ABSTRACT

Title of Dissertation: FROM DEMONSTRATION TO DYNAMIC INTERACTION: ENABLING LONG-TERM ROBOTIC PLANNING

Mara Levy
Doctor of Philosophy, 2025

Dissertation Directed by: Professor Abhinav Shrivastava
Department of Computer Science
University of Maryland, College Park

Robotic learning has seen rapid growth over the past decade, driven by advances in machine learning that have brought real-world deployment of robots closer to reality. Research in this area primarily falls into two categories: reinforcement learning and imitation learning. Despite their promise, both approaches face significant challenges, including limited data availability and the difficulty of obtaining accurate state representations. This thesis explores how we can advance these methods to enable robust performance in real-world, unstructured environments.

We begin by exploring how to redefine state representation, presenting two complementary approaches. The first focuses on human state representation but is easily extendable to robots. It significantly outperforms existing methods in generalizing to unseen states and varying camera viewpoints. The second approach introduces a more concise, keypoint-based representation. We show that this method enables training of robot policies with minimal demonstrations and generalizes effectively to new environments and objects of varying shapes and sizes.

Next, we turn to the problem of learning policies from a single demonstration, without relying on handcrafted reward functions. Remarkably, our method achieves

comparable final performance to existing approaches while using 100× less data. Finally, we demonstrate how these methods can be deployed in dynamic environments, even when trained under static conditions. By layering a lightweight planner on top of a pretrained policy, we achieve substantial improvements over naïve replanning strategies, approaching oracle-level success rates.

FROM DEMONSTRATION TO DYNAMIC INTERACTION: ENABLING LONG-TERM ROBOTIC PLANNING

by

Mara Levy

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:
Professor Abhinav Shrivastava, Chair/Advisor
Professor Shuvra S. Bhattacharyya, Dean's Representative
Professor Lerrel Pinto
Professor Pratap Tokekar
Professor Ruohan Gao

To my amazing parents, who supported me and made this possible.

Acknowledgements

It's hard for me to express how many times this dissertation almost did not get written. Without the support of these people I would have dropped out of this PhD the same year I started. I'm forever grateful to everyone who pushed me to be the absolute best version of myself. I owe this to all of you.

First and foremost, I want to thank my advisor, Abhinav Shrivastava. From the very beginning, he saw something in me that I couldn't see in myself, and he consistently encouraged me to pursue that potential. He built a lab environment centered around collaboration rather than competition, surrounding me with brilliant and supportive peers. Although my research didn't always align directly with his expertise, Abhinav was instrumental in helping me overcome roadblocks offering a fresh set of eyes when I didn't know how to move forward.

Throughout every challenge, he supported me. He has a remarkable ability to look at a problem and make you realize it's not as insurmountable as it feels. He taught me that failure is never personal and showed me how to move forward when things didn't go as planned. I can't begin to describe how thankful I am to have had the opportunity to work with such a smart and caring person.

Next I want to thank Professor Lerrel Pinto, who welcomed me into his lab at a time when I was eager to gain outside experience. Despite not being one of his students, he never made me feel like an outsider. During my time in New York, I learned a tremendous amount—not only from him but also from the students in his lab. My time at NYU was pivotal in my growth as a roboticist, and I'm incredibly thankful for the opportunity to be a part of that community.

I would also like to acknowledge the professors I worked with during my undergrad. Kostas Daniilidis gave me my first real exposure to research by welcoming me into his lab and allowing me to learn alongside his students. Max Mintz, who advised me as an undergraduate, saw a future for me in research long before I did. His encouragement pushed me to explore that path, and without his belief in me, I likely would never have pursued a PhD. And finally, Rajiv Gandhi, who taught my very first computer science course in college and remains the best professor I've ever had. When I told him I wanted to apply to the University of Maryland, he didn't hesitate for a moment before writing an email for me.

Finally, in the spirit of honoring the educators who shaped my journey, I want to thank my high school computer science teacher, Steve Svetlik. More than perhaps anyone I've ever met, he believed that women belong in computer science—and made sure his students knew it too. Without his encouragement and unwavering belief, I might never have discovered my passion for computer science or had the confidence to pursue it this far. Everyone deserves to have someone like him in their lives.

I also want to thank my amazing lab mates, who have stood by me through every twist and turn, always ready to listen and support me. Lillian Huang was one of the

first friendly faces I met—she introduced me to so many people and helped me feel at home when I knew no one else. Nirat Saini not only taught me how to write a paper and make a proper figure, but she was also a steadfast friend and confidant during the moments when everything felt like it was falling apart.

To all of my collaborators—Shirley, Daniel, Eric, Mukund, Pulkit, Siddhant, Anubhav, Saksham, Chuong, Matt G, Vatsal, and Max—thank you. A special thanks to those of you who gave me the opportunity to mentor you; your trust helped me build confidence in my own abilities and played a vital role in getting me to this point. And to the rest of my lab—Namitha, Archana, Sharath, Shishira, Prateksha, Matt W, Kamal, Soumik, Saketh, Gaurav, Hanyu, Bo, Yixuan, and Khoi—thank you for making these years as joyful and memorable as they can be.

I would also like to thank Tom Hurst, Migo Gui, Sharron McElroy, Jodie Gray, Janet Woolery, Matthew Baney, and other UMD CS and UMIACS staff members who have helped with administrative tasks and hurdles.

Finally, I want to thank my incredible family. To my boyfriend, Justin—thank you for bearing the full weight of the stress that comes with a PhD. I'm sorry for all the times I disappeared during deadlines; I hope seeing this work come to life makes some of it feel worth it. I love you so much for always reminding me that no hurdle was worth a breakdown. Even when I didn't believe it, it mattered to hear it.

To my brother, Mitchell, you have been such a role model in my life and always believed that I have what it takes to succeed. Thank you for letting me live in your NYC apartment for six months so I could work at NYU. To my sister, Dana, thank you for the countless hours spent on the phone listening to me vent. I'm so grateful

for your constant support, even when I steal your clothes or buy the same phone case. Being your little sister is such a privilege, and I never take either of you for granted.

Lastly, to my amazing parents—I truly wouldn't be here without you. You've supported me in more ways than I could possibly describe. Your unconditional love, even when I've been difficult, has shaped me into the person I am today. You gave me the freedom to make my own decisions and mistakes while always being there to guide me. You taught me what it means to be determined and driven. I owe you everything.

Table of contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the last decade, research in the fields of computer vision and machine learning has advanced rapidly. This progression has improved the viability of robotic learning, a field that had been largely ignored in favor of a physics-based solution. Despite the reinvigoration of the field, there are still many challenges that are unique to robotic learning. The largest issue in the community is the amount of data available compared to the amount of data required to train generalized algorithms. There are several reasons that collecting data has been a problem in robotics. Although cost is a clear issue, there are several other problems that are less visible. One of these is that operating a robot can be difficult and that means collecting data requires skilled labor. Additionally, even if a person has this training, collecting data for robots is time intensive.

Researchers in robotic learning have largely adopted one of two general approaches to solve these problems. The first approach is reinforcement learning(RL). RL uses a reward function to train robots without requiring a human operator. The robot explores the environment while trying to maximize said reward. RL is beneficial because the robot can find a general approach by exploring the whole range of states. In its pure form, RL does not require demonstrations from a human expert. However,

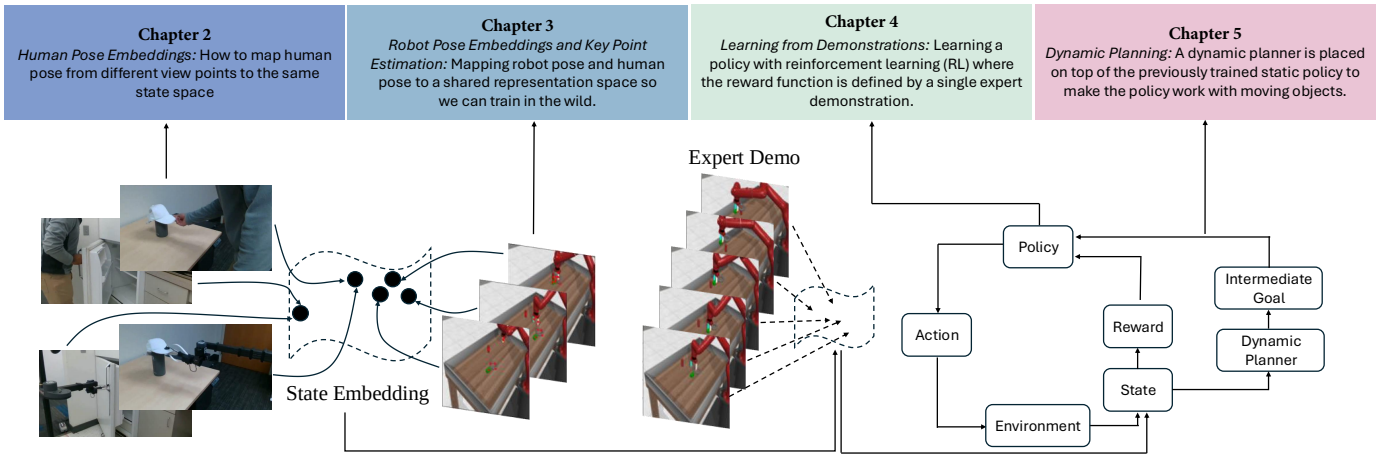


Figure 1.1: A generalized framework for learning robot policies from human videos. On the left we show how human pose [7], robot pose as well as robot key points can be mapped to a state embedding space. We can then retrieve this embedding from an expert demonstration and use that to train a robot policy which can be deployed in the real world via a dynamic planning add on.

RL faces many other issues, such as an inability to effectively explore the entire environment and find success. Additionally, RL still requires a lot of training time on a real robot, and often this training needs to be overseen by a human. Solutions such as training in simulation have been proposed to overcome this challenge, but create a whole new set of problems trying to navigate sim-to-real.

The second approach is imitation learning. Imitation learning can work much quicker than RL, but is known to overfit to the expert demonstrations. Imitation learning approaches are where limited data is most impactful. As a result researchers have focused on how to work around this limitation. For example, one could use data from a different domain, such as a video of a human to do initial training.

In the end, it is likely that researchers will find success by using some combination of these techniques. However, for now it is important to think of what needs to be accomplished in each field so that the pieces are ready to put together.

One important aspect to making both of these methods work is correctly and concisely defining the state of the environment. A common technique in modern approaches is to use an image, or several images, to represent the state. However,

current computer vision techniques to find image features are not applicable to robotics. For example, these features are often invariant to rotation, but the rotation of an object is key to a successful robot maneuver. Even when roboticists have learned their own image features they have struggled to make them generalize outside of the lab environments where the training images were collected. This signals that more work needs to be done to process the images before they can be used as part of the state. On the other hand, preprocessing an image presents its own issues because information will be lost during the processing stage that could be key to the success of the episode.

In Variational View-Invariant Pose Embeddings (V-VIPE) [70], we take a first step toward view-invariant state representation by learning a shared latent space where poses from different cameras are embedded such that proximity in world space corresponds to proximity in latent space. This is important because cameras used at test time are not guaranteed to have the same camera extrinsics as those used during training. This work is done on human poses, but can be easily applied to robot poses as well.

V-VIPE could also be extended to embed both human and robot poses into a shared latent space. From here one could easily pretrain a policy from human demonstrations using the embedding as the state representation. The policy can then be fine-tuned and deployed on a robot with substantially less expert data than would otherwise be required.

V-VIPE employs a generative model to learn pose embeddings, enabling it to generalize more effectively to unseen poses compared to traditional methods. This is especially valuable given the limited size of many human pose datasets, which often leads to overfitting. V-VIPE demonstrates strong interpolation capabilities, easily generating plausible intermediate poses between two given inputs. It also shows signs of extrapolation: adding noise to an embedding and decoding it yields

poses that still adhere to the physical constraints of the human body. As generative techniques continue to improve, future work should explore how to further build on these capabilities.

Beyond representing human state, it's also important to consider how to represent the state of the robot's environment. In Prescriptive Point Priors for Visuo-Spatial Generalization of Robot Policies (P3-PO) [71], we explore effective approaches to this challenge.

As mentioned earlier, training robot policies directly from images often leads to poor generalization. One proposed solution is to use point clouds instead. However, real-world point clouds are often noisy—especially when captured with moving cameras—and processing them is computationally intensive. Moreover, these methods still struggle to generalize across variations in object shape and size. In P3-PO we propose a keypoint based solution that shows generalization to changes in environment as well as changes in the physical appearance of the object.

P3-PO leverages state-of-the-art techniques in semantic correspondence and point tracking to identify a set of pre-labeled keypoints, which are then tracked throughout an episode. These keypoints serve as a compact state representation for the policy. Thanks to this compactness, the policy can achieve high success rates with only a small number of demonstrations. Furthermore, since P3-PO does not rely on visual appearance, it generalizes well across different backgrounds and lighting conditions. Building on this idea, recent work [42] demonstrates that such point-based representations can be used to train policies directly from human video demonstrations, bypassing the need for robot-collected data entirely.

Even with progress in training robots from human videos, data scarcity remains a fundamental challenge. Lessons from other areas of computer science suggest that the volume of data needed for robust generalization is often beyond reach in robotics. This underscores the need for methods that can learn effectively from limited data.

To address this, we introduce Waypoint Exploration from a Single Demonstration (WayEx) [68], which constructs a reward function from a single demonstration without action labels. WayEx not only improves exploration in reinforcement learning but also generalizes across diverse tasks.

A major limitation of current reinforcement learning (RL) methods is their reliance on carefully crafted, task-specific reward functions. These functions are often brittle and difficult to design, diminishing the practical advantages of RL. Alternatively, sparse rewards—triggered only upon task success—can be used, but they make it hard for agents to discover successful behaviors through random exploration. Waypoint Exploration from a Single Demonstration (WayEx) addresses this by using a single, unlabeled demonstration to generate a task-agnostic reward function, enabling application across diverse tasks. While related to inverse reinforcement learning (IRL), which also aims to infer rewards from successful behavior, WayEx is not learning-based and therefore avoids the need for large amounts of expert data. It not only outperforms existing general-purpose RL approaches, but also matches the performance of task-specific methods that require 100 times more expert demonstrations.

Currently, WayEx is constrained by the need for a distance function between states. However, with an appropriate state representation, defining such a function becomes significantly easier. This highlights the importance of methods like V-VIPE and P3-PO. In the Future Work section, we explore how to find the distance between these representations.

Once we develop solutions for simple, static tasks in controlled environments, the next step is adapting these approaches to real-world settings. A critical challenge is moving beyond the assumption that the environment remains static during robot execution. In No-frills Dynamic Planning using Static Planners [67], we show how static policies can be effectively applied to dynamic scenarios without the need for additional training.

This is achieved by predicting the future state of the environment and modifying the policy's input to reflect that anticipated state. To do this, we use an Estimated Time of Arrival (ETA) network to predict how long it will take the robot to reach its goal, alongside a trajectory prediction network that forecasts how objects in the scene will move during that time. This predictive approach enables the robot to perform tasks like trapping a moving ball—something that static replanning alone, without anticipating future dynamics, cannot accomplish.

Chapter 2

Human State Representation

This chapter introduces Variational View-Invariant Pose Embeddings (V-VIPE), a method for learning a latent space that consistently represents human pose across different viewpoints. The core idea is to use a Variational Autoencoder (VAE) to define a smooth latent space with desirable structural properties, and then learn a mapping from observed poses into this space. This approach ensures that poses which are close in 3D space are also close in the latent representation, enabling robust generalization across views. V-VIPE lays important groundwork for human-to-robot learning by aligning pose representations across camera perspectives. Future work will explore extending this alignment across different domains, such as human and robot embodiments.

2.1 Introduction

Learning to represent three dimensional (3D) human pose given a two dimensional (2D) image of a person, is a challenging problem with several important downstream applications such as teaching a person to mimic a video, action recognition and imitation learning for robotics. The key challenge arises from the fact that different camera viewpoints observing the same 3D pose lead to very different projections in a

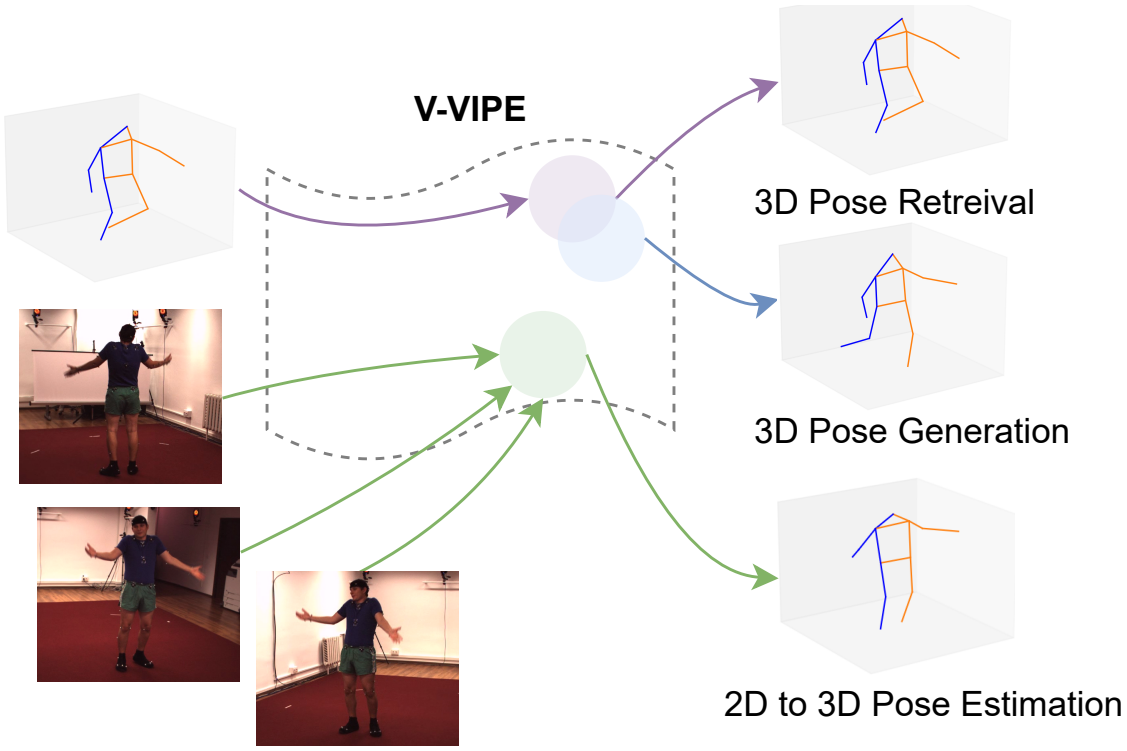


Figure 2.1: The several functions V-VIPE is capable of. The purple path represents 3D pose retrieval. The blue path represents generation by adding noise to the purple path. The result is a variation of the original pose. The green path shows 2D to 3D pose estimation from several viewpoints.

2D image. The common practice is to circumvent this challenge by estimating 3D pose in the camera coordinate space [48, 81, 159]. However, this leads to differences in scale and rotation between the estimated 3D representations from images of the same 3D pose from different camera viewpoints. Without the knowledge of camera parameters, it is not possible to establish correspondence between these 3D representations. This is important as we move towards environments where we have very little control over the camera viewpoint, such as photos taken with a phone or AR glasses. In such scenarios, we can make very few assumptions about the camera space.

In this paper, we address this challenge by separating the problem of estimating 3D pose from 2D images into two steps. First, we learn an embedding to represent 3D poses in canonical coordinate space. Next, we learn to encode 2D poses, from different camera viewpoints, to the embedding from the first step. This leads to a canonical

3D pose embedding that is invariant to camera viewpoints. This view-invariant pose embedding is highly flexible, allowing us to do 3D pose retrieval, 3D pose generation, and most importantly, estimating consistent 3D pose from different 2D viewpoints 2.1.

In our approach we use a variational autoencoder (VAE) to learn an embedding for 3D human poses. This VAE is trained to reconstruct 3D poses and has two key benefits: (a) we can leverage loss functions to ensure similar 3D poses are close in the embedding space, and (b) we learn embeddings that can generalize better to unseen 3D poses due to the variational training paradigm. Next, we learn a mapping from 2D poses (either ground-truth or estimated using off-the-shelf detectors) to this 3D pose embedding space by training a 2D pose encoder that estimates the 3D pose embedding. This embedding is used as input to the pre-trained decoder from the VAE to estimate the corresponding 3D pose, thus leading to "lifting" the 2D pose from different camera viewpoints to 3D [57, 98]. We refer to embedding as variatonal view-invariant pose embedding (**V-VIPE**).

Our proposed V-VIPE is highly flexible and generalizable. We can encode 3D poses and use the embedding for downstream tasks, like retrieval and classification. We can also map 2D poses from unseen camera viewpoints to this embedding. We can estimate 3D poses from these embeddings using the decoder. Finally, we can generate unseen 3D poses. To the best of our knowledge, V-VIPE is the only representation to offer this diversity of applications.

We perform an extensive experimental evaluation over two datasets: Human 3.6M [51] and MPI-3DHP [82]. We show quantitative results on 2D to 3D pose retrieval and qualitative results on 3D pose generation and 2D to 3D pose estimation. We show that V-VIPE performs 1% better than other embedding methods on seen camera viewpoints and about 2.5% better for unseen camera viewpoints. In addition, we show generalization of our approach by training on one dataset and testing on the other.

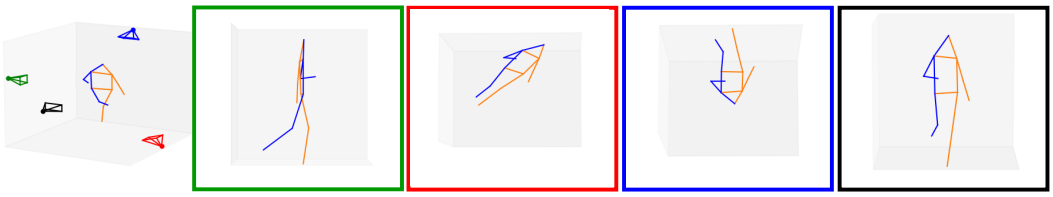


Figure 2.2: On the left we can see the 3D pose in the original global coordinates with 4 different cameras. The next 4 images are the 3D poses as seen from these 4 cameras.

To summarize, our main contributions are as follows:

- We learn a variational view-invariant pose embedding (V-VIPE) by training a VAE to represent 3D poses in canonical coordinate space, which allows it to be camera invariant.
- We propose a model to map from 2D poses to V-VIPE, which enables us to estimate 3D poses of 2D images. Additionally, because V-VIPE is camera invariant, our mapping can generalize to unseen cameras.
- We also estimate and generate 3D poses using V-VIPE via a decoder that can be used for downstream tasks.

In the rest of the paper we expand upon these ideas. We summarize the related works in Section 2.2. In Section 2.3, we describe our proposed method, and Section 2.4 provides the experimental evaluations. Section 2.5 looks at ablations of our method. Finally, Section 2.6 derives the conclusions.

2.2 Related Work

Human Pose Estimation. There are two family of approaches for human pose estimation. One is to directly estimate 3D poses from an 2D images [100, 129], and the other is to lift pre-detected 2D poses to 3D poses [81, 128, 153]. In recent years, state-of-the-art approaches have almost exclusively focused on the lifting strategy.

Our goal is to specifically find correspondence between 2D poses in images from different camera viewpoints without any knowledge of camera parameters or temporal context. Recent works have explored how temporal information can improve 3D pose estimation [19, 157], typically by processing a sequence of images using a transformer [157]. However, our focus is 3D pose estimation using a single 2D image which is similar to [142].

The key distinction between our approach and prior works in estimating 3D poses using 2D images is view-invariant embedding that can be estimated from a monocular viewpoint. Several works have attempted to address view invariant estimation by leveraging many viewpoints [79, 111, 140] because it is much easier to place a person in canonical coordinate space when you have access to many views. However, access to multiple viewpoints of the same scene is an unrealistic assumption in the canonical settings. Therefore, these approaches can only be used in environments that have multiple cameras observing the same scene. In contrast, our approach can be applied to any arbitrary 2D image. Some works only use a single viewpoint during inference time, but still require multiple views for each pose during training [63]. Whereas our method is more flexible and can be trained on any dataset with both 2D and 3D information, even if there is only one camera viewpoint available. Similar to our work, [142] performs view-invariant pose estimation from one view, but their method requires localized transformations that fundamentally change the 3D pose and must be reversed at the end to get the final pose. Our approach, on the other hand, requires only one global rotation to a canonical camera viewpoint that does not change the integrity of the pose.

