



Institut für
Computertechnik
Institute of
Computer Technology

A MASTER THESIS ON

Particle-based Object-centric Motion Policy Learning

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Diplom-Ingenieur

(Equivalent to Master of Science)

in

Automation and Robotic Systems (UE 066 515)

by

Benedikt Frey

12230445

Supervisor:

Univ.-Prof. Dr.-Ing. Dongheui Lee

Co-supervisors:

Univ.Ass. Daniel Jan Sliwowski, M.Sc.

Vienna, Austria

April 2026

Abstract

Learning motor skills from visual observation is fundamental to human intelligence, yet transferring this capability to robotic systems remains an open challenge. Current approaches either develop implicit 3D scene understanding through large-scale robot-specific training on time-synchronized state (image) and action (robot pose) pairs. Cross-embodiment transfer is achieved by scaling the amount of data seen during training; however, collecting robot demonstrations is costly and time-consuming. Other 3D-aware methods rely on curated datasets (e.g., point clouds, simulation), restricting scalability to internet-scale video data. This thesis addresses both limitations with a Learning from Observation framework that learns robot manipulation policies directly from human video demonstrations, requiring no robot-specific training data. Task-relevant objects are segmented from RGB-D video using detection and segmentation models, with the resulting masks used to initialize dense 3D particle tracks propagated by a point-tracking model. These trajectories are used to train a conditional diffusion model that learns object-centric motion distributions conditioned on initial scene particles and task labels. At deployment, the same perception pipeline extracts object geometry from a live camera stream, generates and ranks candidate trajectories, and executes the best candidate via a Cartesian impedance controller. Because the policy operates in object-centric 3D space rather than robot configuration space, it is embodiment-agnostic by design. The framework is evaluated on a pouring task across 77 real-world trials, achieving 66.2% success in a fully open-loop setting and outperforming existing vision-based diffusion policies. Cross-embodiment transfer to a second platform yields 62.5% success without retraining, confirming that the learned representation encodes transferable task-level geometric structure. In generalization experiments with unseen objects, the proposed method achieves up to 50% success, where a visual diffusion policy baseline fails entirely, demonstrating the advantage of explicit 3D object-centric representations over implicit depth perception. These results suggest that particle-based tracking of human demonstrations provides a viable and scalable path toward robust, embodiment-independent manipulation policy learning.

Kurzfassung

Das Erlernen motorischer Fähigkeiten durch visuelle Beobachtung ist ein grundlegender Bestandteil menschlicher Intelligenz, doch die Übertragung dieser Fähigkeit auf robotische Systeme bleibt eine offene Herausforderung. Aktuelle Ansätze entwickeln ein implizites Verständnis der 3D-Szenenstruktur durch roboterspezifisches Training. Der Transfer zwischen verschiedenen Roboterplattformen wird dabei durch die Skalierung der Trainingsdatenmenge erreicht; allerdings ist die Erhebung solcher Demonstrationsdaten kostspielig und zeitaufwendig. Andere 3D-basierte Methoden stützen sich auf kuratierte Datensätze, was ihre Skalierbarkeit auf Internetvideodaten einschränkt. Diese Arbeit adressiert beide Einschränkungen durch einen „Learning-from-Observation“-Ansatz, der Manipulationsstrategien direkt aus menschlichen Videodemonstrationen lernt und keine roboterspezifischen Trainingsdaten erfordert. Aufgabenrelevante Objekte werden aus RGB-D-Videos mithilfe von Detektions- und Segmentierungsmodellen extrahiert, wobei die resultierenden Masken zur Initialisierung 3D-Partikeltrajektorien verwendet werden, die durch ein Point-Tracking-Modell propagiert werden. Diese Trajektorien dienen als Trainingsdaten für ein konditioniertes Diffusionsmodell, das objektzentrierte Bewegungsmuster lernt, konditioniert auf der initialen Szenenbeobachtung und Aufgabenlabels. Im Einsatz extrahiert dieselbe Methode die Objektgeometrie aus einem Live-Kamerabild, generiert und bewertet Kandidatentrajektorien und führt die beste Trajektorie mithilfe eines kartesischen Impedanzreglers aus. Da die Policy im objektzentrierten 3D-Raum anstelle des Roboterkonfigurationsraums operiert, ist sie per Design roboterunabhängig. Der Ansatz wird anhand einer Einschenkaufgabe in 77 realen Versuchen evaluiert und erreicht eine Erfolgsrate von 66,2%. Der Transfer auf eine zweite Roboterplattform erzielt 62,5% Erfolg ohne erneutes Training und bestätigt, dass die gelernte Repräsentation übertragbare geometrische Strukturen auf Aufgabenebene kodiert. In Experimenten mit unbekanntem Objekten erreicht die Methode bis zu 50% Erfolg, während ein visuelles Diffusionsmodell als Baseline vollständig versagt, was den Vorteil expliziter 3D-objektzentrierter Repräsentationen gegenüber impliziter Tiefenwahrnehmung verdeutlicht. Insgesamt legen die Ergebnisse nahe, dass die partikelbasierte Repräsentation menschlicher Demonstrationen einen praktikablen und skalierbaren Ansatz zur Erlernung robuster, plattformunabhängiger Manipulationsstrategien darstellt.

Acknowledgment

The author wishes to acknowledge the use of ChatGPT and Claude for translating the abstract and refining the language. The text was manually revised and reviewed for accuracy; it remains an accurate representation of the author's underlying work and intellectual contributions.

This thesis would not have been possible without the support of many kind people. First, I would like to thank my supervisor, Daniel Sliwowski, whose input was immensely helpful and whose guidance, especially during the robot experiments, was invaluable. I would also like to thank the entire ASL team. Next, I would like to thank my colleagues and friends, Stefan Lechner, Dario Spoljaric, Zoltán Varga, Julia Kuhn, Clemens Hacker, and Julia Kedra, for their support throughout this thesis. Finally, I would like to express my gratitude to my family for their support and encouragement throughout my studies.

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Copyright Statement

I, Benedikt Frey, hereby declare that this thesis is my own original work and, to the best of my knowledge and belief, it does not:

- Breach copyright or other intellectual property rights of a third party.
- Contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
- Contain material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.
- Contain substantial portions of third party copyright material, including but not limited to charts, diagrams, graphs, photographs or maps, or in instances where it does, I have obtained permission to use such material and allow it to be made accessible worldwide via the Internet.

Signature: _____

Vienna, Austria, April 2026

Benedikt Frey

Contents

Abstract	iii
Abstract (German)	iv
Acknowledgment	v
1 Introduction	1
1.1 Motivation and Research Questions	2
2 Related Work	5
2.1 Robotics Foundation Models	5
2.2 Video Prediction as Motion Planning	7
2.3 Particle Tracking & Trajectory-Based Methods	7
2.4 Gap Analysis	9
3 Diffusion Learning	11
3.1 DDPM and DDIM	13
3.2 Neural Networks	14
3.3 Classifier-Free Guidance	15
4 Method	17
4.1 Problem Formulation	17
4.2 Data Representation	20
4.3 Learning Method	22
4.4 Optimization	27
5 Evaluation	31
5.1 Evaluation on the Validation Set	31
5.2 Experimental Setup	34

5.3	Real World Evaluation	36
5.4	Comparative Analysis	40
5.5	Cross-Object Generalization	42
5.6	Cross-Embodiment Generalization	45
6	Conclusion	49
6.1	Answering the Research Questions	50
6.2	Limitations	52
6.3	Outlook	53
	Bibliography	54
A	Model Parameters	63

List of Tables

4.1	Target Measurements in the Dataset: Mean and Standard Deviation	30
5.1	Evaluation Validation Set	32
5.2	Ablation Validation Dataset Performed with Mahalanobis Optimization	33
5.3	Results Test Setup. Rows indicate the bottle position, columns indicate the cup position. Each cell shows the outcome distribution across four trials per configuration, color-coded by result quality. Asterisks denote configurations where one or more trials were not executed due to kinematic infeasibility.	36
5.4	Ablation Study Results.	40
5.5	Results for Visual Policy Trained on Robot Demonstrations	41
5.6	Results: Visual Policy vs. Our Policy on Unseen Objects	44
5.7	Results Test Setup with Kinova Robot	46
A.1	Diffusion Model Configuration	63
A.2	Training Configuration	64
A.3	Object Dimensions	64

List of Figures

1.1	Demonstration learning flowchart [1].	2
1.2	Overview of our Motion Learning Pipeline	4
2.1	Overview: State-of-the-Art Video-based Demonstration Learning	10
3.1	The Forward and Reverse Diffusion Process [2]	15
4.1	Three Stages of the Demonstration Learning Pipeline	17
4.2	Diffusion model learning 3D particle trajectories. (<i>Top</i>) The forward process (Equation (4.4)) gradually corrupts a clean trajectory \mathbf{X}_0 into Gaussian noise \mathbf{X}_K . (<i>Middle</i>) The denoising network ϵ_θ conditions on the scene observation and action label to predict the noise at each diffusion step k , trained via the objective in Equation (4.5). (<i>Bottom</i>) During inference, the reverse process (Equations (4.6)–(4.7)) iteratively denoises \mathbf{X}_K back into a coherent trajectory $\hat{\mathbf{X}}$	19
4.3	Data Collection Process	20
4.4	Example of a Video Created for the Dataset	22
4.5	SpatialTracker Trajectories (left) and Processed Trajectories for Dataset (right)	22
4.6	Diffusion Model UNet Structure	23
4.7	Training Pipeline: From SpatialTracker Data to Policy Learning	24
4.8	Example of Data Sample Augmentation: (a) Pouring from Right to Left; (b) Rotated around vertical axis.	26
4.9	Inference Pipeline: From Extracting Object Information to Robot Execution	26
4.10	Successful Execution on the Franka Emika Robot	28
5.1	Example Trajectory on Validation Set - left: Generated Trajectory, right: Ground Truth Trajectory	32
5.2	Example Failed Trajectory on Validation Set Smaller Model - left: Generated Trajectory, right: Ground Truth Trajectory	33

5.3	Evaluation Test Setup	35
5.4	Example of Successful Pouring Execution	37
5.5	Example of Successful Short-range Pouring Execution	37
5.6	Example of Failed Pouring Execution	38
5.7	Effect of random point cloud ordering on the validation set. Right: trajectory generated from a correctly ordered observation. Left: trajectory generated from the same observation with a randomised point order. Colours indicate index assignment under the grid-based ordering scheme.	38
5.8	Effect of random point cloud ordering on real-world data. Right: trajectory generated from a correctly ordered observation. Left: trajectory generated from a randomly ordered observation. Red points indicate the cup; blue points indicate the bottle.	39
5.9	Pouring Trajectories: Left Visual Diffusion Policy and Right Our Object-centric Policy .	42
5.10	Different Cups used for Cross Object Evaluation (From left to right: Original (ϕ 10cm), White (ϕ 8.5cm), Green (ϕ 8.5cm), Yellow (ϕ 7.2cm)).	43
5.11	Different Bottles used for Cross Object Evaluation (Left – original $H = 20$ cm, $\phi = 3.5$ cm, Right – novel $H = 24.5$ cm, $\phi = 2.8$ cm).	43
5.12	Three Pouring Motions Cross-Object Generalization: Green Cup, White Cup, Yellow Cup	44
5.13	Evaluation Test Setup Kinova Robot	45
5.14	3D trajectory plots on the Kinova robot. Left: short-range configuration with a high arc, showing variation in approach curvature across sampled trajectories. Right: long-range configuration, where trajectory diversity decreases and the pouring point is placed well above the cup, a behaviour driven by the Mahalanobis-based selection, which does not account for the initial inter-object distance.	46
5.15	Example of Successful Pouring Execution on the Kinova Robot	47
5.16	Qualitative Comparison of Multiple Executed Trajectories on the Kinova Robot	48

Acronyms

- ARAP** As-Rigid-As-Possible. 8
- CAD** Computer-Aided Design. 9, 20
- CNN** Convolutional Neural Network. 15
- DDIM** Denoising Diffusion Implicit Model. 14
- DDPM** Denoising Diffusion Probabilistic Model. 13, 14
- DoF** Degrees of Freedom. 5, 20
- ELBO** Evidence Lower Bound. 12
- k-NN** k-Nearest Neighbours. 8
- KL** Kullback-Leibler-Divergenz. 12, 13
- LfD** Learning from Demonstration. 2
- ML** machine learning. 4
- MSE** Mean Square Error. 14
- PCA** Principal Component Analysis. 27
- RGB** Red, Green, Blue. 21, 40
- RGB-D** Red, Green, Blue-Depth. 1, 20, 21, 26, 50
- RMSE** Root Mean Square Error. 8
- ROS** Robot Operating System. 26, 27
- SAM** Segment Anything Model. 20, 50
- SOTA** State-of-the-Art. 4
- ViT** Vision Transformer. 6
- VLA** Vision-Language-Action. 1, 5, 6, 9
- VLM** Vision-Language-Model. 5

Chapter 1

Introduction

It is increasingly common to learn new skills from watching videos, tying a tie from a YouTube tutorial or mastering a cooking technique from an online show, simply by observing and reproducing the actions. We as humans have an exceptional ability to extract essential information from visual demonstrations and reproduce the observed behavior in entirely new contexts, using different objects, environments, or conditions. This capacity, often developed in early childhood as we observe and imitate the actions of others [3]. It enables us to learn complex fine motor skills without direct instruction or identical conditions. Robots today can replicate demonstrated tasks with precision, but they often fail when the environment or object changes [4] or require a large amount of training data [5]. This diploma thesis applies the concept Learning from Observation (LfO) and explores the transfer into robotics. Learning motor skills by observing daily human tasks and repeating the task in a new environment. We propose a pipeline for learning robot manipulation policies directly from human video demonstrations, without requiring robot-specific data. Current approaches such as Vision-Language-Action (VLA) [6] models rely on large-scale internet video data combined with extensive robot demonstrations, developing an implicit understanding of scene depth rather than an explicit geometric one [7]. In contrast, 3D-aware models are trained on narrowly scoped datasets, limiting their ability to generalize across diverse tasks [8]. This work aims to bridge that gap by lifting 2D video into 3D space through 3D particle tracking models [9], leveraging depth estimation models or RGB-D cameras to introduce explicit spatial awareness without sacrificing possible scalability. The motion of task-relevant objects is modeled through sets of 3D particle tracks sampled from their surfaces, forming a compact representation of manipulation tasks. A conditional diffusion model [10] learns to predict these trajectories given an initial scene observation and a task specification. The resulting motion is transferred to a robot via rigid pose estimation and a Cartesian controller. The framework is implemented and evaluated on a pouring task.

1.1 Motivation and Research Questions

As robots are required to perform increasingly complex motions, especially in dynamic environments beyond simple pick-and-place tasks, programming such behaviors becomes increasingly difficult [11]. This has motivated research into more intuitive approaches, such as Learning from Demonstration (LfD), where robots acquire new skills by gathering information from demonstrations rather than relying solely on explicit coding.

In a survey on demonstration learning [1] the demonstration learning paradigm is laid out clearly. Figure 1.1 shows the design process.

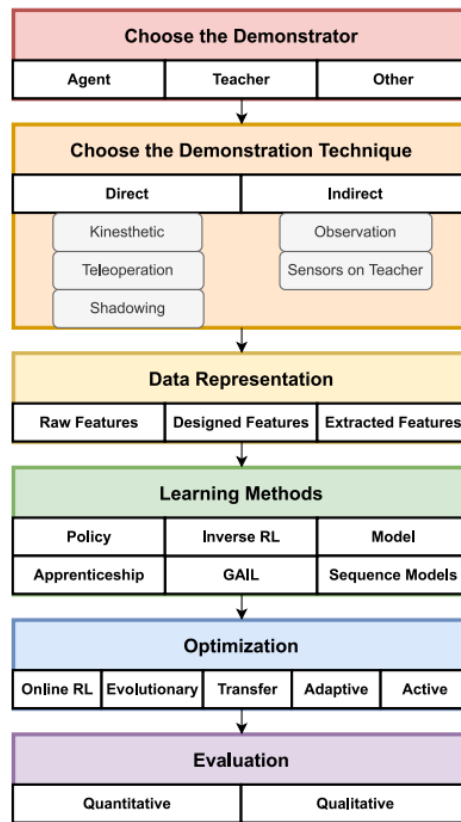


Figure 1.1: Demonstration learning flowchart [1].

Correia et al. [1] split the field into two categories: (1) the direct demonstration technique, e.g., teleoperation and (2) the indirect method, e.g. observation. In indirect demonstration learning a different agent demonstrates the task, it does not capture the demonstration data directly from the learning agent's sensors. As a result, the learning agent cannot directly apply the state–action pairs from the demonstration dataset [1]. This offers a big advantage in reducing time to collect training data, since the demonstrating agent can use their own embodiment to demonstrate the actions (no robotic agent needed) [1]. Additionally, the resulting policy is not bound to a specific operating agent, i.e., the policy can be used on a variety of robots. To properly facilitate skill transfer between agents, an appropriate

data representation needs to be selected. Correia et al. [1] present three different methods: raw features (e.g. images, audio data), designed features (e.g. keypoints, object segmentation) and extracted features (e.g. visual embeddings). While raw features offer the most context for a policy, they also require the highest amount of data to train and insufficient data can often lead to overfitting and poor generalization [4]. A useful feature representation for a motion learning policy should draw primarily on data extracted from interactions between agents and objects within the scene. Point clouds offer a natural fit for this, providing a structured and spatially grounded representation of scene information. In this context, a particle-based representation of a demonstration scenario, constructed solely from scene-relevant point cloud data, constitutes a highly informative feature. Recent advancements in machine learning algorithms in tracking and depth estimation have offered advancements in point cloud generation and tracking in video data. Based on these advancements, we construct the first research question we want to answer in this thesis:

Can we leverage out-of-domain data using particle-based tracking to learn generalizable robot motion policies?

