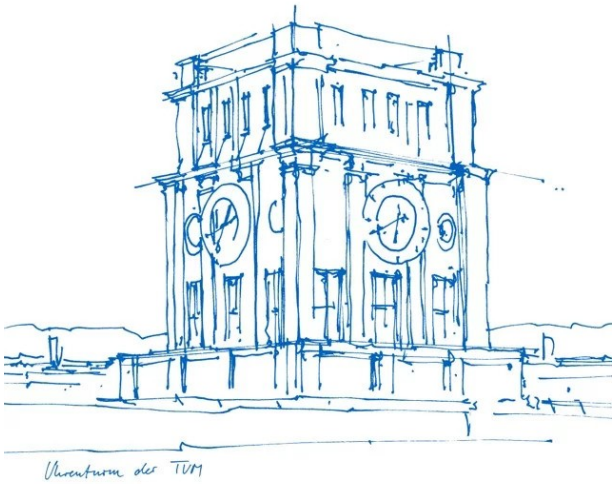
Professorship of Information-Oriented Control
School of Computation, Information and Technology
Technical University of Munich

Learning Abstract Representations of Agent-Environment Interactions

Chaitanya Chawla



Professorship of Information-Oriented Control
School of Computation, Information and Technology
Technical University of Munich

Learning Abstract Representations of Agent-Environment Interactions

Chaitanya Chawla

Thesis for the attainment of the academic degree
Bachelor of Science (B.Sc.)
at the Department of Electrical Engineering and Information Technology

Advisor:	Prof. Dr. Ing. Sandra Hirche
Supervisor:	Dr. Ing. Stefan Sosnowski
External Supervisors:	Prof. Jean Oh and Tanmay Shankar (Carnegie Mellon University, US)
Submission Date:	02.02.2024

I hereby confirm that this Bachelor's Thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources documented in the list of references

Munich, 02.02.2024 Chaitanya Chawla

Acknowledgments

I would like to thank and express my utmost gratitude to my advisor, Prof. Dr. Ing. Sandra Hirche, and supervisor, Dr. Ing. Stefan Sosnowski, whose advice and expertise guided me throughout the duration of this project. Their assistance and unwavering support shaped this work and made it a smooth and enjoyable learning experience. I am truly grateful for the opportunity to write my bachelors thesis under their supervision.

I would also like to thank Prof. Jean Oh, who willfully agreed and accepted me in her group at Carnegie Mellon University for conducting research on this topic. A most heartfelt thanks to Tanmay Shankar, for mentoring me at every stage of the project. He went beyond the usual effort and made sure I understood each concept he used in building TVI. Not only did he generously gave me full access to his code for temporal variational inference, but also offered me valuable advice and feedback during the entirety of the project. It was truly one of the best learning experiences so far.

Furthermore, I will be forever grateful to TUM Promos, the Heinrich and Lotte Mühlfenzl-Stiftung, and the DeutschlandStipendium for their financial as well as non-material support through my bachelors thesis with various scholarships.

Finally, I would like to express my deepest appreciation to my family for their unconditional love and support throughout the bachelors degree. Thank you for always being there for me.

Abstract

In learning from demonstration, tasks are generally well defined and known to a learner, e.g., opening a door or stirring a cup. In this project, we present an unsupervised approach for learning representations of common manipulation tasks such that, given a demonstration of an unknown task, a high-level task strategy can still be transferred from a teacher to a learner. Building on previous work of temporal abstractions of agent demonstrations, e.g., trajectories of a human hand or a robotic arm, we propose interaction abstractions, to extend these to representations of how an agent interacts with the objects around, e.g., a human opening a box or a robot pouring from a cup. To demonstrate the effectiveness of our learnt interaction abstractions, we generated two real-world datasets with robot- and human demonstrations. We train our model on these datasets along with three pre-compiled simulation datasets and compare the learnt abstractions with other representation learning baselines. Our findings show that our approach significantly improves the reconstruction of individual skills as well as entire trajectories. Furthermore, using the learnt abstractions, our approach enables a real-world robot to compose complex interactions to perform novel tasks beyond what it was trained on.

Contents

1. Introduction

With the recent advancement in generative AI, the robotics community has also sought to develop methods to represent robotic skills, and consequently, use them to generate a wide range of new, unseen tasks. These skill representations abstract away the low level controls and instead reason about more complex, high-level skills, such as grasping or transporting an object. This notion of skills has allowed humans to explore new tasks by utilizing previously learnt knowledge. Similarly, reasoning about such high-level skills increases an agent's capabilities for combining different skills from its knowledge base to achieve new tasks unseen during training. For example, an embodied agent equipped with the skills of grasping, pulling, and transporting, can use them to compose complex tasks, such as opening a drawer and taking objects from it.

Aside from the ability of skill composition, humans exemplify another fundamental ability of observing patterns in object-state changes to analyze the progress of a task. Humans, when interacting with objects in the environment, not only focus on the actions that are executed but also on the effects such interactions bring upon the manipulated objects. Although prior work in skill abstraction has shown a strong promise of reusable skills, most of these works either learn abstractions over agent behaviors, or over changes in environment state, but do not consider both together. Using Temporal Variational Inference, this project aims to introduce a framework to learn skill representations from interactions between agents and their environments, depicted in fig. 1.1, from demonstrations of agents interacting with objects, in an unsupervised manner.

1.1. Skills as Latent Variables

Consider an agent trajectory $\tau^r = \{s_1^r, a_1^r, s_2^r, a_2^r, ...s_{n1}^r, a_{n1}^r, s_n^r\}$, where s_t^r is the state of the agent (for example - robot joint states) affected by the agent's action a_{t-1}^r (for example - robot joint velocities) on s_{t-1}^r at time $t-1$. Given a trajectory τ^r, an option ω (also referred to as a *skill*) executed during the trajectory is a temporal abstraction, where the agent behaves in a consistent manner across some period of time. We assume that the identity of an option ω is defined by a latent variable z. The sequence of options in a trajectory is represented through a sequence of latent variables, $\zeta = \{z_t, b_t\}_{t=1}^T$, where b_t defines whether to terminate the current option or not. Note that the terms *latent variables* and *skills* refer to the same entity and are used interchangeably in this project.

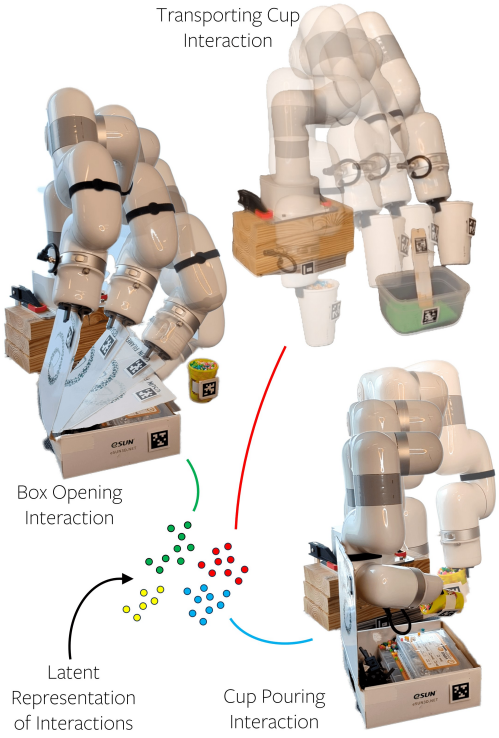


Figure 1.1.: Overview of learning skill abstraction. A depiction of a sample agent-environment interaction space that can be learnt by the model. Actual samples of the interaction space with real data are shown in a later chapter.

1.2. Standard Variational Inference

Temporal Variational Inference originates from Standard Variational Inference (SVI), where a variational distribution $q(z|x)$ (usually a Gaussian) is used to infer latent variables z from observed data x, approximating the unknown conditional $p(z|x)$. The likelihood of observations is then optimized given the predicted latent variables under a learned decoder $p(x|z)$. For our work, x represents the ground truth trajectory of the agent and environment from the demonstrations and z is the latent skill abstraction of these trajectories.

To restate our goal, we want to infer the sequence of latent variables $\zeta = \{z_t, b_t\}_{t=1}^{T}$ from a trajectory $\tau = \{s_t, a_t\}_{t=1}^{T}$. We do this by estimating the conditional probability $p(\zeta|\tau)$ with a variational approximation $q(\zeta|\tau)$. However, SVI requires optimization of $p(\tau|\zeta)$, which is not possible in our case as the distribution of variable ζ can only be inferred at test time. To address this problem, we instead seek to optimize the joint likelihood $p(\tau, \zeta)$ of the trajectory and latent variables with the help of two policies *pi* and *eta*. Section 2.2 on Temporal Variational Inference explains how these two policies can be used to infer the joint likelihood.