3D Pose Generation. Training a model capable of generating new 3D poses is important for representing unseen data in addition to training data. There are two main types of generators that can be used, Generational Adversarial Networks (GANs) and Variational Auto Encoders (VAEs).

Several works have used GANs [142, 145, 149] to generate training data for 3D poses. However, they are not well suited for our task which also requires encoding 3D poses in an embedding space. VAEs, on the other hand, are better suited for learning embedding by auto-encoding 3D poses. [97] learns a latent network, where they go directly from 2D to 3D without using the 3D data as input to the model, whereas [122] learns a latent representation using a variant of VAE and generate 3D poses using 2D pose as a precondition to their decoder. [57] employs a basic autoencoder instead of a VAE, which leads to an inconsistent embedding space that is harder to map to 2D inputs. [38] also learns an autoencoder instead of a VAE, but additionally, they choose to regress on the embedding and perform little normalization prior to training which leads to a poorly regularized output space.

2.3 Proposed Method

Our method consists of three main parts. In 2.3.1 we review the input data pre-processing to ensure that the output is independent of camera view. In section 2.3.2, we describe how we define V-VIPE through a VAE model. In section 2.3.3 we cover how we learn V-VIPE from the detected keypoints. The final model is a network that takes as input a single frame monocular image and estimates a view invariant pose, which can be used to compare any two human poses independent of the context of the original image.

2.3.1 Data Processing

Before we pass any data through our model we perform two key steps. First, we modify the global rotation of the image; second, we scale the keypoints so that the original size does not affect the model.

Global Rotation Realignment. Predicting 3D pose in canonical space is extraor-

dinarily difficult as mentioned in [81]. We believe this is mostly due to the global rotation[1] of any 3D pose. Global rotation is hard to estimate due to its ambiguity. We can see in Figure 2.2 that a pose in global space can have a very different appearance in camera space. Without any information, such as a ground truth pose, which we can align the output to or any camera parameters, it would be difficult to determine that any two of these poses are the same.

We argue that global rotation is irrelevant for human pose comparison. Specifically, when we are trying to determine if two poses are the same we do not need to understand how those are oriented in relation to the world they are in. If one pose is facing the x-axis and the other is facing the y-axis, it is still possible that their overall pose is the same. We thus remove rotation dependence by aligning the coordinates of the left hip, right hip and the spine to the same points in every pose of the dataset. This can be visualized in Figure 2.3. In order to achieve such alignment we find the rotation that minimizes the equation:

$$L(C) = \frac{1}{2}\sum_{i=1}^{n} ||a_i - Cb_i||^2 \tag{2.1}$$

where a_1, a_2, a_3 equal the 3D points representing the left hip, right hip and spine respectively and b_1, b_2, b_3 equal $[[0, -1, 0], [0, 1, 0], [0, 0, 1]]$. Aligning to these points causes the hips to align to the y axis and the spine to the z axis. We specifically align the hips because they are in a straight line so it is easy to align to one axis and the spine because it is directly above the root and therefore can be easily aligned to a perpendicular axis. In order to minimize Equation 2.1, we use the Kabsch algorithm [55].

Scaling and Pose Normalization. In this work, we are only concerned with estimating pose such that it is easy to compare how similar two poses are. This is because pose comparison is what is needed for downstream tasks such as action

[1]By global rotation we mean how a human is rotated in relation to the canonical space.

recognition. To account for this, we scale and normalize the input, such that it becomes independent from factors[2] that should not affect the pose similarity estimation.

We use the universal skeleton provided by the dataset to remove the size factor. In this representation all joints are scaled to the same proportions. This makes the size of the 3D output independent of the inputted 2D image or the original 3D pose.

Moreover, to complete the normalization of the data we use a process similar to [16] where we center the root joint and scale all of the other joints, accordingly.

2.3.2 3D Pose VAE

The proposed model consists of two parts, a 3D Pose VAE Network and a 2D Mapping Network. The 3D Pose VAE Network, Figure 2.4.a, consists of an encoder network and a decoder network, which make up the VAE model. To stay consistent with other papers we choose [81] as the backbone for both our encoder and our decoder.

The benefit of using a VAE for the 3D Pose VAE Network is its ability to generalize to new poses. This is because the goal of a VAE is to synthesize unseen poses. Although this is not our main goal, we do want our network to potentially be able to represent unseen poses, which is a realistic setting in real world applications.

Normalizing the rotation, as defined in the step above, helps the VAE by reducing the range of values that the output can be. We want the VAE to learn all possible human poses within the range and by making that range smaller we make it easier to learn an embedding that spans the whole space. If we omit the rotation realignment then our embedding space would have to learn not only joint location in relation to all other joints, but also joint location in relation to the global space. This is in general unnecessary as location in global space is not relevant when comparing if two poses are equal. Additionally, learning a normalized rotation means that the output is all in one space and can be compared easily without additional alignments.

[2]Intuitively, two people are capable of being in the same pose no matter their height or weight.

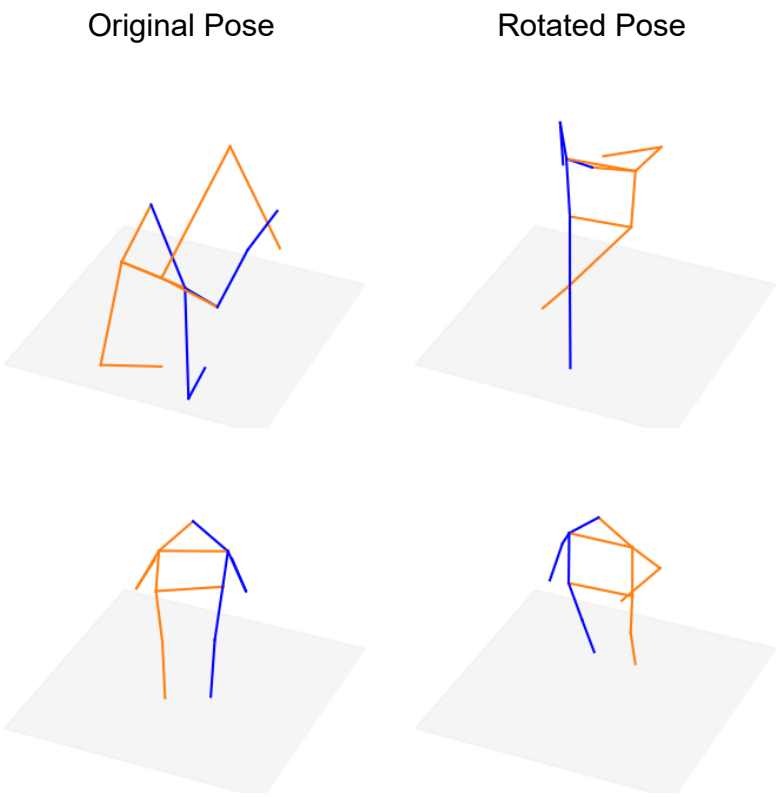


Figure 2.3: How poses change when we align the points and modify the rotation. On the left is the original pose and on the right is the pose after we have rotated it.

The 3D Pose VAE Network has two parts: (i) an encoder, which takes as input a 3D pose, $S_{3D} = \{s_i \in \mathbb{R}^3 | i = 1 \dots N\}$, where N is the number of keypoints, and outputs a mean for possible embeddings, $\mu_e \in \mathbb{R}^n$, and a variance for the embedding, $\sigma_e \in \mathbb{R}^n$. Using these values and a Gaussian distribution prior we take a sample, e. We denote the distribution of the latent space modeled by the encoder with $q(e|S_{3D})$; (ii) a decoder, which takes in input an embedding, e, and outputs an estimation of 3D pose $\hat{S}_{3D} = \{\hat{s}_i \in \mathbb{R}^3 | i = 1 \dots N\}$. The distribution of the decoder is represented as $p(S_{3D}|e)$.

The goal of the 3D Pose VAE Network is to find a V-VIPE space that is representative of the entire range of 3D human poses for a specific scale and normalization. A feature of the 3D Pose VAE Network should be that poses that are close together in

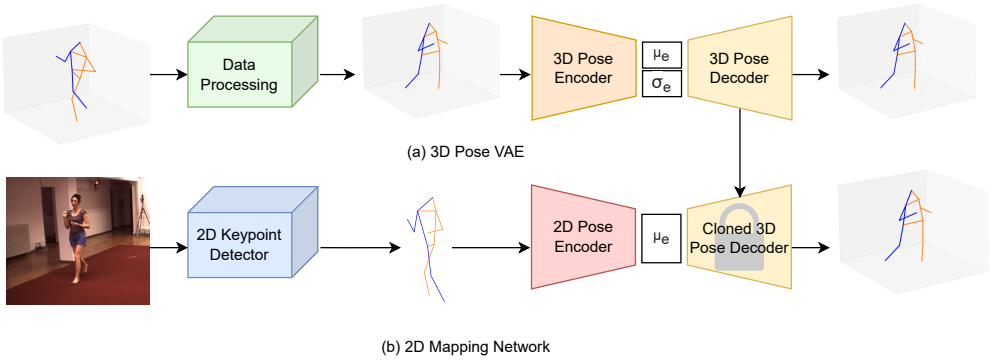


Figure 2.4: The network on top is our "3D Pose VAE Network." First we pass the 3D input through our data processing phase. Once we have the output we can pass that as input to our VAE network, which generates V-VIPE and then attempts to reconstruct the pose. On the bottom is our "2D Mapping Network." 2D keypoints are extracted using a detector. We then pass these through our 2D encoder and then a locked clone of the decoder network from the 3D Pose VAE Network. This reconstructs the original 3D pose.

the original 3D space are close together in the embedding space. An important part of learning an accurate mapping from 2D space is that even if there is a slight error in the V-VIPE estimation the output will still be a pose that is similar to the original 3D pose. Additionally, defining a smooth space for V-VIPE enables us to interpret if two poses are close together in 3D space by observing if they are close together in the embedding space.

We define a distance function, D, which represents the Mean Per Joint Position Error (MPJPE). MPJPE measures the distance between two 3D points by taking the $L2$ distance between each joint location and then computing the mean of those distances for all joints.

During training we thus optimize for three factors:

- A reconstruction loss, which is equivalent to the Mean Squared Error (MSE) loss between S_{3D} and $\hat{S}_{3D}$. $L_{\text{mse}} = \frac{1}{N}\sum(S_{3D} - \hat{S_{3D}})^2$
- The KL Divergence loss $L_{\text{KL}} = \text{KL}[q(z|S_{3D})|p(z)]$. This loss represents the distance between the distribution of the encoder and the prior distribution, $p(z)$. In this work we use a Gaussian distribution as the prior.

- The third is a triplet loss. To compute the triplet loss we first find the 3D distances, $D_{i,j}$ within a batch between all elements. For each pose we then set the closest pose in the batch to be the positive example (j) and the second closest pose to be the negative example(k). We make sure the positive and negative poses are at least .1 apart from each other and if they aren't we select the next closest pose as the negative example. We do this because we want the examples to be hard, but not too hard that they introduce noise. We compute triplet loss between i, j and k by doing $L_{\text{triplet}} = \max[0, D_{i,k} - D_{i,j} + m]$, where m is our margin. This loss is useful because it causes similar poses to move closer together in the embedding space.

This makes the overall loss function to train the 3D Pose VAE:

$$L_{\text{V-VIPE}} = L_{\text{mse}} + L_{\text{triplet}} + L_{\text{KL}} \tag{2.2}$$

2.3.3 2D Mapping Network

Once we have trained the 3D Pose VAE Network we utilize its embedding space to learn a 2D Mapping Network (see Figure 2.4.b). In particular, we take the 3D Pose VAE Network decoder model and we freeze it so that it translates from the pre-defined V-VIPE space to 3D coordinates. Next, we train a new encoder Enc_{2D} for 2D coordinates. The new encoder takes in input $S_{\text{2D}} = \{p_i \in \mathbb{R}^2 | i = 1...N\}$ and outputs a V-VIPE, $e \in \mathbb{R}^n$. We pass e through the frozen decoder to get what the embedding represents in 3D space according to the model trained in the previous phase, $\hat{S}_{\text{3D}} = \{p_i \in \mathbb{R}^3 | i = 1...N\}$.

To train the 2D Mapping Network we use two losses. Given the input, S_{2D}, the output $\hat{S}_{\text{3D}}$ and the ground truth 3D keypoints, S_{3D}, we compute $MSE(S_{\text{3D}}, \hat{S}_{\text{3D}})$. We combine this loss with a triplet loss, which we compute similarly as in Section 2.3.2. The main difference is that we use the output from the 2D encoder and the ground

truth 3D keypoints. We then back-propagate this loss through the whole network, but do not apply the gradient losses to the decoder network. This is because we do not want to change the embedding space, but we just want to train the 2D encoder to make it compliant with the latent space.

We find that it is beneficial to pre-train the decoder as described in 2.3.2 because we want to construct a space for V-VIPE that is smooth, without also needing to learn a 2D to 3D mapping. Because we train our 3D Pose VAE on normalized 3D poses it will only learn how to map to a normalized pose. Therefore the output of the 2D Mapping Network is also normalized. This means the output is rotation and scale invariant, making it easy to compare 2D poses from different camera viewpoints.

2.4 Experiments and Results

2.4.1 Experimental Setup

The model uses a backbone network described described in [81]. We stack 2 blocks of this network together for both the encoder and the decoder network of both the 3D Pose VAE Network and the 2D Mapping Network. We set the linear size to 1024, and we use a 0.1 dropout. The dimension of a V-VIPE is 32 and the margin for the triplet loss is 1.0. Any 2D keypoint detector could be used, but we chose AlphaPose [28, 72, 73, 146]. We use COCO keypoints because they are widely used for 2D detectors. We implemented the model in PyTorch and we trained it on 1 GPU.

2.4.2 Metrics

We evaluate the model using two metrics. The first is a hit metric, inspired from [142], which we use to measure how often we are able to retrieve a pose that is similar to a query pose. Given two normalized keypoints S^i_{3D} and S^j_{3D} we first apply a Procrustes alignment [119] between the two to get $A(S^i_{3D})$ and $A(S^j_{3D})$. Given a dataset with

Table 2.1: Hit metric results for different values of k. The upper part of the table shows the Hit metrics when using ground truth (GT) keypoints. The bottom part of the table shows the metrics when using keypoint detection(D) and augmentation(A). For Pr-VIPE and our method AlphaPose is the keypoint detector. Epipolar Pose uses its own detector. The $*$ version of Epipolar Pose is trained on the Human3.6 dataset and the # version is trained on the 3DHP Dataset. Epipolar pose does not generalize to unseen datasets.

Dataset →	H3.6M			3DHP (All)			3DHP (Unseen)		
k →	1	10	20	1	10	20	1	10	20
PR-VIPE (GT)	**97.6**	**99.9**	**100.0**	42.6	72.8	79.1	43.7	73.2	82.0
Ours (GT)	89.7	98.8	99.4	**45.3**	**76.2**	**83.1**	**47.9**	**77.9**	**84.5**
2D keypoints	28.7	47.1	50.9	9.80	21.6	25.5	-	-	-
Epipolar Pose*	69.0	89.7	92.7	-	-	-	-	-	-
Epipolar Pose$^\#$	-	-	-	24.6	53.2	61.3	-	-	-
PR-VIPE (D)	**72.1**	**94.3**	**96.8**	17.9	44.7	64.1	19.2	46.6	55.6
PR-VIPE (D + A)	70.9	93.1	96.0	25.4	55.6	64.1	27.8	57.7	65.8
Ours (D)	70.0	92.7	95.6	23.5	54.3	64.0	26.2	57.0	66.4
Ours (D + A)	69.0	93.5	96.3	**26.9**	**59.0**	**68.2**	**30.1**	**61.6**	**70.3**

many views we select two camera views. We find all embeddings for the 2D poses from the selected cameras. Then, we query each embedding from camera 1 and find the k nearest neighbors from the set of embeddings for camera 2. We consider a pair of embeddings a hit if their original 3D pose satisfies $\text{MPJPE}(A(S^i_{\text{3D}}), A(S^j_{\text{3D}})) < .1$. We report Hit@k for k=1,10,20 and average over all pairs of cameras. This metric represents view invariance because it shows how well we can match poses from one viewpoint to similar poses from another viewpoint.

The second is the Mean Per Joint Position Error (MPJPE), which we define in Section 2.3.2. This error is used to determine the distance between two sets of 3D keypoints.

2.4.3 Datasets

In all the experiments we train on the standard training set of the Human3.6M dataset (H3.6M) [51]. For our hit metric we use the test set of H3.6M as the validation set and show results on the MPI-INF-3DHP dataset [82] (3DHP).

Human3.6M. The H3.6M dataset [51] contains 3.6 million human poses taken from 4 different cameras. All of these cameras are at chest level. The standard training set for this dataset is made up of subjects 1,5,6,7 and 8. The standard test set contains poses from subjects 9 and 11. For the evaluation of the hit metric, we follow the method described in [128], where they remove poses that are similar.

MPI-INF-3DHP. 3DHP [82] contains 14 different camera angles. For our tasks we remove the overhead cameras, which leaves us with 11 cameras. Of these cameras, 5 are at chest height and the others have a slight vertical angle. This dataset is used to show whether or not our method will generalize to data that is different from the training data.

2.4.4 Augmentation

In order to improve the model's ability to generalize we introduce camera augmentation similar to the work done in [128]. To calculate this augmentation we take the ground truth 3D pose and randomly rotate it. We then project this pose into 2D. We add augmented poses to each of our batches during training time. We found that it was best to add augmented poses for half of the poses in each batch.

2.4.5 Quantitative Results

Similar Pose Retrieval Experiments. We compare our model for hit metrics against 3 baselines. The first baseline is the PR-VIPE model, which attempts to define an embedding space without reconstructing the 3D pose; we adopted their open source code and re-trained their model so we would have results on the same 2D pose detector, i.e., AlphaPose. The second baseline is simply finding the nearest neighbor of the detected 2D keypoints. The third baseline uses Epipolar Pose [63] to detect 3D keypoints. In this case, Procrustes alignment is performed between all poses and the closest aligned pose is selected as the match.

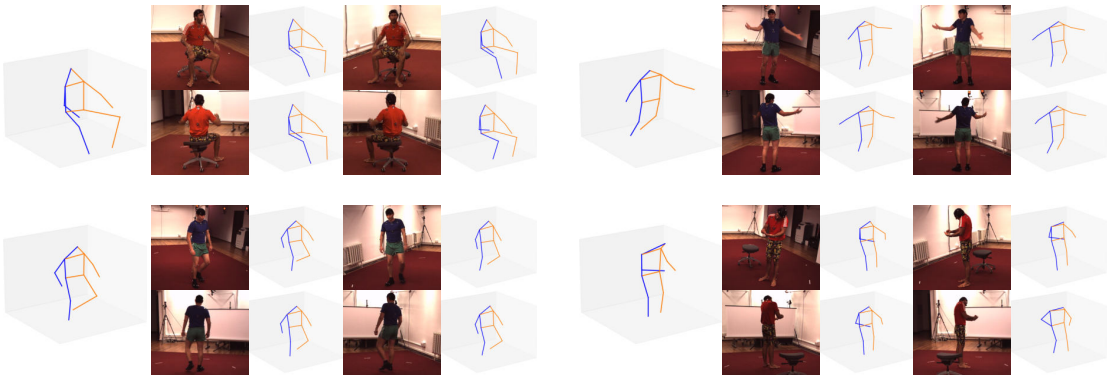


Figure 2.5: Pose Estimation from 2D images of our model applied to different camera viewpoints. We show 4 sets of results. The ground truth is on the left hand side of each example, while on the right we provide the 4 original views as well as our model 3D output for each view.

We show the hit metrics for the different k values in Table 2.1. The top section of the table shows the results of our method and of PR-VIPE when trained and tested with ground truth (GT) 3D keypoints. The left part of the table reports the results on the test set of H3.6M. We can see that our approach is slightly worse than the PR-VIPE approach. This is because we are testing on very similar data to the original training set. Our model, however, is designed to generalize. The generalization of the model is demonstrated in the middle part of the table, where we report the performance on the 3DHP dataset when considering all available cameras. In this case, our model gets higher values for all values of k. Moreover, when we pair one chest camera with a camera that is not at chest height, i.e., unseen cameras with respect to the training data(right part of the table), we can see that the gap is even larger. For example, when considering $k = 1$, the gap between the two models is about 4.2 percent for unseen cameras and 2.7 percent for all cameras. This demonstrates that the latent space we acquired during the VAE training is able to generalize to unseen camera viewpoints better than existing models.

In the bottom section of the table, we show results when the keypoints are automatically detected(D). For PR-VIPE and our model we use AlphaPose. Epipolar Pose detects its own keypoints. Again our method outperforms the PR-VIPE model

Figure 2.6: This figure demonstrates what a query and retrieval look like. On the left of each pair of images is the query pose and the image on the right is the image that is considered the closest match by our model. Each pair of images is labeled with the MPJPE between the two poses. Its easy to see that some poses, such as the one on the far left, are easy to retrieve because they are so distinct. And others, such as the one on the far right, have occluded points as well as other factors that make the nearest neighbor hard to find.

when generalizing to data different from the training set, 3DHP, as well as to unseen cameras. For example, when $k = 10$ our method outperforms PR-VIPE by about 5.6 percent for all 3DHP cameras, and by about 7 percent for the unseen category.

In this section we also show results for detected keypoints plus additional training data generated by augmenting the 3D poses. We see an increase from just our detection model for 3DHP because we have introduced new camera viewpoints to the training data. We see an improvement over PR-VIPE when they use augmented data, although we do not get as much of a boost from augmentation because our model already generalizes better than theirs. For $k = 1$ our model outperforms theirs by 1.5 percent.

Additionally, in the table we report the 2D keypoints and Epipolar Pose results. We can observe that using the 2D keypoints is not effective, as demonstrated by the low hit metric for all k values. The Epipolar Pose$^{\#}$ method performs better than both our method and the PR-VIPE method before any augmentation is applied to the data because it is trained on the 3DHP dataset and does not need to generalize. When you try to run the Epipolar Pose* model on 3DHP data the output does not resemble human pose. We do not report generalized results for Epipolar pose because of this. Despite the fact that Epipolar Pose$^{\#}$ is trained specifically for detection on the 3DHP dataset when we add augmentation of the data to our model we are able to beat their results by about 2 percent.

3D Pose Estimation Experiments. In addition to calculating the hit metric

described above our model also outputs the predicted 3D pose. We find that the average error of this model is 62.1 millimeters. We calculated this number using a model trained on keypoints detected by the Cascaded Pyramid Network [17] as this is commonly [30, 100, 142] used for 3D Pose Estimation. We find that while this number is not competitive with current methods for pose estimation that use more complex models or take in more information, such as sequences, it is similar to the error found in [81], which we use as the backbone for our network.

2.4.6 Qualitative Results

2D to 3D Pose Estimation. Figure 2.5 shows examples of our 3D estimations given a 2D image as input. We show examples of 4 different poses each with 4 different camera angles. In the two examples on the left we have very accurate retrievals. All of the cameras have similar retrievals that allow us to determine that the person is in the same pose despite the very different original camera angles. The examples on the right are the ones where our model struggles to find the whole pose. In the example on the top we are able to find the hand position because the hands are visible in every image, however our model struggles to detect that the body is slightly angled. This is likely because the difference in 2D keypoints between an angled and not angled body are very small and our 2D keypoint detector is not accurate enough. In the example on the bottom our model succeeds with the arms, except for one camera viewpoint where the arm is not visible in the image at all. The other way our model struggles is with the head tilt. This is likely because this is difficult to visualize from most camera angles.

3D Pose Retrieval. We show how our model is able to retrieve similar poses from different view points. In Figure 2.6 you can see the query pose as well as the pose that is retrieved from a different view point. Ideally, the two poses will be identical. This is the visualization of what the Hit metric represents. If the queried pose is sufficiently close to the retrieved pose then we have a hit.