Since we want to explore the indirect demonstration method, no robot needs to be chosen as the demonstrating agent. This offers the possibility to utilize any agent, like a human, as the demonstrator. However, the observation method requires a mapping of the demonstrator agent's actions onto the learner [1]. Here we look again at human behavior, as humans we leverage the understanding of what an object is and how objects interact with each other. We extrapolate our experiences from one example to other problems. This concept of an object centric observation is the basis for the mapping between the demonstrator and the learner. When learning from the demonstrator behavior, the motion policy should specifically learn the object motion. The goal is to find an approach that generalizes towards a policy that is applicable for environments outside the original demonstration data. Also, we want to observe if the policy learns inter-object relationships. Learning how objects move relative to each other, e.g., 'how does a bottle move relative to a glass when pouring'. Subsequently, the learned object behavior can then be mapped into the learners' task space (e.g. using a calibrated vision system). The desired object trajectory in the task space can then be transformed into an executable robot pose.

The second research question we want to answer, then formulates as follows:

How to leverage object-centric information in learning generalizable robot motion policies?

More concretely, this raises the question of whether a policy grounded in object-centric representations can implicitly learn inter-object relationships in the context of manipulation tasks. Rather than explicitly encoding task-specific constraints, can the model discover them on its own, for instance,

learning that a bottle must align with the opening of a cup during a pouring task, without ever being directly taught this relationship? Allowing policies to reason about objects and their interactions in a three dimensional context.

In the following chapters, we lay out the State-of-the-Art (SOTA) in the field of observation-based demonstration learning. We define and assess the machine learning (ML) algorithms responsible for particle-based tracking in video data. We outline the architecture of the learning method. Quantitative and qualitative experimental results derived from a self-created dataset are presented. Finally, this will result in a discussion to conclusively answer the research questions posed in the thesis. Figure 1.2 shows the proposed pipeline from training in the video domain to the execution on the robot.

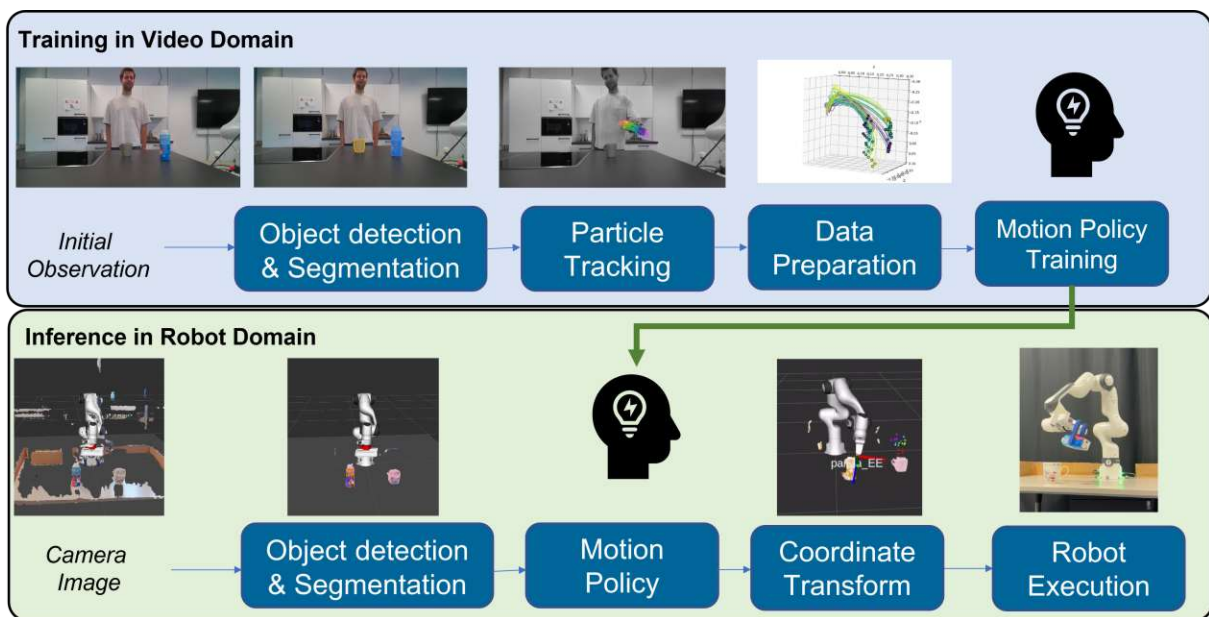


Figure 1.2: Overview of our Motion Learning Pipeline

Chapter 2

Related Work

Generating complex robot motions for manipulation tasks remains a central challenge in robotics, particularly as demands increase in industrial automation and household applications. Traditional approaches struggle to adapt to uncontrolled environments, where real-time generation and responsiveness to dynamic conditions are essential [12]. As discussed in the introduction, learning from video demonstrations offers a promising alternative by enabling robots to acquire skills from human or expert demonstrations without extensive manual programming. This chapter reviews the state-of-the-art in observation learning from video for robotics. We survey existing approaches for learning robot policies from visual data and analyze their respective advantages and limitations. The review progresses from end-to-end vision-language-action models to more specialized trajectory-based methods. Particular attention is given to particle tracking and trajectory representations, as these form the foundation of our approach. We further examine different particle tracking techniques and their role in robot motion generation. The chapter concludes with a gap analysis that motivates our method for generalizable learning from observation.

2.1 Robotics Foundation Models

We first consider Vision-Language-Action (VLA) models, often described as foundation models for robotics, which operate end-to-end across the entire robot system. These models leverage a Vision-Language Model (VLM) conditioned on visual observations and action description to predict robot action (e.g., end-effector pose). The RT-2 architecture [6] exemplifies this approach. It uses a pre-trained Vision-Language-Model (VLM) that is fine-tuned on robot-specific action data, including 6-Degrees of Freedom (6-DoF) end-effector poses, gripper width, and sensor inputs (usually: camera image). The model generates action tokens that map user language inputs to new robot poses [6]. A key advantage of VLA models is their ability to connect high-level visual-semantic reasoning with low-level robot

control. However, these models require large, labeled datasets of robot demonstrations that contain synchronized sensory data. Collecting such data is costly and time-consuming [5]. Furthermore, VLA models are often tied to the specific robot they were trained on. Generalizing across different embodiments remains a major challenge [5]. Recent work aims to reduce this dependency on robot-specific data. For example, LAPA [13] pre-trains motion representations on large-scale human video datasets and fine-tunes them on smaller robot datasets. Similarly, synthetic data generation approaches augment training data using simulation, which can more efficiently generate training samples [14]. The RT-X dataset [5] directly targets cross-embodiment generalization, by significantly scaling up the amount and diversity of the training data. Despite these advances, VLA models remain unsuitable for purely observation-based learning. Although they incorporate human video data, they still rely on robot-specific demonstrations as a foundation.

2.1.1 Human-Robot Retargetting for Policy Learning

Human-robot retargetting focusses on learning models that can transfer human motions directly to robots [15]. In a subsequent step they can be leveraged to compute robot actions required to achieved the demonstrated task. Vid2Robot [16] learns a mapping from human demonstration videos to robot actions using three types of data: robot-only recordings, human-only recordings, and paired human-robot demonstrations. The model encodes the demonstration video and the robot's current observation separately, then fuses them through cross-attention into a shared latent representation. A decoder predicts executable robot actions from this representation. This joint latent space enables implicit alignment between human and robot behavior. Gen2Act [17] extends this idea by replacing real human demonstrations with synthetically generated ones. A pre-trained video generation model produces plausible human demonstrations from a scene image and language instruction. The system encodes both the generated video and robot observations using a Vision Transformer (ViT) encoder and compresses them via a Perceiver Resampler [17]. Instead of explicit cross-attention, Gen2Act adds randomly sampled points to generated videos and trains the model to track how those points move over time, creating an auxiliary loss that forces the model to learn motion patterns explicitly. This motion-awareness objective ensures the policy doesn't just understand the visual appearance of a goal, but also captures the underlying dynamics of how objects and the robot should move to achieve it [17]. Both approaches rely on learning a shared latent space between human and robot observations. While they reduce the need for robot data, they do not eliminate it entirely.

2.2 Video Prediction as Motion Planning

An alternative line of work seeks to eliminate robot-specific training data by decoupling action planning from execution. Rather than directly predicting robot commands, these approaches first generate a video — a sequence of future image frames, depicting how a task should unfold, and then extract the required actions from that visual plan. Several works [18,19] train inverse dynamics models to infer robot actions. Rather than modeling full environment dynamics, the inverse dynamics model learns a simpler mapping: given two consecutive frames o_t and o_{t+1} , it predicts the action $a_t = f(o_t, o_{t+1})$ that caused the transition. It is trained on action-annotated robot trajectories with a mean squared error objective, making it lightweight relative to the video generation backbone. A separate inverse dynamics model can be trained for each robot. UniPi [18] further suggests using simulators to replace real-world data. Other approaches [20,21] derive actions using object tracking or rigid body transformations, assuming known camera geometry. These methods compute robot actions directly from estimated object motion. This paradigm offers two key advantages. First, motion planning becomes independent of the robot platform. Second, execution can be handled separately, either through simulation [20] or analytic control using object poses [21]. Additionally, generated videos are interpretable and allow human verification. However, these methods face important limitations. Generated videos may depict realistic human motion but not physically executable robot trajectories. Moreover, high computational costs, up to one minute per prediction [18] limit their applicability in real-time settings.

2.3 Particle Tracking & Trajectory-Based Methods

Rather than learning policies end-to-end, another class of methods represents motion explicitly through tracked visual features. These approaches use tracking models such as TAPIR [22], CoTracker [23], and LocoTrack [24] to extract dense point trajectories from videos. These trajectories capture the motion of objects, hands, and body parts across frames, providing a structured spatiotemporal representation. Methods either use these trajectories directly for control (e.g., visual servoing) or as features for policy learning. In the following section we give a short introduction how the necessary particle trackers function in two dimensional and three dimensional space.

2.3.1 2D and 3D Particle Tracking

Particle tracking aims to follow the movement of specific points across video frames. Given initial query points, the goal is to estimate their trajectories throughout the sequence. This task is challenging due to deformation, occlusion, lighting changes, and visual ambiguity. Traditional methods relied

on handcrafted features and local motion estimation, which struggled with long-term tracking [25]. Modern approaches instead learn robust representations using deep neural networks, often combining transformers and correlation volumes to model motion and appearance jointly [22,23]. TAPIR [22] uses a two-stage pipeline: it first matches query points independently across frames and then refines trajectories using local correlation. In contrast, CoTracker [23] jointly tracks multiple points using a transformer, exploiting correlations between trajectories. LocoTrack [24] extends this idea by computing dense local correspondences between regions across frames. Recent work also focuses on improving training data. CoTracker3 [26] uses pseudo-labeling on real-world videos, significantly reducing the need for synthetic data while improving robustness.

In robotics, particle tracking provides an interpretable and low-dimensional motion representation. Tracked points preserve geometric consistency and reflect physical constraints, making them well-suited for manipulation tasks. However, most tracking methods operate in 2D and cannot fully capture 3D motion. To address this, 3D-aware tracking methods incorporate depth information [27]. SpatialTracker [28] introduces a triplane encoding that projects 3D points into multiple planes and processes them with convolutional networks. A transformer then predicts point trajectories across frames. The method includes an As-Rigid-As-Possible (ARAP) loss to maintain geometric consistency. Subsequent work extends this approach. DELTA [29] introduces spatio-temporal attention, while TAPIP3D [30] incorporates local neighborhood information via k-NN-based context tokens. Despite recent advances, important limitations remain. In particular, estimating depth from monocular images continues to be a challenging problem [31,32]. Errors in depth estimation can significantly degrade overall system performance [30]. Existing monocular depth methods still exhibit considerable inaccuracies, often reporting RMSE values exceeding 0.3 meters [33]. Additionally, computational complexity limits real-time performance, and rigid constraints reduce applicability to deformable objects. Overall, 3D particle tracking remains less mature than its 2D counterpart, but recent progress suggests strong potential for robotics applications.

2.3.2 2D-Particle Tracking for Trajectory-based Motion Learning

Trajectory-based motion learning using 2D tracking generally follows two approaches: visual servoing [34,35] and rigid transformation methods [36]. Visual servoing uses continuous visual feedback to guide robot actions. RoboTAP [34] demonstrates this approach by generating training data from tracked trajectories and learning motion plans from a small number of demonstrations. ATM [35] extends this idea by introducing a transformer that predicts future trajectories conditioned on visual input and language instructions. A small network then maps these trajectories to robot actions. These methods significantly reduce the amount of required robot data but still depend on embodiment-specific

training.

Rigid transformation approaches instead represent manipulation as object motion. A rigid transformation $T = (R, t)$ defines object movement in 3D space, independent of the robot. The robot then executes this motion using standard controllers. Track2Act [36] follows this paradigm by learning interaction plans from large-scale video data. It predicts point trajectories using a diffusion transformer and lifts them into 3D space using depth estimation. These trajectories define object motion directly. While this enables embodiment-agnostic learning, the method relies on goal images that are not available at inference time. Additionally, it requires a residual policy trained on robot data to correct execution errors.

2.4 Gap Analysis

Figure 2.1 summarizes the state of the art in video-based demonstration learning. These approaches reveal a clear trend: most methods rely on 2D representations with limited spatial fidelity. VLA models and video-conditioned policies implicitly learn 3D structure through robot demonstrations. Existing particle-based visual-servoing methods reduce data requirements but still depend on robot-specific training. Track2Act enables generalization across different robot embodiments when given a goal state image, but depends on embodiment-specific residual policies to handle fine-grained task execution. Video prediction methods eliminate robot data but rely on simulation or inaccurate 3D reconstruction.

Recent 3D-aware approaches [8,37] demonstrate that spatial reasoning is feasible using point cloud representations. However, they require depth sensors, simulation, or CAD models. At the same time, advances in 3D particle tracking suggest that extracting spatial information directly from video is increasingly viable. Methods such as DELTA [29] and TAPIP3D [30] improve robustness and efficiency, while progress in monocular depth estimation reduces reliance on specialized hardware. Diffusion models have shown strong effectiveness for 3D reasoning and manipulation tasks [?, 8, 37] and human motion forecasting [38].

These observations reveal a key gap: no existing method combines robot demonstration-free learning with fully 3D-aware motion representation based on particle tracking. In particular, no approach provides an end-to-end pipeline that learns manipulation tasks directly in 3D space from video data alone. Such a system could enable generalizable, embodiment-agnostic robot learning while preserving spatial accuracy. This would support rapid adaptation across different robotic platforms and real-world environments.