1.3. Related Work

Learning from Demonstration

Learning from Demonstration (LfD) is an approach allowing non-expert demonstrations to be used for solving tasks, avoiding the manual specification of skills as shown by *Stilman et al.* [1] or engineering solutions for definite tasks, shown by works like *Qin et al.* [2], [3], and *Sivakumar et al.* [4], which have developed mappings between demonstrators and robot state to facilitate imitation of human demonstrators. While traditional LfD is an interesting approach to transfer demonstrations from a teacher to a learner, it has some major drawbacks (*Argall et al.* [5]) such as a need for experts, limited scalability, and a dominant rigidity instead of environment adaptability. On the other hand, data-driven methods are more computationally sustainable, data accessibility, and resilient to out-of-distribution errors as shown in *Padalkar et al.* [6]. We also seek to learn a policy network from demonstrations.

Motion Primitives

A popular concept of motion primitives to learn robot behavior, introduced by *Peters et al.* [7], can help form adaptable step-wise trajectory representations. However, the reason for their limited use (*Savariano et al.* [8]) in robot learning is their complex dynamics tuning, limited generalization, and difficulty representing high-dimensional data. Efforts (*Neumann et al. 2014* [9]) have also been made to sequence these primitives for better generalization in downstream tasks. Our work differs in the sense that it jointly learns both the representation of primitives and how they must be sequenced.

State Abstractions

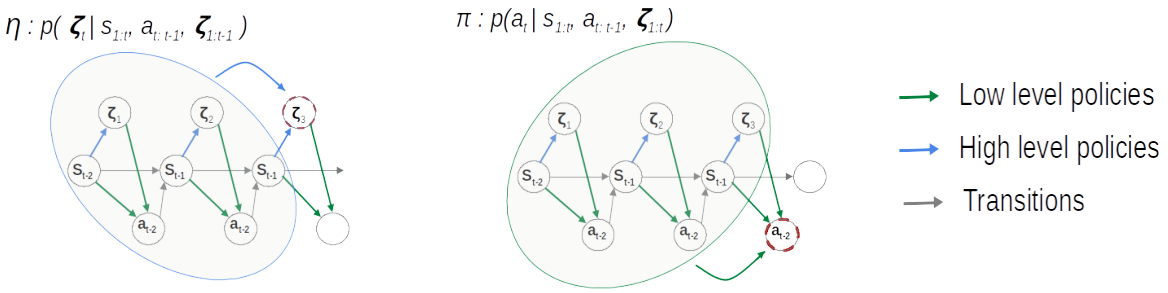
To achieve such a higher level of understanding of robot skills, attempts have been made to represent them as abstractions. *Gelada et al.* [10] and *Zhang et al.* [11] work on learning state abstractions that facilitate ignoring irrelevant components of environments, and making analogies across various environment- and task-instances. Work has also been done in developing temporal abstractions of agent behaviour as shown in *Sharma et al.* [12], *Eysenbach at al.* [13], and *Shankar et al.* [14]. The approaches that learn temporal abstractions of behaviors are often unaware of the patterns of object state changes they induce (i.e., their effects) [13] and [14]. Conversely, most environmental state abstractions are unaware of the behaviors that caused them, shown in *Gelada et al.* [10] and *Zhang et al.* [11]. However, our work argues that it is important to understand interactions between agents and their environment together at an abstract level.

1.4. Problem Statement

Learning skills in an unsupervised, data-driven manner not only avoids having to annotate demonstrations but also enables the use of large-scale and diverse demonstration data in

robotics. In this project, the goal is to study the effect of learning agent- (e.g. robot trajectory) and environment behavior (e.g. object trajectory) together, along with the interactions between them, in an unsupervised manner. By doing so, the model aims to achieve an understanding of the agent behaviors needed to affect the desired environmental changes. Thereafter, we seek to reason upon the improvements by such a model in trajectory prediction compared to frameworks exclusively trained on agent- or environment behavior.

The project is built upon a robot skill learning framework and learns temporal abstractions of both environment state and robot skills in an unsupervised manner. We aim to study the effectiveness of using such an approach in reconstructing demonstrations as well as its usability in composing unseen, downstream tasks.



(a) High Level Policy η (b) Low Level Policy π

Figure 2.1.: Latent variable inference policies (*Shankar et al.* [14], p. 4)

2. Approach

Behavioral abstractions, or skills, are considered to be a representation of an agent acting consistently for a specific time period. For example, in the case of a Pick and Place task, the model should be able to segment skills such as - grasping, lifting, transporting the object and so on. This work adopts the behavioral abstraction framework of [14], which uses Temporal Variational Inference to infer these abstractions.

2.1. Framework Overview

At every timestep, a high-level policy η selects the next skill ζ_t to be executed, as well as the termination variable b_t, using the previous states $s_{1:t}$, actions $a_{1:t}$, and latent variables $\zeta_{1:t-1}$. This helps a low-level policy *pi* infer the next action a_t, utilizing the history of states $s_{1:t}$, actions $a_{1:t-1}$, and latent variables $\zeta_{1:t}$ as depicted in fig. 2.1 (*Shankar et al.* [14], p. 4). The action a_t can be executed on s_t to obtain s_{t+1}. Using this framework, eqn. 2.1 defines the joint likelihood $p(\tau, \zeta)$ of the trajectory and the latent variables as described in [14].

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

2.2. Background on Temporal Variational Inference

Temporal Variational Inference (TVI) trains a variational encoder $q^r(z|\tau, r)$ that takes as input an agent trajectory τ^r and outputs a sequence of skill encodings $z^r = \{z_1^r, z_2^r, ...z_k^r\}$, where

$k < n$ is a learnt number of skills executed. TVI also trains a latent conditioned policy $\pi^r(a^r|s^r, z^r)$ that takes as input agent state s^r, and the chosen skill encoding z^r, and predicts the low-level action a^r that the agent should execute. TVI trains q^r and π^r to reconstruct the actions observed in the trajectory τ^r. Fig. 2.2 shows the overall pipeline of the project along with the environment abstractions.

Unlike Standard Variational Inference (see section 1.2), which optimizes the conditional $p(\zeta|\tau)$, TVI optimizes the joint likelihood $p(\tau, \zeta)$ of trajectories and latent variables with respect to the policies π^r and η^r. This helps obtain usable policies, which can be queried at test time. We start with the standard objective of maximizing the log-likelihood of trajectories across the Dataset D, $L = \mathbb{E}_{\tau \sim D}[log p(\tau)]$. This is lower bound with

$$J = \mathbb{E}_{\tau \sim D}[log\ p(\tau)] - D_{KL}[q(\zeta|\tau)\ ||\ p(\zeta|\tau)], \tag{2.2}$$

where the latter term describes the Kullbeck Liebler (KL) Divergence [15] (see appendix A.1). Substituting eqn. 2.1 in eqn. 2.2, we get the objective:

$$J = \mathbb{E}_{\tau \sim D, \zeta \sim q(\zeta|\tau)} \Bigg[\sum_{t=1} \Big\{ log\ \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) + log\ \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) \\ + log\ p(s_{t+1}|s_t, a_t) \Big\} + log\ p(s_1) - log\ q(\zeta|\tau) \Bigg] \tag{2.3}$$

Taking $\theta, \phi,$ and ω as parameters for $\pi, \eta,$ and q, we can compute the gradient for the objective J to optimize the policies:

$$\Delta J = \Delta_{\theta,\phi,\omega}\ \mathbb{E}_{\tau \sim D, \zeta \sim q(\zeta|\tau)} \Bigg[\sum_{t=1} \Big\{ log\ \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) + log\ \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) \Big\} \\ - log\ q(\zeta|\tau) \Bigg] \tag{2.4}$$

For a more detailed derivation of TVI, please refer to the appendix A.2.

Policy Parameters

Policies π and η are parameterized as LSTMs, with 8 layers and 128 hidden units per layer, as they only utilize the past data of agent states, actions and latent variables to make the next predictions. On the other hand, q is parameterized as a bi-directional LSTM, since q predicts the entire sequence of latent variables ζ given the complete trajectory τ.