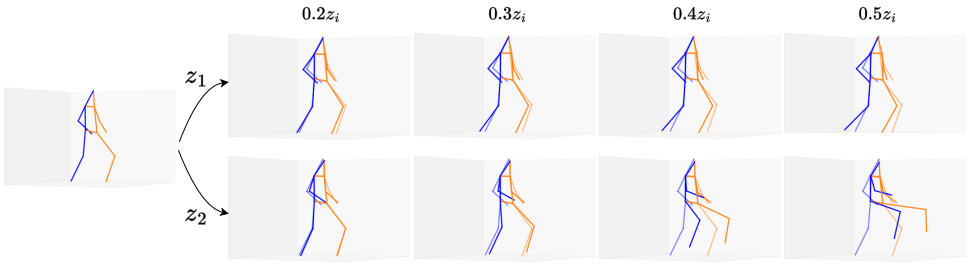


Figure 2.7: Pose generation, starting from a 3D pose we select two random noise directions z_i and generate poses using increasing magnitudes of noise αz_i, where $\alpha \in \{0.2, 0.3, 0.4, 0.5\}$. V-VIPE leads to smooth pose variations and can be used to generate unseen 3D poses.

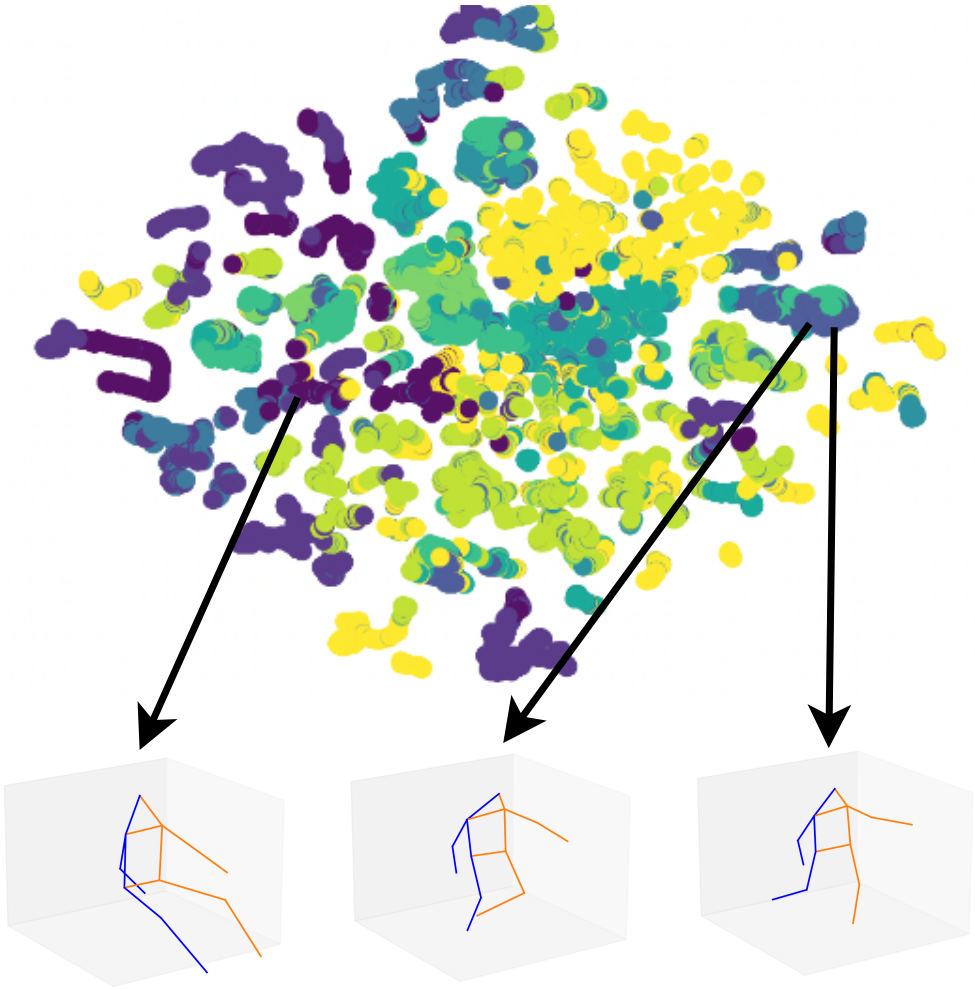


Figure 2.8: t-SNE visualization of the V-VIPE space of our model for poses in the H3.6M dataset. Each color represents similarity to one of 10 "key" poses that we selected. In the expansion, three different poses and their place in the visualization are shown.

Visualizing V-VIPE. Figure 2.8 shows a t-SNE visualization, which we use to show the smoothness of the learned V-VIPE space, where each dot represents a V-VIPE. In order to properly show the clustering we select 10 visually different 3D poses and color our visualization based on which of the 10 poses is the most similar to the pose that each point represents. It is easy to see from this graph that similar colors are typically found in clusters. This means that the space well represents the notion of similarity between poses. We can see this even clearer in the expansion of the visualization where we show three poses and their locations in the cluster. The two poses on the right are colored the same and are very close together. These are slightly different, but the overall pose is very similar. We then select a point that is very far away and here we can see that the pose is quite different.

3D Pose Generation. Our model is able to generate new poses by adding noise to the embedding space of an existing pose. In Figure 2.7 we define a noise array z and add it to an embedding with increasing magnitudes. The pose continues to move in one direction as we increase magnitude showing that our embedding space is smooth.

2.5 Ablation Study

We performed an ablative analysis in order to understand which of our design choices best contributed to our results. **Triplet Loss.** First we examine how important it is that we include the triplet loss term in our method. We remove it from the loss term and find that the new Hit@1 value is 17.41 with no augmented data. This is a drop of 6.1 from the Hit@1 value when triplet loss is included. Therefore the triplet loss value is important to the overall loss term.

Data Processing. We examine how important it is for us to rotate the 3D pose before training on our model. This step is important because it enables us to compare the similarity of poses with two different global rotations without needing to do a

time consuming Procrustes Alignment between every pair of poses. We find that the Hit@1 value on 3DHP with no augmentation obtained when using non rotated points is 18.0 percent, a 5.5 percent decline from our approach.

Pretraining the Decoder. Finally, we studied whether or not pretraining a VAE and using a defined embedding space contributed to our final hit metric. We found that the Hit@1 value for the model with no pretraining is 23.4 versus the 23.5 we obtained by completing the pretraining step. However, this step is important anyways because it enables the model to do 3D Pose Retrieval. Without it we would not be able to map our 3D poses to our embedding space. Therefore we would not be able to generate similar poses to a given 3D pose or query a 3D pose to find a similar 2D pose from a set of images.

2.6 Conclusion

In this work we showed that by using only 3D poses to define a V-VIPE space we can define a better camera invariant space than if we were to only use 2D poses. We defined a procedure made of two steps: first we train a VAE model to learn a latent space of 3D poses; then, we train a 2D keypoints encoder that is linked to the VAE decoder to allow 3D reconstructions of 2D images. We adopted a VAE model as it creates a smooth latent space that can generalize better towards unseen poses during training. In order to achieve this goal, we train a VAE with a three component loss function. We performed an extensive experimental evaluation, by using two datasets, i.e., Human3.6M and MPI-INF-3DHP. We demonstrated that the latent space is modeling a meaningful notion of similarity of the embeddings. This is reflected in the Pose Retrieval experiments where we improve about 2.5 percent in the Hit@1 metric when considering unseen cameras. We also showed qualitative examples demonstrating the capability of our embedding space to capture the notion of similarity of poses.

This is important in downstream tasks. In the future we believe that this approach has a lot of promise for application to downstream tasks such as action segmentation and detection.

Acknowledgements: This work was partially supported by NSF CAREER Award (#2238769) to AS and the DARPA SAIL-ON (W911NF2020009) program.

2.7 Supplementary

Included in the supplementary material are several extra pages of results. In Figure 2.9 we include several results that show the effect of adding noise to an embedding for a pose. In Figure 2.10 we show that by finding embeddings in between two poses we can generate way point poses. Figure 2.11 shows the results of querying several images.

Figure 2.9: In this figure we show what happens when we add random noise to the embedding space for a pose. For each original pose we sample noise and then add it to the original embedding in larger magnitudes. The smallest magnitude is on the left and we increase the magnitude of noise as the images move to the right.

Figure 2.10: Given the frame on the left and the frame on the right for each example we show our models ability to generate frames in between two poses. We take the mean for the embedding space of two poses, calculate the distance between those two means and then add a portion of the distance to the embedding on the left at each step. The images seen in the middle three columns are all generated. We can see that we are able to generate a sequence of poses that lead from one random pose to another.

Figure 2.11: Several retrievals from the 3DHP dataset. On the left is the query pose and on the right is the retrieved pose. On top of each pair of images is the distance between the two poses.

Chapter 3

Robot State Representation

This chapter introduces Prescriptive Point Priors for Visuo-Spatial Generalization of Robot Policies (P3-PO), which proposes a novel state representation based on keypoints and an effective algorithm for extracting this representation in unseen environments. Keypoints offer a robust representation for robotics, as they are not directly tied to raw visual features and are therefore less susceptible to issues like overfitting to background textures or lighting conditions. Future work will explore how this representation can be integrated with other approaches to enable more effective learning and generalization in real-world settings.

3.1 Introduction

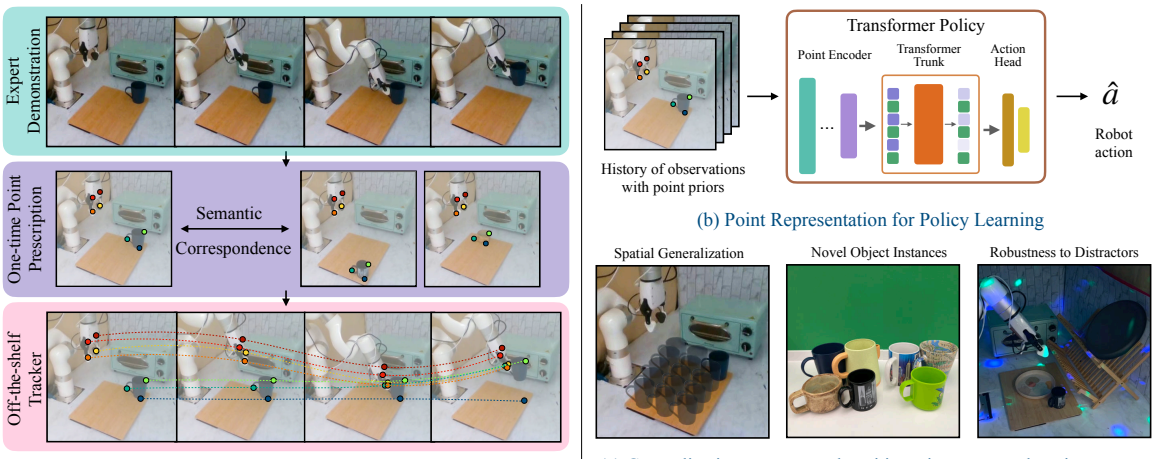
A long standing goal in robotics has been to develop robot policies that are robust to environmental changes and can operate across variations in spatial configurations and object instances. While significant advances have been made in this direction for computer vision [12, 116] and natural language processing [2, 110, 135], the majority of robot policies remain confined to controlled laboratory environments with carefully designed settings. Robotic policies struggle to generalize to real-world scenarios because of the challenges and high costs associated with gathering diverse, high-quality

Figure 3.1: A human prescribes key points one time for one instance of an object and those points are transferable to all other instances of the same object.

robotic data.

Recent efforts aim to address this data problem by either aggregating existing robot datasets under a common framework [94] or collecting extensive real-world datasets through easy-to-use teleoperation tools [21, 27, 120, 156]. However, aggregated datasets suffer from inconsistencies across actions recorded for different projects [94], while large teleoperated collections, though useful, are often specific to a single robot type [120, 156] and it is unclear whether these approaches would scale for different robot morphologies. Consequently, developing generalized robot models still largely depends on collecting more expert demonstrations.

One way to get around this data problem is to use strong representation priors



(a) Prescriptive Point Prior Annotation for Robot Data

(b) Point Representation for Policy Learning

(c) Generalization across novel positions, instances and environments

Figure 3.2: Overview of the Prescriptive Point Priors for Policies (P3-PO) framework. (a) A human annotator prescribes a set of semantically meaningful key points on a **single demonstration frame**, typically in under 5 seconds. Off-the-shelf computer vision models are then used to automatically propagate these key points throughout the entire dataset without further human input. (b) The derived key points are leveraged by a transformer policy to predict the action. (c) P3-PO enables learning policies with improved generalization capabilities, including spatial generalization (i.e. generalization to new locations), generalization to novel object instances, and robustness to background distractors. P3-PO combines the strengths of vision and policy prediction methods through simple yet effective human-prescribed semantic guidance.

that transfer across scenarios and feed these representations as input into existing policy architectures. While priors such as object proposals [125, 160, 161] and pose estimation [96, 139] have been used in prior work, they often lose information which makes policy learning harder or require accurate modeling of the object poses to make the policy work. In this work, we explore if there exists a representation that is flexible, serves as a strong prior, and can provide the object-centric abstraction of a scene to enhance generalization. Compared to segmentation and object models, a point-based representation retains fine-grained spatial information without requiring accurate modeling of object boundaries or poses. By representing objects and scenes as a set of unstructured points, these representations extract only the essential geometric relationships between relevant elements in the scene. This allows the policy to focus exclusively on the key spatial interactions.

We present Prescriptive Point Priors for Policies or P3-PO, a novel framework

that leverages the generalization capabilities of state-of-the-art computer vision models alongside state-of-the-art robot policy architectures. Through P3-PO, we demonstrate improved spatial generalization, the ability to generalize to novel object instances, and robustness to large environmental changes. P3-PO is built on three key ideas. First, a human annotator prescribes a set of semantically meaningful points on a single demonstration frame, a process that often requires less than 5 seconds. Second, a diffusion-based visual correspondence model [131] and a state-of-the-art point tracker [56] are used to seamlessly transfer these points to the entire dataset without further human involvement. Third, the derived points are combined with state-of-the-art policy architectures like BAKU [41] for learning robust robot policies. The novelty of P3-PO lies in strategically combining the strengths of vision models and policy methods, yielding a simple yet effective approach for generalizable robot learning.

We demonstrate the effectiveness of P3-PO through experiments on four real-world tasks in an xArm Kitchen environment. Our main findings are summarized below:

1. P3-PO exhibits an overall 43% improvement over prior state-of-the-art policy learning algorithms across 4 real world tasks (Section 3.5.5).

2. P3-PO generalizes to novel object instances, exhibiting a 58% improvement on a set of held-out objects as compared to prior work (Section 3.5.6).

3. Policies trained with P3-PO are robust to the presence of background distractors (Section 3.5.7) and work with both true depth and predicted metric depth from state-of-the-art models like Depth Anything [147, 148] (Section 3.5.8).

All of our datasets, and training and evaluation code are publicly available. Videos of our trained policies can be seen here: point-priors.github.io.

3.2 Related Works

3.2.1 Imitation Learning (IL)

Imitation Learning (IL) [50] refers to training policies with expert demonstrations, without requiring a predefined reward function. In the context of reinforcement learning (RL), this is often referred to as inverse RL [1, 90], where the reward function is derived from the demonstrations and used to train a policy [43, 44, 47, 69, 87]. While these methods reduce the need for extensive human demonstrations, they still suffer from significant sample inefficiency. As a result of this inefficiency in deploying RL policies in the real world, behavior cloning (BC) [107, 115, 118, 134] has become increasingly popular in robotics. Recent advances in BC have demonstrated success in learning policies for both long-horizon tasks [18, 80, 123] and multi-task scenarios [13, 14, 41, 94]. However, most of these approaches rely on image-based representations [13, 20, 41, 53, 94, 155], which limits their ability to generalize to new objects and function effectively outside of controlled lab environments. In this work, we propose P3-PO, which attempts to address this reliance on image representations by directly using points priors as an input to the policy instead of raw images. Through extensive experiments, we observe that such an abstraction helps learn robust policies that generalize across varying scenarios.

3.2.2 Object-centric Representation Learning

Object-centric representation learning aims to create structured representations for individual components within a scene, rather than treating the scene as a whole. Common techniques in this area include segmenting scenes into bounding boxes [23, 26, 29, 80, 161] and estimating object poses [136, 138]. While bounding boxes show promise, they share similar limitations with non object-centric image-based models, such as overfitting to specific object instances. Pose estimation, although

less prone to overfitting, requires separate models for each object in a task. Another popular method involves using point clouds [9, 160], but their high dimensionality necessitates specialized models, making it difficult to accurately capture spatial relationships. In contrast, P3-PO leverages point prescription, eliminating the need to learn a representation because it is predefined by a human. This enables zero-shot generalization to both new objects and new spatial configurations. Similar to our method prior work [54] uses correspondence to identify where to interact with an object, however, this method relies on anygrasp [29], which is limited to a certain set of objects. Additionally, this method requires learning the affordance of an object, which can introduce errors that are less likely with point prescription.

3.3 Background

3.3.1 Behavior Cloning

Behavior cloning [106, 121] aims to learn a behavior policy π^b given access to either the expert policy π^e or trajectories derived from the expert policy $\mathcal{T}^e$. This work operates in the setting where the agent only has access to observation-based trajectories, i.e. $\mathcal{T}^e \equiv \{(o_t, a_t)_{t=0}^{T}\}_{n=0}^{N}$. Here N and T denote the number of demonstrations and episode timesteps respectively. We choose this specific setting since obtaining observations and actions from expert or near-expert demonstrators is feasible in real-world settings [52, 156] and falls in line with recent work in this area [20, 22, 41, 66, 156].

3.3.2 Semantic Correspondence

Finding corresponding points across multiple images of the same scene is a well-established problem in computer vision [76, 164]. Correspondence is essential for solving a range of larger challenges, including 3D reconstruction [58, 83], motion tracking [24, 45, 56, 158], image registration [164], and object recognition [61]. In

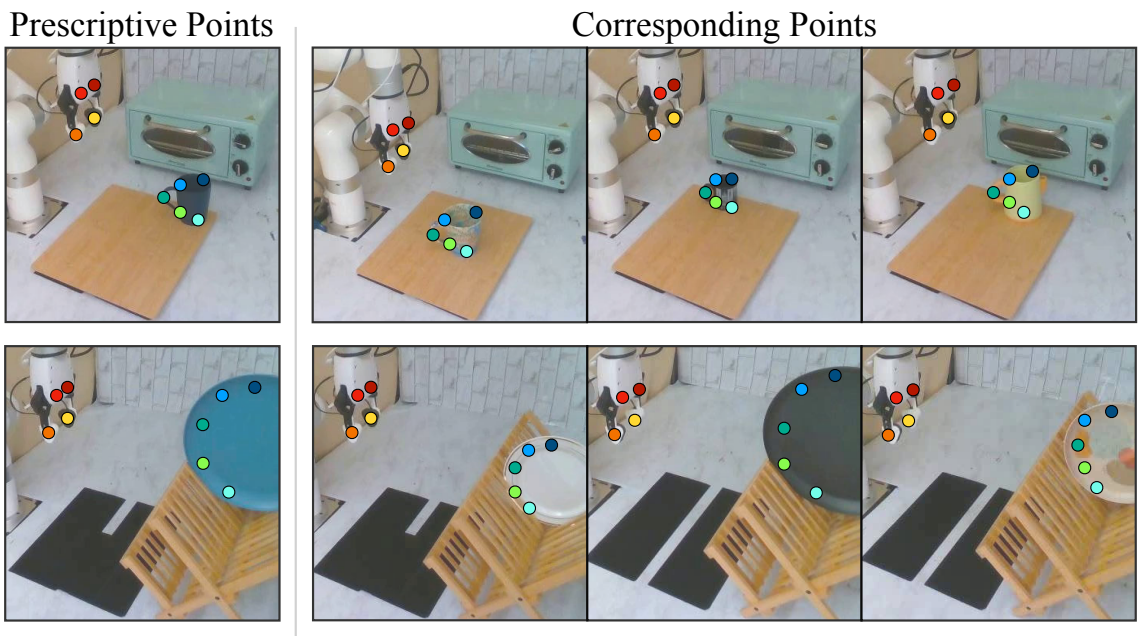


Figure 3.3: Results of the correspondence model when used on the pick mug and plate off rack tasks. On the left is the frame that is annotated by a human. On the right we show that semantic correspondence [131] is able to identify the same points across a variety of instances of each object.

contrast, semantic correspondence focuses on matching points between a source image and an image of a different scene (e.g., identifying the left eye of a cat in relation to the left eye of a dog). Traditional correspondence methods [76, 164] often struggle with semantic correspondence due to the substantial differences in features between the images. Recent advancements in semantic correspondence utilize deep learning and dense correspondence techniques to enhance robustness [37, 39, 49] across variations in background, lighting, and camera perspectives. In this work, we adopt a diffusion-based point correspondence model, DIFT [131], to establish correspondences between a reference and an observed image, which is illustrated in Figure 3.3.

3.3.3 Point Tracking

Point tracking across videos is a problem in computer vision, where a set of reference points are given in the first frame of the video, and the task is to track these points

across multiple frames of the video sequence. Point tracking has proven crucial for many applications, including motion analysis [3], object tracking [150], and visual odometry [92]. The goal is to establish reliable correspondences between points in one frame and their counterparts in subsequent frames, despite challenges such as changes in illumination, occlusions, and camera motion. While traditional point tracking methods rely on detecting local features in images, more recent advancements leverage deep learning and dense correspondence methods to improve robustness and accuracy [45, 56, 158]. In this work, we use Co-Tracker [56] to track a set of reference points defined in the first frame of a robot's trajectory. These points tracked through the entire trajectory are then used to train generalizable robot policies for the real world. This can be visualized in Figure 3.2.a.

3.4 Prescriptive Point Priors for Policies (P3-PO)

Given demonstrations for robot manipulation tasks that cover a small set of possible object configurations and types, we seek to learn a generalizable robot policy that is robust to significant environmental variations and applicable to diverse object locations and types. To achieve this, we introduce P3-PO, an algorithm that decouples perception and planning to promote generalization. P3-PO operates in two phases. First, given a small set of robot demonstrations, the user annotates a single task frame with a set of semantically meaningful points. These reference points are propagated to the rest of the dataset using a combination of semantic correspondence and point tracking. The points obtained are fed into a transformer-based policy model for action prediction. An overview of our method has been provided in Figure 3.2. Below, we describe each component in detail.

3.4.1 One-Time Point Prescriptions

Our method begins by collecting robot demonstrations for a task through robot teleoperation [52]. The user then randomly selects one demonstration and annotates semantically meaningful points on the first frame that are relevant to performing the task, such as points on the robot and the objects being manipulated. This process often requires less than 5 seconds. These user-annotated points serve as priors for the rest of the data generation process. Using an off-the-shelf semantic correspondence model called DIFT [131], we transfer the points on the first frame to the corresponding locations on the first frames of all other demonstrations in the dataset. This allows us to initialize the key points across the entire dataset without additional human effort. For each demonstration, we then employ an off-the-shelf point tracking algorithm, Co-Tracker [56], to automatically track the initialized key points through the entire trajectory. In this way, by leveraging existing vision models for correspondence and tracking, we can efficiently compute the key points on every frame in the dataset while requiring the user to only annotate a single frame. This process, visualized in Figure 1, takes advantage of large, internet-scale pre-training of the vision models to generalize to new object instances and scenes without additional training. We prefer point tracking over correspondence at every frame due to its faster inference speed and ability to handle occlusions by continuing to track points. During inference, DIFT is used to mark the corresponding key points on the first frame, followed by Co-Tracker tracking the points during execution.