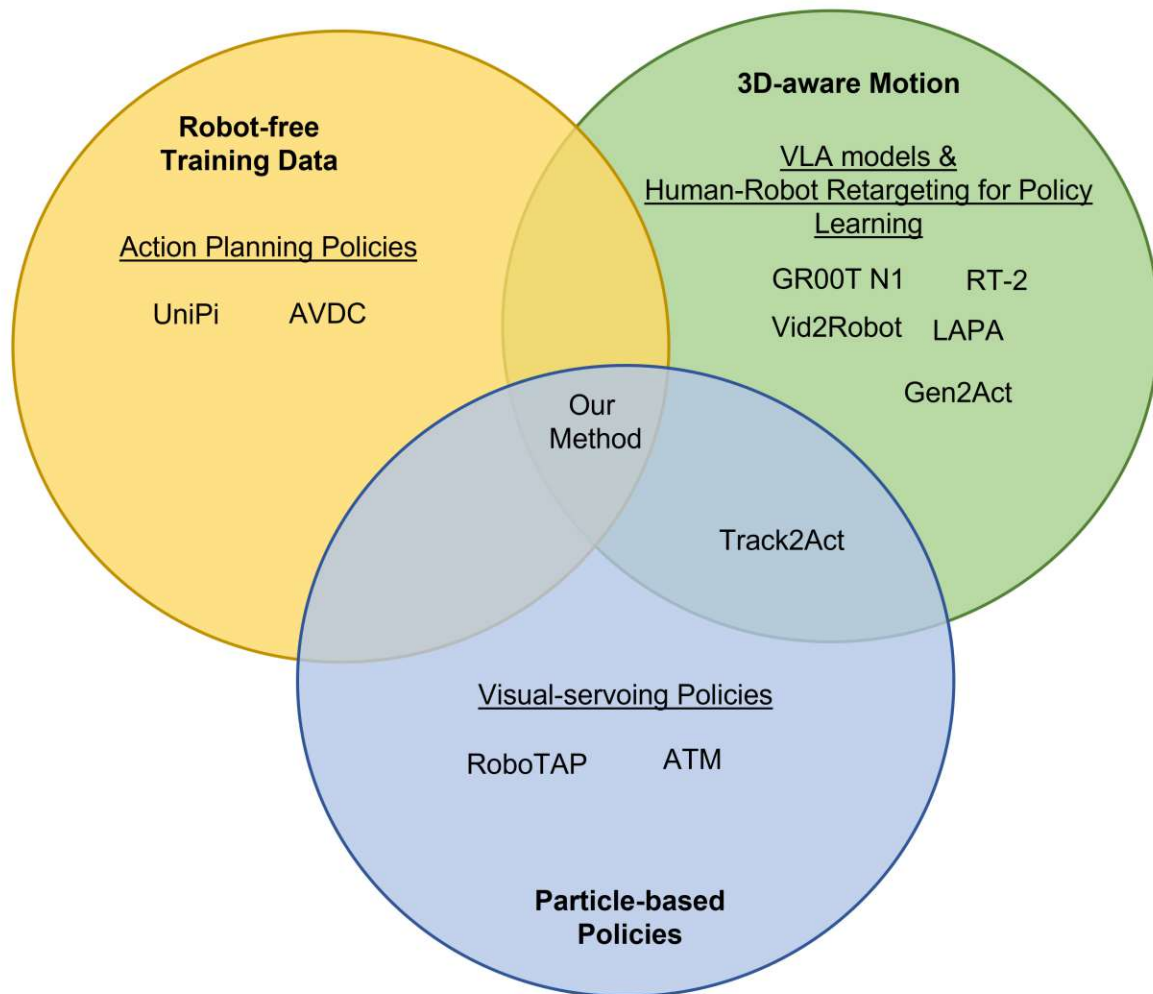


Figure 2.1: Overview: State-of-the-Art Video-based Demonstration Learning

Chapter 3

Diffusion Learning

This chapter explains the mathematical basics of diffusion probabilistic models, more commonly known as diffusion models [39]. Diffusion is inspired by the physical principle in non-equilibrium thermodynamics: the process in which particles mix independently, compensating for concentration differences until a concentration equilibrium is reached. For example, when dropping food colouring into water, the blend gradually mixes until a uniform colour is achieved. In machine learning, the diffusion process is mimicked by iteratively adding Gaussian noise to the original data. This forward diffusion destroys the original data structure. Stochastically, we define it as a Markov chain process that gradually maps data to a noise distribution. We define the Markov chain as follows:

$$q(x_{1:T} | x_0) = \prod_{t=1}^T q(x_t | x_{t-1}) \quad (3.1)$$

$$q(x_t | x_{t-1}) = \mathcal{N}\left(\sqrt{1 - \beta_t} x_{t-1}, \beta_t I\right) \quad (3.2)$$

Where x_0 is the original data sample and x_t is the noisy sample at timestep $t \in T$. The added noise is controlled by the noise schedule β_t , a scalar value. Each step of the Markov chain depends only on the previous state x_{t-1} .

To generate a new sample, we need to learn to reverse this process. This allows us to input Gaussian noise (here: x_T) and generate a sample (x_0). However, estimating $q(x_{t-1} | x_t)$ directly is not feasible, since it requires integrating over the entire data distribution $q(x_0)$, a quantity we cannot compute in closed form. Instead, we need to learn a model p_θ to approximate the conditional probabilities:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^T p_\theta(x_{t-1} | x_t) \quad (3.3)$$

$$p_\theta(x_{t-1} | x_t) = \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (3.4)$$

Where $\mu_\theta(x_t, t)$ is the predicted mean of the denoising step and $\Sigma_\theta(x_t, t)$ is the predicted variance. To learn optimal parameters θ such that p_θ represents the data distribution, we use the Maximum Likelihood Estimation of p_θ :

$$p_\theta(x_0) = \int p_\theta(x_{0:T}) dx_{1:T} \quad (3.5)$$

This term is intractable, because integrating over all possible intermediate noisy states from $t = 0$ to $t = T$ is computationally too expensive, as the number of possible noise trajectories grows exponentially with T . However, we can use the forward diffusion process $q(x_{1:T} | x_0)$ (see Eq. 3.1), which is fully known during the training phase. We rewrite the term (Eq. 3.5) using the logarithm for convenience:

$$\log p_\theta(x_0) = \log \int q(x_{1:T} | x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} | x_0)} dx_{1:T} \quad (3.6)$$

Applying Jensen's Inequality, we derive the evidence lower bound (ELBO) as follows:

$$\log p_\theta(x_0) \geq \mathbb{E}_q \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} | x_0)} \right] =: \mathcal{L} \quad (3.7)$$

We can then optimise the ELBO \mathcal{L} instead, which expands to:

$$\begin{aligned} \mathcal{L} = & \underbrace{\mathbb{E}_q [\log p_\theta(x_0 | x_1)]}_{\text{reconstruction term}} - \underbrace{D_{\text{KL}}(q(x_T | x_0) \| p(x_T))}_{\text{prior matching}} \\ & - \underbrace{\sum_{t=2}^T D_{\text{KL}}(q(x_{t-1} | x_t, x_0) \| p_\theta(x_{t-1} | x_t))}_{\text{denoising matching}} \end{aligned} \quad (3.8)$$

The advantage of this formulation is that each term is analytically tractable, since all components are known from the forward diffusion process. By minimising the KL divergences in the denoising

matching term, we directly train $p_\theta(x_{t-1} | x_t)$ to match the tractable posterior $q(x_{t-1} | x_t, x_0)$, which is Gaussian and has a closed-form expression. With this method, the model learns both $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$. Intuitively, this means the model learns to reverse each individual noising step rather than the entire process at once – a much simpler objective. The full derivation can be found in Sohl-Dickstein et al. [39]. Figure 3.1 gives a visual explanation of the forward and reverse diffusion process.

3.1 DDPM and DDIM

In an extension of the original diffusion paper, the Denoising Diffusion Probabilistic Model (DDPM) [40] reformulates the reverse diffusion process in a simplified way. Returning to the forward Markov chain from Eq. 3.2:

$$q(x_t | x_{t-1}) = \mathcal{N}\left(\sqrt{1 - \beta_t} x_{t-1}, \beta_t I\right), \quad (3.9)$$

this implies the closed-form expression

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad (3.10)$$

where ϵ is the noise and we define the auxiliary variables:

$$\alpha_t = 1 - \beta_t \quad \text{and} \quad \bar{\alpha}_t = \prod_{s=1}^t (1 - \beta_s). \quad (3.11)$$

We can then simplify by fixing the predicted variance $\Sigma_\theta(x_t, t) = \sigma_t^2 I$ to a diagonal matrix, where the scalar σ_t^2 depends only on the noise schedule β_t . This simplifies the denoising matching term from Eq. 3.8 to:

$$D_{\text{KL}} = \frac{1}{2\bar{\beta}_t} \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2 \quad (3.12)$$

After reparameterising the mean in terms of the noise prediction ϵ_θ , we can rewrite the objective as the standard DDPM loss function:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{x_0, \epsilon, t} \left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right] \quad (3.13)$$

Because the covariance matrices are fixed, the KL divergence between the true posterior and the

learned reverse process reduces to a scaled squared error between their means. This turns variational inference into simple regression using the Mean Square Error (MSE) between the true noise and the predicted noise. The DDPM simplification allows for stable and scalable training, achieving high quality output during the inference [40].

The Denoising Diffusion Implicit Model (DDIM) [41] further builds on the DDPM framework. While both approaches use the same trained model, the sampling processes differ. DDPM defines a stochastic sampling process:

$$x_{t-1} = \mu_{\theta}(x_t, t) + \sigma_t z, \quad z \sim \mathcal{N}(0, I), \quad (3.14)$$

where the variance σ_t depends on the noise schedule β_t . DDIM, on the other hand, generalises the reverse process into a non-Markovian process by introducing a free parameter $\eta \in [0, 1]$. The update rule becomes:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \epsilon_{\theta}(x_t, t) + \sigma_t z, \quad z \sim \mathcal{N}(0, I), \quad (3.15)$$

where

$$\sigma_t = \eta \sqrt{\frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \left(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}\right)}. \quad (3.16)$$

The parameter η controls the stochasticity of the process:

- $\eta = 1$ reproduces DDPM.
- $\eta = 0$ yields a fully deterministic process.
- $0 < \eta < 1$ produces partially stochastic dynamics.

Intuitively, the DDIM denoising process can be imagined as a smooth trajectory backwards through noise levels, while the DDPM process is a random, noisy walk backwards. Since DDIM is deterministic or only partially stochastic, comparable sample quality can be achieved with a fraction of the timesteps required by DDPM [40].

3.2 Neural Networks

During the forward diffusion process, noise is progressively added to the data. In the reverse process, the trained denoising model predicts the removal of noise from a Gaussian noise sample (x_T) to recover a clean sample (x_0).

As shown in Figure 3.1, the denoising neural network is typically a U-Net [42]. The U-Net consists of a contracting path, also called encoder, followed by a symmetrical expanding path, also called decoder,

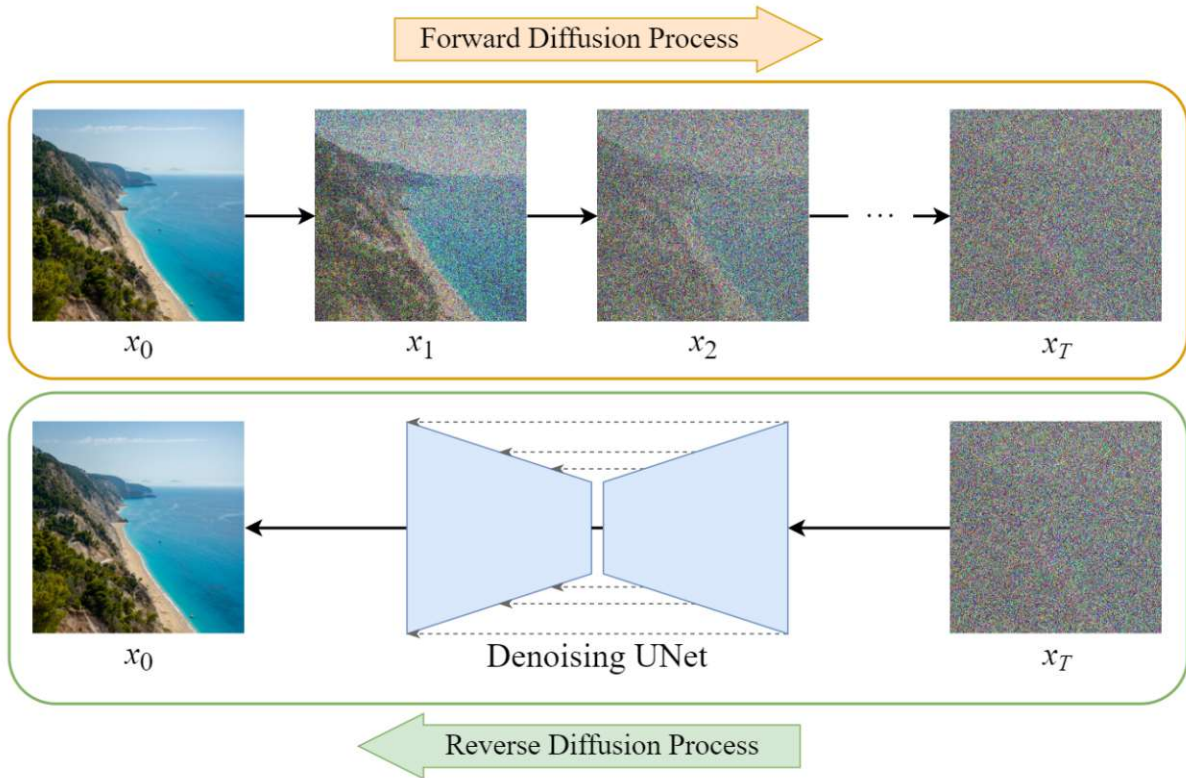


Figure 3.1: The Forward and Reverse Diffusion Process [2]

with skip connections joining corresponding layers from the contracting to the expanding side. The layers consist of a Convolutional Neural Network (CNN) block as well as cross-attention blocks in more recent implementations [43]. In recent years, transformer models have also been used as the neural network backbone of the diffusion process [44].

3.3 Classifier-Free Guidance

What makes diffusion models particularly applicable to our problem is the technique of classifier-free guidance [10]. Conditioning a diffusion model to generate a specific outcome originally required training a separate classifier [45]. Classifier-free guidance instead trains the diffusion model jointly, by conditioning a large proportion of training samples on the condition (c) and training roughly 10–20% of samples without any condition (\emptyset).

During inference, the model runs twice — once with the condition and once without — producing:

- $\epsilon_\theta(x_t, c)$: the conditional noise prediction,
- $\epsilon_\theta(x_t, \emptyset)$: the unconditional noise prediction.

Both estimates are combined using a guidance scale w :

$$\hat{\epsilon} = \epsilon_{\theta}(x_t, \emptyset) + w (\epsilon_{\theta}(x_t, c) - \epsilon_{\theta}(x_t, \emptyset)) \quad (3.17)$$

Depending on the strength of the guidance w , the model produces a sample that ranges from weakly aligned with the condition ($w \approx 0$) to strongly steered towards it ($w > 15$). Classifier-free guidance enables the conditioning of diffusion models on a wide variety of factors, driving advances in, for example, text-to-image generation [10].

Chapter 4

Method

This chapter presents our manipulation policy framework that addresses the gap identified in the Related Work section where no current approach combines three-dimensional reasoning with learning from video datasets without explicit robot training data. The overall framework consists of three stages, (1) Data Representation, (2) Learning Method, and (3) Optimization, illustrated in Figure 4.1, and follows the Learning from Demonstration pipeline introduced in Figure 1.1.

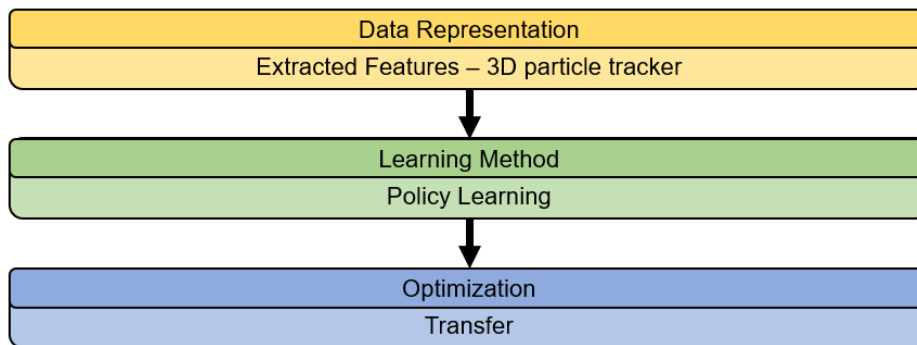


Figure 4.1: Three Stages of the Demonstration Learning Pipeline

The goal is to learn a manipulation policy from video demonstrations that generates object-level motion plans executable on a robot, without requiring robot-specific demonstrations.

4.1 Problem Formulation

We design our policy to generate viable object motions directly from the scene representation. The policy receives an initial observation \mathcal{O} of the task scene, which we represent as a set of object-centric

point clouds extracted for all n task-relevant objects:

$$\mathcal{O} = \{P_1, P_2, \dots, P_n\}, \quad P_i \in \mathbb{R}^{N \times 3} \quad (4.1)$$

To specify which motion to generate, we define a discrete action label $a \in \{a_1, a_2, \dots, a_L\}$ over all available actions and encode it as a one-hot vector $e_a \in \{0, 1\}^L$. We represent object motion in the scene as a sequence of 3D particle trajectories. Concretely, we pack all trajectories into a tensor $\mathbf{X} \in \mathbb{R}^{3 \times N \times T}$, where N is the number of tracked surface particles and T is the number of trajectory timesteps. The three channels carry the Cartesian coordinates (x, y, z) of each particle at each timestep. We flatten each point cloud $P_i \in \mathbb{R}^{N \times 3}$ into $\bar{P}_i \in \mathbb{R}^{3N}$ and concatenate the flattened scene observation, the desired action and the diffusion step k (we use k to index diffusion steps and reserve T exclusively for trajectory timesteps) into a single condition vector:

$$c = \phi([\bar{P}_1, \dots, \bar{P}_n, e_a, k]) \quad (4.2)$$

where $\phi : \mathbb{R}^{d_{in}} \rightarrow \mathbb{R}^{d_c}$ is a learned linear projection that maps the concatenated inputs into the cross-attention conditioning space of the diffusion model. We build our motion-learning policy as a conditional diffusion model that learns the distribution over object trajectories given the scene observation and task action:

$$\pi_\theta : (c, \epsilon) \mapsto \hat{\mathbf{X}}, \quad \hat{\mathbf{X}} \sim p_\theta(\mathbf{X} | c) \quad (4.3)$$

During training, we corrupt a clean trajectory \mathbf{X}_0 by injecting Gaussian noise $\epsilon \sim \mathcal{N}(0, I^{3 \times N \times T})$ at each diffusion step k :

$$\mathbf{X}_k = \sqrt{\bar{\alpha}_k} \mathbf{X}_0 + \sqrt{1 - \bar{\alpha}_k} \epsilon \quad (4.4)$$

where $\bar{\alpha}_k$ are predefined noise-schedule parameters (see: Chapter Diffusion Learning). As illustrated in the top panel of Figure 4.2, this forward process progressively corrupts clean particle trajectories across K diffusion steps until they collapse into pure noise \mathbf{X}_K . The network then learns to predict the injected noise via the denoising score-matching objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{X}_0, \epsilon, k} [\|\epsilon - \epsilon_\theta(\mathbf{X}_k, c, k)\|^2] \quad (4.5)$$

The middle panel of Figure 4.2 depicts this model architecture, the typical UNet structure consisting of four down and up blocks. The first and respective last three blocks use cross attention to encode the condition vector c , from Equation (4.2), into the neural network. The UNet takes a noised trajectory at the current diffusion step k and the conditioning vector c , as input, and predicts the noise to be

removed. At inference time, we draw an initial noise sample $\mathbf{X}_K \sim \mathcal{N}(0, I^{3 \times N \times T})$ and iteratively denoise it over K DDIM steps ($k = K, K-1, \dots, 1$). At each step, the model first estimates the clean trajectory:

$$\hat{\mathbf{X}}_0^{(k)} = \frac{\mathbf{X}_k - \sqrt{1 - \bar{\alpha}_k} \epsilon_\theta(\mathbf{X}_k, c, k)}{\sqrt{\bar{\alpha}_k}} \quad (4.6)$$

and then takes a reverse-diffusion step:

$$\mathbf{X}_{k-1} = \sqrt{\bar{\alpha}_{k-1}} \hat{\mathbf{X}}_0^{(k)} + \sqrt{1 - \bar{\alpha}_{k-1}} \epsilon_\theta(\mathbf{X}_k, c, k) \quad (4.7)$$

As shown in the bottom panel of Figure 4.2, this reverse process iteratively recovers structure from \mathbf{X}_K until it produces the final denoised, task-consistent trajectory $\hat{\mathbf{X}} = \mathbf{X}_0$.