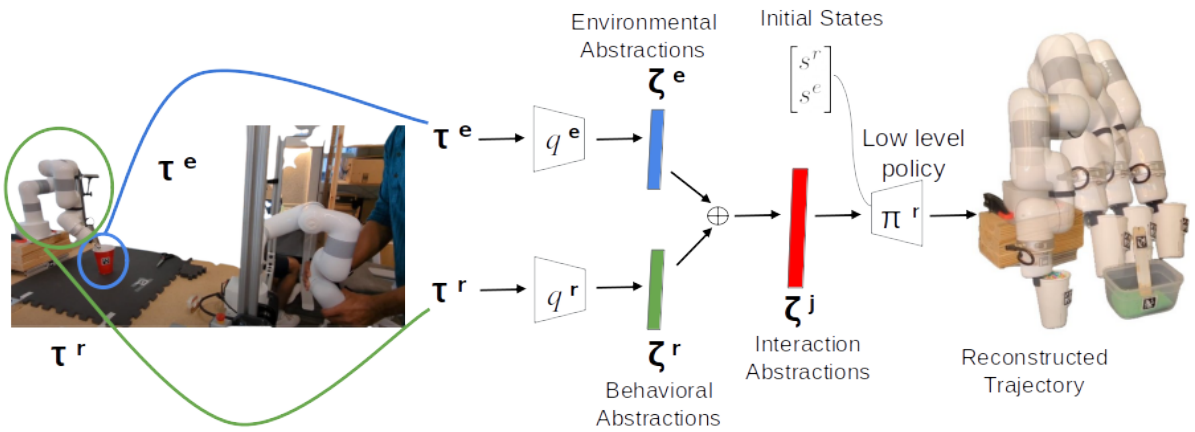


Figure 2.2.: Overview of the pipeline.

Advantages of TVI

1. Joint training of $\pi, \eta,$ and q allows training of high-level policy η based on the available options, as well as the adaption of low-level policy π's actions based on how effective they are in reconstructing demonstrations.

2. TVI allows a continuous parameterization of options z^j, eliminating the need to pre-specify the number of options required. Thus, it may learn as many options required to capture the complete trajectory information.

3. Furthermore, π and η are causally conditioned, i.e. they only depend on past information of states. This makes them directly usable for querying at test time.

2.3. Building Interaction Abstractions from Agent and Environmental Abstractions

As described in section 2.2, we record a similar trajectory for the objects in the environment $\tau^e = \{s^e_1, a^e_1, s^e_2, a^e_2, ...s^e_{n1}, a^e_{n1}, s^e_n\}$, with a^e_t describing the change in environment state at time t, rather than the action of the agent. An equivalent variational encoder $q^e(z|\tau^e)$ for the environment was trained to predict the sequence of latent encodings $z^e = \{z^e_1, z^e_2, ..., z^e_k\}$.

After obtaining the latent agent $\{z^r\}^n_{t=1}$ and environment $\{z^e\}^n_{t=1}$ abstractions, we concatenate them to form abstractions of the interactions $\{z^j\}^n_{t=1}$ and condition the agent policy π^r on these interaction abstractions. At test time, given a sequence of desired interactions $\{z^j_1, z^j_2, ..., z^j_k\}$, we can feed the agent policy $\pi^r(a|s, z^j)$ these desired interactions, and the current state s^r, s^e, and query for the next agent action to execute a^r. We depict this pictorially in fig. 2.2. During training, we can update encoders q^r and q^e, and policy π^r, to optimize the

evidence lower bound to the loglikelihood of observed state changes in trajectories τ^r and τ^e under the policy π^r.

3. Datasets and Pre-Processing

We wanted to study how effectively the model can learn the structure of the demonstrations, withstanding the out-of-domain performance in the real world. We generated two real-world datasets - a robot dataset and a human dataset. Furthermore, we also tested our model on pre-generated simulation datasets. Collectively, these datasets spanned a variety of morphologies of robots, and interactions, including pushing, transporting, reorienting, grasping, rotating, etc.

3.1. Real World Datasets

3.1.1. Robot Dataset

We generated a real-world dataset of an X-ARM Lite6 robot performing 5 different tasks: Pouring, Stirring, Box-Opening, Drawer-Opening, and Pick-Place. We collected 10 demonstrations for each task (except for drawer opening, where we collected 6 demonstrations). For each demonstration, we collected data on a puppet robot, that mimics a kinesthetically controlled master bot, depicted to the left of fig. 2.2. We recorded the 6 DoF joint state, and gripper state as the robot state s_t, and treated joint velocities and gripper opening and closing as the robot action a_t. We placed Apriltags [16] [17] on each of the two objects involved in a task for object-state estimation, and collected the 6-D poses (position and 4D quaternion orientation) of both objects as the object state (also pictured in fig. 2.2). This 13-D state was the input for the model.

3.1.2. Human Dataset

A similar real-world dataset was generated with a human performing the above 5 tasks and additionally 3 combinations of these tasks: Drawer-Opening + Pick-Place, Pouring + Stirring, and Box-Opening + Pouring. This was implemented to study how the model reasons between a complex task and the individual tasks it was composed of. For the objects, we again used Apriltags [17] to retrieve the poses. As different tasks had varying numbers of objects, the number of objects was fixed for all tasks as 4 (excluding ground), which is the maximum number of objects present in any task. For a demonstration having less than 4 objects (i.e. the base five tasks), dummy tags were included in the dataset, with negative tag IDs and null poses. This helped maintain consistency across the tasks and the model was able to learn the difference between the varying number of objects as shown later in the experiments section.

I. Human Pose Estimation

To capture human motion, we required 3D data of human hand poses. However, off-the-shelf 3D pose estimation models (FrankMocap [18], MediaPipe [19], and DexYCB [20]) provide poses, which are relative to a point in the hand, e.g. the base of the palm. Whereas absolute world coordinates are required to obtain the trajectory of the hand. To tackle this issue, we deployed a 2D hand pose estimation model - MMPose [21] - estimating 21 2D keypoints of a human hand, and mapped RGB frames with corresponding depth frames received from a RealSense D4325i (RGBD) camera. This allowed us to compute the respective depths of each keypoint detected in every time step. Fig. 3.1 shows the real-time inference results from MMPose.

Figure 3.1.: 21 2D keypoint inference from MMPose

3.2. Simulation Datasets

Our model was also trained and tested on some publicly available simulated robot datasets Roboturk [22] (collected on a Sawyer robot), and RoboMimic [23] and FrankaKitchen [24][25], (both collected on a Franka). For each dataset, we use the robot joint states as robot state, gripper state and joint-velocities as robot actions, and the 6-D object pose of the primary object as the object state.

3.3. Pre-Processing

Collecting data from the real world is often erroneous and unreliable due to various reasons, such as sensor inconsistency and out-of-domain performance of pose estimation models. To reduce outliers and obtain smoother trajectories across datasets, we use the following preprocessing methods -

I. Pose Transformation

To bring all pose information to the same world coordinates, we transform objects' positions and orientations from camera coordinates to ground object coordinates using:

$$\begin{bmatrix} {}^{gnd}R_{obj} & {}^{gnd}t_{obj} \\ \underline{0} & 1 \end{bmatrix} = \begin{bmatrix} {}^{cam}R_{gnd} & {}^{cam}t_{gnd} \\ \underline{0} & 1 \end{bmatrix}^{-1} \begin{bmatrix} {}^{cam}R_{obj} & {}^{cam}t_{obj} \\ \underline{0} & 1 \end{bmatrix} \tag{3.1}$$

where ${}^{S}R_{A}$ denotes the rotation matrix of A in the S coordinate frame and ${}^{S}t_{A}$ denotes the translation of A in the S coordinate frame. Similarly, in the human dataset, the 3D positions of the keypoints are transformed to the world coordinates.

This is also essential as we fuse data from two cameras, to obtain a union set of the valid poses from both cameras and increase the number of frames with valid tag poses. Invalid poses are not fed to the model for training directly, instead, we skip these and interpolate the valid poses to compensate for them and maintain the original demonstration length. For more details, refer to section 3.3 (V).

As mentioned later in the chapter 4, experiments showed that a further decrease in the state reconstruction was possible by feeding the model object poses relative to the agent, instead of poses in world coordinates.

II. Hand Pose Validation Filter

We also implemented two filters based on the natural properties of a human hand.

Firstly, if the distance between keypoints in any frame did not reflect the keypoint distances of a normal human hand within an acceptable range, that particular frame was marked invalid and skipped during interpolation. Note that this filter was only applied after transformation to world coordinates, hence it did not depend on the distance from the camera.

Secondly, it was noted that MMPose [21] returned a pose estimate, even when no hands were present in the frame. The above filter could not filter these false poses, as they sometimes conform to real human hands. However, it was noted that these hand poses were always detected at a specific location in a frame. Thus, we bypassed the issue by marking the poses detected in that specific location invalid.

III. Gaussian Weighted-Average Filtering

Our approach of mapping RGB data on depth frames to compute depth had a major limitation. A minimally deviated keypoint estimation in RGB frame could predict a keypoint outside the actual hand, leading to highly divergent estimated depth.

Instead of using a pure point estimate for the depth of the keypoint, we calculated a Gaussian weighted average of K=5 points on either side of the specific keypoint along both the X and Y axes in the camera frame. This helped slightly reduce the anomalies in depth estimation, but there was still room for improvement.