3.4.2 Policy Architecture

We employ the BAKU [41] architecture for policy learning. Instead of feeding in raw images, we use points derived from the previous section as input to the policy. Each 2D image point is first back-projected to 3D using the depth information from the camera. The points are flattened into a single vector in the order in which they are

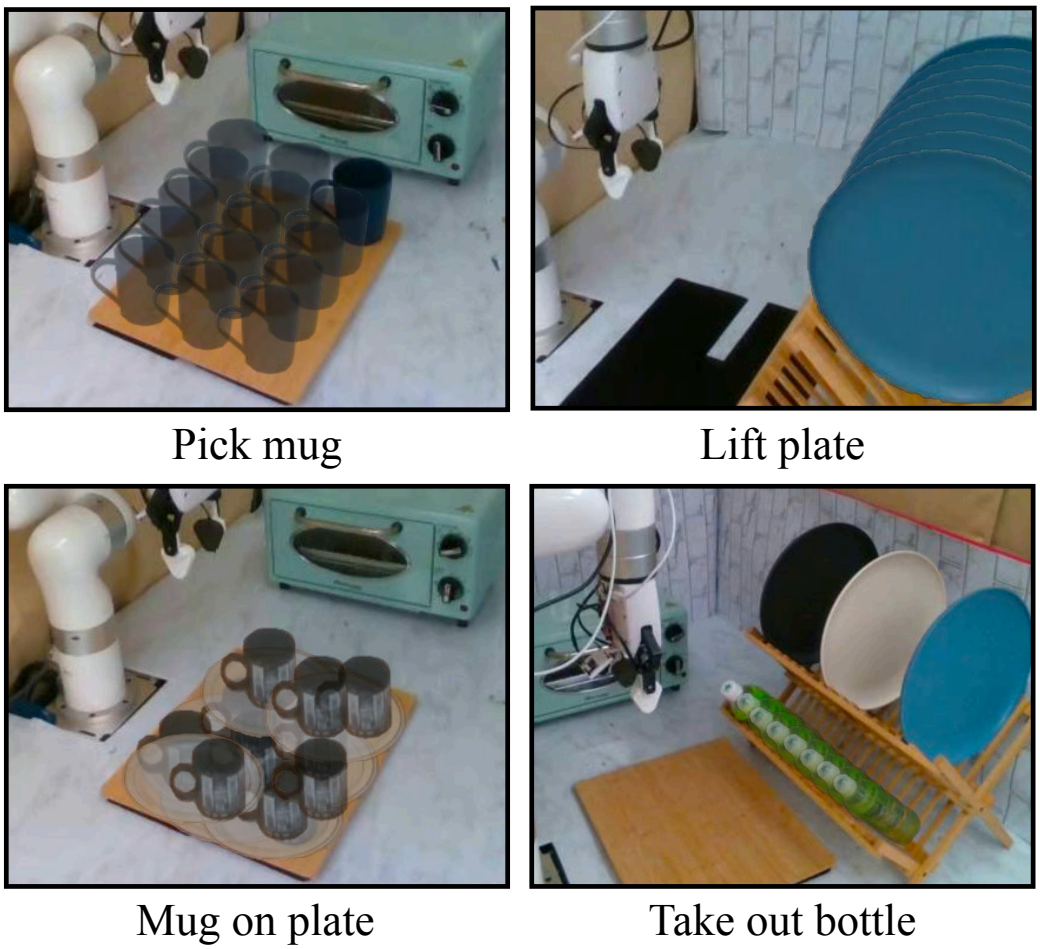


Figure 3.4: Illustration of spatial variation used in our experiments.

annotated. This point representation is then encoded using a multilayer perceptron (MLP) encoder. Given the noise in real-world depth sensing, we aggregate a history of observation features and feed them as separate tokens into the BAKU causal transformer policy. The policy predicts the action corresponding to each historical token using a deterministic action head and action chunking with exponential temporal averaging [156].

3.5 Experiments

Our experiments are designed to answer the following questions: (1) How well does P3-PO work for policy learning? (2) How well does P3-PO work for novel object

instances? (3) Can P3-PO handle background distractors? (4) How does P3-PO perform with estimated depth? (5) Can P3-PO be improved with stronger priors?

3.5.1 Experimental Setup

Our experiments are performed on a Ufactory xArm 7 robot with an xArm Gripper in a kitchen environment. The policies are trained with RGB-D images from a third-person camera view and robot proprioception as input. The action space is comprised of the robot end effector pose and the gripper state. We collect a total of 160 demonstrations across 4 real-world tasks with varied object positions and types. The demonstrations are collected using a VR-based teleoperation system [52] at a 30Hz frequency, which are then subsampled to 5Hz. The learned policies are deployed at 5Hz.

3.5.2 Task Descriptions

We experiment with four manipulation tasks that exhibit significant variability in object position, type, and background context. Figure 3.6 provides rollouts of the tasks performed in our real-world setup. For each task, we collect expert demonstrations across a variety of object sizes and appearances. We refer to objects and environments seen in our collected data as in-domain. During evaluations, we add novel object instances that are unseen in the training data. The variations in positions and object instances for each task are depicted in Figure 3.4 and Figure 3.5 respectively. We provide a brief description of each task below.

Pick mug The robot arm picks up a mug placed on the kitchen counter. The position of the mug is varied for each evaluation. We collect 15 demonstrations for 4 different mugs, resulting in a total of 60 demonstrations for the task. During evaluation, we introduce 3 novel mugs.

Figure 3.5: Illustration of objects used in our experiments. In each image, the left pile depicts the in-domain objects while on the right are novel objects used in our generalization experiments.

Lift plate The robot arm lifts a plate placed on the upper level of a rack. We collect 8 demonstrations for 3 different plates, resulting in a total of 24 demonstrations. During evaluation we introduce 2 novel plates.

Mug on plate The robot arm picks up a mug placed on the kitchen counter and places it on a plate. We collect 15 demonstrations for 4 different mugs placed on the same plate, resulting in a total of 60 demonstrations for the task. During evaluation we use 1 novel mug and 1 novel plate.

Take out bottle The robot arm takes a bottle out from the lower level of a rack. We collect 8 demonstrations for 2 different bottles, leading to a total of 16 demonstrations.

During evaluation we introduce 2 novel bottles.

For all tasks, the xArm is initialized at its home position, while the object locations are varied across trials. During evaluation, the objects are placed in a held-out set of positions to keep comparisons fair across baselines.

3.5.3 Baselines

We compare P3-PO with three primary baselines.

Full RGB Representation This method utilizes the BAKU transformer architecture [41], which takes the full RGB image of the scene and robot proprioception as input.

RGB-D Representation This is a depth-based extension of BAKU that separately processes the depth image using an encoder and appends a depth token to the policy input for action prediction.

GROOT [160] GROOT is a transformer-based imitation learning algorithm that constructs an object-centric 3D representation using Segment Anything [61] and a clustered point cloud which makes it robust to background distractors and novel objects. We refer the reader to the paper [160] for more details about GROOT.

3.5.4 Considerations for policy learning

P3-PO generates a point-based representation from 512x512 pixel images. For correspondence, P3-PO leverages DIFT [131], using the first layer of the hundredth time step with an ensemble size of 4. Point tracking in P3-PO is performed by a modified version of online Co-Tracker that enables tracking one frame at a time, rather than in chunks. P3-PO and GROOT utilize observation history while the RGB and RGB-D baselines do not [41]. Additionally, the RGB and RGB-D baselines

Table 3.1: In-domain policy performance

Method	Pick mug	Lift plate	Mug on plate	Take out bottle
RGB [41]	13/40	22/30	8/40	5/20
RGB-D	19/40	22/30	7/40	6/20
GROOT [160]	7/40	7/30	0/40	0/20
P3-PO	**39/40**	**29/30**	**32/40**	**13/20**

incorporate robot proprioception as an input, while we follow GROOT and do not use robot proprioception as an input to P3-PO.

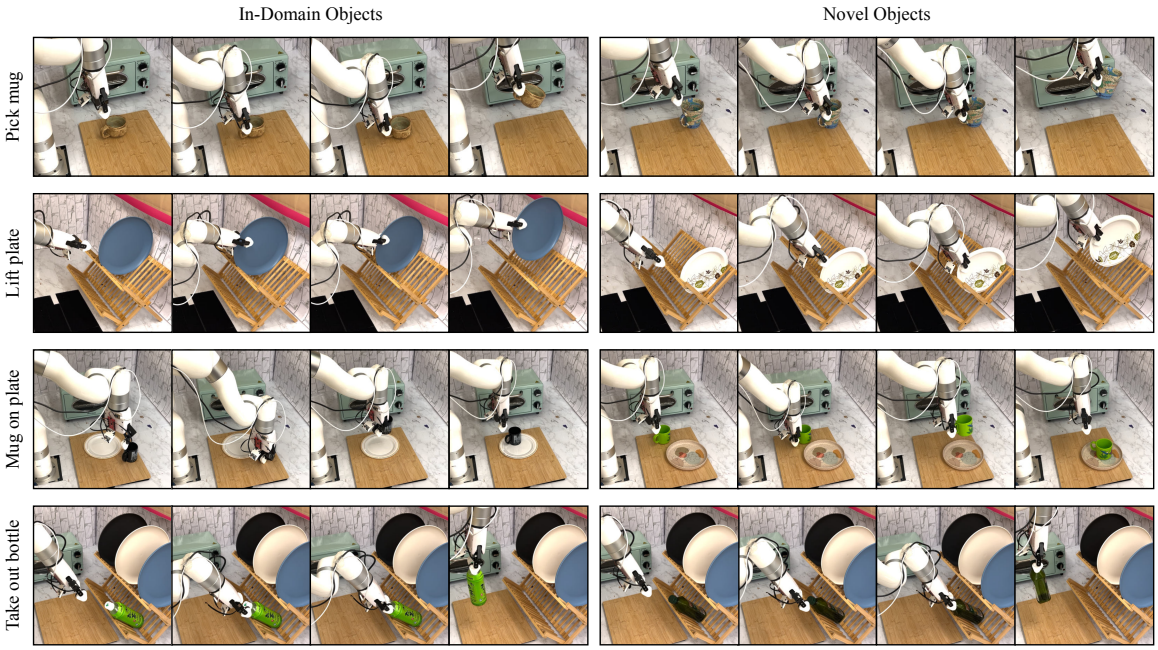


Figure 3.6: Real-world rollouts showing P3-PO's capability on **(a)** in-domain objects and **(b)** novel objects.

3.5.5 How well does *P3-PO* perform for policy learning?

We first evaluate P3-PO's performance on the in-domain environment configurations using in-domain objects. We conduct 10 trials per object per task resulting in a variable number of total trials per task. The results have been reported in Table 3.1. We observe that P3-PO outperforms the strongest baseline on average by 43% across our four real-world tasks. It must be noted that we only use 8-15 demonstrations per

Table 3.2: Policy performance on novel object instances

Method	Pick mug	Lift plate	Mug on plate	Take out bottle
RGB	6/30	5/20	0/20	2/20
RGB-D	3/30	10/20	1/20	4/20
GROOT	4/30	5/20	0/20	0/20
P3-PO	**29/30**	**18/20**	**16/20**	**10/20**

object per task, which is much smaller than prior works studying spatial generalization in robot policy learning [21, 120, 156]. For the full RGB and RGB-D baselines, most failures stem from minor errors, suggesting these policies could be improved with additional demonstrations. In the case of GROOT, having introduced larger variations in both the rotations and spatial locations of objects than the original paper [160], we observe that the policy is unable to generalize. We believe this limitation arises because GROOT normalizes each object-specific point cloud by its centroid, losing information about its position in space. To address this, GROOT adds the positional embedding of the centroid to the processed representation. However, this may not optimally reinforce the positional information. Videos on our website provide examples supporting these hypotheses.

3.5.6 How well does *P3-PO* work for novel objects?

Table 3.2 compares the performance of P3-PO with the baselines when tested on novel objects. These objects can be seen in Figure 3.5. We conduct 10 trials for each novel object for each task resulting in a variable number of total trials per task. We observe that P3-PO's visual representation allows it to effectively generalize to novel object instances, outperforming the strongest baseline by 58% across all tasks. The RGB and RGB-D baselines exhibit reduced performance on novel objects due to their reliance on visual features learned from the training set. While designed for generalization, GROOT struggles with spatial variations resulting in lower accuracy. These results

Table 3.3: Policy performance with background distractors

Method	**Pick mug**		**Lift plate**	
	In-domain	Novel object	In-domain	Novel object
RGB	0/5	0/5	4/5	0/5
RGB-D	1/5	0/5	2/5	0/5
GROOT	0/5	1/5	1/5	1/5
P3-PO	**5/5**	**5/5**	**5/5**	**5/5**

suggest that P3-PO, with its point-based non-image specific representation, is better equipped to generalize to novel objects than prior methods.

3.5.7 Can *P3-PO* handle background distractors?

We evaluate the performance of P3-PO in the presence of distractors in the task background. An illustration of the distractors has been included in Figure 3.2(c). We study this on two tasks - *pick mug* and *lift plate*. For each, we evaluate the performance using 5 trials each on an in-domain object and a novel object. Table 3.3 provides these results. We observe that P3-PO outperforms the strongest baseline by 80%. The image-based baselines exhibit low accuracy due to their reliance on visual features while GROOT struggles with spatial generalization. These results reinforce that P3-PO's point representation, decoupled from raw pixel values, enables policies that are robust to environmental perturbations.

3.5.8 How does *P3-PO* without ground truth depth?

Given recent advances in monocular depth prediction [147, 148], we investigate the importance of true depth values for the performance of P3-PO. To evaluate this, we compare P3-PO when using true depth from an RGB-D camera versus predicted depth from an off-the-shelf monocular depth estimation model, Depth Anything 2 [147]. As shown in Table 3.4, we observe that P3-PO achieves equivalent performance on

Table 3.4: Effect of camera vs. predicted depth on P3-PO

P3-PO	**Pick mug**		**Lift plate**	
	In-domain	Novel object	In-domain	Novel object
Camera Depth	5/5	5/5	5/5	5/5
Depth Anything	5/5	5/5	5/5	5/5

two tasks, with one in-domain object and one novel object, regardless of whether true or predicted depth was provided. This is an interesting result, as it implies that P3-PO may be applicable to large-scale robot datasets [59, 94] which might not always include depth data.

3.5.9 Can *P3-PO* be improved with stronger priors?

Prior work has shown that encoding relational structure between inputs can improve policy learning generalization [64, 104, 124, 151]. In this section, we investigate whether encoding the spatial relationships between key points as a graph prior could further enhance P3-PO's performance. Specifically, we represent the key points as a fully connected graph, with edges encoding the 3D distance between each pair of points. Leveraging the annotation order, we flatten this graph into a vector representation encoding the spatial relations between all point pairs. Policies are then trained on this graph-structured input. As shown in Table 3.5, encoding the key points as a graph prior results in similar performance to directly using the key points as input. While additional structure did not provide clear benefits in this case, future work could explore more sophisticated relational encodings or combining our approach with other structural priors.

3.5.10 Can *P3-PO* complete more complex tasks?

In this section, we demonstrate that in addition to the tasks shown above, P3-PO excels at tasks that are more complex and dexterous than simple pick-and-place operations.

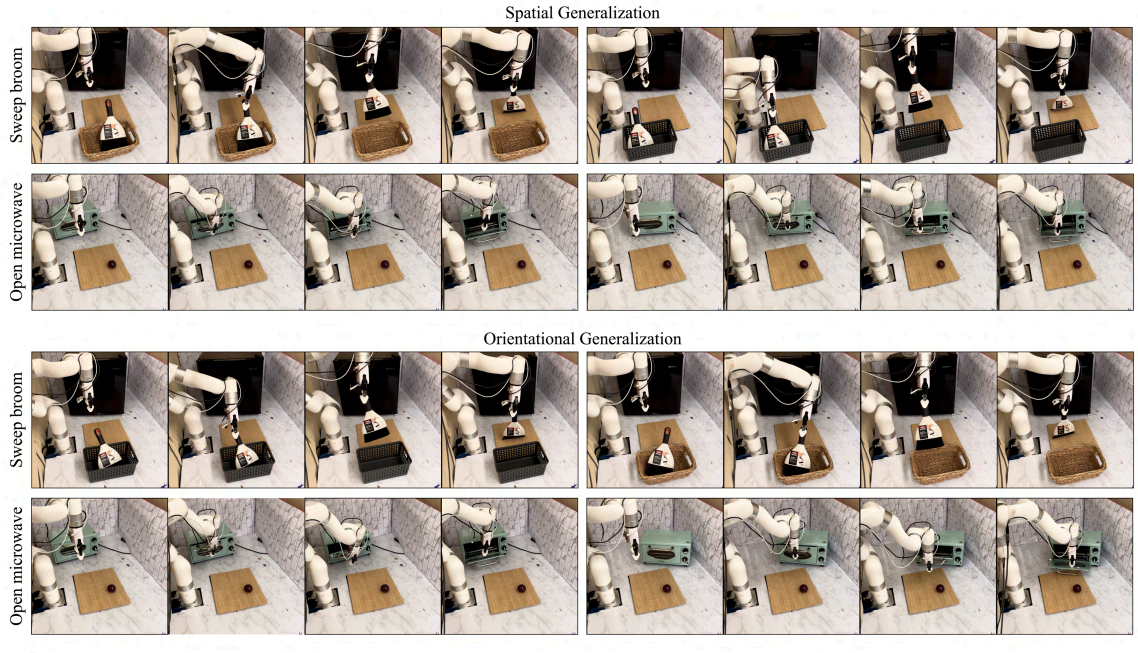


Figure 3.7: Rollouts of the two complex tasks. On the top we show that for both of these tasks P3-PO can generalize to the object being in a different location. On the bottom we show that P3-PO can also generalize to different orientations of the object.

First, we present results for a sweeping task where the robot picks up a broom and sweeps a nearby cutting board (Fig 3.7). This task is challenging for two reasons - (1) The broom's handle is rounded, requiring precise handling to prevent slipping. This demonstrates that the point-based context provides sufficient environmental understanding for precise manipulation. (2) The task is long-horizon, consisting of two stages: lifting the broom and sweeping the board. P3-PO is able to understand its place in the sequence and act accordingly.

To evaluate P3-PO's adaptability, we tested the task with two different baskets, showcasing its ability to handle varying vertical and horizontal angles. This is demonstrated in Fig 3.7 The model achieves an **80**% success rate when trained on 30 demonstrations and evaluated on 10 trials.

Next, we present results for an open-microwave task, which requires the robot to navigate into the thin opening between the handle and the microwave. Despite relying solely on point-based input, the robot is able to succeed with a high level of

precision. This task is further challenging due to the wide range of orientations and variations in the microwave's initial placement. The results demonstrate P3-PO's ability to generalize to new locations and accurately interpret object orientations. P3-PO achieves an **80**% success rate on this task when evaluated on 10 trials and trained on 22 expert demonstrations. Variations in microwave locations and task execution are shown in Figure 3.7.

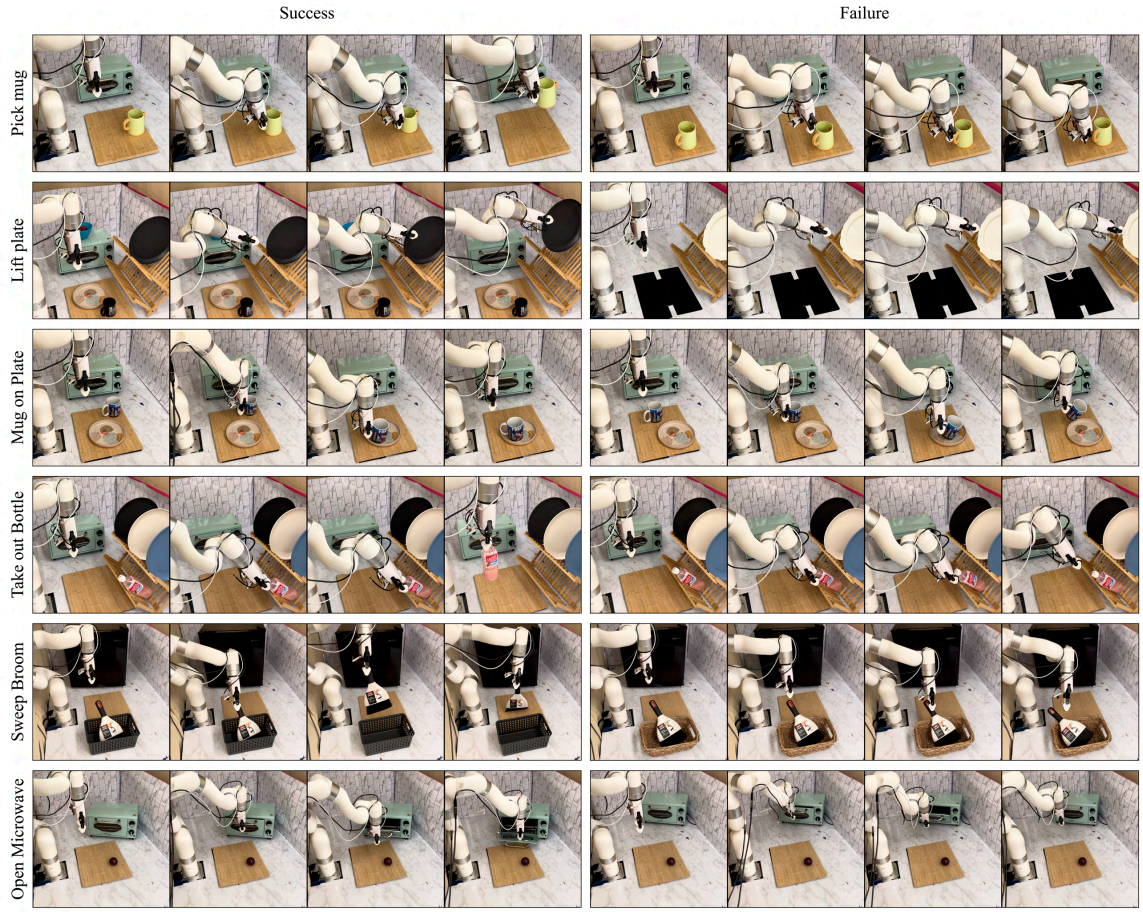


Figure 3.8: Demonstrations of success and failure for each task. On the left we show successful demonstrations and on the right demonstrations of episodes that fail.

3.5.11 What do failures look like in *P3-PO*?

In Figure 3.8, we present examples of both successful and failed episodes across all six tasks. These examples highlight that while P3-PO does not always succeed,

Table 3.5: Effect of stronger priors on P3-PO

Input	**Lift plate**		**Take out bottle**	
	In-domain	Novel object	In-domain	Novel object
Point	5/5	4/5	3/5	3/5
Graph	5/5	5/5	4/5	3/5

its failures come close to achieving the task objectives. For instance, in the "open microwave" task, the failures occur when the robot cannot open the microwave door far enough for the door to remain open. Similarly, tasks like "pick mug" and "lift plate" likely fail due to noise in the depth measurements. Notable these failures are infrequent and we believe they can be addressed in future iterations of this work.

3.6 Conclusion and Limitations

In this work, we presented Prescriptive Point Priors for Policies (P3-PO), a simple yet effective framework that leverages human-provided semantic key points to enable more robust policy learning. P3-PO demonstrates improved generalization to spatial variations, novel objects, and distracting backgrounds compared to prior state-of-the-art methods. We recognize a few limitations in this work: (a) P3-PO's reliance on existing vision models makes it susceptible to their failures. For instance, point tracking failures under occlusion hurt policy performance. However, we believe that continued advances in computer vision will serve to further strengthen performance of P3-PO. (b) While point abstractions facilitate better generalization, they lose information about scene context that could be important for navigation amid obstacles or clutter. Future work developing algorithms to retain sparse contextual cues while maintaining P3-PO's object-centric representation may help address this. (c) In this work, we primarily study the single task performance of point prior policies. Extending the framework to multitask learning would be an interesting research

direction. Overall, we believe P3-PO takes an important step toward developing general, data-efficient robot policies suitable for real-world deployment by grounding them in human point priors.

Chapter 4

Learning From Demonstrations

This chapter introduces a new method for learning reinforcement learning (RL) policies without relying on handcrafted reward functions. As discussed throughout this thesis, a major challenge in robotics is the limited availability of data and the high cost of collecting it. While RL offers the advantage of requiring less human supervision during training, its reliance on task-specific reward functions undermines its practicality. Designing these rewards is often time-consuming and brittle, negating many of RL's benefits. Prior efforts such as inverse reinforcement learning (IRL) [36, 47, 109] aim to learn reward functions from expert demonstrations, but typically require large datasets—bringing the data scarcity issue back into focus. In Waypoint Exploration from a Single Demonstration (WayEx), we demonstrate that a single unlabeled demonstration can effectively replace a reward function, achieving performance on par with state-of-the-art methods that require 100× more data.

4.1 Introduction

Humans have a natural ability to learn tasks by observing a single demonstration which they can follow step-by-step. For instance, we can watch a video and grasp how to open a vault, then practice until we succeed without requiring further instructions.

Drawing inspiration from this ability, the combination of learning from demonstrations and reinforcement learning techniques has become a popular and potent approach for training robots [86, 103, 117, 162]. However, compared to humans who can learn simple tasks from a single demonstration, robots require a multitude of diverse expert instances. For example, to learn how to open a vault, the robot must observe successful demonstrations for different views and locations of the vault handle with respect to the robot's location. Moreover, each demonstration must contain information about the location of the vault (state), the precise joint rotations (action) to reach the vault, and knowledge about how close it is to completing the task (reward).