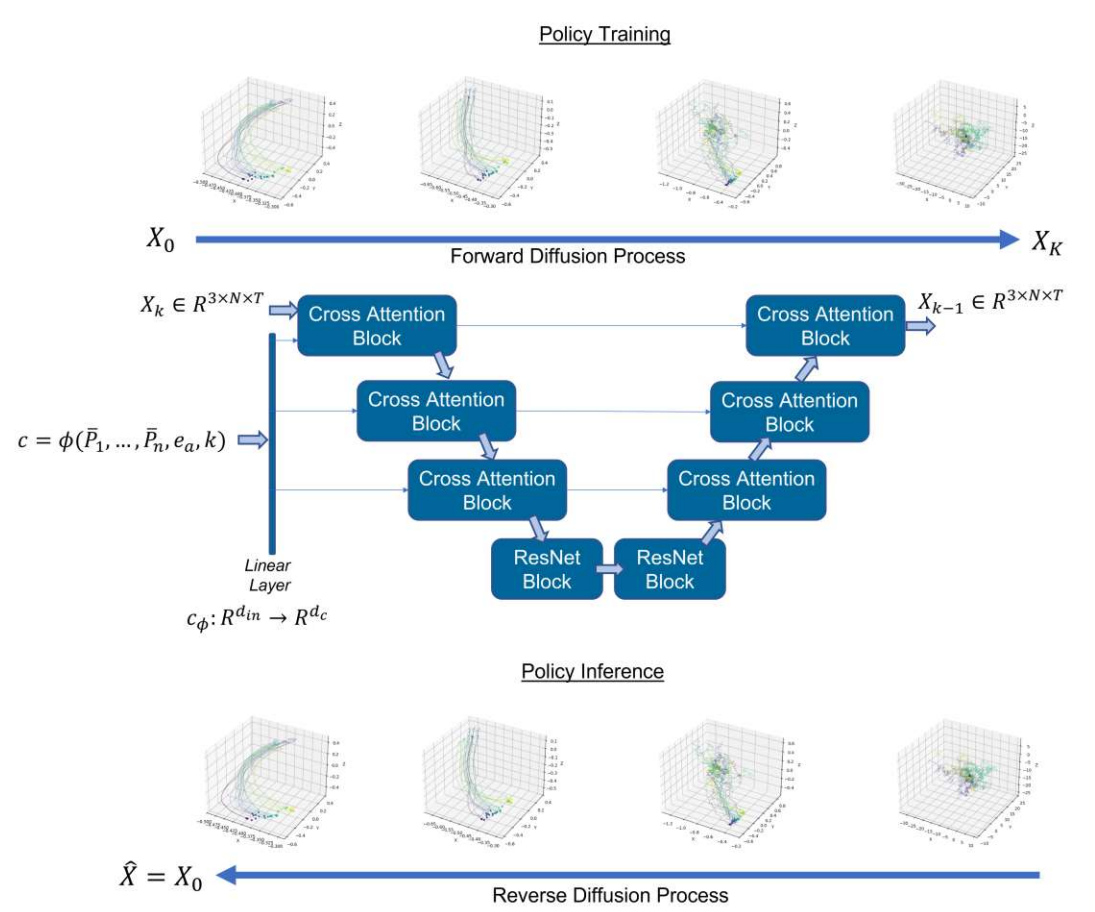


Figure 4.2: **Diffusion model learning 3D particle trajectories.** (Top) The forward process (Equation (4.4)) gradually corrupts a clean trajectory \mathbf{X}_0 into Gaussian noise \mathbf{X}_K . (Middle) The denoising network ϵ_θ conditions on the scene observation and action label to predict the noise at each diffusion step k , trained via the objective in Equation (4.5). (Bottom) During inference, the reverse process (Equations (4.6)–(4.7)) iteratively denoises \mathbf{X}_K back into a coherent trajectory $\hat{\mathbf{X}}$.

In the following sections we will explain in detail how we extract the 3D particle trajectories for the training dataset, what pre-processing steps we perform before training the model, and finally, how

we use optimization techniques to generate the most accurate 3D particle trajectories.

4.2 Data Representation

Videos provide a rich and easily accessible source of demonstrations, capturing object interactions in a wide range of real-world environments. However, raw video observations contain significant amounts of irrelevant visual information and encode motion only in two-dimensional image space. Learning robot manipulation policies requires a representation that isolates task-relevant object motion and expresses it in three-dimensional space. While prior approaches [46] estimate object-centric representations such as 6-Degrees of Freedom (DoF) poses using methods like FoundationPose [47], these methods often rely on multiple viewpoints or accurate prior CAD models of the object. As a result, they can struggle to generalize to novel objects and unseen geometries. Instead we use a particle representation of objects, each particle corresponds to a tracked point on the surface of an object. The temporal evolution of the set of the object particles describes the object’s motion during the interaction. This representation captures the geometric structure of objects and the relative motion between objects. We extract these trajectories using SpatialTracker [9], which estimates temporally consistent 3D particle tracks from RGB-D videos. Using either monocular vision combined with depth estimation (see: Related Work) or RGB-D videos directly. To focus the tracking process on task-relevant objects, we implement an object-centric pre-processing pipeline illustrated in Figure 4.3.

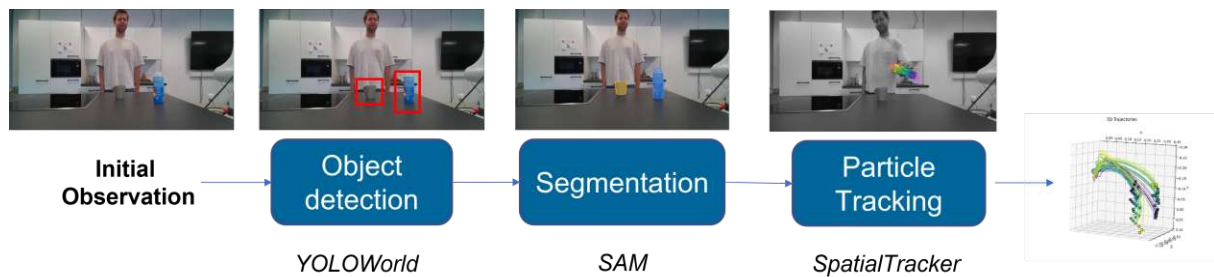


Figure 4.3: Data Collection Process

We first define the set of task-relevant objects from the input video. Given the first frame, we employ YOLOWorld [48], an open-vocabulary object detection model capable of recognizing a wide range of object categories, to identify and localize the objects of interest. We segment the detected objects using the Segment Anything Model (SAM) [49] to obtain precise binary masks of their surfaces. These masks define the regions from which we initialize particle tracks in a uniform spatial pattern. We then use SpatialTracker to track these regions over time, producing dense particle trajectories for each segmented object. Because the tracker only follows particles inside the initial masks, the resulting trajectories correspond directly to the movement of the object surfaces involved in the manipulation. The

particles in the initial frame represent our scene observation \mathcal{O} . The final dataset consists of synchronized three-dimensional trajectories describing the motion of all relevant objects in the scene. These trajectories form the primary representation used for our motion policy in the subsequent stages of the framework.

4.2.1 Dataset

Initially, the (Im)PerfectPour dataset was considered as a potential training source. This dataset was collected at the Autonomous Systems Lab at TU Wien and contains more than 400 teleoperated demonstrations of a Franka Emika Panda robot [50] performing manipulation tasks such as picking, placing, pouring, and whipping [51].

(Im)PerfectPour is an RGB-only dataset and does not provide ground-truth depth information, which is required by the SpatialTracker. To address this, we estimate depth using the ZoeDepth ZoeD-M12-NK model [32], as proposed by the authors of SpatialTracker. The tracker then outputs trajectories in a (u, v, d) representation, consisting of pixel coordinates (u, v) and the estimated depth d . However, the predicted depth is not always consistent across different videos, and in some cases varies even within a single sequence. As metric depth estimation methods continue to improve, this limitation is expected to become less significant. As a result, the tracked trajectories cannot be converted into 3D-consistent object movements to learn a policy. Since accurate three-dimensional coordinates are required for robot execution, a new dataset was recorded using an RGB-D sensor with accurate metric depth information. For this purpose, 50 videos were collected using an Azure Kinect structured-light depth camera [52]. The dataset captures a human performing a pouring task, consisting of picking up a bottle, pouring into a cup, and placing the bottle back on the table. Recording human demonstrations also enables evaluating the transfer of motion patterns from human demonstrations to robot execution. Figure 4.4 illustrates a sequence of frames from one recorded video.

Each recorded video was processed using the previously described object-centric tracking pipeline. SpatialTracker extracted particle trajectories for both the bottle and the cup objects. Each pixel coordinate (u, v) was first undistorted using the Brown-Conrady model, correcting for radial and tangential lens distortion over ten iterations using the Kinect’s calibrated intrinsic matrix K and distortion coefficients D . The undistorted coordinates were then lifted to 3D by scaling with the corresponding depth value d and dividing by the focal lengths, yielding metric points $(x, y, z) = (d \cdot x' / f_x, d \cdot y' / f_y, d)$ expressed in meters. For robustness, trajectories containing invalid depth readings (zero, NaN, or infinite values) were discarded. Remaining 3D points were then filtered for outliers by computing each point’s distance from the per-trajectory median, and removing any point exceeding three standard deviations from the mean distance. The remaining trajectories were then manually segmented into



Figure 4.4: Example of a Video Created for the Dataset

three distinct manipulation phases corresponding to our action labels $a \in \{\text{pick, pour, place}\}$. Each video therefore produced three separate trajectory samples representing individual manipulation actions. The cup object was tracked independently from the bottle. Because both trajectories are synchronized and extracted using identical tracking parameters, they can be combined to represent the relative motion between the manipulated object and the target object. Figure 4.5 shows the particle trajectories extracted by SpatialTracker corresponding to the video in Figure 4.4.

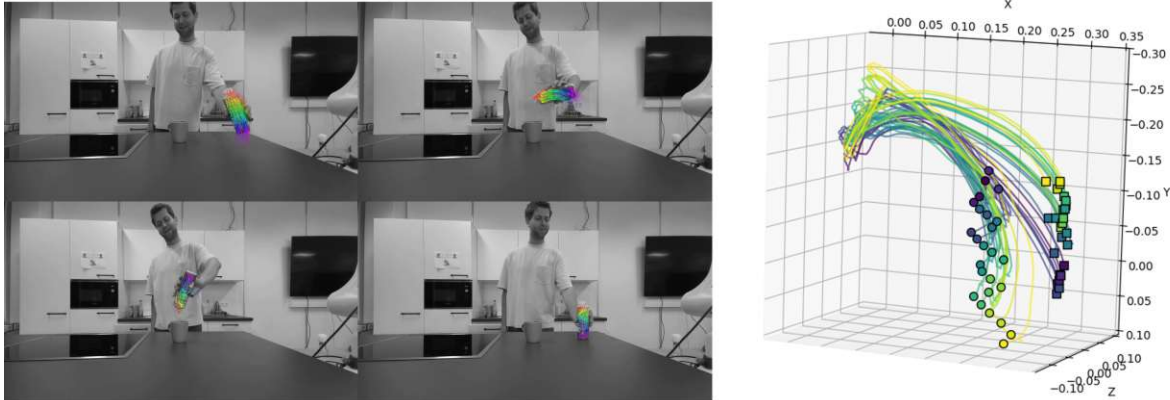


Figure 4.5: SpatialTracker Trajectories (left) and Processed Trajectories for Dataset (right)

After filtering invalid samples and incomplete trajectories, the final dataset contains 138 trajectory samples across the three manipulation actions.

4.3 Learning Method

We implement our policy as a conditional diffusion model built on a UNet architecture [42], following the standard design used in image generation models. The model structure is illustrated in Figure 4.6. We use the HuggingFace Diffusers library for the implementation [53]. While UNets are tradition-

ally applied to images, we explicitly exploit a structural analogy between images and our trajectory representation. Recall from the Problem Formulation Section that we represent object motion as a ten-

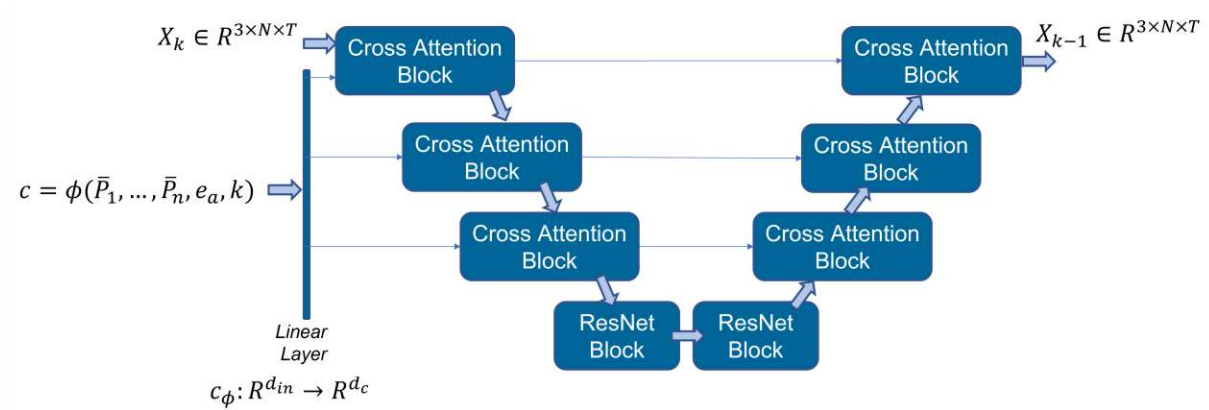


Figure 4.6: Diffusion Model UNet Structure

sor $\mathbf{X} \in \mathbb{R}^{3 \times N \times T}$, where the three channels encode the Cartesian coordinates (x, y, z) , N denotes the number of particles, and T the number of timesteps. This tensor directly matches the format of an image: the channel dimension corresponds to color channels, while the two remaining dimensions define a spatial grid. We therefore interpret \mathbf{X} as an 'image', where one axis indexes particles and the other indexes time. Given that diffusion models—and UNet backbones in particular—excel at modeling structured signals in image space, we expect them to capture both local geometric structure and temporal correlations in particle trajectories. We adopt the standard encoder–decoder UNet design with skip connections between corresponding resolution levels to preserve fine-grained details throughout the denoising process. The encoder progressively downsamples the input tensor \mathbf{X}_k to extract higher-level features, while the decoder upsamples and reconstructs the trajectory, combining intermediate representations through skip connections. We incorporate conditioning on the scene observation \mathcal{O} , action label e_a , and diffusion step k through the condition vector c (Equation (4.2)) using cross-attention layers embedded within the UNet blocks. These layers inject task-specific context at multiple resolutions, allowing the network to guide the denoising process toward trajectories that are both geometrically consistent with the scene and aligned with the desired manipulation action. Overall, the model operates directly on the trajectory tensor in the same way as an image diffusion model, enabling us to reuse well-established architectural components while adapting them to structured 3D motion generation. We implement the diffusion process using a DDIM scheduler, which enables efficient inference with as few as 50 denoising steps, while training is performed over 1000 diffusion timesteps. The full model configuration is provided in Table A.1 in the Appendix.