IV. Temporal Filtering

Another filter to remove detection anomalies from the human dataset used a temporal filter, which marked a frame invalid if the distances between the pose keypoints from the current and preceding frame exceeded a given threshold.

V. Pose Interpolation

With the previously mentioned filters, many frames across the demonstrations were marked as invalid, thus inapt for the final dataset. Furthermore, cameras do not always capture the most optimum data due to various environmental factors, such as lighting, reflections, and occlusion. This produces sub-optimal images, leading to unacceptable hand pose estimation or AprilTag detections, which must be omitted from the final dataset.

To compensate for these omitted frames and maintain the original demonstration length, we interpolate across the valid frames upon the position and orientation of hand keypoints and the objects in the environment. Two interpolate the two types of data - 3D positions and 4D orientation quaternions, we employed the following techniques:

1D Interpolation

For positions of hand keypoints and environment objects, we used a vanilla linear 1D interpolation.

$$y = y_0 + (x - x_0) * \frac{y_1 - y_0}{x_1 - x_0} \tag{3.2}$$

Slerp Interpolation

For orientations of the environment objects, we used a slerp method. Slerp - 'spherical linear interpolation' [26], describes an interpolation (with constant angular velocity) along the shortest path on the unit hypersphere between two quaternions q_1 and q_2. Eqn. 3.3 defines the slerp function:

$$Slerp(q_1, q_2; u) = q_1(q_1^{-1}q_2)^u \tag{3.3}$$

where the parameter u moves from 0 (simplifying the expression to q_1) to 1 (simplifying the expression to q_2).

4. Experiments and Results

We conducted four main experiments to test the skill learning efficacy of the model, quantitatively and qualitatively. The four experiments seek to answer the following questions:

1. How well can the learnt interaction abstractions model individual skill segments?
2. Can the model understand the similarities and dissimilarities of structure in different, individual skills?
3. Can the learnt interaction abstractions accurately reconstruct the original demonstrations?
4. Can the learnt skills be used for combinatorial, downstream tasks?

4.1. Reconstructing Individual Skills

We train the model separately on the previously mentioned datasets, sample a set of individual skills (a trajectory segment from the test set of respective datasets), and obtain the latent skill abstraction z^j. By feeding the latent abstraction and the start state $s_{t=0}$ to the policies π and η, we obtain a reconstructed trajectory of the individual skills.

For a quantitative comparison, we calculate the state reconstruction error across three parameters: robot joint angles, object cartesian positions, and object cartesian orientations. All three parameters are firstly normalized, and then compared with an L2-norm between the normalized ground truth and the predicted trajectory. We compare our approach against other baselines based on a hierarchical Dynamic Motion Primitives based approach [28] and TVI [20], the skill representation learning approach. The results have been shown in 4.1. Both TVI and our approach reconstruct skills with significantly low reconstruction errors. However, the additional information on agent-environment interactions in our approach allows for a better representation.

We qualitatively visualize the results by firstly rolling out the reconstructed skill trajectories on a Gazebo Simulator (see fig. 4.1) and then on the actual robot (see fig. 4.2).

4.2. Latent Space Representation

In addition to reconstructing individual skills, we visualize a set of hand-picked skill samples across each dataset in a latent space representation. Using T-SNE [38], a non-linear dimen-

Table 4.1.: State Reconstruction Error. Lower is better.

	Agent-Env State Reconstruction Error		
Dataset	H-DMP	TVI	IntAbs (Ours)
RealWorld (Ours)	1.12	0.09	**0.04**
RTurk	3.48	0.72	**0.62**
RMimic	2.09	0.84	**0.58**
FrankaKit	1.85	0.53	**0.32**

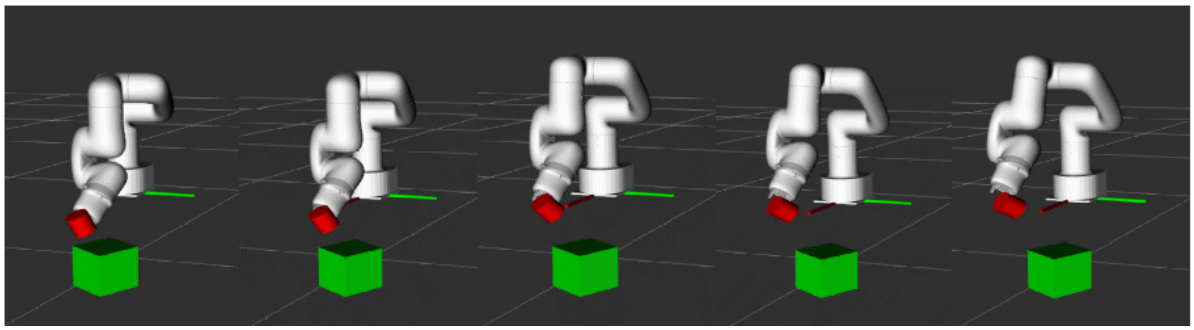


Figure 4.1.: Sequence of reconstructed Box Opening skill on Gazebo. The red cube shows the movement of the lid of the box.

Figure 4.2.: Sequence of reconstructed Box Opening skill on real robot

sionality reduction technique, we were able to map the 16-dimensional latent variables z^j's of different skills in a 2D space. Fig. 4.3 shows such a representation for the real-world robot dataset by plotting a single frame from each individual skill z^j for visualization. The advantage of using T-SNE over PCA [39], is the property of T-SNE to preserve the local clustering of data points in lower dimensions as well, whereas PCA strives to preserve the variance of the data. Note the rough clustering of similar interactions into similar parts of the space. Please refer to the appendix A.3 for further results on latent space visualizations.

To analyze the latent space quantitatively, we label a set of 200 individual interaction trajectory segments with one of 17 semantic labels of the type of interaction occurring. This is split into two sets: a base set for creating a latent space representation, and a test set for 50 samples for querying to the space. We evaluate the clustering efficacy by using a K-nearest neighbor classifier (with $K = 5$) over the latent space with 50 queries from the test set. It was observed that our unsupervised model achieves a 39.5% accuracy on this 17-way classification problem, compared to a random classifier with 5.8% accuracy. This confirms the models ability to cluster similarly structured demonstrations together.

4.3. Entire Demonstration Reconstruction

We also qualitatively evaluate the model's ability to reconstruct entire task demonstrations. In a similar manner to section 4.1, except in this case we feed in a sequence of N interaction abstractions, i.e. $\{z_j^n\}_{n=1}^N$, comprising of 15-20 skills for the robot. Fig. 4.4 visualizes the rollouts of two reconstructed tasks: Pick-Place and Drawer-Opening. It was also observed in some reconstructions that the robot failed to grasp the object, such as the drawer, due to significant divergence from the training data.

4.4. Combinatorial Skills

As a final experiment, we showed that the individual latent representations z^j's can be used to construct novel, downstream tasks, unseen by the model during training. For example, using our robot dataset, a novel task could be defined as firstly opening a box and then pouring a liquid inside it.

To evaluate the combinatorial property of the interaction abstractions, we manually select a set of learnt abstractions from two tasks: $\{z^j_{Task1}\}$ and $\{z^j_{Task2}\}$ and concatenate them to form a new, unseen task representation: $\{z^j_{Task3}\} = \{z^j_{Task2}, z^j_{Task2}\}$. We then feed in this new task specification to our policy π, along with a starting state $[s^r, s^e]_{t=0}$, and generate a new trajectory to execute for the composite task. This was tested for two sets of tasks as pictured in figures 4.5 and 4.6: firstly, opening a box and pouring a liquid, and secondly, pouring a liquid from a cup and then stirring.