Hence, most common methods in learning from demonstrations, such as Imitation learning [93, 102, 103] and Inverse reinforcement learning [36, 47, 109] require a set of expert demonstrations, with a defined state, action and reward space. Collecting all the data in real time and computing the definitive action and reward space is impractical and inefficient. Therefore, in this work, we strive to reduce these inefficiencies by using only a single demonstration for training. Additionally, our setup does not require knowledge about the action space, relying on just the state space and a corresponding reward function.

Prior works identify key states (waypoints [93, 102]) along the robot's trajectory, to help it navigate towards the goal. We also employ waypoints to solve the task, without having access to the action space. Each observation within the demonstration is defined as an individual waypoint. While other approaches [93] need access to waypoints during the testing phase, WayEx only requires waypoints during training. This prevents the need for training a model to predict the waypoints during inference. We leverage the waypoints during training via an augmented reward structure based on the known Q-Values [141] associated with each waypoint.

A common approach to solving a task without a dataset of expert trajectories is to use dense rewards, based on the key steps of a task. Existing studies in the field of

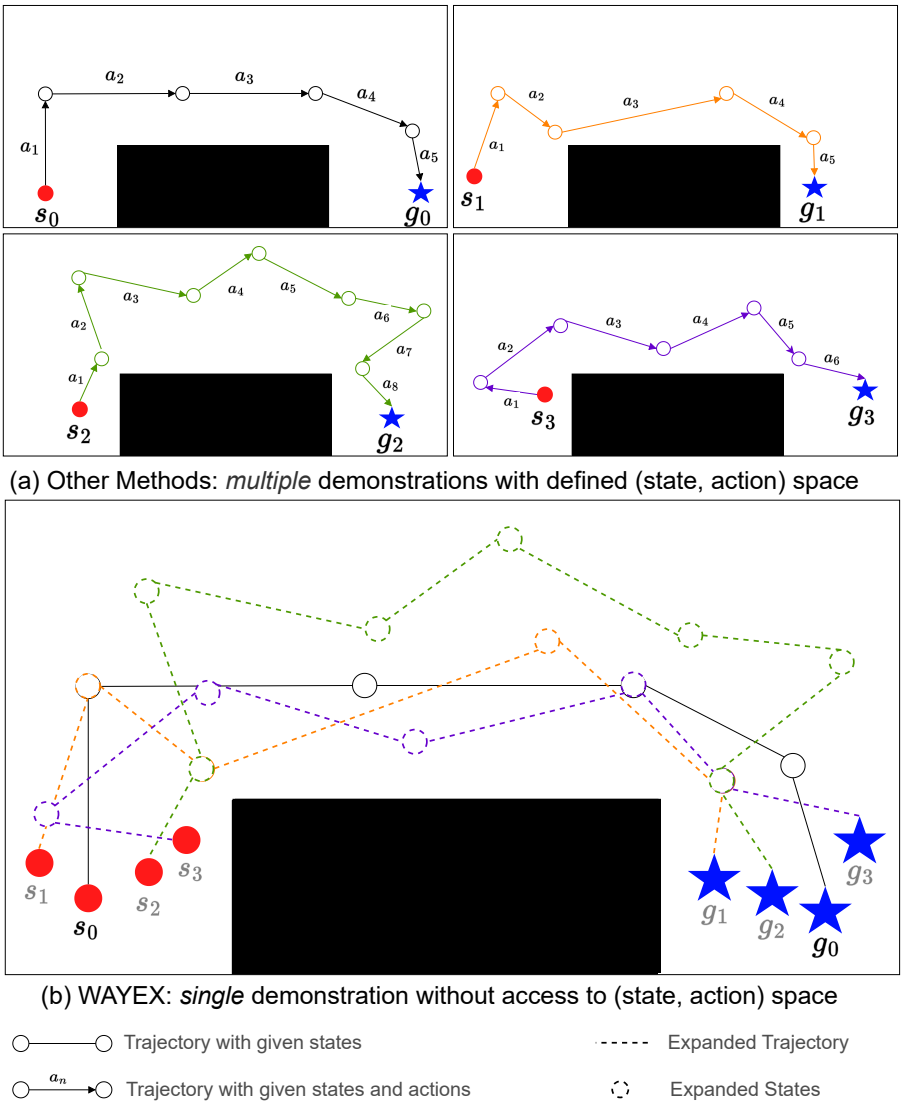


Figure 4.1: A comparison of our approach to general imitation learning techniques. **(a)** Traditional Imitation learning approaches require multiple expert trajectories with a known action space for training (4 shown here). **(b)** For our proposed method WayEx we use only one expert trajectory, and expand knowledge from this one trajectory to learn how to solve the task. During training with a single initial state (s_0) and a single goal state (g_0), our model learns to navigate back to the expert trajectory from points that are not part of the trajectory (all dotted states, which can be a combination of 4 expert trajectories shown on the top). This enables the model to successfully reach the goal state. We further introduce additional start and goal states ($[s_1, g_1]$,[s_2, g_2], $[s_3, g_3]$).

reinforcement learning acknowledge that employing dense rewards is difficult since it requires practitioners to engineer specific reward functions for each task. Additionally these rewards, if ill-designed, can lead to unforeseen behavior. To circumvent these challenges, we assume a sparse reward structure when determining the new reward. With sparse rewards, the robot receives a reward of 0 if the actions lead to a goal state; otherwise, the reward is -1. Acquiring this reward solely through the process of exploration can prove to be highly challenging, making certain tasks unachievable with sparse rewards alone [85]. Our approach strikes a balance, allowing the model to receive frequent rewards without exposing it to the typical risks associated with dense rewards.

By utilizing this new reward structure, our model can solve tasks that closely resemble the demonstration setup. However, despite its ability to learn from a single example, our approach still encounters a common limitation of learning from demonstrations. If the robot encounters a state beyond the scope of what it has previously encountered it will not know how to proceed. To overcome this challenge, the robot needs to acquire experience beyond the confines of the provided demonstrated space. A commonly employed method to achieve this is to integrate learning from demonstrations with conventional model-free reinforcement learning algorithms [40, 75]. Strict model-free reinforcement learning involves learning optimal state-action pairs through trial and error rather than relying on expert trajectories. We combine model-free reinforcement learning with our waypoint reward and introduce an additional expansion method that further enhances the model's knowledge.

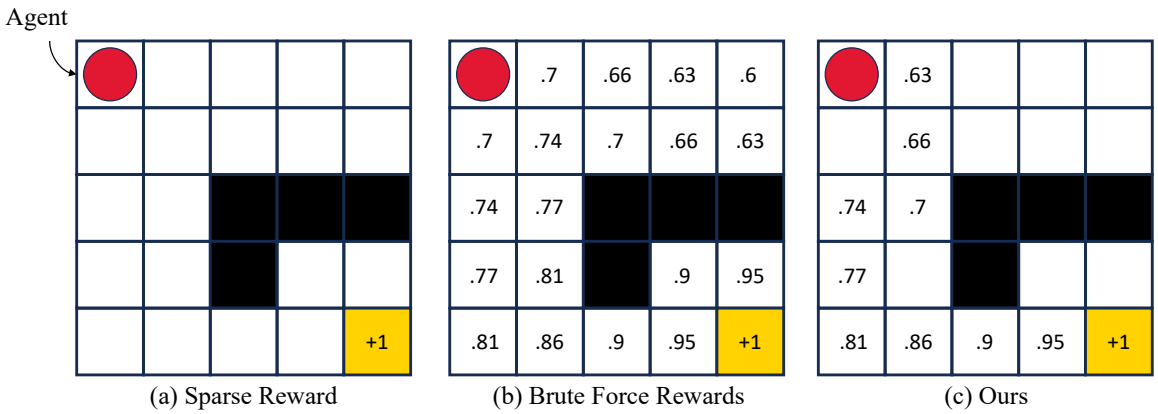
In this work, we propose WayEx, derived from **Way**point **Ex**ploration, a novel approach that enables the training of a reinforcement learning model using a single expert demonstration and without any prior knowledge of the action space. It can serve as a wrapper around any reinforcement learning algorithm, facilitating its applicability as the field advances. Our primary technical contributions are: (1)

the introduction of a new reward function based on sparse rewards, which provides additional rewards without introducing unforeseen consequences, and (2) a method for expanding knowledge beyond a single demonstration to encompass the entire spectrum of both the state and goal spaces. We demonstrate that our approach enables faster learning of tasks compared to previous reinforcement learning methods while requiring minimal additional information.

4.2 Related Work

Goal-Conditioned Reinforcement Learning. We are interested in investigating tasks that involve a robot reaching a specific end state specified by an initial "goal." In the reinforcement learning (RL) community, these problems are known as Goal-Conditioned Tasks. Prior works have studied how to use RL in many different ways in order to solve these tasks [6, 31, 77, 89, 105, 130]. Early works such as [105, 130] show that it is possible to use standard RL methods such as [40, 75, 84], but it can be time-consuming, and there are some types of tasks that these methods alone cannot solve. To combat this, other works have suggested the use of hindsight re-labeling [6, 15], which speeds up the process, but still requires a large amount of data to reach a successful trajectory [105].

Imitation Learning. Learning from demonstrations, also known as imitation learning, is a common approach for solving goal conditioned tasks. Our approach uses ideas similar to prior works such as [86, 95, 102, 103, 117, 162]. These works take recorded successful episodes of a task and use them to aid in training the RL model. Several of these papers operate by adding successful demonstrations to the replay buffer [86, 95]; however, these papers require accurate knowledge of the action space. This type of information is only accessible in datasets specifically designed for robot training. In order to learn from other sources, such as videos, the field must evolve



(a) Sparse Reward (b) Brute Force Rewards (c) Ours

Figure 4.2: **Visualization of Grid World Toy Example. (a)** shows the environment setup with a sparse reward. **(b)** shows the reward for each state once the entire environment has been solved using the bellman equation [11]. **(c)** represents WayEx where with a single demonstration, we compute a close approximation of the reward for each state along the path to the goal.

past this method. Additionally, aside from [102, 117], these works all require a large number of demonstrations. [102] can use a smaller number, but needs a task-specific reward function. [117] only uses one example, but requires different start states. [103] proposes pre-training the model and then fine tuning for each individual task, but their pre-training is data intensive and not always generalizable.

Inverse Reinforcement Learning. Another closely related area, Inverse Reinforcement Learning (IRL), uses a model to predict the reward values based on the state and action space. Methods such as [36, 47, 109], employ a hybrid approach involving both IRL and adversarial learning to solve a task with limited demonstrations. Similar to our method, IRL methods are useful because they do not require a defined reward function. Nonetheless, despite their claim of using a small number of demonstrations, these methods still require at least 50 demonstrations. Additionally, these methods struggle to generalize beyond the initial demonstrations. Our approach differs from IRL approaches because we do not require the training of a model to define our generalized reward, which leaves less room for error and requires less data.

Modified Sparse Reward. Several methods [78, 112, 137, 143] attempt to

modify the sparse reward function in different ways to make learning more efficient. Despite employing a similar reward structure to ours and looking at the maximum of two different functions, [143] still requires a large number of demonstrations as well as access to the action space. Another way to learn with sparse rewards is to split multi-step tasks into several different tasks [112]. This allows the sparse rewards to be more frequent, but requires a precise definition of each auxiliary task which makes creating a generalized model more difficult.

4.3 Method

4.3.1 Preliminaries

We formulate our problem as a Markov Decision Process (MDP) consisting of an [n]-tuple $(S, A, R, \tau_D, \gamma)$. The elements of this tuple are the state space S, the action space A, the reward function $R\colon S \times A \to \mathbb{R}$, the demonstration trajectory $\tau_D\colon (s_0^*, \ldots, s_N^*)$ and the discount factor γ. We will refer to a random episode trajectory as τ, where $\tau = (s_0, a_0, \ldots, s_N, a_N)$. N is the total number of states and actions to reach every state. A policy is represented as $\pi_\theta\colon S \to A$, with parameters θ. We use a sparse reward paradigm for our method, where the action space is unknown. A sparse reward is defined as a reward function R that receives a reward of 0 when in the goal state, g. At all other times the reward is -1. If g represents the goal state, and s_n and a_n are the n^{th} state and action respectively, then the sparse reward function R can be represented as

$$R(s_n, a_n) = \begin{cases} 0 & \text{if } s_n = g, \\ -1 & \text{if } s_n \neq g. \end{cases} \tag{4.1}$$

4.3.2 Overview

In this section, we describe our method WayEx for learning goal-conditioned skills from a single demonstration with no information about the demonstration's action space. An illustrative overview of our method is provided in Figure 4.2, which shows a much simpler grid world demonstration. The leftmost grid in Figure 4.2 represents the environment, where an agent must traverse the boxes to find the goal, which gives a sparse reward of 0. The middle grid shows the reward for each box using a bellman equation [11], which requires access to rewards for all states. Finally, the right grid shows our approach which traverses a single path and then recursively determines the reward of each state along the path. Access to one successful demonstration allows WayEx to determine the pseudo ground truth rewards for each box along the path.

WayEx uses a sparse reward, along with bellman's equation to compute the value for each waypoint along the demonstration path. During exploration, a new state obtains a reward if its distance from a known waypoint is less than a threshold d_{thresh} (captured by `is_prox_wp`()). We use Nearest Neighbors to determine the waypoint the new state is compared to. After training and achieving some success, we improve our method's ability to generalize to unseen start and goal states, by expanding on possible start and goal states. We present an algorithmic overview of WayEx in Algorithm 1, followed by a comprehensive breakdown of each step for a clearer understanding.

4.3.3 Proximal Waypoint

Each state of the environment can be represented as $s_i = \{p_1, p_2, \ldots, p_K\}$, $\forall$ i $\in (1, \ldots, N)$, where p_k represents an environmental parameter such as object pose and gripper velocity and K represents the number of environmental parameters. Note that each parameter, p_k, is a relative value, with respect to the object's location, instead of an absolute value with respect to the world coordinate system. This ensures better generalizability of our method. We use our policy π_θ to determine an action that allows

us to reach an unseen state s_e. Following other reinforcement learning algorithms [40, 75], we add random noise to the action space, in order to increase the amount of exploration done during training. Once we have reached s_e we will determine if it is within close proximity of a waypoint along our demonstration trajectory τ_D. To do this we first use the Nearest Neighbor function to find the waypoint with the smallest total L2 distance between the parameters in the waypoint and the parameters in s_e as

$$s_t^* = \mathbf{NN}(s_e, \tau_D) = \arg\min_{\forall s_i^* \in \tau_D} \|s_e - s_i^*\|_2 \,, \tag{4.2}$$

where $s_t^* \in \tau_D$ is the closest state to (or the proximal waypoint for) s_e from the states within the expert trajectory τ_D. For an agent to receive a reward at state s_e, the distance between the parameters of s_e and s_t^* should be less than a threshold. For instance, if $d_{\text{thresh}} = \{d_1, d_2, \ldots, d_K\}$ is the corresponding threshold for parameters between $s_e = \{p_1, p_2, \ldots, p_K\}$ and $s_t^* = \{p_1^*, p_2^*, \ldots, p_K^*\}$, then we compute the Boolean `hasProxWP` as

$$\texttt{is\_prox\_wp}(s_e, s_t^*) = \begin{cases} \text{True}, & \text{if } \|p_i^* - p_i\| \leq d_i, \forall i \in K, \\ \text{False}, & \text{otherwise.} \end{cases}$$

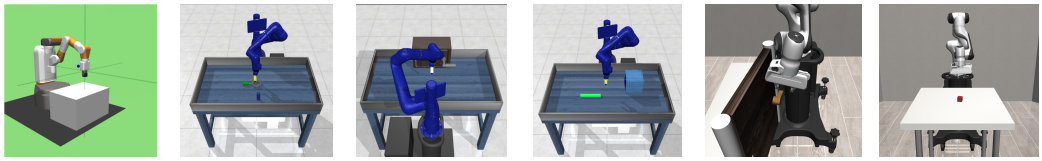
We define one d_{thresh} for each of the waypoints in τ_D. For instance s_t^* has its own threshold, d_t^*, and when s_t^* is the nearest neighbor to s_e, we use d_t^* as the threshold. In order to encourage progression, if `hasProxWP` is False 10 consecutive times for a waypoint s_t^*, then we increase d_t^* by ϵ $(= 0.001)$. We repeat this until we explore a point that falls within the threshold of the waypoint s_t^*.

4.3.4 New Reward Function

Given the nearest neighbor waypoint, s_t^*, and the boolean result, `hasProxWP`, we can solve for the reward function, r, for our state action pair (s_e, a_e). $l_{\max}$ represents the

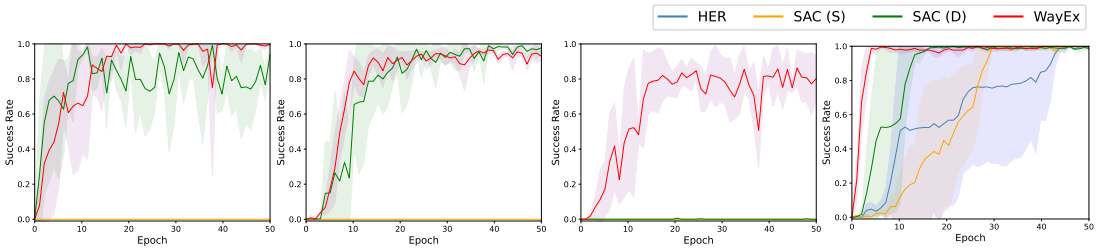
Algorithm 1 WayEx, prior to expanding knowledge

$\tau_D = (s_0^*, \ldots, s_T^*)$: A successful demonstration
π_θ: The policy that we will follow and update
while true **do**
    $\tau \leftarrow (s_0, a_0, \ldots, s_N, a_N)$: an episode roll out
    for all $s_n, a_n \in \tau$ **do**
        $s_t^* \leftarrow \mathbf{NN}(s_n, \tau_D)$, where $s_t^* \in \tau_D$
        `hasProxWP` $\leftarrow$ `is_prox_wp`(s_e, s_t^*)
        $r \leftarrow \text{reward}(\texttt{hasProxWP},\ t,\ l_\text{D}, l_\text{max})$
        $R \leftarrow \max(r,\ \gamma * \text{critic}(s_{n+1}))$
    end for
end while



(a) Pick and Place (b) Peg Assembly (c) Open Door (d) Peg Insertion (e) RS-Open Door (f) RS-Lift

Figure 4.3: The environments that we experimented on with WayEx. We show results on 4 different tasks: (a) pick and place, (b) peg assembly, (c) open door and (d) peg insertion. These tasks are ideal because they have a clear definition of success and therefore a clear sparse reward. However, most of these tasks cannot be solved with sparse rewards alone.



(a) Door Open (b) Peg Insertion (c) Peg Assembly (d) Pick and Place

Figure 4.4: **(a,b,c)** shows the results of the Meta World [152] environments when trained using SAC [40] and a batch size of 2048. **(a)** Open Door Task, **(b)** Peg Insertion Task, **(c)** Peg Assembly Task. **(d)** Pick and Place task is from OpenAI [105].

Table 4.1: Requirements for different baselines and WayEx in terms of state, action and number of expert demonstrations.

Pre-requisites	SAC (S)	SAC (D)	HER	SAC + RB	SAC + MCAC	AWAC	AWAC + MCAC	**WayEx**
Requires Action Space	✗	✗	✗	✓	✓	✓	✓	✗
Requires Pre-training	✗	✗	✗	✗	✗	✓	✓	✗
# Expert Demonstrations	0	0	0	1 or 100	1 or 100	1 or 100	1 or 100	**1**

maximum length of an episode, l_{D} represents the length of the demonstration τ_D and t represents the time at which s_t^* occurs. We compute the proposed reward, r as

$$r = \begin{cases} \sum_{i=0}^{l_{\mathrm{D}}-t} -\gamma^i & \texttt{hasProxWP}, \\ \sum_{i=1}^{l_{\max}} -\gamma^i & \text{not } \texttt{hasProxWP}. \end{cases} \tag{4.3}$$

If `hasProxWP` is false, we still want to account for the possibility that our state, s_e, is on a successful trajectory. To do this, we say that the final reward $R = \max(r, \gamma * \text{critic}(s_{e+1}))$. In order for this to work we need to warm up the critic. We do this by training it for 1000 time steps on just r before we include the critic reward. For more information on actor critic methods please refer to [40, 75].

4.3.5 Expanding Knowledge

Following the demonstration, WayEx teaches the policy how to solve the task from a fixed start and goal state. We now need to expand to every possible start and goal location. To achieve this, we slowly increase the number of possible start locations and goal locations, by adding random noise to the initial state space.

Given our current demonstration trajectory τ_D with states s_0^* and goal g^*, let $\mathcal{N}^*(\mu^*, \sigma^*)$ be the distribution representing the amount of noise we will add to the start state and goal state. We will set the mean, μ^*, to 0 and only increase the standard deviation σ^*.

At the start of training σ^* is set to 0. σ^* uses a modification strategy applied every 25 episodes. The updated value of σ^* is described as σ', where:

$$\sigma' = \begin{cases} \sigma^* + 0.001, & \text{if success rate} \geq 0.05, \\ \sigma^*, & \text{otherwise.} \end{cases} \tag{4.4}$$

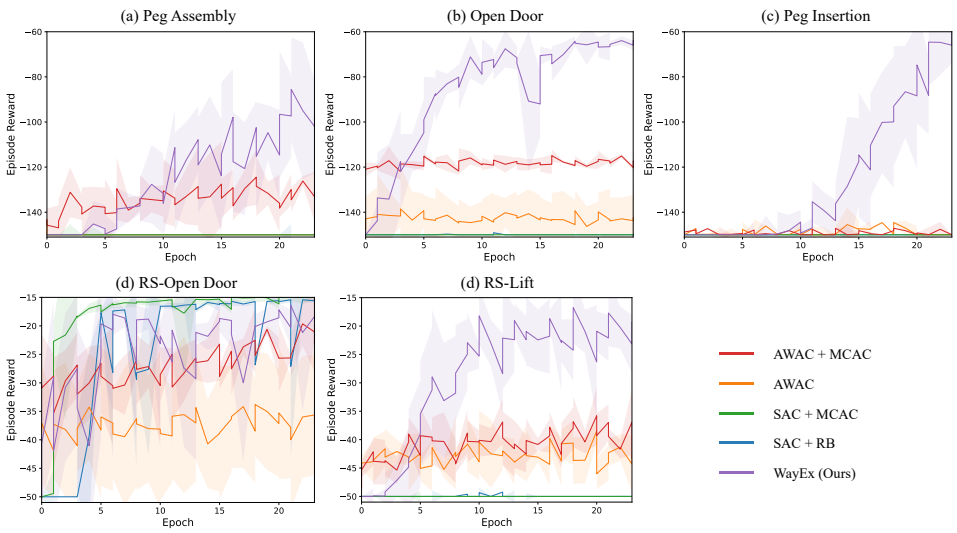


Figure 4.5: This figure shows the results of several baselines when the are given one expert demonstration **(a,b,c)** shows the results of the Meta World [152] environments when trained using AWAC [88], MCAC [143], SAC + RB and AWAC + MCAC. **(d,e)** shows the results of these same baselines on the robosuite tasks [163]

4.4 Experiments and Results

4.4.1 Environment Setup

We use MuJoCo [132] to simulate our tasks. The implementation of our reinforcement learning algorithms was based on modified versions of the open-source code provided by stable-baselines3 [108] and our baselines were based on [143]. Although we used only one demonstration for each task, we varied the starting demonstration across different seeds to demonstrate the adaptability of our method to different initial demonstrations. To ensure robustness, we conducted each experiment four times with different seeds and present the mean and standard deviation of these seeds in our graphs. Each epoch consists of 40,000 simulated timesteps in MuJoCo. The pick and place environment, the robosuite door environment and the robosuite lift environment have 50 time steps per episode, and the remaining environments have 150 time steps per episode. We pre-train AWAC for 25000 timesteps.

4.4.2 Tasks

WayEx is trained on six distinct tasks, each designed to showcase the robot's capability to accomplish simple goal-conditioned objectives. The pick and place environment is from the OpenAI Fetch tasks [105], the next three (open door, peg insertion, and peg assembly) tasks are from Meta World [152]. The final two tasks, Robosuite (RS)-Open Door and RS-Lift come from [163]. Images of these tasks can be seen in Figure 4.3. Our tasks are: **(a) Pick and Place:** The task is to grasp a box and move it towards a goal in the air. **(b) Peg Assembly:** The task is to pick up a round nut and then place the round part over a peg. **(c) Open Door:** The task is to grasp a door handle and then open the door until it reaches a goal location. **(d) Peg Insertion:** The task is to pick up a peg and insert it into a hole. **(e) RS-Open Door:** The task is to grasp a door handle and then open the door very slightly. **(f) RS-Lift:** The task is to pick up a block and lift it into the air.