4.3.1 Training

We cannot directly feed the SpatialTracker trajectories into the diffusion model during training. The trajectories have a varying amount of valid points and timesteps. Instead, we need a series of pre-processing steps before training begins. The pipeline facilitating this data pre-processing is illustrated in Figure 4.7.

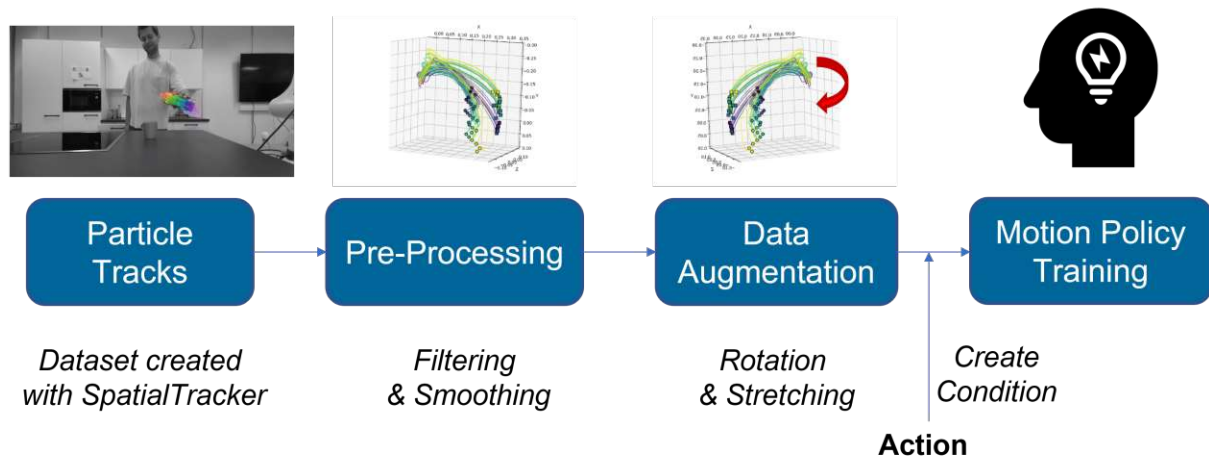


Figure 4.7: Training Pipeline: From SpatialTracker Data to Policy Learning

The diffusion model expects a constant number of points N per point cloud and timesteps T per action. We smooth and downsample all trajectories to a consistent size. Raw tracked trajectories contain high-frequency noise that can impede learning. We first remove all outliers more than 20 cm away from the mean of the point cloud. Afterwards, we apply a Savitzky-Golay filter (window length 5, polynomial order 2) independently to each point's x , y , and z coordinate trajectory, which attenuates noise while preserving the broader motion shape. Following smoothing, trajectories are resampled to the target dimensions. For the temporal axis, sequences longer than T are downsampled via linear interpolation to T evenly spaced frames; sequences shorter than T have their final frame repeated until the target length is reached. For the spatial axis, if a point cloud contains more than N points, it is downsampled by uniformly sampling indices over the full range of the original point cloud array. In the uncommon case where a sample contains fewer than N points, synthetic points are generated by averaging three randomly selected existing points at each timestep and appending them to the cloud until the target count is met. This is valid under the assumption that the object is convex. The average of three points is a convex combination, which lies within the convex hull of the point set. For convex objects, the convex hull coincides with the object itself, so the generated points remain within the object. Because the diffusion model treats the point cloud as an ordered sequence, a spatially consistent ordering must be established so that the same index always refers to the same spatial location on the object surface

across samples. We sort points at $t = 0$ using a grid-based scheme. The xy -plane is divided into a $\lceil\sqrt{N}\rceil \times \lceil\sqrt{N}\rceil$ regular grid, and each point is assigned to its corresponding cell. Cells are visited in row-major order, sorted first by y -cell index (top to bottom) and then by x -cell index (left to right). For each cell we select a point closest to its center. This produces a consistent ordering that is applied only to the initial frame $t = 0$. Since SpatialTracker inherently maintains temporal consistency across video frames, this initial ordering is preserved throughout the remainder of each trajectory without any additional sorting. For our use-case we choose $N = 25$ points and $T = 100$ timesteps, based on the mean values of the respective factors in the dataset. Then all trajectory values are normalised to the range $[-1, 1]$ as required for diffusion training. To extract the condition we need the initial object point clouds from the scene (\mathcal{O}) and the discrete actions a encoded using one-hot vectors. In our case $e_a \in \{0, 1\}^3$, making one-hot vectors a natural choice given the small, fully discrete action space. Alternatively, semantic embeddings such as CLIP [54] could be used. The point cloud observations get flattened to a vector of size $3N$, concatenated with the one-hot action vector, and projected into the cross attention conditioning of the diffusion model through a linear layer. Additionally, for our case, we translate the trajectories into a coordinate frame centred at the mean of the cup point cloud, placing the cup at the origin of the scene and making the policy invariant to its absolute position across different configurations, since the cup remains static during the action and thus serves as a natural anchoring point.

Given the limited dataset size of 138 samples, data augmentation is essential to ensure generalisation and robustness. We apply augmentation both to the observation (initial point clouds) and the trajectory samples. The one-hot action vector requires no augmentation. The data augmentation pipeline scales the initial point clouds uniformly between 90% and 110% of their original volume and adds per-point Gaussian noise to improve robustness to sensor noise and detection inaccuracies at inference time. We augment trajectory samples using rotations and isotropic scaling. Each trajectory is transformed by rotating it and applying a scaling factor, producing additional valid training samples. This augmentation encourages the model to become invariant to rotation and scale variations at inference time. Since our trajectories are expressed relative to the cup centroid, rotations about the vertical axis produce valid training samples that preserve the physical constraints of the task. Figure 4.8 shows an original trajectory alongside an augmented version, illustrating the effect of rotation and isotropic scaling on the pouring motion.

The full training configuration is summarised in Table A.2. The model is trained for 30,000 epochs.

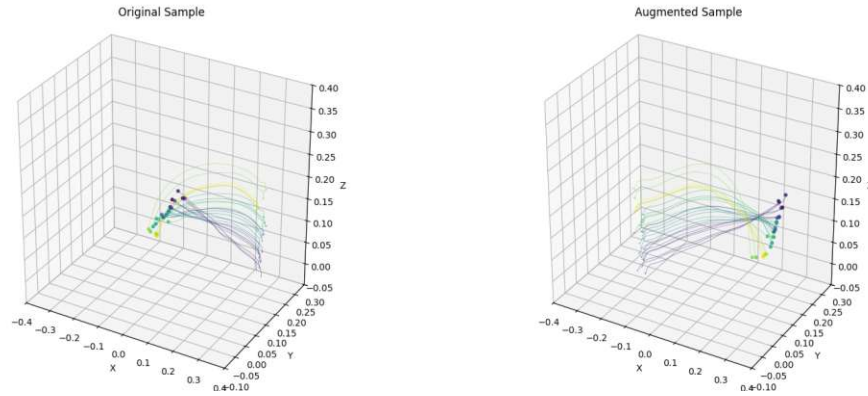


Figure 4.8: Example of Data Sample Augmentation: (a) Pouring from Right to Left; (b) Rotated around vertical axis.

4.3.2 Inference

At inference time, the learned motion policy is deployed within the Robot Operating System (ROS) environment to enable real-world execution on a robotic platform. The pipeline follows the same object-centric representation used during training, ensuring consistency between training and deployment. The system first acquires visual observations from an Azure Kinect RGB-D camera integrated as a ROS node. Using the same perception pipeline applied during dataset generation, task-relevant objects are identified and segmented. The resulting segmentation masks together with depth measurements are used to extract object-specific point clouds in the camera coordinate frame. A visualization of this process is shown in Figure 4.9.

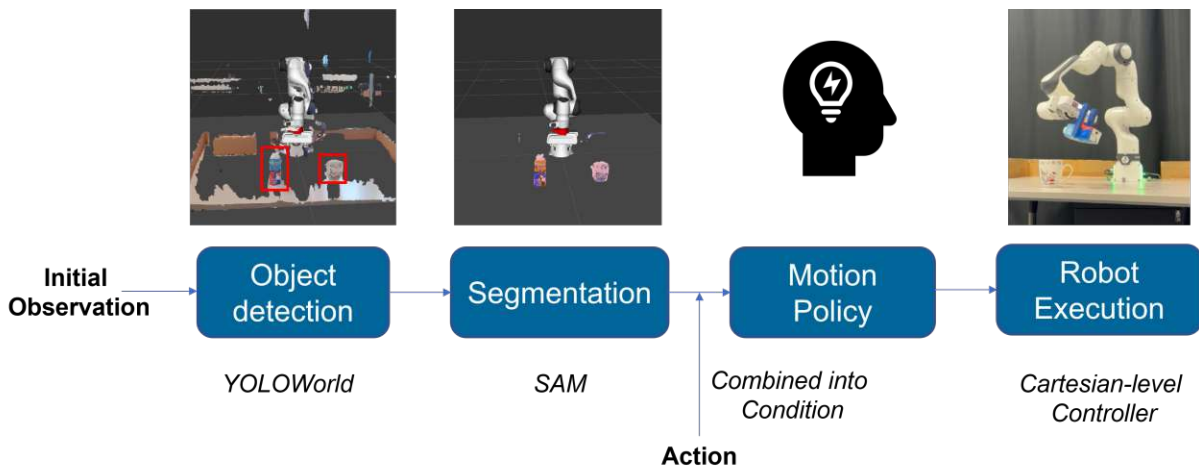


Figure 4.9: Inference Pipeline: From Extracting Object Information to Robot Execution

At inference time, observations O are sampled from the object point cloud using the same spatial ordering described in the pre-processing step, ensuring that point indices correspond to consistent spatial locations, for instance, points on the top or bottom of the object always map to the same indices as

seen during training. We then construct M conditioning vectors from this point cloud by appending the action embedding e_a to each point, drawing M independent samples to account for uncertainty in perception and scene variability. In this work, batch inference is performed by sampling $M = 20$ variations of the conditioning input. A good trade-off between inference time and output quality (Inference time on H100 GPU - 1 Sample: 1.6 seconds, 20 Samples: 3.4 seconds, 100 Samples: 14.1 seconds). For each condition, the diffusion-based policy generates M three-dimensional point cloud sequences. An optimization strategy (described in the following section) selects the most suitable candidate. Once selected, the candidate must be converted into a representation suitable for robot execution. Since the robot operates in Cartesian space and cannot directly follow particle-set trajectories, the predicted point cloud sequence is transformed into a sequence of rigid object poses. This conversion assumes that the manipulated object behaves approximately as a rigid body over the trajectory. The initial object pose is estimated from the first point cloud using Principal Component Analysis (PCA), which provides both object orientation—derived from the principal axes—and object position, computed as the centroid. To resolve sign ambiguity in the principal components, the sign of each axis is consistently fixed by enforcing a positive direction for the first component (i.e., ensuring the first element of each principal axis vector is positive). For subsequent timesteps, rigid transformations between consecutive point clouds are computed using the Kabsch–Umeyama algorithm [55]. This method estimates the optimal rotation matrix that minimizes the root mean squared deviation between corresponding point sets. The translation component is obtained from the displacement of the point cloud centroids. By iteratively applying this procedure from the first to the last timestep, a sequence of object poses is constructed. The resulting pose trajectory is expressed in the camera coordinate frame and must therefore be transformed into the robot’s coordinate system. A standard hand–eye calibration procedure provides this transformation. The ROS TF library [56] manages and applies these coordinate transformations within the system. Finally, the transformed pose trajectory is executed using the Cartesian impedance controller from RE-ASSEMBLE [57]. The controller tracks the computed end-effector motion over time, enabling the robot to reproduce the predicted manipulation behavior. Figure 4.10 illustrates a successful execution of the pouring task on the Franka Emika Panda robot [50].

4.4 Optimization

The proposed motion policy operates in an open-loop setting, meaning that no feedback is incorporated during execution to correct deviations from the planned trajectory. As a result, the quality of the generated trajectory must be assessed prior to execution to reduce the likelihood of task failure. To address this limitation, an optimization strategy is introduced that selects the most suitable trajectory

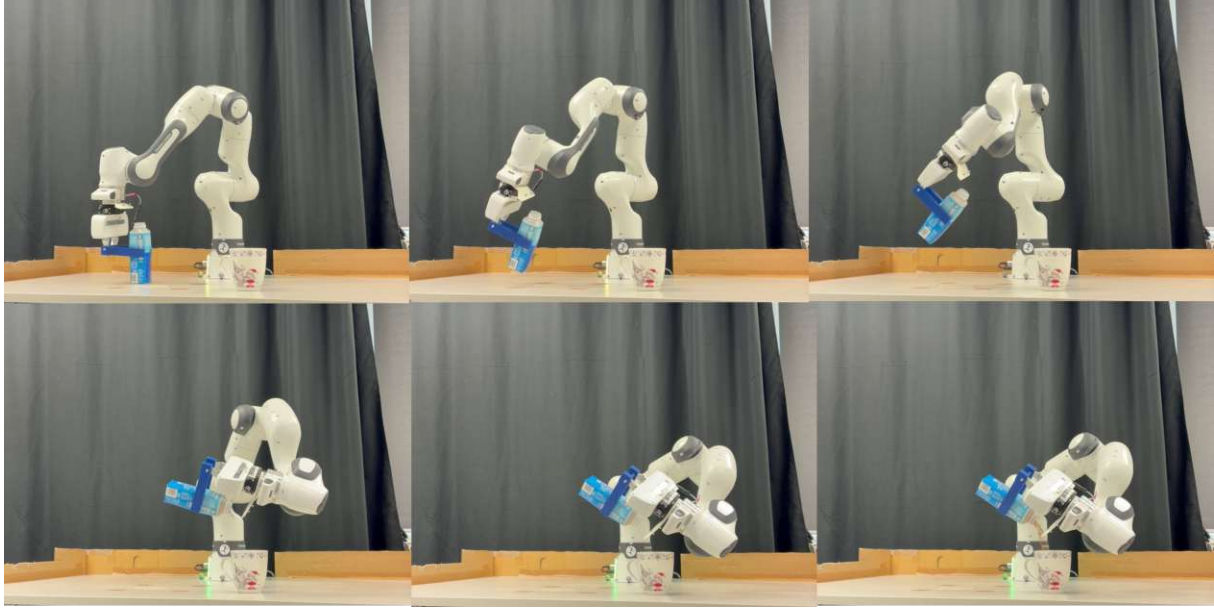


Figure 4.10: Successful Execution on the Franka Emika Robot

from a set of candidates generated during batch inference. The key idea is to evaluate each candidate trajectory based on how closely it aligns with the statistical properties of successful demonstrations in the training dataset. Rather than relying on a single heuristic, the method considers multiple geometric features that are indicative of successful task execution. At inference time, M candidate trajectories $\{\hat{\mathbf{X}}^{(i)}\}_{i=1}^M$ are generated via batch inference. Each candidate is scored using the Mahalanobis distance [58] with respect to the action-specific distribution of successful demonstrations:

$$D_M^2(\hat{\mathbf{X}}^{(i)}) = \left(\mathbf{f}(\hat{\mathbf{X}}^{(i)}) - \boldsymbol{\mu}_a\right)^\top \boldsymbol{\Sigma}_a^{-1} \left(\mathbf{f}(\hat{\mathbf{X}}^{(i)}) - \boldsymbol{\mu}_a\right) \quad (4.8)$$

where $\mathbf{f}(\hat{\mathbf{X}})$ denotes a feature vector of geometric metrics, and $\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a$ represent the action-specific mean and covariance computed from the training dataset. Based on empirical analysis of our use case, three metrics were identified as particularly relevant:

1. **Inter-object distance** d_{inter} : The distance between the manipulated object and the target object over the trajectory.
2. **Object rotation magnitude** θ_{rot} : The total rotation undergone by the manipulated object throughout the trajectory.
3. **Relative orientation** θ_{rel} : The angle between the principal axes of the manipulated object and the target object.

Accordingly, the geometric feature vector is defined as:

$$\mathbf{f}(\hat{\mathbf{X}}) = \begin{bmatrix} d_{\text{inter}} \\ \theta_{\text{rot}} \\ \theta_{\text{rel}} \end{bmatrix} \quad (4.9)$$

The score defined in Eq. (4.8) reflects how well a candidate trajectory matches the distribution of successful demonstrations. The selected trajectory is defined as:

$$\hat{\mathbf{X}}^* = \arg \min_i D_M^2(\hat{\mathbf{X}}^{(i)}) \quad (4.10)$$

As shown in Eq. (4.10), the trajectory with the smallest Mahalanobis distance is selected for execution. The Mahalanobis distance (Eq. (4.8)) measures how likely a candidate trajectory is under the distribution of successful demonstrations, with lower values indicating better alignment. Unlike the Euclidean distance, it normalizes each feature by its variance and accounts for correlations between features, making it well-suited for this application. The feature vector defined in Eq. (4.9) captures complementary aspects of the manipulation process, including spatial proximity and geometric alignment. For instance, successful pouring requires not only a specific relative distance between the bottle and the cup, but also a characteristic rotation and alignment between their principal axes. To model the variability of these features, statistical summaries are computed separately for each manipulation action (*pick*, *pour*, *place*). The mean vector $\boldsymbol{\mu} = [\mu_d, \mu_{\text{rot}}, \mu_{\text{rel}}]$ and the covariance matrix $\boldsymbol{\Sigma}$ are estimated from the dataset, capturing both expected values and correlations between features. Table 4.1 reports the mean and standard deviation for each metric. It is important to note that the relevance of individual metrics varies across tasks. For example, the relative orientation during the *place* action exhibits a high variance, indicating that it is less informative for distinguishing successful executions. This observation motivates the use of a multivariate distance measure (Eq. (4.8)) that accounts for feature correlations and differing variances.