The execution of both the compositional tasks shows the model's ability to complete complex tasks, despite never having seen these sequences in training. However, certain limita-

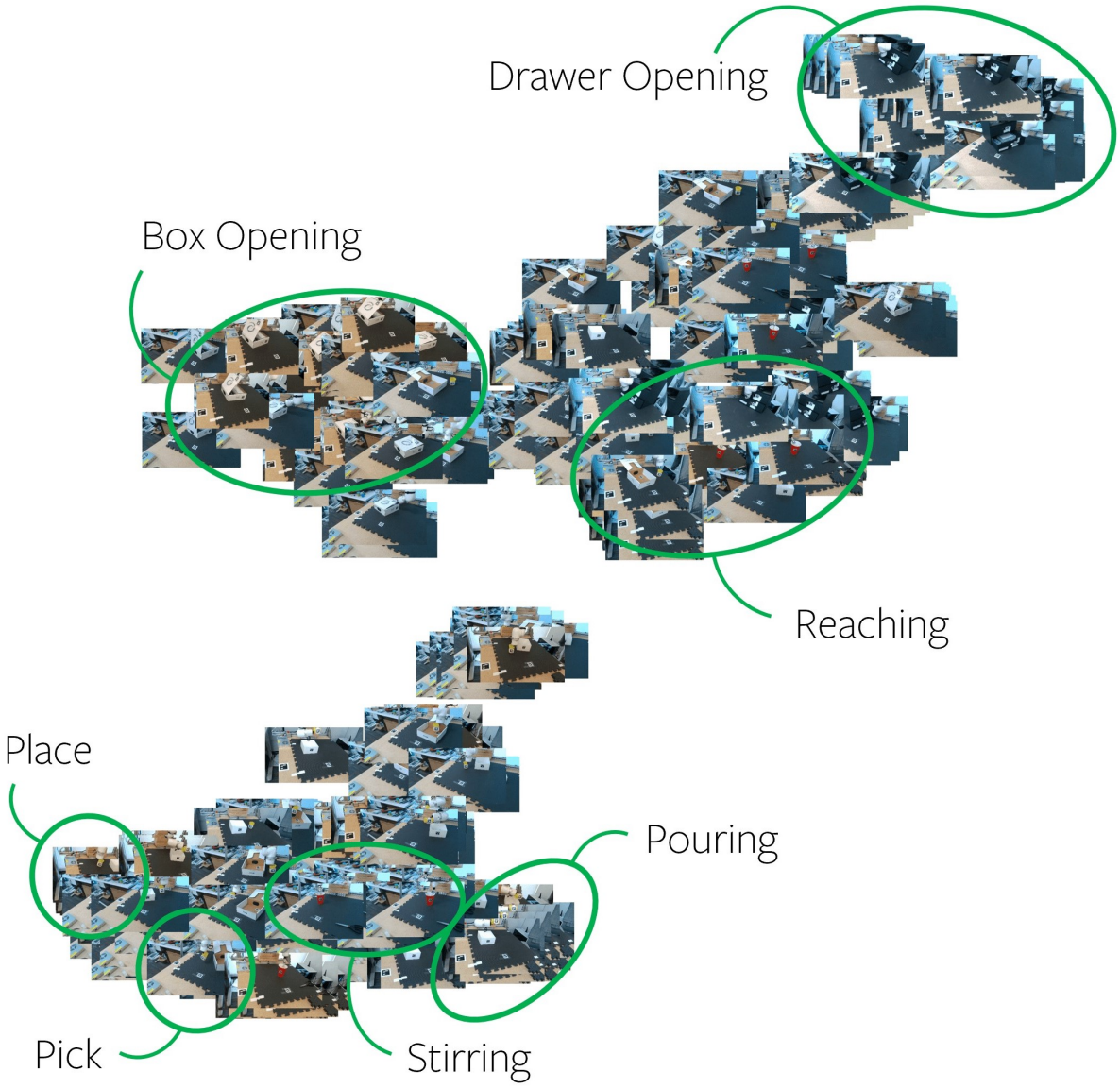


Figure 4.3.: The learnt latent representation space from the real-world robot dataset.

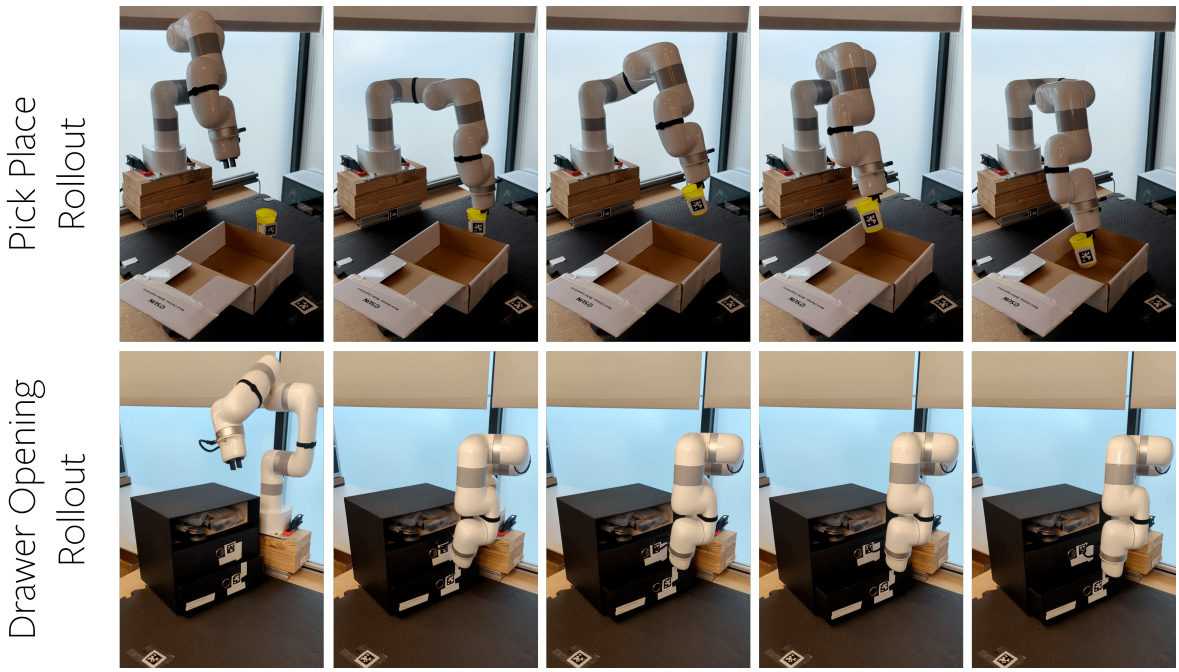


Figure 4.4.: Rollouts of the reconstructed trajectories on the real robot for two tasks, PickPlace (top) and DrawerOpening (bottom).

tions were also observed during experiments. As can be seen in fig. 4.6, the robot pours some beads outside the container and even drops the cup after pouring the beads, which suggests room for improvement.

4.5. Limitations

One significant limitation of the model is its inability to succeed when there is a considerable difference (>5cm) between the current environment states and the training data. As seen in sections 4.3 and 4.4, the robot fails to complete the tasks at hand due to imperfect grasps during reconstruction and fails to recover from these failures.

Another limitation of the model is the manual selection of skill abstractions in composing new downstream tasks. A potential method to solve this would be to incorporate the final object states in the training and test phases, to make the robot capable of trying different strategies to reach the final goal.

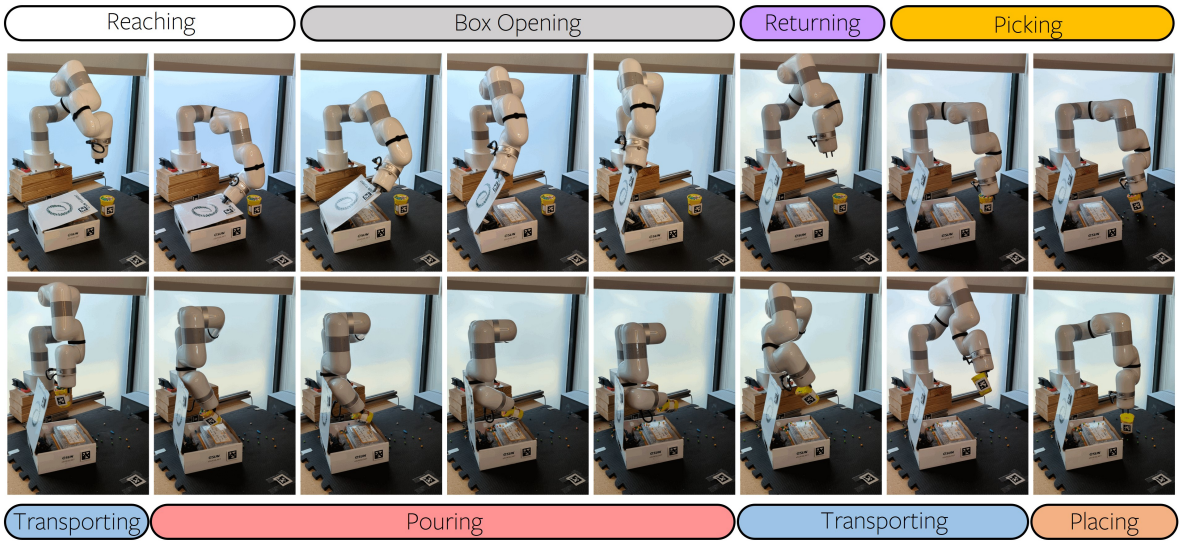


Figure 4.5.: Real-world robot executing the compositional task of opening a box, then pouring beads from a cup into the box. We label which interaction occurs in the frames for better visualization; the model does not have access to such labels.