4.4.3 Baslines

We test our method (Table 4.1) against following baselines:

- **SAC**. This baseline uses the Soft Actor Critic (SAC) algorithm [40] as the reinforcement learning algorithm. It can be initialized in three ways: (1) Sparse Reward **(SAC (S))**, (2) Dense Reward **(SAC (D))**, (3) SAC + Replay Buffer **(SAC + RB)**: we initialize the replay buffer with expert demonstrations.

- **Hindsight Experience Replay (HER)** is a well known technique [6], which uses the previous experiences to learn overtime.

- **SAC+MCAC**. This baseline uses the soft actor critic (SAC) algorithm [40] as well as Monte Carlo augmented Actor-Critic [143].

- **Advantage Weighted Actor Critic (AWAC)** follows the methods described

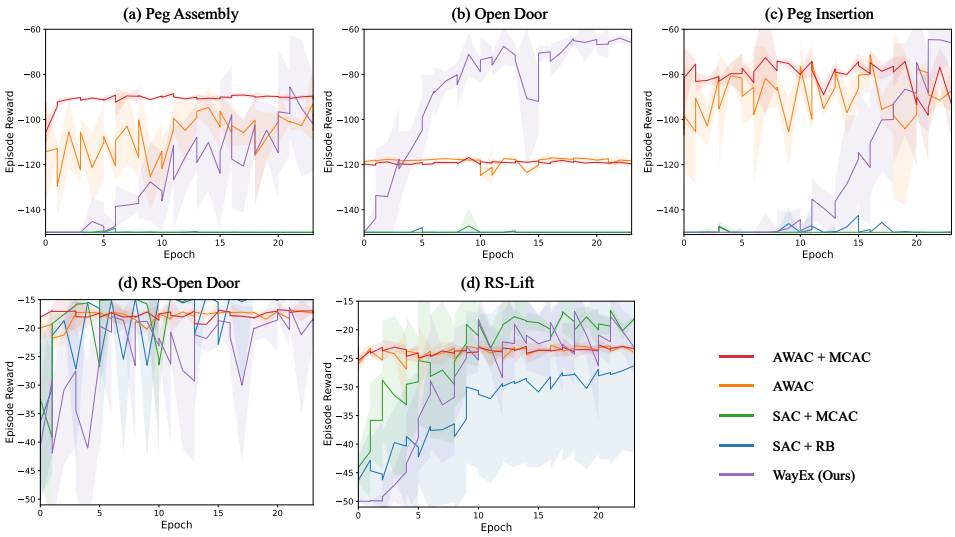


Figure 4.6: This figure shows the results of several baselines when they are given one hundred expert demonstrations **(a,b,c)** shows the results of the Meta World [152] environments when trained using AWAC [88], MCAC [143], SAC + RB and AWAC + MCAC. **(d,e)** shows the results of these same baselines on the robosuite tasks [163]. Our method continues to use only one demonstration.

in [88].

- **AWAC+MCAC** uses [88] along with a modified reward as describe in [143].

4.4.4 Results

We evaluate several different combinations of baselines each of which have different requirements in comparison to our method.

No Action Space

First we look at methods that do not require access to the action space of an expert demonstration in order to learn. In Figure 4.4(d), we analyze the performance of our method compared to other conventional reinforcement learning algorithms for the pick and place task. These graphs reveal two significant observations: (1) WayEx exhibits a remarkable acceleration in the learning process with just a single demonstration, over SAC. (2) WayEx demonstrates a considerably lower standard deviation compared

to the other algorithms. We hypothesize that in general, methods relying on sparse rewards requires a degree of luck. For that, the environment must be explored extensively until a reward is obtained, allowing the method to learn the task. However, WayEx circumvents this by guiding the agent towards the goal, irrespective of the initial start point.

In Figure 4.4(a,b,c), we examine the outcomes of training three Meta World environments [152] using the SAC algorithm. These results are compared against a sparse reward and an episode-specific dense reward. Notably, we encountered difficulties in implementing hindsight experience replay with these environments. The Figure 4.4(a) represents the results of the open door task, (b) displays the results of the peg insertion task, and (c) showcases the peg assembly task. Across all these environments, we observed that SAC was unable to solve the tasks effectively when utilizing sparse rewards. For the open door and peg insertion task WayEx performs similar or better than the dense reward. The dense rewards have been finely tuned to each task and can require a lot of trial and error to finalize, while WayEx is a general reward that can be applied to any method. Regarding the peg assembly task, we discovered that SAC was unable to solve the task even with the dense reward during the training period. This is due to increased complexity of the task, which results in a noisier reward signal. This task is more difficult because there are a greater number of objectives to be accomplished. However, our method proved capable of swiftly solving the task despite these challenges.

One Expert Demonstration

Next, we look at our method compared to baselines that require just 1 example. We compare against AWAC, AWAC + MCAC, SAC + RB and SAC + MCAC when just one expert demonstration is used. The results of this are shown in Figure 4.5. We find that for the three meta world tasks our approach significantly outperforms the

other approaches. AWAC and AWAC + MCAC work in the Peg Assembly and Door Open tasks, but they are not able to solve the problem as well as our approach is. For the RS-Door Open task, we find that our approach performs very similar to all other baselines. This is likely because the task is very easy and requires only a small amount of data. The RS-Lift task results look similar to the MetaWorld results where our method significantly outperforms the others. This task was more difficult than others due to the rotation of the block being different in each episode, but our method is able to handle it nonetheless.

100 Expert Demonstrations

Finally, we look at our method compared to the same baselines when the baselines use 100 expert demonstrations. The results are shown in Figure 4.6. Note that AWAC, which is the method that performs the best in these scenarios has to be pre-trained in addition to the online training. We find that although our method takes more online training time with just one demonstration, versus 100 demonstrations, our method performs equal to or better than the baselines in all of the tasks.

4.5 Conclusion

We present WayEx, a new approach that enables training reinforcement learning models using a single demonstration. Unlike other imitation learning methods, which typically rely on multiple demonstrations and access to detailed action information, WayEx can operate with limited information and single demonstration. In order to achieve this, we introduce a novel universal reward function and leverage a knowledge expansion technique that extends beyond initial start and goal states. This makes it highly suitable for learning tasks with minimal information across different environments. We show that WayEx is faster than standard reinforcement learning models, in cases

where the rewards are sparse or dense, and showcase its ability to succeed where other approaches fall short. Additionally, we show that WayEx performs similar to or better than a variety of imitation learning methods when these methods use either one or one hundred expert demonstrations. In future research, we aim to explore the use of expansion for non-linear states and investigate the utilization of image-based state spaces rather than joint locations.

Acknowledgements: This work was partially supported by DARPA SAIL-ON (W911NF2020009) program and NSF CAREER Award (#2238769) to AS.

Chapter 5

Dynamic Interaction

This chapter explores how simple robotic skills, typically developed in controlled lab settings, can be adapted for real-world use through dynamic planning. The central contribution is a novel planning framework that combines three components: a static policy trained in a fixed environment, an Estimated Time of Arrival (ETA) network that predicts how long the robot will take to reach its goal, and a trajectory prediction network that forecasts the motion of objects in the scene. This approach addresses a core limitation of current imitation and reinforcement learning methods, which often assume a static environment i.e., objects remain stationary unless moved by the robot. In real-world scenarios, however, such assumptions rarely hold. Since collecting real-world training data is costly and challenging, leveraging planning to adapt static policies to dynamic settings is a promising step toward robust, real-world robotic deployment.

5.1 Introduction

Humans have an innate ability to understand the world around us. We do not have to think twice when we dodge people on the street or prevent something from falling off the table. Robots, on the other hand, are relatively inept at performing the same

tasks. They cannot adapt to dynamic situations in the same way people do. In the past, robots have been utilized in specialized environments, where assumptions can be made about the surrounding environment and the object location. The goal of this project is to leverage recent advancements in robot learning, path planning, and trajectory forecasting to enable robots to act in dynamic environments.

In a typical robotic planning problem, the robot observes the environment, plans to complete a specific goal, and then executes one step of that plan, repeating this process until the goal is met. However, in a dynamic environment, accomplishing this is infeasible – as a robot plans to meet a specific target, the real target will have already moved. Employing a typical planning approach to solve this (e.g., pure pursuit control) results in a robot following behind the target, often without reaching it. In this work, we will address this problem by planning for a possible future location of the object and then adjusting this plan as necessary, to reach the final target position *at the same time as the actual object*.

The field of dynamic targets is relatively under-explored, other than tasks like catching a ball [60, 65, 113, 114, 154]. However, the task of catching does not require the exact timing; for example, an agent can go to the goal location early and wait for the object. Compare this to our setup of trapping a rolling ball with a box. If the robot reaches the location early, it misses the ball entirely. We demonstrate that synchronizing the agent's timing and the moving object is critical in completing such dynamic tasks. With the notable exception of [154], most other approaches for solving dynamic tasks rely on solving for the physics that represents an object, which fails to generalize beyond the specific task these equations are designed for.

Solving the simple trapping task, raises several research challenges. The robot needs to know a goal location to go to, but predictions of object movements in the real-world are noisy. The standard solution of re-planning needs to work in tandem with the future location predictions. The speed at which we get the predictions needs

to be in sync with the speed at which the robot needs them. The approach needs to be real-time, and therefore should use as little computation as possible. The forecasting approach should implicitly learn the underlying physics in the environment, such as the initial velocity of the object and friction, which dictate the movement of objects. Finally, the solution should not require re-training for every possible dynamic setup.

Our key contribution is an approach to utilize a static planner for dynamic tasks using a **Dynamic Planning add-on**; i.e., if we can successfully solve a task with a static target, then our approach can solve the same task when the target is moving. Our planner integrates a given static planner with two new networks, a trajectory forecasting network, and a robot's estimated arrival network. Using these three networks, our planner is able to determine an appropriate goal location and roll out actions that the robot should take at each step of an episode to interact with a dynamic object. We demonstrate the effectiveness of our approach on the task of trapping a moving ball and dynamic FetchReach [32].

5.2 Related Work

Motion forecasting: Forecasting the trajectory of an object to improve robotic manipulation has been studied for decades (e.g., at least as early as 1995 [101]). Since then, the interplay between prediction and robotics has been studied in many different scenarios. In recent years, a popular paradigm to study trajectory forecasting is social situations with many moving actors [4, 91]. Most of these approaches build on Recurrent Neural Networks, or variants, to predict actors' future movements. Generally speaking, the input and output sequences are of the same short length, limiting their applications in robotics problems, which require long-horizon predictions. Such approaches require a prohibitive amount of memory and time to make accurate long-horizon predictions.

A popular use of trajectory forecasting in robotics is predicting how a robot will interact with objects to perform specific tasks [33–35]. However, much of this comes down to learning, or modeling, the physics of an object and the robot, and predicting how the objects will interact with each other. To accomplish these tasks, the algorithms must simulate thousands of possible futures to find the right one. This makes them unlikely to work in environments with dynamic objects, because by the time an action is decided on, the required interaction will likely need to be different.

Due to the issues highlighted above, many recent robotics works have resorted to using physics to accomplish their tasks [65, 113]. For the most part, current techniques require large amounts of domain-specific data pre-processing and post-processing, to manually extract velocity, acceleration, position, etc. [127]. Such processing may require information about the environment, that ideally, we would want to learn.

Dynamic tasks: There are a few recent works that address the task of moving targets. In [32] authors take the original OpenAI Gym environments and develop an end-to-end solution, where a reinforcement learning model tries to hallucinate the future. However, they move the objects along a straight predictable path, not taking into account any physics. Additionally, [62, 144] address playing ping pong and catching a ball; but as discussed above, they both rely on accurate physics modeling to plan future paths.

Similar to our approach, [25, 154] solve for dynamic goals using future prediction. However, [25] predicts the future, taking both actions and object movement into account, as opposed to just the object movement. This exponentially increases the number of possible futures and adds complexity, without a clear benefit. [154] only addresses the problem of catching, which doesn't require as accurate timing as our task. Additionally, they feed their entire future prediction into the action policy. This implies that not only do they have to run their forecaster at each step, which is computationally expensive, but for each task, they have to train both the forecaster

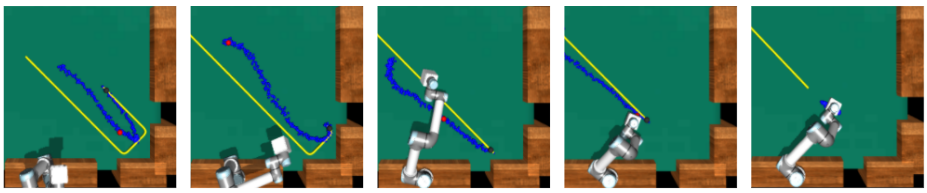


Figure 5.1: Our goal in the Trapper task is to train the robot arm to trap the ball, shown in grey, while it is still moving. The velocity and direction of the ball are randomly set at the beginning of each episode, as is the starting location of the ball. The yellow line represents the known trajectory of the ball, which can be found by running the same episode twice, and the blue line represents the predicted trajectory given only past locations. The arm utilizes the predicted trajectory when selecting a future location, shown in red, to target.

and the model-predictive controller; making it difficult to evaluate their generalization to other tasks. We decouple these two tasks, which leads to a more general solution that can be applied to many different problems.

5.3 Method

Our key contribution is a mechanism to retool a static planner using a Dynamic Planning add-on. This implies that if you can successfully solve a task with a static target, then you can solve the same task if the target is moving. A task-centric interpretation would be that our proposed add-on 'reduces' a dynamic task to static task, which can be solved by off-the-shelf static planners.

Our proposed add-on, Dynamic Planner, has three main components that work in tandem to complete a dynamic task. First we describe the dynamic task used in this paper (Section 5.3.1), followed by the building blocks of our model (Sections 5.3.2, 5.3.2, 5.3.2), and how they are integrated to work together (Section 5.3.2).

5.3.1 Task definition

The goal in our task, **Trapper**, is to trap a moving ball using a robotic arm, visualized in Figure 5.1. To be successful, the agent needs to correctly synchronize the time when the attached box covers the goal location and when the ball is at that specific

location. If the box traps the ball, by covering it, the episode is considered a success. We chose the UR5 robot because of its widespread use and several successful sim2real demonstrations [99, 126]. We use a box attached to the UR5 arm, instead of a gripper, to focus on the dynamic aspect of the Trapper task, and omit the added difficulty of getting the gripper in *just* the right position for manipulation. We used the MuJoCo [133] physics simulator to model Trapper.

At the start of each episode, the ball is placed at a random location on a billiard table. A total of eight pockets imply that the ball can roll off the table at any point. This forces the robot to trap the ball while it is still moving, as opposed to the shortcut of waiting for the ball to come to a halt before attempting to trap it. In addition to considering the ball rolling off the table, the planner must learn to predict and deal with the ball bouncing off the walls of the table. Across episodes, we randomize the starting location and initial velocity of the ball. Also, during some test episodes, we change the friction of the table.

Next, we describe how we utilize the static planner for the Trapper-Static task (trap a ball that is not moving) to solve the Trapper-Dynamic task (trap a rolling ball).

5.3.2 Overview

Our approach builds on three building blocks: a static planner, a trajectory forecaster, and a network that predicts how long it takes the arm to reach any particular goal from its' current location (or an estimated time of arrival network). We first describe each of these components and then present how they interact with each other to solve dynamic tasks.

Static Planner

Our approach utilizes a static planner. To demonstrate the different possible applications for our algorithm, we use two types of static planners, utilizing model-free reinforcement learning and inverse kinematics, to plan a trajectory towards the goal location.

Model-free Reinforcement Learning Planner (RL): We use Deep Deterministic Policy Gradients [74] with Hindsight Experience Replay (HER) [5], to solve for the static planner for the Trapper-Static task. Our policy maps the state space S and the goal space G of the environment to an action: $\pi : S, G \rightarrow A$. We will designate the states corresponding to the robot arm as s^R and the current state of the object as s^O. Given a static goal, $g \in G$, our policy for the arm is $\pi_g(s^R)$, for any $s^R \in S$. In Section 5.3.2, we will describe how this policy can be utilized to solve dynamic tasks.

We define the reward space used to obtain $\pi_g(s^R)$ as $Q^{\pi_g}(S, A)$. HER typically works best with a sparse reward, $Q^{\pi_g}(S, A) = 1$ if we have reached goal state, g, and $Q^{\pi_g}(S, A) = 0$ otherwise. However, in the specific case of the Trapper task, it is difficult for the arm to learn how to align the box horizontally with the table to trap the ball without an additional reward. We address this by solving for a polynomial reward $R(\theta_{\text{box}})$, where θ_{box} is the angle between the surface normal of the open-face of the box and the surface normal table. We boost this reward by multiplying by a function β, which increases as the box nears the ball. This ensures that the box is in the right orientation to trap the ball. In addition, we give a sparse reward based on whether or not the ball is successfully trapped. Therefore, our final reward function is:

$$Q^{\pi_g}(s^R, a) = \left[\mathbb{1}\left(s^R(\text{box}) == g\right) + \beta R(\theta_{\text{box}})\right], \tag{5.1}$$

where $s^R(\text{box})$ is the location of the box and $==$ checks if the box has reached the goal, with some margin of error.

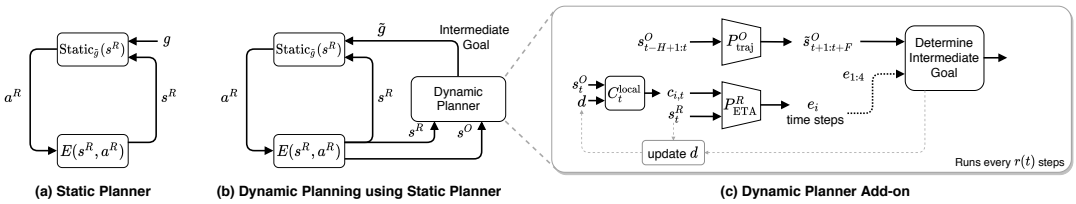


Figure 5.2: **Planner Overview:** (a) How a simple planner would be run in a static situation. At each step the planner, Static$_g$ (π_g or IK_g), takes in s^R, the state of the robot, as well as g, the goal location. It then outputs a^R, the action for the robot to take in the environment. The action is simulated in E and the new state is outputted. (b) The Dynamic Planner is integrated with the static planner, taking in s^R as well as s^O, the state of the object and outputting the $\tilde{g}$ to be used in the static planner. (c) The Dynamic Planner is extended to show how P^R_{ETA}, the ETA network for the robot, and P^O_{traj}, the trajectory forecasting network are used to find $\tilde{g}$.

We use the Stable Baselines [46] implementation of DDPG [74] and HER [5] to train this static planner on the Trapper-Static task. After training, this planner achieves $\sim 75\%$ success rate on the Trapper-Static task. Failure cases typically occur if the ball is too close to the arm's base or out of the arms reach. We do not re-train this static planner when adapting it for dynamic tasks. Figure 5.2(a) illustrates a static planner during inference. Since we use an off-the-shelf static planner we refer the reader to [5, 46] for more details.

Inverse Kinematics Planner (IK): For the second static planner, we use the UR5 inverse kinematics and the dampened least squares method to solve for the trajectory of the arm [10]. Ideally, when using an inverse kinematics planner you would want to plan the entire arm trajectory to the goal each time the goal location changes, however this is too time consuming for a dynamic problem. To mitigate this we solve for 5 steps of the path at a time. This gives the flexibility to change the goal location as the episode rolls out. We will refer to this planner as $IK_g(s^R)$ where, the planner takes in the state of the robot, s^R, and outputs a^R, the next action for the robot to take. Unlike the RL planner, this planner reaches $\sim 100\%$ success rate.

Trajectory Forecasting

Trajectory forecasting has been widely studied in computer vision and autonomous vehicle communities (Section 5.2). However, many of its applications (e.g., [4]) do not require the kind of long-horizon real-time forecasting that is needed for dynamic robotics. Most trajectory forecasting techniques use Recurrent Neural Networks (RNN) or variants, which have memory and time limitations for long-horizon predictions.

Instead, we opt for a faster, albeit potentially less accurate, approach [91] that requires little knowledge about the actual environment and tries to predict F steps into the future given H steps of past information, where $F >> H$. Therefore, we sacrifice some accuracy to gain a real-time long-horizon prediction. This is necessary because, unlike in autonomous driving, the episode happens in a very short period of time. We cannot wait to gather a lot of prior experience. In our specific example, we use 24 steps of prior information to predict 300 steps of future information. The lower prediction accuracy has modest impact on the success rate at high velocities, but it satisfies the necessary time constraints, making it an attractive choice for this scenario.

Our trajectory forecasting network uses [91], which uses Convolutional Neural Networks (ConvNets). Unlike RNNs, which make sequential predictions, ConvNets can predict long-horizons in a single-shot leading to faster prediction in our setup. Our selection was also, in part, inspired by the findings of [8]. The state of the object (i.e., its location) at time t, is denoted by $s_t^O = (x_t, y_t, z_t)$. At the current time t, the network takes H past object locations, $s^O_{t-H+1:t}$, and the current location as inputs and outputs predictions for future locations for F time steps. This network can be defined as

$$P^O_{traj}(s^O_{t-H+1:t}) = \tilde{s}^O_{t+1:t+F} \tag{5.2}$$

where $\tilde{s}^O_{t+1:t+F}$ represents the predicted object locations at $[t+1, t+F]$ time steps .

These predicted locations are relatively accurate for earlier time steps, but get more inaccurate as the time gets further from t. This inaccuracy can be seen in the illustrated dashed blue line in Figure 5.3. However, given the design of our dynamic planner, the predicted trajectory need not be exact as long as it is generally in the right direction. If this condition holds, our algorithm moves the arm in the right direction, so it is always getting closer to the goal location. We quantify this by showing that results using our predicted path are close to the results when we use the oracle future path.

Estimated Time of Arrival (ETA) of the Arm

In previously studied dynamic environments, it did not matter when the robot arrived at the goal location, so long as it did so before the object. For example, when catching a ball as long as the robot is there when the ball reaches the desired location, the robot can successfully catch it. This is also true in other popular dynamic problems, such as ping pong.

Trapper-Dynamic is different because it requires coordinated arrival timing between the object and the robot. If the robot reaches the target location too early, the box will already be on the table by the time the ball arrives, causing the ball to collide with the box instead of properly trapping it. Likewise, if the robot reaches the target location too late, it will miss its chance to trap the ball entirely. Therefore, the crux of our approach is that we have to time the drop of the box with the exact time the ball will be at the drop location. In order to coordinate this, we define an ETA function $P^R_{ETA}(s^R, g)$, which for every $s^R \in S$ and a given goal location, g, provides an estimation for how long it will take our static solver to reach g from state s^R.

We use a simple Multi-layer Perceptron (MLP) network to model P^R_{ETA}. To get the data to train this network, we run Trapper-Static using the static solver multiple times, for many different g, sampling five intermediate states of the robot between

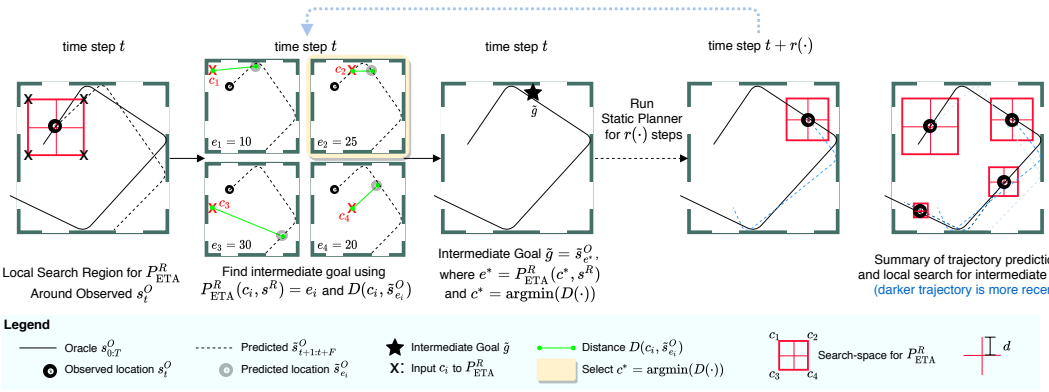


Figure 5.3: The progression of how our dynamic planner selects the new target location.The first image shows the box with edges of size $2d$ drawn. From here we run the ETA network on the corners and find the corresponding points on the predicted trajectory. We select the point on the trajectory closest to the corresponding corner as the new $\tilde{g}$. This is repeated every t_{TTR} steps.

the initial state and the goal location. We then calculate how long it takes to reach the goal location from each of those states. We discretize the ETA window of 500 time steps into 100 bins (e.g., 0-4 steps, 5-9 steps, etc.). In order to figure out which of these bins a state falls into we structure our network with 3 linear layers each with a ReLu activation network and a final layer which is also linear and outputs a probability to 100 different buckets using softmax. At training time, we use both the state and the goal as input and classify each pair into one of the bins. We use softmax cross-entropy loss for optimization. We observed that the discretized ETA network led to better convergence than a regression model, and the possible error of up to 4 time steps for each classification did not have any practical impact.