Action	Mean Value	Standard Deviation
(1) the distance between objects	distance (m)	m
Pick	0.333	0.104
Pour	0.208	0.036
Place	0.318	0.071
(2) the object rotation over the entire trajectory	angle (deg)	deg
Pick	3.17	2.17
Pour	134.99	12.76
Place	133.21	15.02
(3) the angle between the primary direction of each object	angle (deg)	deg
Pick	5.95	6.57
Pour	132.04	16.13
Place	123.76	75.44

Table 4.1: Target Measurements in the Dataset: Mean and Standard Deviation

During inference, the diffusion model generates multiple candidate trajectories $\{\hat{\mathbf{X}}^{(i)}\}_{i=1}^M$. For each candidate, the geometric feature vector $\mathbf{f}(\hat{\mathbf{X}}^{(i)})$ (Eq. (4.9)) is computed, and the Mahalanobis distance (Eq. (4.8)) is evaluated with respect to the corresponding action-specific distribution. The trajectory with the smallest distance, as defined in Eq. (4.10), is selected for execution. To further improve execution robustness, an additional post-processing step is applied to the selected trajectory. Specifically, the deviation between the trajectory’s relative orientation and the dataset mean is computed. This deviation is then gradually corrected by adjusting the rotation component over the final 25% of the trajectory timesteps. This correction ensures more consistent alignment between objects during critical phases of the manipulation, such as pouring.

While this optimization strategy improves the reliability of open-loop execution, it remains limited by the absence of real-time feedback. A closed-loop approach incorporating online perception and trajectory adaptation could further enhance performance, but is currently not feasible due to the computational cost of dense 3D particle tracking. The effectiveness of the proposed open-loop framework, including the trajectory selection strategy, is evaluated empirically in the following chapter.

Chapter 5

Evaluation

This chapter evaluates the proposed framework with respect to our two research questions. First, we ask whether out-of-domain data processed through particle-based tracking can be leveraged to learn robot motion policies that generalize across a variety of spatial configurations and robot embodiments. We evaluate the learned policy across a structured grid of 24 bottle–cup configurations using a Franka Emika Panda robot to assess the spatial coverage. We then transfer the same policy, without retraining, to a Kinova robot arm to examine the embodiment-independence enabled by particle-based trajectory learning. Second, we investigate whether grounding predictions in object-centric geometric representations, rather than raw visual observations, enables generalization to novel scenes and previously unseen objects. To this end, we compare against a visual diffusion policy baseline and assess cross-object generalization to unseen cups and bottles, evaluating the degree to which object-centric representations confer robustness to appearance-level variation and novel object geometry.

5.1 Evaluation on the Validation Set

The validation set consists of nine pouring motions. All models were trained for 30,000 epochs and evaluated against ground truth trajectories. Four configurations were compared: a baseline model using a scaled linear scheduler; a variant using the cosine scheduler from the diffusers library ('squared-cos_cap_v2') [53]; a third model combining the cosine scheduler with stronger data augmentation, expanding the trajectory scaling range from 90–110% to 75–125% and condition noise from 1% to 5% Gaussian; and finally a smaller architecture with one CrossAttention block removed and remaining block sizes halved.

The Endpoint Error measures Euclidean distance between predicted and ground truth positions at the final timestep; since the end pose is critical for task success (e.g., correctly positioning a pouring vessel), this metric directly reflects task-level accuracy. The Trajectory Length Error captures the differ-

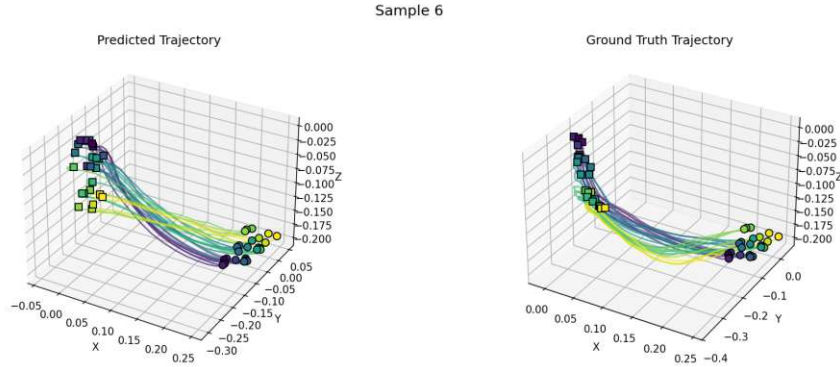


Figure 5.1: Example Trajectory on Validation Set - left: Generated Trajectory, right: Ground Truth Trajectory

ence in total path length. The Centroid Trajectory Error measures deviation of the predicted centroid path across all timesteps; it evaluates the coarse motion profile. The Mean Per-Timestep Error evaluates average per-point displacement aggregated over all timesteps, providing a measure of how well the full point cloud geometry is preserved throughout the motion. The Chamfer Distance measures nearest-neighbour distances in both directions between predicted and ground truth point clouds, capturing overall shape similarity independent of point correspondence. Together, these metrics are complementary: endpoint and length errors assess global trajectory properties, the centroid error tracks coarse motion profile, and the per-timestep and Chamfer metrics evaluate fine-grained geometric fidelity at each step. The final four rows of the table reports metrics for the optimization algorithm. The Mahalanobis Distance calculates how far a predicted sample lies from the training distribution across these three metrics, accounting for their covariance; lower values indicate trajectories more consistent with the training distribution. The metrics are reported in metres and degrees.

Metric	Baseline	Cosine Scheduler	Larger Augmentation	Small model
Endpoint Error	0.0740	0.0732	0.0841	31.5152
Trajectory Length Error	0.0640	0.0691	0.0723	34.4971
Centroid Trajectory Error	0.0050	0.0048	0.0057	326.0633
Mean Per Timestep Error	0.0712	0.0714	0.0760	15.7407
Chamfer Distance	0.0396	0.0397	0.0480	15.1538
Hausdorff Distance	0.0749	0.0811	0.0962	33.7253
Mean Rotation	109.5958	109.1133	109.0838	66.4012
Mean Inter-Object Distance	0.1998	0.2000	0.2033	31.4735
Mean Relative Rotation	109.6837	108.4714	108.8098	38.6051
Mahalanobis	5.2226	5.1658	5.3665	1 082 582.2500

Table 5.1: Evaluation Validation Set

The smaller model fails entirely, producing errors several orders of magnitude larger than the remaining configurations, as reflected both numerically and in the degenerate trajectories shown in Fig-

ure 5.2. The three remaining models perform comparably. Notably, the larger augmentation does not improve accuracy on unseen data, suggesting the baseline models do not suffer from significant overfitting. For the best-performing models, endpoint error is approximately 7.3cm and trajectory length error approximately 6.4cm, reasonable but non-negligible values, which motivates the optimization strategy described in the previous chapter. The centroid trajectory error is low at around 5mm, indicating the overall motion path is well captured, while the Chamfer distance of 3.96cm reflects some deviation in internal point cloud structure. While the overall motion is accurately captured, as reflected by the low centroid trajectory error, the internal point cloud exhibits more noticeable discrepancies, as indicated by the higher Chamfer distance. This explains the results of the endpoint error, which calculates particle-level deviations from the ground truth at the end point of the trajectory. The predicted mean rotation and mean relative rotation are approximately 20 degrees below the dataset mean (see: Table 4.1: Mean: 135 degrees), suggesting that single-sample inference tends to underestimate rotational quantities. We perform an ablation study, using the optimization strategy, to analyse the effects on the validation dataset.

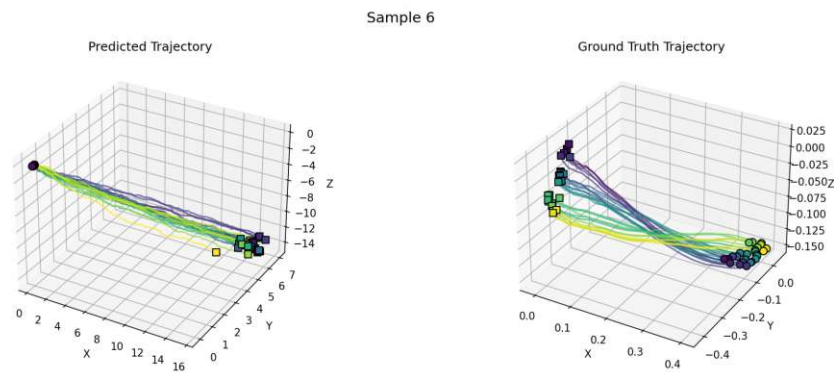


Figure 5.2: Example Failed Trajectory on Validation Set Smaller Model - left: Generated Trajectory, right: Ground Truth Trajectory

Metric	Baseline (No Optimization)	Baseline	Cosine Scheduler	Larger Augmentation	Small model
Endpoint Error	0.073998	0.0844	0.0715	0.0942	6.2058
Trajectory Length Error	0.063950	0.0395	0.0374	0.0425	7.4809
Centroid Trajectory Error	0.004987	0.0052	0.0031	0.0055	12.0783
Mean Per Timestep Error	0.071199	0.0731	0.0624	0.0786	3.1086
Chamfer Distance	0.039607	0.0447	0.0377	0.0547	2.9420
Hausdorff Distance	0.074946	0.0836	0.0723	0.0877	6.6606
Mean Rotation	109.5958	119.9283	119.8370	122.9668	62.4874
Mean Inter-Object Distance	0.1998	0.2059	0.2149	0.2204	6.0935
Mean Relative Rotation	109.6837	117.3613	116.8935	121.4805	51.2736
Mahalanobis	5.222622	1.8902	1.8946	1.8066	37994.9531

Table 5.2: Ablation Validation Dataset Performed with Mahalanobis Optimization

With optimization, the cosine scheduler model consistently outperforms the others across all tra-

jectory metrics. Batch inference substantially reduces trajectory length error from 6.4cm to 3.7cm, consistent with the distance-sensitive nature of the Mahalanobis scoring, and all other metrics improve as well. The Mahalanobis distance drops from 5.17 to 1.89, confirming that selected samples are more representative of the training distribution. While the geometric metrics show improvement under optimization, they may not fully capture the performance gains. Metrics such as Chamfer Distance and Mean Per-Timestep Error measure positional deviation of point cloud geometry, but are relatively insensitive to rotational correctness when the overall shape and position are approximately right, a bottle rotated slightly incorrectly may still produce a low Chamfer distance if its spatial location is accurate, as it only compares each point to the nearest neighbour from the ground truth point cloud. The mean rotation rises from approximately 109 degrees in single-sample inference to approximately 120 degrees under Mahalanobis optimization, closing roughly half the gap toward the dataset mean (see Table 4.1). On the other hand, the endpoint error increases by 1 cm, rising from 7.4 cm to 8.4 cm. This can be attributed to a shift in the mean inter-object distance, which moves closer to the dataset mean (20.8 cm), increasing from 19.9 cm to 20.6 cm. By favouring samples that lie closer to the training distribution across all three scoring metrics jointly, the selection process implicitly biases toward more representative rotational behaviour. This suggests the true performance benefit of optimization may be larger than the geometric metrics alone indicate, particularly for task success where the final rotational pose of the bottle is critical to task success. To further evaluate the importance of the optimization strategy an ablation study is performed on the robot as well.

5.2 Experimental Setup

To assess the performance of the proposed motion policy in real-world settings, a structured evaluation environment is defined in which the relative spatial configuration between the manipulated object (bottle) and the target object (cup) is systematically varied. The setup consists of a discrete grid with six positions, labeled A–C along the horizontal axis and 1–2 along the depth axis, as illustrated in Figure 5.3. Adjacent positions are spaced 20 cm apart, with the central column (B) aligned with the robot base. Position B1 is located 32 cm from the base.

The pouring task is chosen as the primary evaluation behaviour, as it represents the most challenging skill within the proposed framework. While pick-and-place sub-tasks were considered for comparison, they were ultimately deemed less suitable. Our policy assumes a rigid connection between the end-effector and the manipulated object. The pickup step could be readily automated using grasp pose estimation methods such as Contact-GraspNet [59], which has been successfully applied in similar manipulation pipelines [8]; however, it was excluded due to time constraints. The placement action is

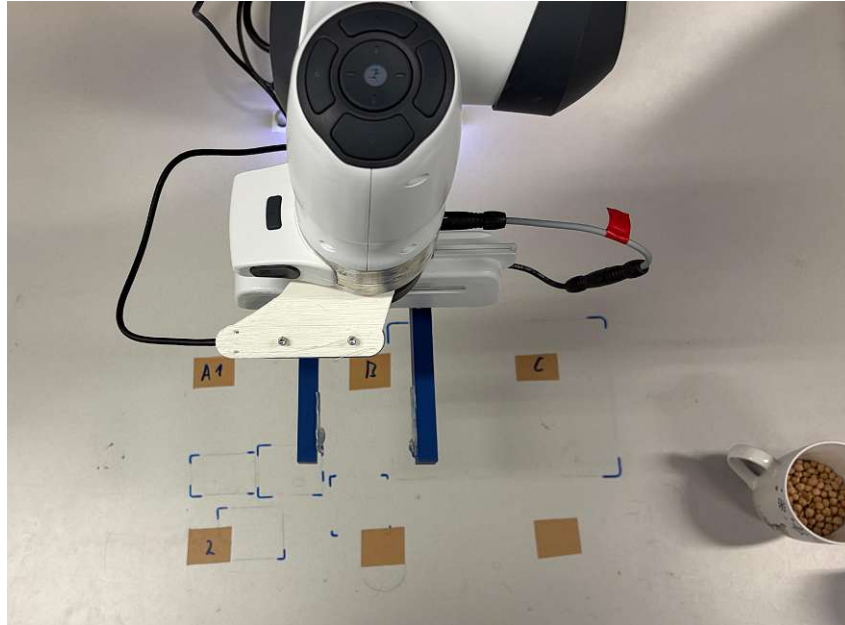


Figure 5.3: Evaluation Test Setup

also difficult to evaluate consistently, as point cloud sampling at the target position is not always feasible depending on the camera angle. Consequently, we omit these experiments. In contrast, pouring requires coordinated control of position, orientation, and relative object alignment. It also involves a dynamic inter-object relationship between the bottle and the cup, making it both the most demanding and the most meaningfully measurable task. Experiments are conducted using a Franka Emika Panda robot [50], equipped with custom 3D-printed fingertips designed to ensure stable grasping of the bottle. For simplicity, the bottle is grasped manually by moving the robot in gravity compensation mode. The grasping side is determined by the spatial configuration of the bottle and the cup, as well as the associated kinematic constraints: for transitions from left to right (A to B, A to C, B to C), the bottle is gripped from the left; for the reverse directions (C to A, C to B, B to A), it is gripped from the right. To prevent water damage to the hardware, dried chickpeas are used instead of liquid. While they do not fully replicate fluid dynamics, they exhibit granular flow characteristics that approximate key aspects of pouring while eliminating the risk of hardware damage. All valid combinations of bottle and cup placements are evaluated under two constraints: (i) both objects must occupy different grid positions, and (ii) they must not lie within the same column, as an object in the foreground would otherwise occlude the object behind it from the camera's perspective. This results in 24 distinct configurations. Each configuration is executed four times using the full policy, including batch inference (20 samples), Mahalanobis-based trajectory selection, and rotation correction. A trial is considered successful if the majority of the bottle contents reaches the cup. This is assessed qualitatively by ensuring that less than 10% of the chickpeas are spilled. Minor spillage is tolerated, as it can result from the bouncing

behaviour of the chickpeas.

5.3 Real World Evaluation

	A1				B1				C1				A2				B2				C2			
A1					25%	50%	25%	0%	25%	50%	25%	0%					0%	0%	0%	100%*	0%	100%	0%	0%
B1	0%	0%	50%	50%*					0%	50%	25%	25%*	0%	0%	0%	100%*					0%	0%	0%	100%*
C1	75%	25%	0%	0%	50%	25%	25%	0%					0%	100%	0%	0%	0%	0%	25%*	75%*				
A2					75%	0%	25%	0%	25%	75%	0%	0%					25%	0%	75%	0%	0%	25%	50%	25%
B2	0%	0%	100%	0%					0%	50%	50%	0%	50%	25%	25%	0%					0%	50%	50%	0%
C2	25%	75%	0%	0%	50%	0%	50%	0%					25%	75%	0%	0%	25%	25%	50%	0%				

Table 5.3: Results Test Setup. Rows indicate the bottle position, columns indicate the cup position. Each cell shows the outcome distribution across four trials per configuration, color-coded by result quality. Asterisks denote configurations where one or more trials were not executed due to kinematic infeasibility.