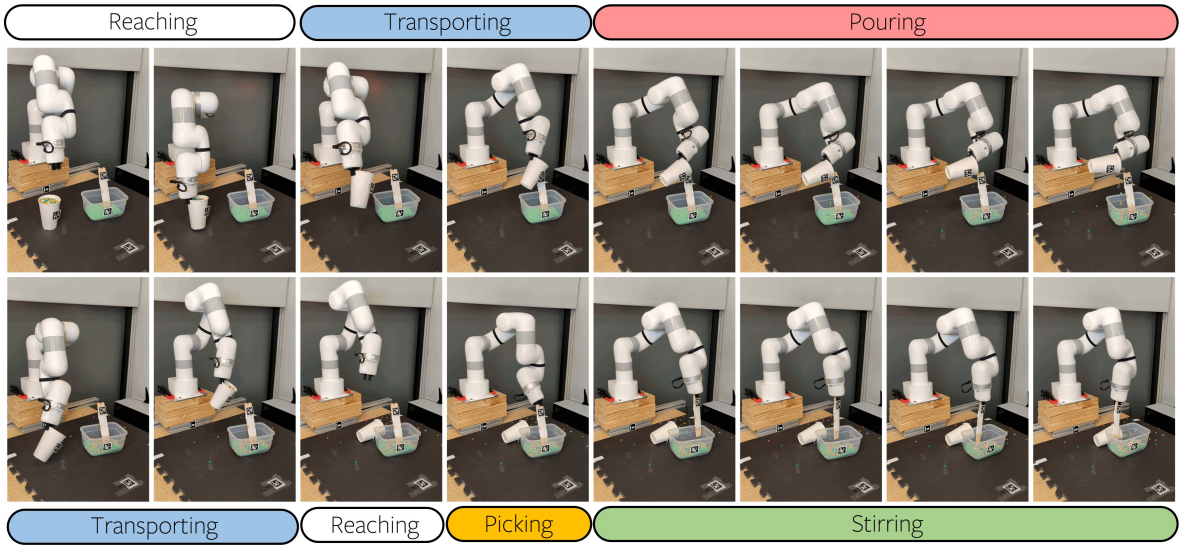


Figure 4.6.: Real-world robot executing the compositional task of pouring beads from a cup, then stirring them with a spoon.

5. Conclusion

This project presents a framework for jointly learning agent and environment behavior from various task demonstrations and evaluates the efficacy of such an approach in reconstructing individual skills as well as entire robotic manipulation demonstrations in an unsupervised manner. The experimental results indicate that incorporating environment behavior and agent-environment interactions significantly improves the modelling of robot demonstrations, yielding much higher similarity between ground truth and reconstructed trajectories. It is also shown that by using such an approach we can compose interactions in novel ways to complete tasks beyond its existing knowledge base.

In the future, extending the model's capabilities further with the following features would improve its effectiveness in real-world applications. Firstly, by incorporating human instructions in selecting interaction abstractions for complex tasks (e.g., "pour water, then stir it"). This would exponentially increase the scope of tasks that can be executed by the model. Secondly, we believe these abstract representations of skills could also simplify the process of translating task strategies across different robot morphologies. Representing latent variables in a morphology-independent manner would help the transfer of skills across robots, significantly reducing the effort and time required to learn new skills.

A. Appendix

A.1. Kullback-Liebler Divergence

Kullback-Liebler divergence (often referred to as KL Divergence) is a statistical way to compare two probability distributions. It is often in machine learning that we replace the observed data or a complex distribution with a simpler, approximating distribution. KL divergence helps us measure the amount of information lost while approximating such a distribution. To quantify, how much information is in the data, an important metric used in information theory is Entropy, denoted as H:

$$H = -\sum_{i=1}^{N} q(x_i).\ log\ q(x_i) \tag{A.1}$$

H describes the theoretical lower bound of data units (bits if log_2) required to encode our distribution q with N variables x_n. Entropy doesn't specify the optimal encoding scheme to achieve this compression but gives a way to quantify the amount of information in the data. This helps us quantify how much information is lost by substituting the observed distribution for a parametrized approximation.

KL divergence is a modification of entropy, where we add the approximating distribution p and look at the difference of the log values for each:

$$H = -\sum_{i=1}^{N} q(x_i).\ (log\ q(x_i) - log\ p(x_i)) \tag{A.2}$$

Essentially, what we're looking at with the KL divergence is the expectation of the log difference between the probability of data in the original distribution and the approximating distribution. Our goal in section 2.2 is to find an approximate of the true distribution $p(z)$ by comparing it with a similar variational distribution $q(z)$, or minimizing the KL divergence between them.

$$D_{KL}(q||p) = \mathbb{E}\left[\frac{log\ q(x)}{log\ p(x)}\right] \tag{A.3}$$

A.2. Temporal Variational Inference

Eqn.2.2 in section 2.2 states the lower bound of the objective for maximizing the log-likelihood of trajectories τ belonging to Dataset D:

$$J = \mathbb{E}_{\tau \sim D}[log\ p(\tau)] - D_{KL}[q(\zeta|\tau)\ ||\ p(\zeta|\tau)] \tag{A.4}$$

The second term refers to the KL Divergence (section A.1) between the variational approximation $q(\zeta|\tau)$ and the unknown distribution $p(\zeta|\tau)$. Substituting eqn. A.3 in the above eqn.:

$$J = \mathbb{E}_{\tau \sim D}[log\ p(\tau)] - \mathbb{E}_{\tau \sim D,\, \zeta \sim q(\zeta|\tau)}\Big[\frac{log\ q(\zeta|\tau)}{log\ p(\zeta|\tau)}\Big] \tag{A.5}$$

$$J = \mathbb{E}_{\tau \sim D,\, \zeta \sim q(\zeta|\tau)}\Big[log\ p(\tau) - log\ q(\zeta|\tau) + log\ p(\zeta|\tau)\Big] \tag{A.6}$$

$$J = \mathbb{E}_{\tau \sim D,\, \zeta \sim q(\zeta|\tau)}\Big[log\ p(\tau, \zeta) - log\ q(\zeta|\tau)\Big] \tag{A.7}$$

where $log\ p(\tau) + log\ p(\zeta|\tau) = log\ p(\tau, \zeta)$.
Substituting the joint decomposition from eqn. 2.1 above, gives us:

$$J = \mathbb{E}_{\tau \sim D, \zeta \sim q(\zeta|\tau)}\Bigg[\sum_{t=1}\Big\{log\ \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) + log\ \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) + log\ p(s_{t+1}|s_t, a_t)\Big\} + log\ p(s_1) - log\ q(\zeta|\tau)\Bigg] \tag{A.8}$$

A.3. Additional Visual Results

A.3.1 Individual Skills Reconstruction

Figures A.1 - A.4 show reconstruction of individual skills on different datasets.

A.3.2 Latent Space Representation

Figure A.3 shows the latent space representation of RoboTurk Dataset.

A.3.3 Entire Demonstration Reconstruction

Figures A.6 - A.8 show reconstruction of complete trajectories on different datasets.

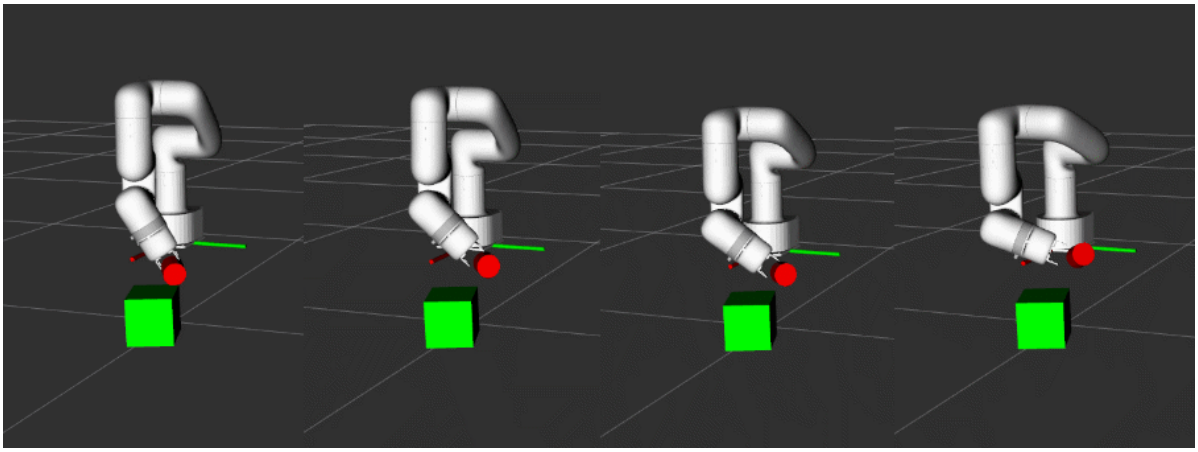


Figure A.1.: Sequence of reconstructed Pouring skill on Gazebo. The red cylinder shows the movement of the pouring cup.