Dynamic Planner

At each time step, our static planner takes in the observed state and the goal, and outputs an action. To utilize this planner for dynamic tasks, we can modify the goal location g input to planner, and there-in lies the challenge. The trajectory forecasting network can give potential future locations of the ball, but its unclear which location should be used as the goal. A simple solution is to use the current location of the

ball at every time step s_t^O (i.e., change the goal every time step) and expect that the arm and ball will eventually converge (e.g., pure pursuit control). However, as we demonstrate in experiments, this results in the arm following the ball and is unable to deal with bounces and drop-offs effectively. Another extreme option is to use a long-horizon future (s_T^O, where $T >> t$) as a goal and have the arm wait for the ball. However, this assumes precise trajectory prediction, which is erroneous. In our approach, we aim to achieve a balance between these extremes, by utilizing the trajectory forecasting and ETA network together.

Our dynamic planner uses the trajectory network and the ETA network to determine an appropriate goal location, which should ideally reflect the location where the ball and the arm will reach at the same time. However, since the outputs of trajectory network and ETA network are noisy, we need to repeat this process, or re-plan, as time progresses. That said, deciding when to re-plan, is a problem within itself. Ideally, we want to re-plan as little as possible, because it is time-consuming. On the other hand, re-planing means that our initial target location does not have to be accurate, as long as it is in the vicinity of where the ball will be.

Re-planning: To address this, we propose a simple approach that finds a balance between these two by shrinking the re-plan window each time, that is, as the arm gets closer to the ball, we re-plan more often, allowing the algorithm to zero-in on an accurate target. Our time to re-plan is given by $t_{\text{TTR}} = \max(\gamma t_{\text{TTR}}, 25)$. We initially set t_{TTR} to be 75 and γ to 0.9. We don't allow t_{TTR} to be less than 25. This method allows the target to be fairly inaccurate at the beginning of each episode and zero in on the correct location later on. Therefore, our approach determines an appropriate goal location, which we will refer to as intermediate goal $\tilde{g}$, and t_{TTR}, it then runs the static planner for t_{TTR} time steps before it re-plans. This process is illustrated in Figure 5.2(b).

Finding intermediate goals: The discussion so far alluded that, given the trajectory

network and the ETA network, determining an appropriate goal location is straightforward, which is untrue. Next, we describe why this is the case and our approach to determine $\tilde{g}$.

At each re-planning step, the trajectory forecasting network predicts $\tilde{s}^O_{t+1:t+F}$ and we want to find the new value of $\tilde{g}$ from this trajectory such that the following is true, for a current state s^R,

$$\tilde{s}^O_{e^*} = \tilde{g} \text{ and } P^R_{ETA}(s^R, \tilde{g}) = e^*, \tag{5.3}$$

that is, both the arm and the ball will reach $\tilde{g}$ in e^* time steps. The problem with these constraints is that they have cyclic dependence, where we need e^*, which is some unknown future time step on the trajectory prediction to find $\tilde{g}$, and we need $\tilde{g}$ to find what e^* should be from the ETA network. To resolve this, we could use a brute force approach and get e_t at every time t on the predicted trajectory until we find an e_t that satisfies eq. 5.3. However, this is not practical for a real-time setup, at least not without employing a computational cluster. Instead, we devise the following algorithm to approximate $\tilde{g}$.

As illustrated in Figure 5.3, we begin by considering a local search region (shown as a red box) around the current observed location of the ball s^O_t. To determine the size of this box, parameterized by d, we solve

$$d = D(\tilde{s}^O_{e_t}, s^O_t), \text{ where } e_t = P^R_{ETA}(s^R, s^O_t) \tag{5.4}$$

where D is the distance function. The intuition is to shrink the local search region proportionally as the arm nears the ball's current location. If the arm is getting closer to the ball, then this number should shrink as the episode progresses.

Once we have d, we take the current location $s^O_t = (x, y, z)$ and find the four corners of our square search region, located at $(x + d, y + d, z), (x + d, y - d, z), (x - d, y - d, z), (x - d, y + d, z)$. Note that for our current experiment, there is no need

to modify z because everything is in on a 2D plane, but it's easy to do so. Let these four locations be $c_1, c_2, c_3, c_4 \in C$. We now solve for $e_i = P^R_{ETA}(s^R, c_i)$ and look at the trajectory to select the following points $\tilde{s}^O_{e_i}$. Out of these, we select the point in $\tilde{s}^O_{e_i}$ that satisfies

$$c^* = \arg\min_{c_i}\{D(c_i, \tilde{s}^O_{e_i}); \forall c_i \in C\} \tag{5.5}$$

c^* is our best guess at a location close to the actual trajectory where the arm and the ball will reach the same location at the same time. We use c^* to find a good approximation for e^*, so we can select a point on our estimated trajectory where we predict the arm and ball will meet at the same time. This gives us $e^* \approx P^R_{ETA}(s^R, c^*)$. We can determine the intermediate goal as $\tilde{g} = \tilde{s}^O_{e^*}$ from the original constraint, eq. 5.3, and pass that into our static planner. Simplified versions of the Dynamic Planner and the intermediate goal finding algorithm are illustrated in Figure 5.2(c) and 5.3 respectively.

5.4 Experiments

We first present experiments on the Trapper-Dynamic task, described in Section 5.3.1, and then report results on a dynamic version of the FetchReach environment from [32].

Implementation Details: We train our model in two separate stages. First, we obtain Trapper-Static using either online reinforcement learning or by solving for the inverse kinematics of the arm. After we have a functioning static model, we simultaneously collect data from MuJoCo for our ETA network and our trajectory prediction network. We do this by using Trapper-Dynamic, but keeping the value of g static, so we can observe both the trajectory of the ball and the path of the arm towards one location. We collected ETA data, as discussed in Section 5.3.2. In order to get diverse data for the trajectory prediction network, we run each episode four times longer than usual and then sample five random trajectories of length $F + H$.

We use this data to train our ETA and trajectory prediction network offline.

5.4.1 Baselines

Target Pursuit

This baseline, inspired by pure pursuit control, is the simplest way to solve the dynamic problem. At each frame t, we update $\tilde{g}$ to be s_t^O.

Oracle Trajectory

We use an oracle for the trajectory, instead of predicting the trajectory. We obtain the oracle by running the same episode twice. On the first run, we don't move the arm so that we can collect the ball's trajectory with no arm interference. On the second run, instead of feeding in the output of P_{traj}^O to determine the new $\tilde{g}$, we instead feed in the oracle trajectory. Note that if the arm interacts with the ball (e.g., touches it), then the ground-truth trajectory will be different from the oracle trajectory.

5.4.2 Experiment Setup

We run two different experiments. First, we change the ball's initial velocity across episodes, randomly selecting the x and y components. Second, we change the friction loss of the ball, which is a simple way to increase the friction between the ball and the table. Note that all networks were trained with the first variant, but the second variant is never observed during training, and thus, evaluates generalization.

Number of episodes and time steps

To obtain a data point, we run our algorithm on 100 episodes of Trapper-Dynamic with the same initial settings. For each initial setting, we collect 10 data points and plot the average and standard deviation on the resulting graph. We ran all our

experiments over 500 time steps in MuJoCo.

Metrics and 'Solvable' episodes

For each data point, we collect two different metrics: (1) the success rate over all 100 episodes, and (2) success rate over all 'solvable' episodes of those 100 episodes. To determine if an episode is solvable, we use an oracle ETA network, which knows the exact amount of time it takes from the initial arm location to reach every point on the table. We first get the oracle trajectory, and then, observe if the arm can reach any point on the trajectory before the ball reaches the same point. If this is possible we say the episode can be solved, that is an episode is solvable if $e_t \leq t$, where $e_t = P^R_{\text{ETA}}(s^R, s^O_t)$.

5.4.3 Results on Trapper-Dynamic

Our results are visualized in Figure 5.4. Except for cases where the ball barely moves (velocity < 1.5 or friction > 5), both our models outperform the target pursuit baseline by about 20%. In the target pursuit approach, the arm trails behind the ball, and because the arm moves slower than the ball, it never catches up. Our approach accounts for this by moving towards a future location, saving time that might have been spent following the ball. This allows it to reach certain locations before the ball. As the ball gets faster, the value of $\tilde{g}$ has to be more accurate; this is why we see our success rate fall when we increase speed or decrease friction.

The results of 'oracle trajectory' and our predicted trajectory are similar across both the RL and the IK static planner, except for high-velocity episodes utilizing the IK planner where using the 'oracle' trajectory is better. This is potentially because the inverse kinematics solvers are more exact and can benefit from very precise trajectories. However, since the results are still relatively close, we can infer that accurately synchronizing the arm's timing with the ball is the difficult part of

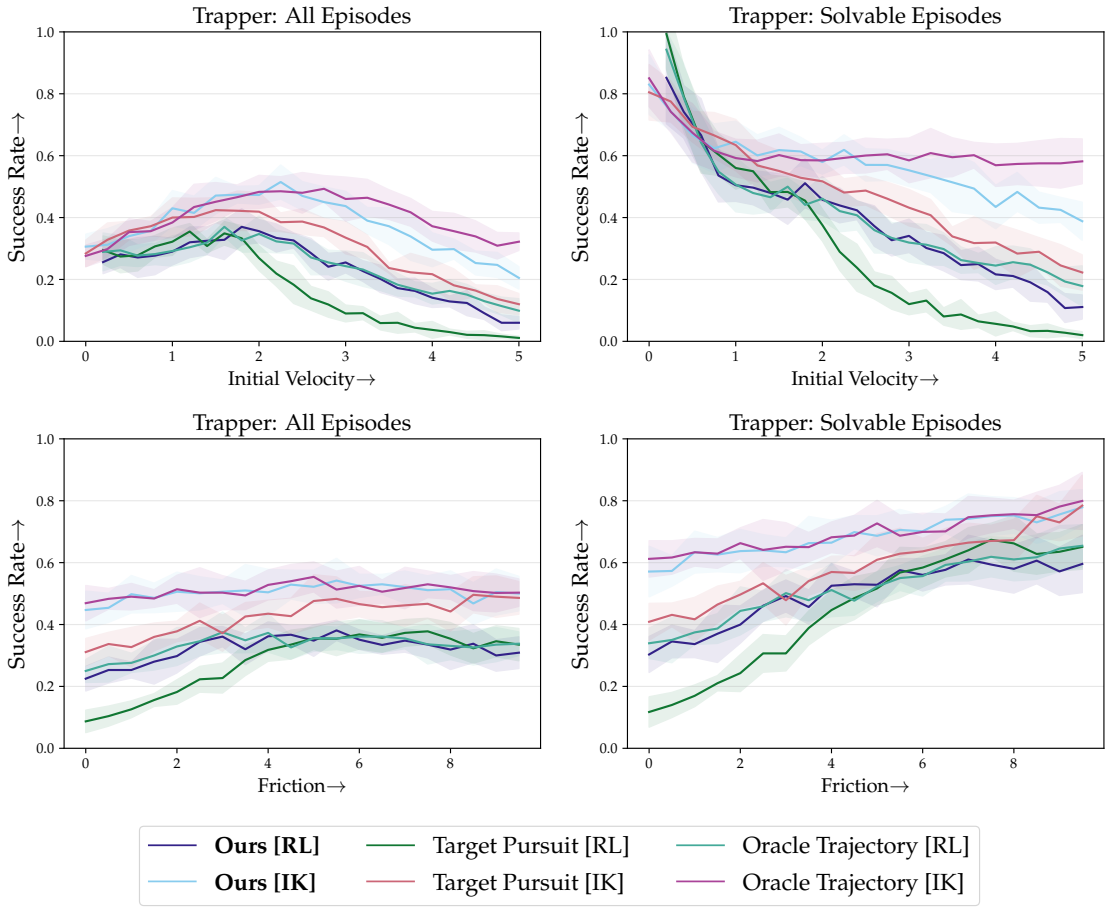


Figure 5.4: Success rates for our Trapper-Dynamic environment. On the top we show the results for initial velocities between 0 and 5. On the bottom we show results for friction loss between 0 and 10. On the right are the success rates for both experiments for episodes we deem solvable. On the left are the success rates for the experiments over all episodes.

the problem. In other words, noise in the trajectory prediction (or $\tilde{g}$) is not very significant as long as the overall predicted direction is correct.

5.4.4 Results on Dynamic FetchReach

To compare our approach to a recent state-of-the-art reinforcement learning algorithm, we run our preliminary set of experiments on a dynamic version of the FetchReach environment. This is one of the environments used by [32] to evaluate Dynamic Hindsight Experience Replay (DHER). This task is much easier than Trapper-Dynamic because there is no friction, and there are no collisions. As can be seen in Figure 5.5(right), our approach significantly outperforms [32]. At the largest gap, our approach outperforms it by almost 60%. This demonstrates that our approach can easily be successfully applied to other environments. Note that we were unable to get any recent state-of-the-art approach for dynamic tasks work on our Trapper-Dynamic task.

5.4.5 Ablation Analysis

RL vs. IK Static Planners

We evaluate the differences between the RL and the IK static planners for our task. Given our current setup, the IK planner outperforms the RL planner by ∼20%. The key challenge for solving Trapper using RL was finding a static policy that could change its target location mid episode. We anticipate that the RL policy not being agile enough to change target locations explains a large percentage of the gap between the two planners. However, we still see value in experimenting with the RL planner. One advantage is speed, the RL planner takes <1ms to find the next action, while the IK planner takes ∼15ms. Additionally, the RL planner is better able to recover from a variety of failure modes. E.g., if the box hits the table and/or misses the ball, the

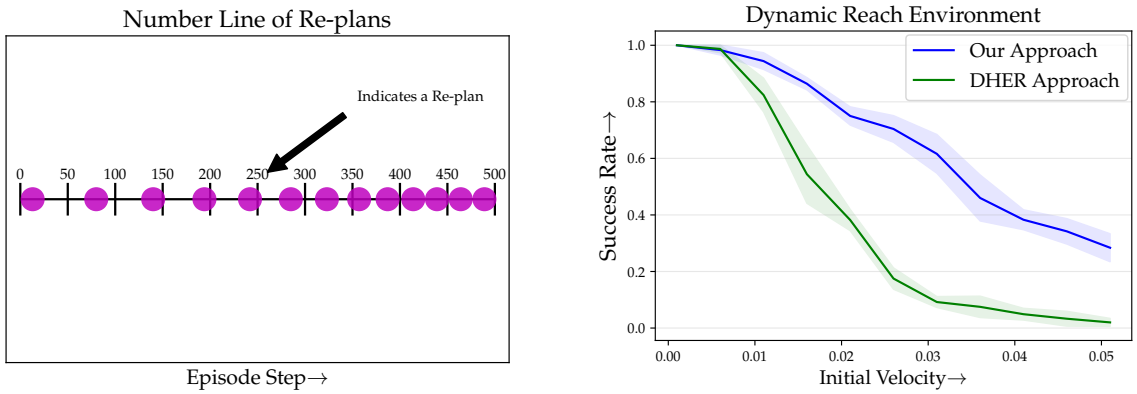


Figure 5.5: **(left):** A visualization of when re-plans occur during an episode. **(right):** Success rate of our algorithm as well as the DHER [32] algorithm on a dynamic version of the FetchReach environment from OpenAI Gym [5]. We evaluate both methods for different velocities of the ball.

RL planner has learned to lift the arm and try again, but the IK planner is unable to recover.

Timing Analysis

Since the entire episode occurs in $<$ 2sec, the timing of the entire planning process is important. When the re-plan strategy is fixed, it leads to 13 re-plans over the course of a 500-step episode (visualized in Figure 5.5(left)). On average the intermediate goal search takes 35ms running on 1 GPU, which includes 4 forwards passes of the ETA network, 1 forward pass of the Trajectory Network, and all of the data processing required to find the target location. Given these numbers, we can see that our algorithm uses an appropriate number of re-plans and that re-planning at each time step is too slow for this task.

5.5 Conclusion

We presented an approach to adapt static models to work with dynamic targets. Our approach is intuitive, easy to implement, and can easily be adapted to newer environments. We showed that we could synchronize the timing between a moving target

object and a moving robot, which has has not been widely studied in learning-based robotics. Results demonstrate that the approach generalizes to different environment parameters and that our approach can work with noisy future predictions.

Acknowledgements: This project was partially funded by DARPA SAIL-ON program (W911NF2020009). AS would like to thank James Davidson and Rahul Sukthankar for helpful early discussions.

Chapter 6

Future Work and Conclusion

6.1 Keypoint Based Reinforcement Learning

Building on my work in few-shot reinforcement learning and state representation from Chapter 4 and Chapter 3, I will explore how these methods can be combined to enable the deployment of WayEx in a real-world robotic setting. A key challenge in the original formulation of WayEx was its reliance on privileged state information available only in simulation to compute the distance between two states. In the real world, this distance is difficult to measure accurately, as comparing image features lacks the precision required for WayEx to function effectively. However, it is feasible to compute the distance between two sets of keypoints, which may provide a practical alternative.

We propose a method that seamlessly integrates the approaches from both papers by using P3-PO features to train a reinforcement learning model with the WayEx reward function. To enable generalization across different object locations, we represent the state as a graph of keypoints rather than using the keypoints directly. In this graph, edges encode the pairwise xx, yy, and zz distances between keypoints. This graph-based representation introduces translation invariance, as it captures the spatial

relationships between objects rather than their absolute positions in the frame.

6.2 3D Based Keypoint Representation

To strengthen the keypoint-based representation, it must be made more robust to occlusions of both objects and the robot—limitations that the current setup cannot adequately address. One potential solution, particularly when using multiple cameras, is to construct a 3D point cloud and perform tracking in 3D rather than 2D. This approach would likely mitigate issues caused by occlusion. However, 3D tracking would likely be both computationally intensive and time-consuming. Moreover, there are limited current computer vision methods that track directly in 3D space.

Another way to improve robustness to occlusions is to track the same keypoints across multiple cameras, as described in P3-PO, and develop a consensus-based approach for determining their locations. With this method, even if a point is occluded in one or two views, it can still be reliably tracked using information from the remaining cameras. Furthermore, this approach allows for correction of the point's location in views where it was previously occluded, enabling those cameras to resume contributing to tracking once the point reappears.

6.3 Learning From Humans

Finally, I will extend the work presented in this paper to develop an improved training pipeline for learning from human demonstration videos. I hypothesize that keypoints in human videos can be tracked similarly to P3-PO, with the human taking the place of the robot. However, unlike P3-PO, this approach cannot rely on access to the actions taken between states, as the human and robot action spaces are not directly comparable. This limitation motivates the use of keypoint-based reinforcement learning, which enables learning from state-only observations without requiring explicit

action labels.

We hypothesize that collecting multiple demonstrations, rather than relying on a single one as in WayEx, could significantly accelerate training by enabling a denser and more informative reward structure. If these demonstrations are provided by humans, the data collection process becomes much more scalable and cost-effective, as it eliminates the need for labor-intensive teleoperation. By gathering a diverse set of "optimal" states from human demonstrations, we should be able to train a robot to perform the same task efficiently and with minimal supervision.

6.4 Conclusion

In this thesis, we explored a range of approaches to address the current limitations in robotic learning. In Chapter 2, we investigated the use of a generative model for representing human poses invariant to the camera pose. We found that this model performs well for both interpolation and extrapolation of pose. However, we also observed that many pose representation methods are prone to overfitting to specific datasets, making it challenging to generate poses outside the distribution of the training data. To mitigate this, we found that incorporating synthetic data can help improve generalization and reduce overfitting.

Chapter 3 explores an alternative approach to state representation in robotic environments. Unlike the more human-centric focus of earlier work, this chapter centers on simplifying environmental representations to enhance the generalization of robot policies. We found that by significantly reducing the complexity of the state, we could substantially improve policy performance. In comparison with other generalization focused methods such as point cloud based approaches our method achieved improvements of over 70%. This work also highlights the advantages of a keypoint based representation over purely image based policies, which tend to be more

sensitive to factors like lighting changes and background clutter.

In Chapter 4, we addressed the difficulty of training policies efficiently via reinforcement learning. Many existing RL methods are either prohibitively slow to train or require so much expert supervision that they become indistinguishable from imitation learning. Additionally, many approaches depend on hand-tuned reward functions, which are often hard to define, especially for real-world tasks. In contrast, we propose using expert demonstrations in combination with a sparse reward to define a more general and scalable reward mechanism. While most prior work assumes access to both states and actions, we focused on learning solely from expert states, a critical step toward learning from human demonstrations, where action labels are typically unavailable. We hope this work inspires further research in state-only learning paradigms.

Finally, we examine how policies developed and refined in controlled lab settings can be adapted for deployment in the dynamic, unpredictable real world. If a robot only reacts to changes as they occur, it will inevitably lag behind the environment. In Chapter 5, we demonstrate how anticipating future events, even with some level of uncertainty, allows the robot to plan for multiple possible outcomes. This predictive approach enables rapid reactions, and we show that it is effective enough to stop a moving object before it leaves the robot's reachable workspace.

Robotic learning remains an open and evolving field, with many challenges yet to be solved and numerous promising directions to explore. I hope that the work presented in this thesis encourages others to broaden their perspective on these problems and inspires new approaches toward advancing the field.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 1. ISBN: 1581138385.

[2] Josh Achiam et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[3] Jake K Aggarwal and Quin Cai. "Human motion analysis: A review". In: *Computer vision and image understanding* 73.3 (1999), pp. 428–440.

[4] A. Alahi et al. "Social LSTM: Human Trajectory Prediction in Crowded Spaces". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[5] Marcin Andrychowicz et al. "Hindsight Experience Replay". In: *Advances in Neural Information Processing Systems*. 2017.

[6] Marcin Andrychowicz et al. "Hindsight Experience Replay". In: *CoRR* abs/1707.01495 (2017). arXiv: 1707.01495. URL: http://arxiv.org/abs/1707.01495.

[7] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. "Human-to-Robot Imitation in the Wild". In: 2022.

[8] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *arXiv:1803.01271* (2018).

[9] Dominik Bauer, Timothy Patten, and Markus Vincze. *ReAgent: Point Cloud Registration using Imitation and Reinforcement Learning*. 2021. arXiv: 2103.15231 [cs.CV].

[10] Trevor Bekolay and Ben Morcos. *abr_control*. https://github.com/abr/abr_control. 2013.

[11] Richard Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[12] James Betker et al. "Improving image generation with better captions". In: *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf* 2.3 (2023), p. 8.

[13] Homanga Bharadhwaj et al. “Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking”. In: *arXiv preprint arXiv:2309.01918* (2023).

[14] Homanga Bharadhwaj et al. “Track2Act: Predicting Point Tracks from Internet Videos enables Diverse Zero-shot Robot Manipulation”. In: *arXiv preprint arXiv:2405.01527* (2024).

[15] Yevgen Chebotar et al. “Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills”. In: *CoRR* abs/2104.07749 (2021). arXiv: 2104.07749. URL: https://arxiv.org/abs/2104.07749.

[16] Ching-Hang Chen et al. “Unsupervised 3D Pose Estimation with Geometric Self-Supervision”. In: *CoRR* abs/1904.04812 (2019). arXiv: 1904.04812. URL: http://arxiv.org/abs/1904.04812.

[17] Yilun Chen et al. “Cascaded Pyramid Network for Multi-Person Pose Estimation”. In: *CoRR* abs/1711.07319 (2017). arXiv: 1711.07319. URL: http://arxiv.org/abs/1711.07319.

[18] Yuanpei Chen et al. *Sequential Dexterity: Chaining Dexterous Policies for Long-Horizon Manipulation*. 2023. arXiv: 2309.00987 [cs.RO].

[19] Yu Cheng et al. “3D Human Pose Estimation using Spatio-Temporal Networks with Explicit Occlusion Training”. In: *CoRR* abs/2004.11822 (2020). arXiv: 2004.11822. URL: https://arxiv.org/abs/2004.11822.