In Table 5.3, rows correspond to the bottle position and columns to the cup position. In total, 96 trials are conducted, of which a subset could not be executed due to kinematic infeasibility (joint limitations). Since the proposed policy does not incorporate the robot’s kinematic model, it may generate motion plans that violate joint limits or cause self-collisions. This issue is most pronounced in configurations where the cup is placed in the central column B, such as A1 to B1, C1 to B1, and A2 to B1, where the required motions push the robot toward the boundaries of its reachable workspace. Future work could address this limitation by improving trajectory selection within batch inference by incorporating the kinematics of the underlying robot platform. In particular, inverse kinematics could be solved explicitly, and trajectories without feasible solutions could be rejected a priori. The infeasible trials are marked with asterisks in Table 5.3, leaving 77 valid trials. Across the 77 valid trials, the policy achieves a success rate of 66.2% (51/77 trials), comprising 24.7% perfect executions (no spill) and 41.6% near-successful executions (less than 10% spill). The remaining 33.8% are classified as failures, including 32.5% with significant spill and 1.3% with no chickpeas entering the cup. Considering that the policy operates in a fully open-loop manner, predicting the entire trajectory without online feedback, this performance is notable. Errors occurring during execution cannot be corrected, yet the results indicate that the learned policy captures relevant geometric relationships and task dynamics required for successful pouring. Qualitatively, the generated motions exhibit human-like characteristics. Trajectories typically follow elevated, arc-shaped paths and approach the cup from above, reflecting patterns present in the training data. This behaviour is evident in both long-range and short-range executions (Figures 5.4 and 5.5).

Failure cases, not produced by kinematic infeasibility, are primarily associated with overshooting, or undershooting the target position. This is consistent with the stochastic nature of the generative model, where small deviations in predicted trajectories can lead to large task-level errors. An example of such a failure is shown in Figure 5.6. These observations suggest that incorporating feedback or

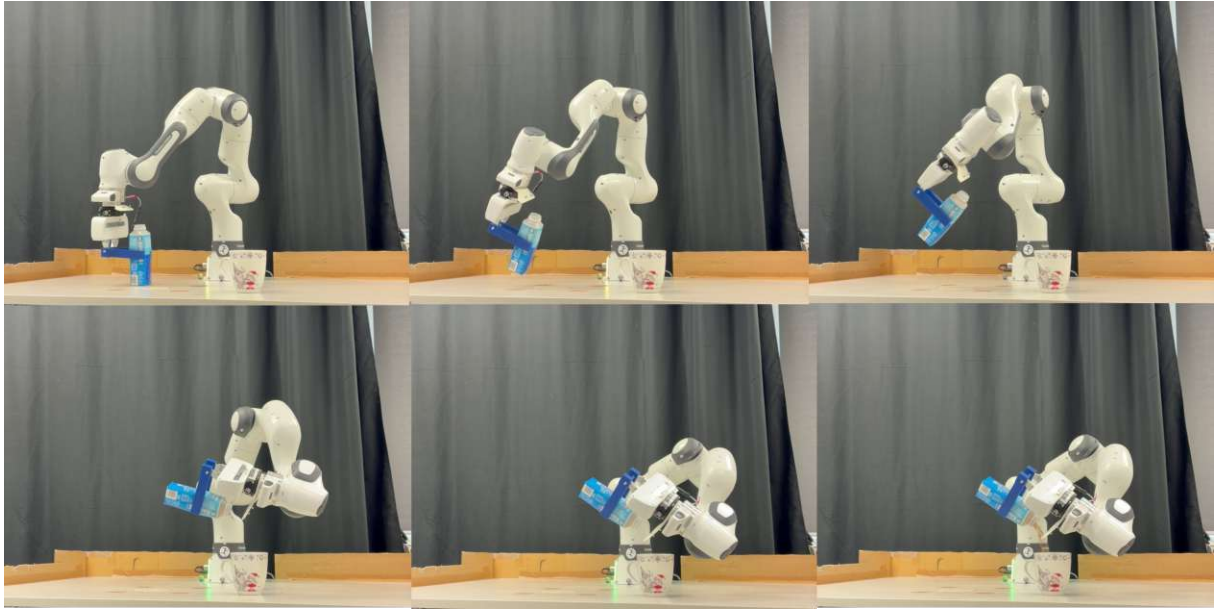


Figure 5.4: Example of Successful Pouring Execution

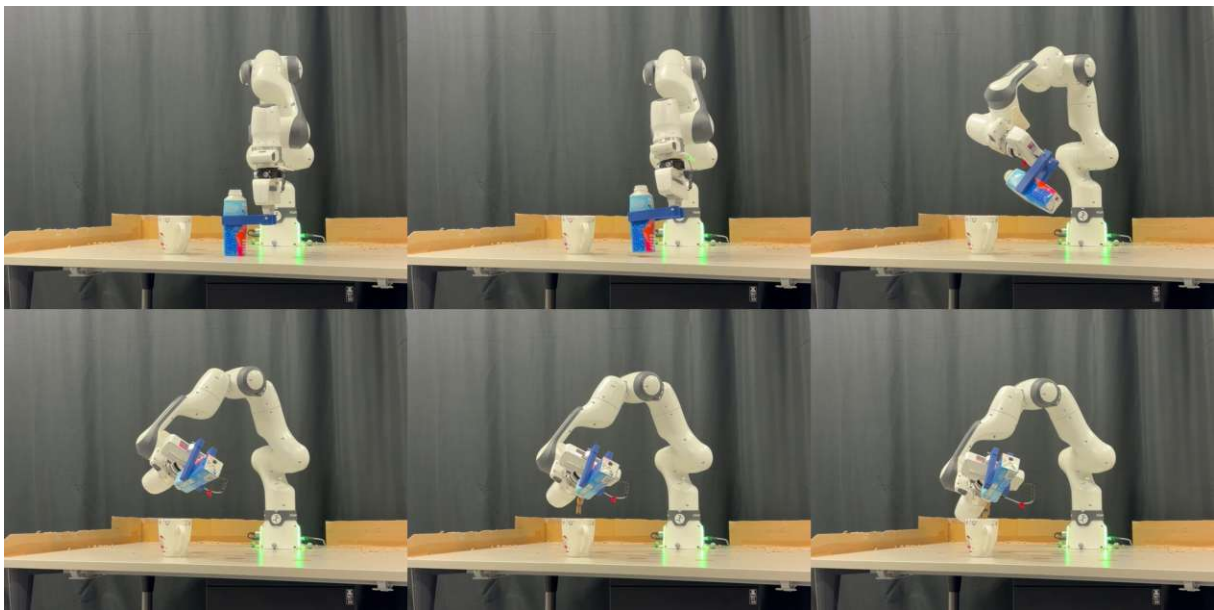


Figure 5.5: Example of Successful Short-range Pouring Execution

adaptive control mechanisms could further improve robustness.

5.3.1 Effects of Point Cloud Ordering

As described in the Method section, point clouds in both training and inference are sorted using the same grid-based algorithm, ensuring that point indices carry consistent spatial meaning across all observations. To isolate the contribution of this ordering, a controlled experiment is conducted on the validation set in which the point cloud order is replaced with a random permutation. Figure 5.7 illus-

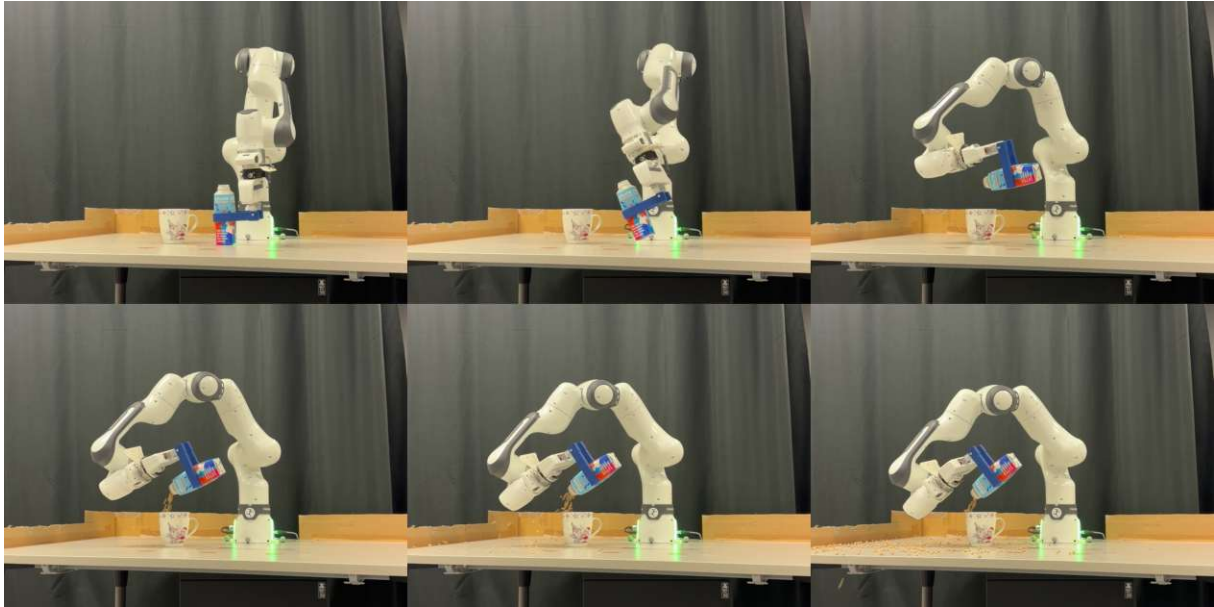


Figure 5.6: Example of Failed Pouring Execution

trates the result. The right side shows the ground truth trajectory generated from a correctly ordered point cloud, where colours encode the index assignment produced by the grid-based scheme. The left side shows the model output when the same observation is presented with a randomised index order. Although the predicted trajectory preserves the correct translational motion, the rotational component is entirely disrupted. The model consistently assigns upper-bottle trajectories to early-indexed points (shown in purple) and lower-bottle trajectories to later-indexed points (shown in yellow), without regard to their actual spatial locations. Consequently, a purple point located at the base of the bottle may be assigned a trajectory intended for the top region, and vice versa. This leads to motion plans that cannot guarantee rigidity, as internal points of the point cloud exhibit significant relative motion.

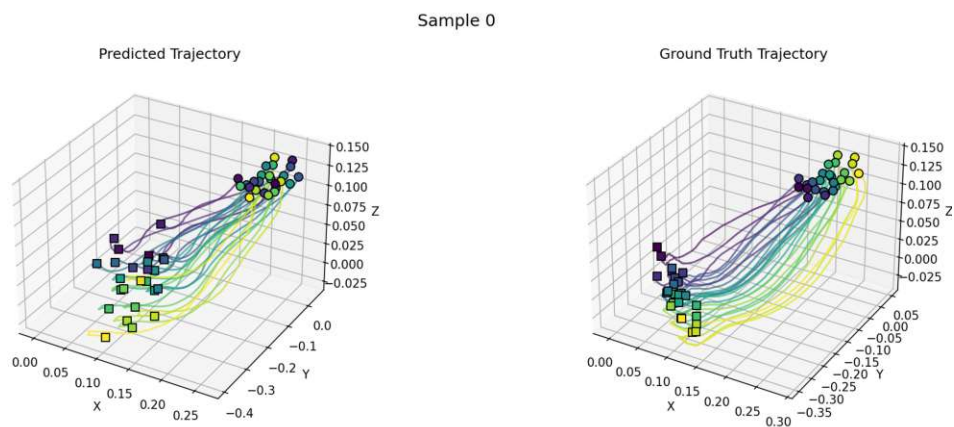


Figure 5.7: Effect of random point cloud ordering on the validation set. Right: trajectory generated from a correctly ordered observation. Left: trajectory generated from the same observation with a randomised point order. Colours indicate index assignment under the grid-based ordering scheme.

The consequences become considerably more severe when this disruption occurs on real-world observations. Figure 5.8 shows the point clouds perceived by the robot (red: cup, blue: bottle) alongside the trajectories generated under each condition. With correct ordering (right), the model produces the smooth, arc-shaped motion characteristic of successful pouring executions. With a randomised ordering (left), individual point trajectories become highly erratic, bearing no resemblance to the expected motion pattern. Executing such a trajectory on the physical robot is infeasible.

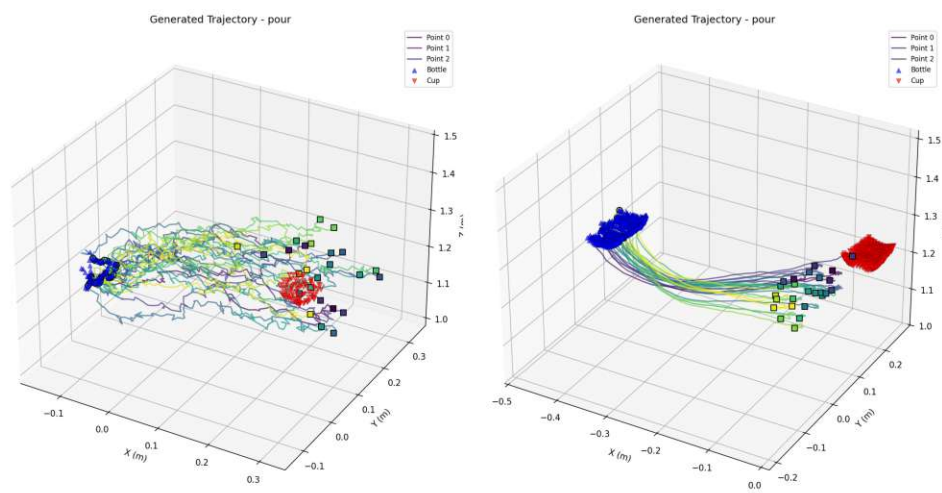


Figure 5.8: Effect of random point cloud ordering on real-world data. Right: trajectory generated from a correctly ordered observation. Left: trajectory generated from a randomly ordered observation. Red points indicate the cup; blue points indicate the bottle.

These results confirm that spatially consistent point cloud ordering is a critical prerequisite for correct model behaviour. The translational component of the predicted motion is relatively robust to index permutation, as it is governed primarily by the aggregate spatial position of the point cloud. The rotational component, however, depends on stable point-to-index correspondence and degrades immediately when that correspondence is broken. The grid-based ordering scheme described in the Method section ensures this correspondence is maintained reliably across all observations. Several directions could address this limitation in future work. One approach would be to associate each point explicitly with a fixed location on the object surface by augmenting its representation with an additional attribute encoding surface correspondence (i.e., a fourth dimension: (x, y, z, s)). This would make the representation invariant to permutations of the point ordering. Another would be to encode the point cloud into a permutation-invariant latent object representation prior to conditioning the diffusion model. We leave the exploration of these strategies to future work.

5.3.2 Ablation Study

To analyse the contribution of the optimization strategy of the proposed framework, we conduct an ablation study focusing on batch inference and the trajectory optimization strategy described in the Method chapter. We already showed the validity of batch inference in on the validation dataset in our previous section. To further assess the real-world impact, we perform additional experiments on two representative configurations (A1 to C1 and A2 to C2). We selected these configurations due to their high feasibility and performance in the main experiment. Each configuration is executed four times. Specifically, we compare three variants: (1) the full model with batch inference (20 samples), Mahalanobis-based selection, and rotation correction; (2) Optimization without rotation correction and (3) a single-sample inference setting without batch selection.

	A1 to C1				A2 to C2			
Our Policy	25%	75%	0%	0%	75%	25%	0%	0%
No Rotation Correction	25%	50%	25%	0%	50%	50%	0%	0%
Single Inference	100%	0%	0%	0%	50%	0%	50%	0%

Table 5.4: Ablation Study Results.

As summarized in Table 5.4, the full model achieves a 100% success rate across both scenarios. Removing rotation correction reduces performance to 87.5%, indicating that accurate alignment of the bottle orientation during the final pouring phase is critical for minimizing spillage. When batch inference is disabled, performance decreases further to 75%, confirming that sampling multiple trajectory candidates and selecting the most plausible one significantly improves robustness. Batch inference effectively mitigates the inherent variability of diffusion-based generation. Overall, the results demonstrate that both batch inference and trajectory refinement contribute meaningfully to performance, with the former improving robustness and the latter enforcing consistency with the training distribution.