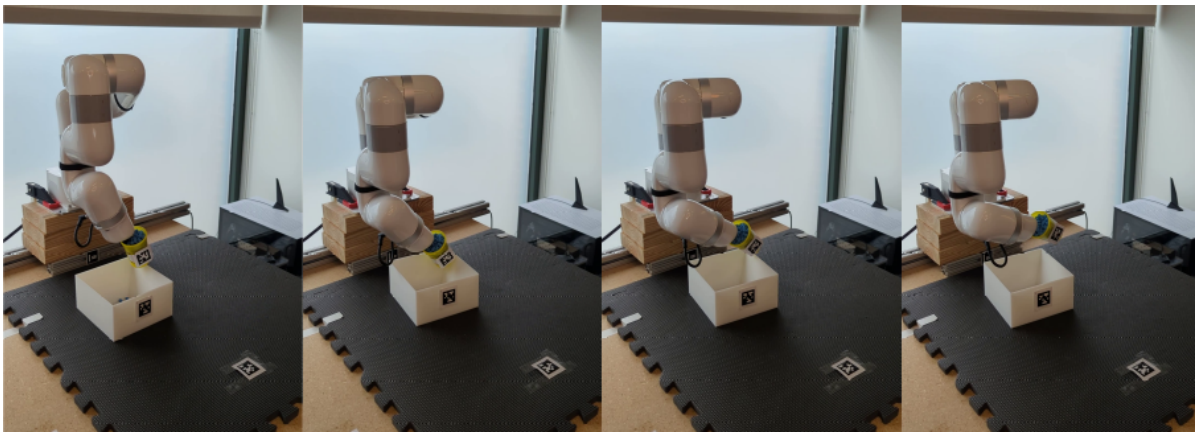


Figure A.2.: Sequence of reconstructed Pouring skill on real robot.

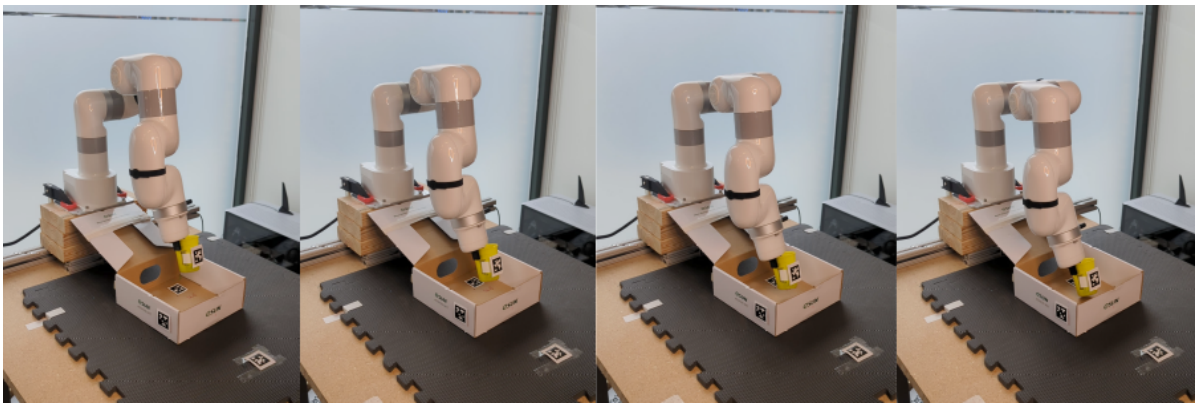


Figure A.3.: Sequence of reconstructed Placing skill on real robot.

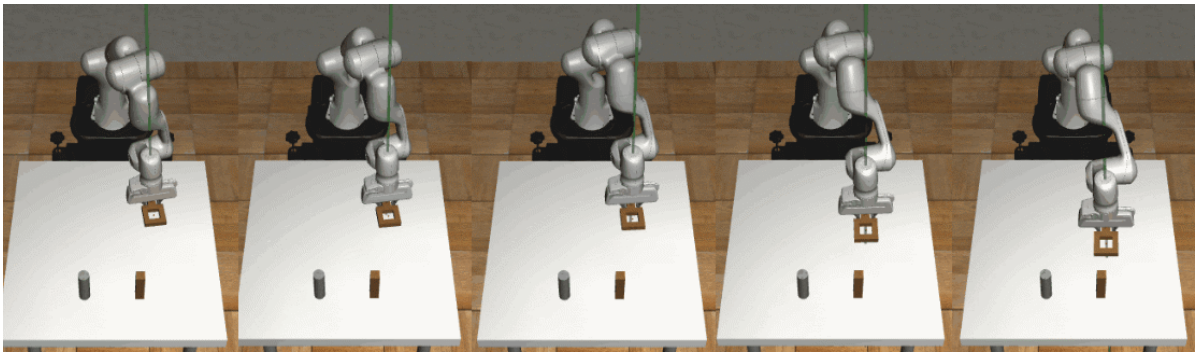


Figure A.4.: Sequence of reconstructed Lifting skill from RoboMimic Dataset.

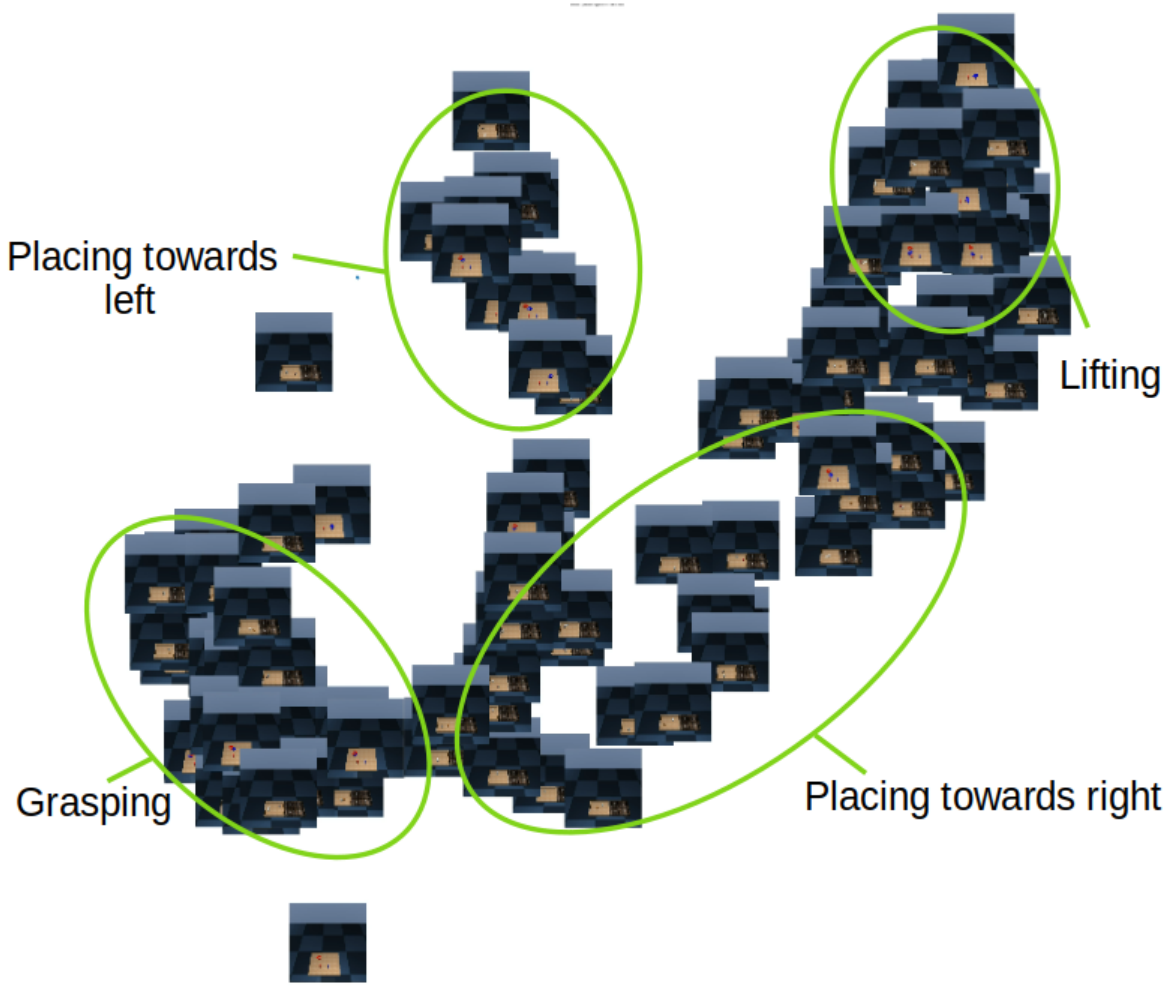


Figure A.5.: Latent Space Representation of RoboTurk Dataset.