[20] Cheng Chi et al. “Diffusion Policy: Visuomotor Policy Learning via Action Diffusion”. In: *Proceedings of Robotics: Science and Systems (RSS)*. 2023.

[21] Cheng Chi et al. “Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots”. In: *arXiv preprint arXiv:2402.10329* (2024).

[22] Zichen Jeff Cui et al. “From play to policy: Conditional behavior generation from uncurated robot data”. In: *arXiv preprint arXiv:2210.10047* (2022).

[23] Coline Devin et al. “Deep Object-Centric Representations for Generalizable Robot Learning”. In: *CoRR* abs/1708.04225 (2017). arXiv: 1708.04225.

[24] Carl Doersch et al. “Tapir: Tracking any point with per-frame initialization and temporal refinement”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 10061–10072.

[25] Alexey Dosovitskiy and Vladlen Koltun. “Learning to Act by Predicting the Future”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[26] Yan Duan et al. “One-Shot Imitation Learning”. In: *CoRR* abs/1703.07326 (2017). arXiv: 1703.07326.

[27] Haritheja Etukuru et al. “Robot Utility Models: General Policies for Zero-Shot Deployment in New Environments”. In: *arXiv preprint arXiv:2409.05865* (2024).

[28] Hao-Shu Fang et al. “RMPE: Regional Multi-person Pose Estimation”. In: *ICCV*. 2017.

[29] Hao-Shu Fang et al. “AnyGrasp: Robust and Efficient Grasp Perception in Spatial and Temporal Domains”. In: *IEEE Transactions on Robotics (T-RO)* (2023).

[30] Haoshu Fang et al. “Learning Knowledge-guided Pose Grammar Machine for 3D Human Pose Estimation”. In: *CoRR* abs/1710.06513 (2017). arXiv: 1710.06513. URL: http://arxiv.org/abs/1710.06513.

[31] Meng Fang et al. “Curriculum-guided Hindsight Experience Replay”. In: *Neural Information Processing Systems*. 2019.

[32] Meng Fang et al. “DHER: Hindsight Experience Replay for Dynamic Goals”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[33] Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. “Unsupervised Learning for Physical Interaction through Video Prediction”. In: *Advances in Neural Information Processing Systems*. 2016.

[34] Chelsea Finn and Sergey Levine. “Deep Visual Foresight for Planning Robot Motion”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016.

[35] Katerina Fragkiadaki et al. “Learning Visual Predictive Models of Physics for Playing Billiards”. In: *International Conference on Learning Representations (ICLR)*. 2016.

[36] Justin Fu, Katie Luo, and Sergey Levine. *Learning Robust Rewards with Adversarial Inverse Reinforcement Learning*. 2018. arXiv: 1710.11248 [cs.LG].

[37] Yabo Fu et al. “Deep learning in medical image registration: a review”. In: *Physics in Medicine & Biology* 65.20 (2020), 20TR01.

[38] Rohit Girdhar et al. “Learning a Predictable and Generative Vector Representation for Objects”. In: *CoRR* abs/1603.08637 (2016). arXiv: 1603.08637. URL: http://arxiv.org/abs/1603.08637.

[39] Kamal Gupta et al. “ASIC: Aligning Sparse in-the-wild Image Collections”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 4134–4145.

[40] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *CoRR* abs/1801.01290 (2018). arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

[41] Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. “BAKU: An Efficient Transformer for Multi-Task Policy Learning”. In: *arXiv preprint arXiv:2406.07539* (2024).

[42] Siddhant Haldar and Lerrel Pinto. *Point Policy: Unifying Observations and Actions with Key Points for Robot Manipulation*. 2025. arXiv: 2502.20391 [cs.RO]. URL: https://arxiv.org/abs/2502.20391.

[43] Siddhant Haldar et al. *Teach a Robot to FISH: Versatile Imitation from One Minute of Demonstrations*. 2023. arXiv: 2303.01497 [cs.RO].

[44] Siddhant Haldar et al. "Watch and match: Supercharging imitation with regularized optimal transport". In: *Conference on Robot Learning*. PMLR. 2023, pp. 32–43.

[45] Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. "Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories". In: *ECCV*. 2022.

[46] Ashley Hill et al. *Stable Baselines*. https://github.com/hill-a/stable-baselines. 2018.

[47] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *CoRR* abs/1606.03476 (2016). arXiv: 1606.03476.

[48] Mir Rayat Imtiaz Hossain and James J. Little. "Exploiting temporal information for 3D pose estimation". In: *CoRR* abs/1711.08585 (2017). arXiv: 1711.08585. URL: http://arxiv.org/abs/1711.08585.

[49] Shuaiyi Huang et al. "Learning Semantic Correspondence with Sparse Annotations". In: *Proceedings of the European Conference on Computer Vision(ECCV)*. 2022.

[50] Ahmed Hussein et al. "Imitation Learning: A Survey of Learning Methods". In: *ACM Comput. Surv.* 50.2 (2017). ISSN: 0360-0300.

[51] Catalin Ionescu et al. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014).

[52] Aadhithya Iyer et al. "OPEN TEACH: A Versatile Teleoperation System for Robotic Manipulation". In: *arXiv preprint arXiv:2403.07870* (2024).

[53] Eric Jang et al. "Bc-z: Zero-shot task generalization with robotic imitation learning". In: *Conference on Robot Learning*. PMLR. 2022, pp. 991–1002.

[54] Yuanchen Ju et al. "Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation". In: *European Conference on Computer Vision*. Springer. 2025, pp. 222–239.

[55] W. Kabsch. "A discussion of the solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A* 34.5 (1978), pp. 827–828. DOI: 10.1107/S0567739478001680. URL: https://doi.org/10.1107/S0567739478001680.

[56] Nikita Karaev et al. *CoTracker: It is Better to Track Together*. 2023. arXiv: 2307.07635 [cs.CV].

[57] Isinsu Katircioglu et al. "Learning Latent Representations of 3D Human Pose with Deep Neural Networks". In: *International Journal of Computer Vision* 126 (2018), pp. 1326–1341.

[58] Bernhard Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering.” In: *ACM Trans. Graph.* 42.4 (2023), pp. 139–1.

[59] Alexander Khazatsky et al. “Droid: A large-scale in-the-wild robot manipulation dataset”. In: *arXiv preprint arXiv:2403.12945* (2024).

[60] Seungsu Kim, Ashwini Shukla, and Aude Billard. “Catching Objects in Flight”. In: *IEEE Transactions on Robotics*. 2014.

[61] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV].

[62] Okan Koç, Guilherme Maeda, and Jan Peters. “Online optimal trajectory generation for robot table tennis”. In: *Robotics and Autonomous Systems (RAS)*. 2018.

[63] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. “Self-Supervised Learning of 3D Human Pose using Multi-view Geometry”. In: *CoRR* abs/1903.02330 (2019). arXiv: 1903.02330. URL: http://arxiv.org/abs/1903.02330.

[64] Sateesh Kumar et al. “Graph inverse reinforcement learning from diverse videos”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 55–66.

[65] R. Lampariello et al. “Trajectory planning for optimal robot catching in real-time”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011.

[66] Seungjae Lee et al. “Behavior Generation with Latent Actions”. In: *arXiv preprint arXiv:2403.03181* (2024).

[67] Mara Levy, Vasista Ayyagari, and Abhinav Shrivastava. *No-frills Dynamic Planning using Static Planners*. 2021. arXiv: 2106.09714 [cs.RO]. URL: https://arxiv.org/abs/2106.09714.

[68] Mara Levy, Nirat Saini, and Abhinav Shrivastava. *WayEx: Waypoint Exploration using a Single Demonstration*. 2024. arXiv: 2407.15849 [cs.RO]. URL: https://arxiv.org/abs/2407.15849.

[69] Mara Levy, Nirat Saini, and Abhinav Shrivastava. *WayEx: Waypoint Exploration using a Single Demonstration*. 2024. arXiv: 2407.15849 [cs.RO].

[70] Mara Levy and Abhinav Shrivastava. *V-VIPE: Variational View Invariant Pose Embedding*. 2024. arXiv: 2407.07092 [cs.CV]. URL: https://arxiv.org/abs/2407.07092.

[71] Mara Levy et al. *P3-PO: Prescriptive Point Priors for Visuo-Spatial Generalization of Robot Policies*. 2024. arXiv: 2412.06784 [cs.RO]. URL: https://arxiv.org/abs/2412.06784.

[72] Jiefeng Li et al. “Crowdpose: Efficient crowded scenes pose estimation and a new benchmark”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 10863–10872.

[73] Jiefeng Li et al. “Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3383–3393.

[74] Timothy P. Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *International Conference on Learning Representations (ICLR)*. 2016.

[75] Timothy P. Lillicrap et al. “Continuous control with deep reinforcement learning.” In: *ICLR*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15.

[76] Tony Lindeberg. “Scale Invariant Feature Transform”. In: vol. 7. May 2012. DOI: 10.4249/scholarpedia.10491.

[77] Minghuan Liu, Menghui Zhu, and Weinan Zhang. *Goal-Conditioned Reinforcement Learning: Problems and Solutions*. 2022. arXiv: 2201.08299 [cs.AI].

[78] Xingyu Lu, Stas Tiomkin, and P. Abbeel. “Predictive Coding for Boosting Deep Reinforcement Learning with Sparse Rewards”. In: *ArXiv* abs/1912.13414 (2019).

[79] Haoyu Ma et al. “TransFusion: Cross-view Fusion with Transformer for 3D Human Pose Estimation”. In: *CoRR* abs/2110.09554 (2021). arXiv: 2110.09554. URL: https://arxiv.org/abs/2110.09554.

[80] Ajay Mandlekar et al. “Learning to Generalize Across Long-Horizon Tasks from Human Demonstrations”. In: *CoRR* abs/2003.06085 (2020). arXiv: 2003.06085.

[81] Julieta Martinez et al. “A simple yet effective baseline for 3d human pose estimation”. In: *CoRR* abs/1705.03098 (2017). arXiv: 1705.03098. URL: http://arxiv.org/abs/1705.03098.

[82] Dushyant Mehta et al. “Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision”. In: *3D Vision (3DV), 2017 Fifth International Conference on*. IEEE. 2017. DOI: 10.1109/3dv.2017.00064. URL: http://gvv.mpi-inf.mpg.de/3dhp_dataset.

[83] Ben Mildenhall et al. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[84] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: (2013). cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. URL: http://arxiv.org/abs/1312.5602.

[85] Volodymyr Mnih et al. “Playing Atari with Deep Reinforcement Learning”. In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[86] Ashvin Nair et al. “Overcoming Exploration in Reinforcement Learning with Demonstrations”. In: *CoRR* abs/1709.10089 (2017). arXiv: 1709.10089. URL: http://arxiv.org/abs/1709.10089.

[87] Ashvin Nair et al. “Awac: Accelerating online reinforcement learning with offline datasets”. In: *arXiv preprint arXiv:2006.09359* (2020).

[88] Ashvin Nair et al. *AWAC: Accelerating Online Reinforcement Learning with Offline Datasets*. 2021. arXiv: 2006.09359 [cs.LG].

[89] Soroush Nasiriany et al. “Planning with Goal-Conditioned Policies”. In: *CoRR* abs/1911.08453 (2019). arXiv: 1911.08453. URL: http://arxiv.org/abs/1911.08453.

[90] Andrew Y. Ng and Stuart J. Russell. “Algorithms for Inverse Reinforcement Learning”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. ISBN: 1558607072.

[91] Nishant Nikhil and Brendan Tran Morris. “Convolutional Neural Network for Trajectory Prediction”. In: *ECCV Workshops: Anticipating Human Behavior*. 2018.

[92] David Nistér, Oleg Naroditsky, and James Bergen. “Visual odometry”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. Ieee. 2004, pp. I–I.

[93] Kei Ota et al. “Deep Reactive Planning in Dynamic Environments”. In: *CoRR* abs/2011.00155 (2020). arXiv: 2011.00155. URL: https://arxiv.org/abs/2011.00155.

[94] Abhishek Padalkar et al. “Open x-embodiment: Robotic learning datasets and rt-x models”. In: *arXiv preprint arXiv:2310.08864* (2023).

[95] Tom Le Paine et al. “Making Efficient Use of Demonstrations to Solve Hard Exploration Problems”. In: *CoRR* abs/1909.01387 (2019). arXiv: 1909.01387. URL: http://arxiv.org/abs/1909.01387.

[96] Chuer Pan et al. “Tax-pose: Task-specific cross-pose estimation for robot manipulation”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 1783–1792.

[97] Aditya Panda and Dipti Prasad Mukherjee. “Monocular 3D Human Pose Estimation by Multiple Hypothesis Prediction and Joint Angle Supervision”. In: *2021 IEEE International Conference on Image Processing (ICIP)*. 2021, pp. 3243–3247. DOI: 10.1109/ICIP42928.2021.9506722.

[98] Sungheon Park, Jihye Hwang, and Nojun Kwak. “3D Human Pose Estimation Using Convolutional Neural Networks with 2D Pose Information”. In: *CoRR* abs/1608.03075 (2016). arXiv: 1608.03075. URL: http://arxiv.org/abs/1608.03075.

[99] Alexander Pashevich et al. “Learning to Augment Synthetic Images for Sim2Real Policy Transfer”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019.

[100] Georgios Pavlakos et al. “Coarse-to-Fine Volumetric Prediction for Single-Image 3D Human Pose”. In: *CoRR* abs/1611.07828 (2016). arXiv: 1611.07828. URL: http://arxiv.org/abs/1611.07828.

[101] P. Payeur, Hoang Le-Huy, and C. M. Gosselin. “Trajectory prediction for moving objects using artificial neural networks”. In: *IEEE Transactions on Industrial Electronics* (1995).

[102] Xue Bin Peng et al. “DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills”. In: *ACM Trans. Graph.* 37 (2018), p. 143.

[103] Karl Pertsch et al. “Demonstration-Guided Reinforcement Learning with Learned Skills”. In: *CoRR* abs/2107.10253 (2021). arXiv: 2107.10253. URL: https://arxiv.org/abs/2107.10253.

[104] Francesca Pistilli and Giuseppe Averta. “Graph Learning in Robotics: A Survey”. In: *IEEE Access* PP (Jan. 2023), pp. 1–1. DOI: 10.1109/ACCESS.2023.3323220.

[105] Matthias Plappert et al. “Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research”. In: *CoRR* abs/1802.09464 (2018). arXiv: 1802.09464. URL: http://arxiv.org/abs/1802.09464.

[106] D Pomerleau. “An autonomous land vehicle in a neural network”. In: *Advances in Neural Information Processing Systems* 1 (1998).

[107] Dean Pomerleau. “ALVINN: An Autonomous Land Vehicle In a Neural Network”. In: *Proceedings of (NeurIPS) Neural Information Processing Systems*. Ed. by D.S. Touretzky. Morgan Kaufmann, 1989, pp. 305–313.

[108] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[109] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. “SQIL: Imitation Learning via Regularized Behavioral Cloning”. In: *CoRR* abs/1905.11108 (2019). arXiv: 1905.11108. URL: http://arxiv.org/abs/1905.11108.

[110] Machel Reid et al. “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context”. In: *arXiv preprint arXiv:2403.05530* (2024).

[111] Edoardo Remelli et al. “Lightweight Multi-View 3D Pose Estimation through Camera-Disentangled Representation”. In: *CoRR* abs/2004.02186 (2020). arXiv: 2004.02186. URL: https://arxiv.org/abs/2004.02186.

[112] Martin A. Riedmiller et al. “Learning by Playing - Solving Sparse Reward Tasks from Scratch”. In: *CoRR* abs/1802.10567 (2018). arXiv: 1802.10567. URL: http://arxiv.org/abs/1802.10567.

[113] Marcia Riley and Christopher G. Atkeson. “Robot Catching: Towards Engaging Human-Humanoid Interaction”. In: *Autonomous Robots* (2002).

[114] R. Ritz et al. “Cooperative quadrocopter ball throwing and catching”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[115] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. “A reduction of imitation learning and structured prediction to no-regret online learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[116] Chitwan Saharia et al. “Photorealistic text-to-image diffusion models with deep language understanding”. In: *Advances in neural information processing systems* 35 (2022), pp. 36479–36494.

[117] Tim Salimans and Richard Chen. “Learning Montezuma’s Revenge from a Single Demonstration”. In: *CoRR* abs/1812.03381 (2018). arXiv: 1812.03381. URL: http://arxiv.org/abs/1812.03381.

[118] Stefan Schaal. “Learning from demonstration”. In: *Advances in neural information processing systems* 9 (1996).

[119] Peter Schönemann. “A generalized solution of the orthogonal procrustes problem”. In: *Psychometrika* 31.1 (1966), pp. 1–10. URL: https://EconPapers.repec.org/RePEc:spr:psycho:v:31:y:1966:i:1:p:1-10.

[120] Nur Muhammad Mahi Shafiullah et al. “On bringing robots home”. In: *arXiv preprint arXiv:2311.16098* (2023).

[121] Nur Muhammad Mahi. Shafiullah et al. *Supervised Policy Learning for Real Robots*. Tutorial presented at the Robotics: Science and Systems (RSS), Delft. 2024. URL: https://supervised-robot-learning.github.io.

[122] Saurabh Sharma et al. “Monocular 3D Human Pose Estimation by Generation and Ordinal Ranking”. In: *CoRR* abs/1904.01324 (2019). arXiv: 1904.01324. URL: http://arxiv.org/abs/1904.01324.

[123] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. “CLIPort: What and Where Pathways for Robotic Manipulation”. In: *CoRR* abs/2109.12098 (2021). arXiv: 2109.12098.

[124] Maximilian Sieb et al. “Graph-Structured Visual Imitation”. In: *Conference on Robot Learning*. 2019. URL: https://api.semanticscholar.org/CorpusID:196470945.

[125] Maximilian Sieb et al. “Graph-structured visual imitation”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 979–989.

[126] Robin A. M. Strudel et al. “Learning to combine primitive skills: A step towards versatile robotic manipulation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019.

[127] H. Su et al. “Trajectory Prediction of Spinning Ball Based on Fuzzy Filtering and Local Modeling for Robotic Ping–Pong Player”. In: *IEEE Transactions on Instrumentation and Measurement* (2013).

[128] Jennifer J. Sun et al. “View-Invariant Probabilistic Embedding for Human Pose”. In: *CoRR* abs/1912.01001 (2019). arXiv: 1912.01001. URL: http://arxiv.org/abs/1912.01001.

[129] Xiao Sun et al. “Integral Human Pose Regression”. In: *CoRR* abs/1711.08229 (2017). arXiv: 1711.08229. URL: http://arxiv.org/abs/1711.08229.

[130] Richard Sutton et al. “Horde : A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction Categories and Subject Descriptors”. In: vol. 2. Jan. 2011.

[131] Luming Tang et al. “Emergent Correspondence from Image Diffusion”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[132] Emanuel Todorov, Tom Erez, and Yuval Tassa. “MuJoCo: A physics engine for model-based control.” In: *IROS*. IEEE, 2012, pp. 5026–5033. ISBN: 978-1-4673-1737-5. URL: http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.

[133] Emanuel Todorov, Tom Erez, and Yuval Tassa. “MuJoCo: A physics engine for model-based control.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[134] Faraz Torabi, Garrett Warnell, and Peter Stone. “Recent advances in imitation learning from observation”. In: *arXiv preprint arXiv:1905.13566* (2019).

[135] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: *arXiv preprint arXiv:2302.13971* (2023).

[136] Jonathan Tremblay et al. “Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects”. In: *CoRR* abs/1809.10790 (2018). arXiv: 1809.10790.

[137] Alexander Trott et al. “Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards”. In: *CoRR* abs/1911.01417 (2019). arXiv: 1911.01417. URL: http://arxiv.org/abs/1911.01417.

[138] Stephen Tyree et al. “6-DoF Pose Estimation of Household Objects for Robotic Manipulation: An Accessible Dataset and Benchmark”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2022.

[139] Pietro Vitiello, Kamil Dreczkowski, and Edward Johns. “One-shot imitation learning: A pose estimation perspective”. In: *arXiv preprint arXiv:2310.12077* (2023).

[140] Tao Wang et al. “Direct Multi-view Multi-person 3D Pose Estimation”. In: *CoRR* abs/2111.04076 (2021). arXiv: 2111.04076. URL: https://arxiv.org/abs/2111.04076.

[141] Christopher Watkins and Peter Dayan. “Technical Note: Q-Learning”. In: *Machine Learning* 8 (May 1992), pp. 279–292. DOI: 10.1007/BF00992698.

[142] Guoqiang Wei et al. “View Invariant 3D Human Pose Estimation”. In: *CoRR* abs/1901.10841 (2019). arXiv: 1901.10841. URL: http://arxiv.org/abs/1901.10841.

[143] Albert Wilcox et al. *Monte Carlo Augmented Actor-Critic for Sparse Reward Deep Reinforcement Learning from Suboptimal Demonstrations*. 2022. arXiv: 2210.07432 [cs.LG].

[144] Erwin Wu and Hideki Koike. “FuturePong: Real-Time Table Tennis Trajectory Forecasting Using Pose Prediction Network”. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020.

[145] Hailun Xia and Meng Xiao. "3D Human Pose Estimation With Generative Adversarial Networks". In: *IEEE Access* 8 (2020), pp. 206198–206206. DOI: 10.1109/ACCESS.2020.3037829.

[146] Yuliang Xiu et al. "Pose Flow: Efficient Online Pose Tracking". In: *BMVC*. 2018.

[147] Lihe Yang et al. "Depth Anything V2". In: *arXiv:2406.09414* (2024).

[148] Lihe Yang et al. "Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data". In: *CVPR*. 2024.

[149] Wei Yang et al. "3D Human Pose Estimation in the Wild by Adversarial Learning". In: *CoRR* abs/1803.09722 (2018). arXiv: 1803.09722. URL: http://arxiv.org/abs/1803.09722.

[150] Alper Yilmaz, Omar Javed, and Mubarak Shah. "Object tracking: A survey". In: *Acm computing surveys (CSUR)* 38.4 (2006), 13–es.

[151] Zhao-Heng Yin and Pieter Abbeel. "Offline Imitation Learning through Graph Search and Retrieval". In: *Robotics: Science and Systems* (2024).

[152] Tianhe Yu et al. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning". In: *Conference on Robot Learning (CoRL)*. 2019. arXiv: 1910.10897 [cs.LG].

[153] Ailing Zeng et al. "SRNet: Improving Generalization in 3D Human Pose Estimation with a Split-and-Recombine Approach". In: *CoRR* abs/2007.09389 (2020). arXiv: 2007.09389. URL: https://arxiv.org/abs/2007.09389.

[154] Kuo-Hao Zeng et al. "Visual Reaction: Learning to Play Catch with Your Drone". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[155] Tianhao Zhang et al. *Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation*. 2018. arXiv: 1710.04615 [cs.LG].

[156] Tony Z Zhao et al. "Learning fine-grained bimanual manipulation with low-cost hardware". In: *arXiv preprint arXiv:2304.13705* (2023).

[157] Ce Zheng et al. "3D Human Pose Estimation with Spatial and Temporal Transformers". In: *CoRR* abs/2103.10455 (2021). arXiv: 2103.10455. URL: https://arxiv.org/abs/2103.10455.

[158] Yang Zheng et al. "PointOdyssey: A Large-Scale Synthetic Dataset for Long-Term Point Tracking". In: *ICCV*. 2023.

[159] Xingyi Zhou et al. "Weakly-supervised Transfer for 3D Human Pose Estimation in the Wild". In: *CoRR* abs/1704.02447 (2017). arXiv: 1704.02447. URL: http://arxiv.org/abs/1704.02447.

[160] Yifeng Zhu et al. "Learning generalizable manipulation policies with object-centric 3d representations". In: *7th Annual Conference on Robot Learning*. 2023.

[161] Yifeng Zhu et al. “Viola: Imitation learning for vision-based manipulation with object proposal priors”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 1199–1210.

[162] Yuke Zhu et al. “Reinforcement and Imitation Learning for Diverse Visuomotor Skills”. In: *CoRR* abs/1802.09564 (2018). arXiv: 1802.09564. URL: http://arxiv.org/abs/1802.09564.

[163] Yuke Zhu et al. “robosuite: A Modular Simulation Framework and Benchmark for Robot Learning”. In: *arXiv preprint arXiv:2009.12293*. 2020.

[164] Barbara Zitova and Jan Flusser. “Image registration methods: a survey”. In: *Image and vision computing* 21.11 (2003), pp. 977–1000.