912.04443.

[109] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 2170–2179. PMLR,

09–15 Jun 2019. URL https://proceedings.mlr.press/v97/gelada19a.html.

[110] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for mdps. In International Symposium on Artificial Intelligence and Mathematics, AI&Math 2006, Fort Lauderdale, Florida, USA, January 4-6, 2006, 2006. URL http://anytime.cs.umass.edu/aimath06/proceedings/P21.pdf.

[111] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR, abs/1703.10593, 2017. URL http://arxiv.org/abs/1703.10593.

[112] From One Hand to Multiple Hands: Imitation Learning for Dexterous Manipulation from Single-Camera Teleoperation. Qin, yuzhe and su, hao and wang, xiaolong, 2022.

[113] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. In Robotics: Science and Systems, 2023.

[114] Homanga Bharadhwaj, Abhinav Gupta, Vikash Kumar, and Shubham Tulsiani. Towards generalizable zero-shot manipulation via translating human interaction plans, 2023.

[115] Benjamin Freed, Siddarth Venkatraman, Guillaume Adrien Sartoretti, Jeff Schneider, and Howie Choset. Learning temporally AbstractWorld models without online experimentation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pages 10338–10356. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/freed23a.html.

[116] Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction, 2023.

[117] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization, 2020. URL https://openreview.net/forum?id=ryl-RTEYvB.

[118] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: explicit invariance during feature extraction. In Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, page 833–840, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[119] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 2, pages 1735–1742, 2006. doi: 10.1109/CVPR.2006.100.

[120] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In Computer Vision and Pattern Recognition (CVPR), 2016.

[121] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf.

[122] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. 2014.

[123] 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. In Proceedings of Robotics: Science and Systems (RSS), 2018.

[124] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos, 2021.

[125] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020.

[126] Danylo Malyuta, Christian Brommer, Daniel Hentzen, Thomas Stastny, Roland Siegwart, and Roland Brockers. Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition. Journal of Field Robotics, page arXiv:1908.06381, August 2019. doi: 10.1002/rob.21898. URL https://doi.org/10.1002/rob.21898.

[127] Christian Brommer, Danylo Malyuta, Daniel Hentzen, and Roland Brockers. Long-duration autonomy for small rotorcraft UAS including recharging. In IEEE/RSJ International Conference on Intelligent Robots and Systems, page arXiv:1810.05683. IEEE, oct 2018. doi: 10.1109/iros.2018.8594111. URL https://doi.org/10.1109/iros.2018.8594111.

[128] John Wang and Edwin Olson. AprilTag 2: Efficient and robust fiducial detection. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4193–4198. IEEE, oct 2016. ISBN 978-1-5090-3762-9. doi: 10.1109/IROS.2016.7759617.

[129] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. Conference on Robot Learning (CoRL), 2019.

[130] G. Kaimal, K. Carroll-Haskins, J. L. Mensinger, R. M. Dieterich-Hartwell, E. Manders, and W. P. Levin. Outcomes of art therapy and coloring for professional and informal caregivers of patients in a radiation oncology unit: A mixed methods pilot study. European Journal of Oncology Nursing, 2019.

[131] E.B. Elimimian, L. Elson, E. Stone, R.S. Butler, M. Doll, S. Roshon, C. Kondaki, A. Padgett, and Z.A. Nahleh. A pilot study of improved psychological distress with art therapy in patients with cancer undergoing chemotherapy. BMC Cancer, 2020.

[132] González-Bernal JJ, Jahouh M, González-Santos J, Mielgo-Ayuso J, Fernández-Lázaro D, and Soto-Cámara R. Influence of the use of wii games on physical frailty components in institutionalized older adults. Int J Environ Res Public Health, 2021.

[133] Roshni Kaushik and Reid Simmons. Effects of feedback styles on performance and preference for an exercise coach. International Symposium on Robot and Human Interactive Communication, 2024.

[134] Roshni Kaushik and Reid Simmons. Older adults' preferences for feedback cadence from an exercise coach robot. International Symposium on Robot and Human Interactive Communication, 2024.

[135] Tanmay Shankar, Chaitanya Chawla, Almutwakel Hassan, and Jean Oh. Translating agent environment interactions across humans and robots. International Conference on Intelligent Robots and Systems, IROS, 2024.

[136] Peter Schaldenbrand, James McCann, and Jean Oh. Frida: A collaborative robot painter with a differentiable, real2sim2real planning environment, 2022. URL https://arxiv.org/abs/2210.00664.

[137] Peter Schaldenbrand, Gaurav Parmar, Jun-Yan Zhu, James McCann, and Jean Oh. Cofrida: Self-supervised fine-tuning for human-robot co-painting, 2024. URL https://arxiv.org/abs/2402.13442.