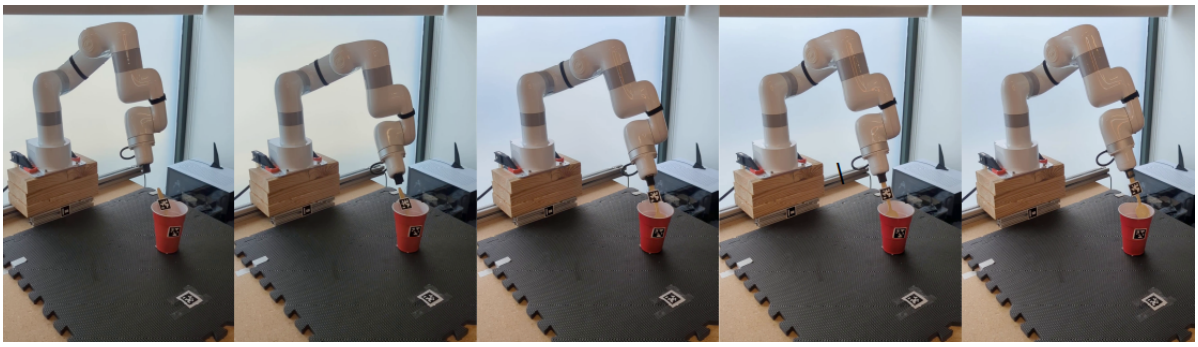


Figure A.6.: Rollout of the reconstructed trajectory for Stirring on the real robot.

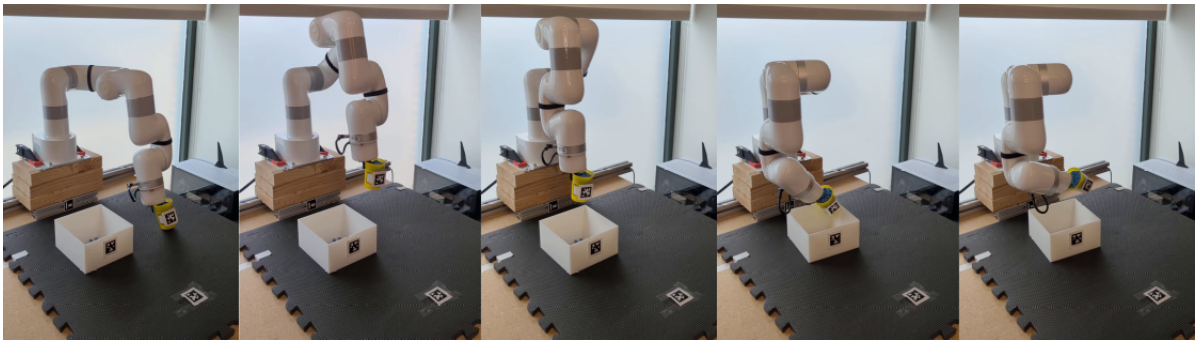


Figure A.7.: Rollout of the reconstructed trajectory for Pouring on the real robot.

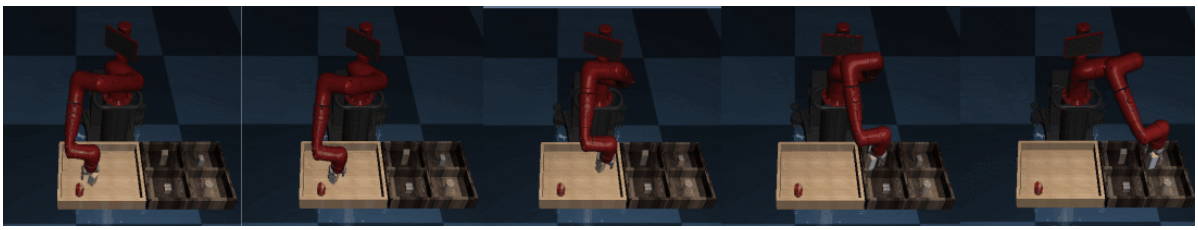


Figure A.8.: Rollout of the reconstructed trajectory for Bin Picking from RoboTurk Dataset.

Bibliography

[1] M. Stilman. "Global Manipulation Planning in Robot Joint Space With Task Constraints". In: *IEEE Transactions on Robotics* 26 (2010), pp. 576–584.

[2] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. *DexMV: Imitation Learning for Dexterous Manipulation from Human Videos*. 2022. arXiv: 2108.05877 [cs.LG].

[3] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox. "AnyTeleop: A General Vision-Based Dexterous Robot Arm-Hand Teleoperation System". In: *Robotics: Science and Systems*. 2023.

[4] A. Sivakumar, K. Shaw, and D. Pathak. "Robotic Telekinesis: Learning a Robotic Hand Imitator by Watching Humans on Youtube". In: *RSS* (2022).

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[6] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. "Open x-embodiment: Robotic learning datasets and rt-x models". In: *arXiv preprint arXiv:2310.08864* (2023).

[7] J. Peters, J. Kober, K. Mülling, O. Krämer, and G. Neumann. "Towards robot skill learning: From simple skills to table tennis". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 627–631.

[8] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel. "Dynamic movement primitives in robotics: A tutorial survey". In: *The International Journal of Robotics Research* 42.13 (2023), pp. 1133–1184.

[9] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters. "Learning movement primitive libraries through probabilistic segmentation". In: *The International Journal of Robotics Research* 36.8 (2017), pp. 879–894.

[10] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. "DeepMDP: Learning Continuous Latent Space Models for Representation Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2170–2179. URL: https://proceedings.mlr.press/v97/gelada19a.html.

[11] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, and X. Wang. "Learning Cross-Domain Correspondence for Control with Dynamics Cycle-Consistency". In: (2021). URL: https://openreview.net/forum?id=QIRlze3I6hX.

[12] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. "Dynamics-Aware Unsupervised Discovery of Skills". In: (2020). URL: https://openreview.net/forum?id=HJgLZR4KvH.

[13] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. "Diversity is All You Need: Learning Skills without a Reward Function". In: (2019). URL: https://openreview.net/forum?id=SJx63jRqFm.

[14] T. Shankar and A. Gupta. "Learning Robot Skills with Temporal Variational Inference". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 8624–8633. URL: https://proceedings.mlr.press/v119/shankar20b.html.

[15] J. Shlens. "Notes on kullback-leibler divergence and likelihood". In: *arXiv preprint arXiv:1404.2000* (2014).

[16] J. Wang and E. Olson. "AprilTag 2: Efficient and robust fiducial detection". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2016, pp. 4193–4198. ISBN: 978-1-5090-3762-9. DOI: 10.1109/IROS.2016.7759617.

[17] D. Malyuta, C. Brommer, D. Hentzen, T. Stastny, R. Siegwart, and R. Brockers. "Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition". In: *Journal of Field Robotics* (Aug. 2019), arXiv:1908.06381. DOI: 10.1002/rob.21898. URL: https://doi.org/10.1002/rob.21898.

[18] Y. Rong, T. Shiratori, and H. Joo. "Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1749–1759.

[19] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. "Mediapipe hands: On-device real-time hand tracking". In: *arXiv preprint arXiv:2006.10214* (2020).

[20] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal, S. Birchfield, J. Kautz, and D. Fox. "DexYCB: A Benchmark for Capturing Hand Grasping of Objects". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[21] M. Contributors. *OpenMMLab Pose Estimation Toolbox and Benchmark*. https://github.com/open-mmlab/mmpose. 2020.

[22] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. "RoboTurk: A Crowdsourcing Platform for Robotic Skill Learning through Imitation". In: *Conference on Robot Learning*. 2018.

[23] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. "What Matters in Learning from Offline Human Demonstrations for Robot Manipulation". In: *arXiv preprint arXiv:2108.03298*. 2021.

[24] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. "Relay Policy Learning: Solving Long Horizon Tasks via Imitation and Reinforcement Learning". In: *Conference on Robot Learning (CoRL)* (2019).

[25] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2020. arXiv: 2004.07219 [cs.LG].

[26] K. Shoemake. "Animating rotation with quaternion curves". In: *SIGGRAPH Comput. Graph.* 19 (1985), pp. 245–254. URL: https://doi.org/10.1145/325165.325242.

[27] S. Krishnan, R. Fox, I. Stoica, and K. Goldberg. "Ddco: Discovery of deep continuous options for robot learning from demonstrations". In: *arXiv preprint arXiv:1710.05421* (2017).

[28] E. B. Dam, M. Koch, and M. Lillholm. *Quaternions, interpolation and animation*. Vol. 2. Datalogisk Institut, Københavns Universitet Copenhagen, Denmark, 1998.