5.4 Comparative Analysis

The proposed approach is compared against a visual diffusion policy baseline that operates on a raw camera input directly. Rather than reasoning over 3D object-centric point clouds, the baseline takes as observation the camera images from two RGB cameras together with the robot’s current end-effector position. The policy trains on a short action horizon. Learning to predict the next 16 end-effector positions, based on the last 8 observations. It operates in a closed-loop fashion, continuously re-observing the scene and refining its predictions based on updated visual feedback throughout execution. This stands in contrast to our approach, which constructs an explicit 3D scene representation from object-centric point clouds and predicts the full object trajectory in a single open-loop pass. Where the

baseline depends on visual consistency between training and deployment and corrects errors reactively, our method reasons about the spatial geometry of the scene upfront, committing to a complete motion plan before execution begins. In general, both methods are trained with data originally obtained from video data. The baseline is trained on 50 robot demonstrations for 1,000 epochs, compared to 30,000 epochs in our approach. While this disparity may appear to disadvantage the baseline, it reflects the differing convergence characteristics of the two models: our policy requires substantially more training steps to learn a shared representation across three distinct actions, whereas the visual policy, trained on a single pouring task, reaches convergence well within 1,000 epochs. The comparison should therefore be understood not as a controlled ablation, but as a realistic assessment of what each approach delivers under its own representative training conditions. Due to the visual nature of the baseline, the visual appearance at inference time must closely match the training distribution. To satisfy this, the fingertips are permanently fixed to the bottle during both training and evaluation, constraining the baseline to a subset of configurations where this setup is consistently feasible (A to B, A to C, and B to C). After excluding trials which violated joint limitations, the visual diffusion policy baseline achieves a success rate of 59.4%, consisting of 43.8% perfect executions and 15.6% near-successful executions, with a failure rate of 40.6%. Full results are shown in Table 5.5.

	A1				B1				A2				B2			
B1	0%	0%	0%	100%*					0%	0%	0%	100%*				
C1	50%	0%	50%	0%	50%	25%	25%	0%	0%	0%	0%	100%*	0%	0%	0%	100%*
B2	0%	0%	50%	50%					0%	0%	25%	75%				
C2	100%	0%	0%	0%	50%	50%	0%	0%	100%	0%	0%	0%	0%	50%	50%	0%

Table 5.5: Results for Visual Policy Trained on Robot Demonstrations

On the same subset of configurations, the proposed method achieves a higher overall success rate of 72.2%, comprising 33.3% perfect executions and 38.9% near-successful executions, reducing the failure rate to 27.8%. Looking more closely at the nature of successes and failures, the visual diffusion baseline achieves a higher rate of perfect executions (43.8%) relative to executions with a minor spill (15.625%). We attribute this to its closed-loop control strategy, which enables continuous correction during execution. Additionally, collected training demonstrations poured from a position close to the cup (low inter-object distance), providing strong visual priors for the pouring motion demonstrated by the policy at inference. When the baseline fails, it typically pours in front of rather than into the cup, which we interpret as errors in implicit depth estimation or the presence of visual features at inference time that were not well represented in the training data. As the baseline learns depth implicitly from image observations alone, it would likely benefit from a larger and more diverse set of demonstrations. Our proposed method, by contrast, consistently produces smooth and visually well-structured trajectories. Its failure modes are predominantly overshooting or undershooting the target, which we attribute

to the open-loop nature of execution, once a trajectory is committed to, no corrective action is possible. An additional source of discrepancy is that the pouring patterns in our training data differ from those in the visual policy’s demonstration set, meaning the two methods have learned qualitatively different motion strategies for the same task. These behavioral differences are clearly visible in Figure 5.9. The visual diffusion policy (left) exhibits a continuously re-adjusting motion plan, with a notable mid-approach correction that reflects the closed-loop nature of the controller. Depth is perceived only implicitly through end-effector pose observations, and the bottle rotation is deferred until the very end of the trajectory. Our object-centric policy (right), by contrast, produces a single smooth, arc-shaped trajectory that is planned once and executed without modification. The bottle already begins rotating toward the cup during the approach phase, which accounts for the arc’s characteristic shape but can also lead to early spillage or undershooting in failure cases.

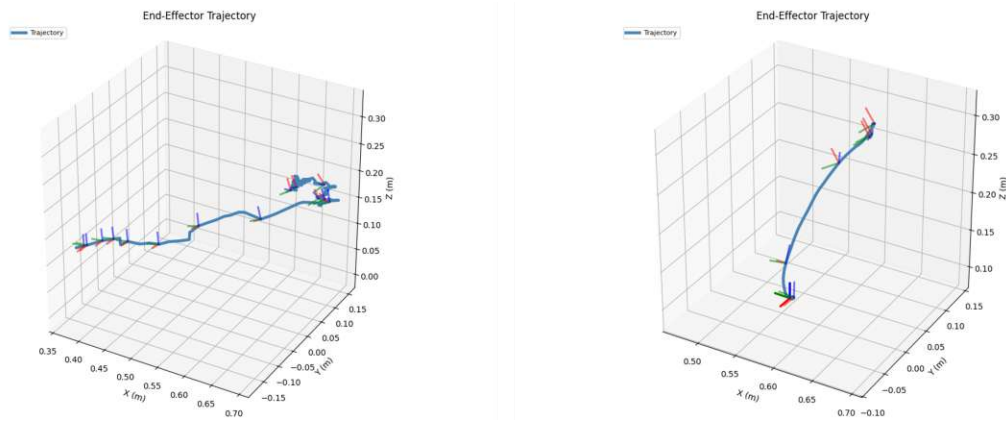


Figure 5.9: Pouring Trajectories: Left Visual Diffusion Policy and Right Our Object-centric Policy

5.5 Cross-Object Generalization

To evaluate generalization beyond the training distribution, experiments are conducted with previously unseen cups and a modified bottle. Three novel cups (white, green, and yellow) are introduced (see Figure 5.10), differing in color and geometry, along with a bottle of altered shape (see Figure 5.11). The object dimensions are shown in the appendix table A.3.

Evaluations are performed on the same configurations used in the ablation study (A1 to C1 and A2 to C2) to ensure consistency. Both the baseline visual policy and the proposed method are tested under identical conditions. The results (Table 5.6) reveal a clear difference in generalization capability. The baseline fails completely, achieving a 0% success rate across all cup variations. In contrast, the proposed method retains partial functionality, achieving 25% success for the white cup and 50% for the green cup. Performance drops to 0% for the yellow cup, which can be attributed to its smaller diameter:



Figure 5.10: Different Cups used for Cross Object Evaluation (From left to right: Original (ϕ 10cm), White (ϕ 8.5cm), Green (ϕ 8.5cm), Yellow (ϕ 7.2cm)).



Figure 5.11: Different Bottles used for Cross Object Evaluation (Left – original $H = 20$ cm, $\phi = 3.5$ cm, Right – novel $H = 24.5$ cm, $\phi = 2.8$ cm).

the reduced opening increases the precision required for successful pouring, meaning small trajectory deviations that would succeed with a larger target instead lead to overshooting or undershooting. This interpretation is consistent with the endpoint and trajectory length errors observed on the validation set (Table 4.1), which indicate that while the overall motion profile is well captured, non-negligible positional deviations remain, deviations that become increasingly consequential as target size decreases.

For the modified bottle, only the proposed method can be evaluated, as the baseline requires a fixed grasp configuration as described in the previous section. Figure 5.12 illustrates the generated pouring motions for three of the unseen cup variants. Despite never encountering these objects during training, the policy produces trajectories that follow the same elevated, arc-shaped approach paths characteristic of the successful executions observed in the main experiment. The bottle begins rotating toward the cup during the approach phase rather than at the final position, and the overall motion profiles remain

	Visual Policy								Our Policy							
	A1/C1				A2/C2				A1/C1				A2/C2			
White Cup	0%	0%	0%	100%	0%	0%	100%	0%	0%	50%	50%	0%	0%	0%	50%	50%
Green Cup	0%	0%	100%	0%	0%	0%	50%	50%	50%	0%	50%	0%	50%	0%	50%	0%
Yellow Cup	0%	0%	0%	100%	0%	0%	50%	50%	0%	0%	0%	100%	0%	0%	50%	50%
Different Bottle	-	-	-	-	-	-	-	-	25%	0%	25%	50%	0%	25%	75%	0%

Table 5.6: Results: Visual Policy vs. Our Policy on Unseen Objects

smooth and structured across all three cases. This qualitative consistency with the training distribution suggests that the object-centric representation captures task-relevant geometric relationships that transfer across changes in object color and shape, rather than overfitting to the specific visual or geometric properties of the training objects. The limited success rate nevertheless reflects the precision constraints identified above.

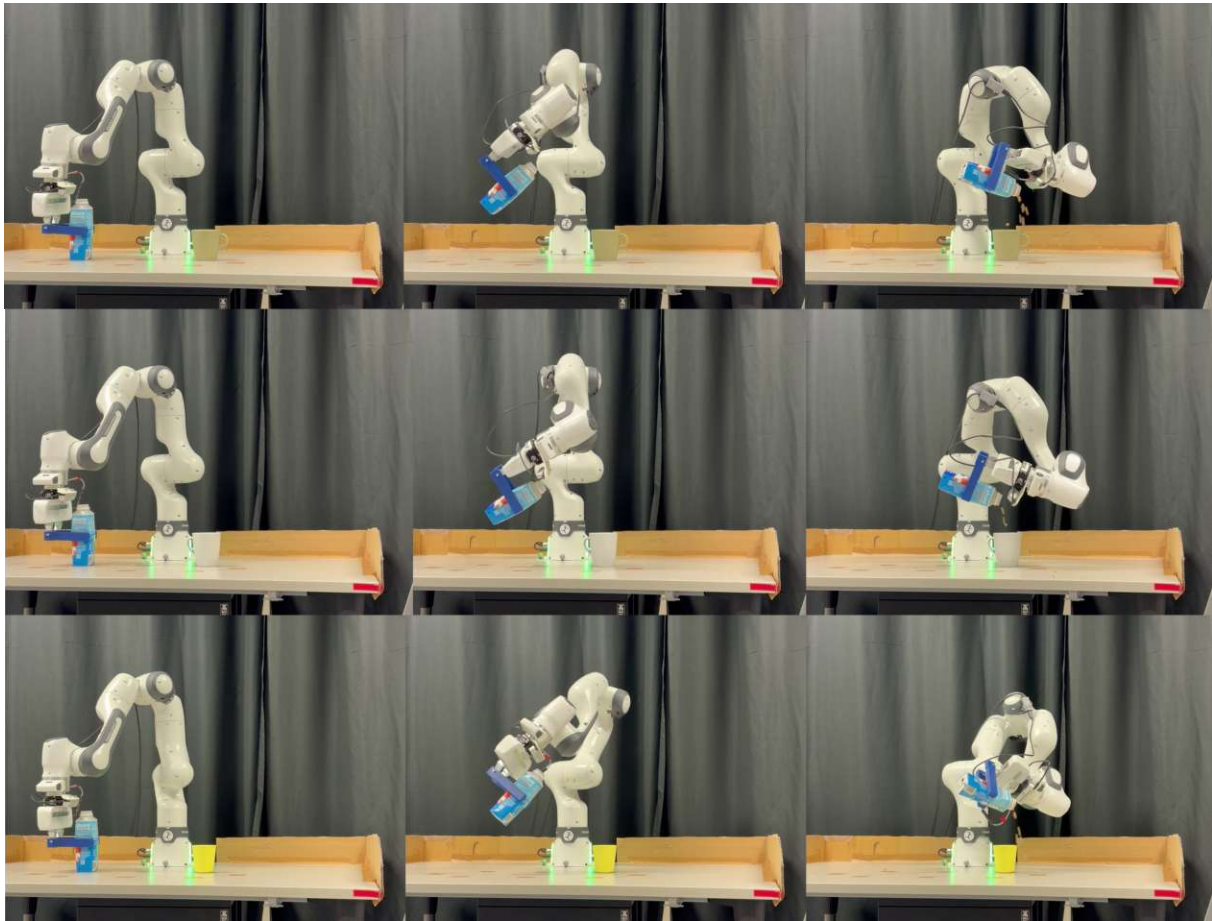


Figure 5.12: Three Pouring Motions Cross-Object Generalization: Green Cup, White Cup, Yellow Cup

These findings are further contextualized by comparing the sensitivity of the two approaches to environmental variation. The baseline is noticeably affected by changes in background elements, for example, removal of a curtain during evaluation leads to complete failure, whereas the proposed object-centric representation remains unaffected by such changes. This confirms that reliance on global vi-

sual features limits generalization, while grounding predictions in geometric object structure provides robustness to appearance-level variation. Taken together, the cross-object results suggest that the proposed method’s object detection, point sampling, and trajectory planning components generalize meaningfully to unseen objects, but that performance degrades as accuracy demands increase, whether due to smaller target geometry or subtle misalignment between the planned surface trajectory and the executed gripper path. Addressing these remaining precision gaps, through tighter optimization or a closed-loop correction mechanism, represents the most promising avenue for further improving cross-object generalization.

5.6 Cross-Embodiment Generalization

To evaluate whether the learned policy transfers across robotic platforms without retraining, experiments are conducted using a Kinova robot arm [60]. Due to workspace constraints arising from differences in gripper geometry and kinematic reach, evaluation is limited to bottle-cup configurations in the second row (A2, B2, C2), as illustrated in Figure 5.13. Unlike the Franka setup, where the bottle is grasped from the side using custom fingertips, the Kinova gripper approaches from behind, which introduces a positional offset that must be accounted for when translating the predicted point cloud trajectory into executable end-effector commands. To do so, the generated trajectory is shifted by 3.5 cm in the depth direction, moving it closer to the robot in accordance with the bottle depth. No other modifications to the policy are made.



Figure 5.13: Evaluation Test Setup Kinova Robot

The results across the evaluated configurations are reported in Table 5.7. Across all valid trials, the

policy achieves a success rate of 62.5%, including 33.3% perfect executions. This exceeds the performance of the Franka robot evaluated on the same subset of configurations, which achieves 54.2%, suggesting that the cross-embodiment transfer does not meaningfully degrade task performance relative to the platform on which the policy is executed. The overall success rate nonetheless remains below the 66.2% observed across all 24 configurations on the Franka, which is expected given the reduced workspace coverage and the additional positional offset introduced by the differing grasp geometry.

	A2				B2				C2			
A2					50%	25%	25%	0%	25%	50%	25%	0%
B2	50%*	25%	25%	0%					25%	0%	50%	25%
C2	50%	50%	0%	0%	0%	25%	75%	0%				

Table 5.7: Results Test Setup with Kinova Robot

Qualitatively, the trajectories generated on the Kinova retain the characteristic arc-shaped structure observed throughout the Franka experiments. The bottle follows an elevated approach path and begins rotating toward the cup during the arc, consistent with the motion patterns present in the training data. This qualitative consistency across platforms suggests that the learned representation captures task-relevant geometric relationships. To further illustrate this, Figure 5.14 shows 3D plots of end-effector trajectories and object point clouds across two representative configurations. The diversity of trajectories sampled from the diffusion model is visible, individual rollouts vary in the height of the arc and the onset of rotation, reflecting the stochastic nature of the generative policy.

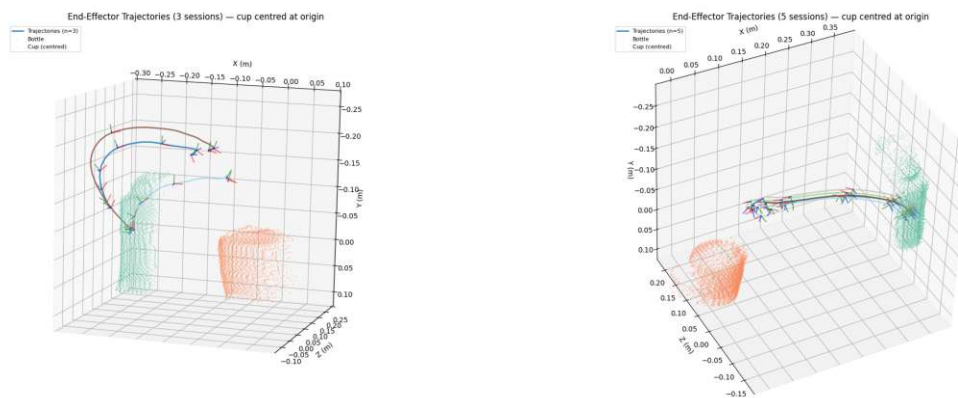


Figure 5.14: 3D trajectory plots on the Kinova robot. Left: short-range configuration with a high arc, showing variation in approach curvature across sampled trajectories. Right: long-range configuration, where trajectory diversity decreases and the pouring point is placed well above the cup, a behaviour driven by the Mahalanobis-based selection, which does not account for the initial inter-object distance.

The long-range configuration (Figure 5.14, right) exposes a limitation of the current optimization strategy. As the initial distance between the bottle and cup increases, the selected trajectory places the pouring point at a fixed height above the cup regardless of the spatial separation. This occurs because

the Mahalanobis-based selection scores candidate trajectories against the training distribution without conditioning on the initial configuration, meaning that trajectories suitable for short-range motions may differ in their geometric properties from those required for longer-range motions. A more adaptive selection mechanism that accounts for the initial inter-object distance could address this, for instance by conditioning the scoring function on the observed separation at the start of execution. An example of a successful pouring execution on the Kinova is shown in Figure 5.15, with additional executions across multiple configurations illustrated in Figure 5.16.



Figure 5.15: Example of Successful Pouring Execution on the Kinova Robot

Failure modes on the Kinova are consistent with those observed on the Franka, with overshooting and undershooting the target position remaining the dominant causes of task failure. This consistency further supports the interpretation that failure is attributable to the open-loop nature of execution and the resulting positional errors, rather than to platform-specific factors. Taken together, these results confirm that the proposed policy generalizes across robot embodiments without retraining, demonstrating that grounding the motion representation in object-centric point cloud geometry, rather than in robot-specific training, enables a degree of platform independence that would not be achievable with image-based approaches alone.

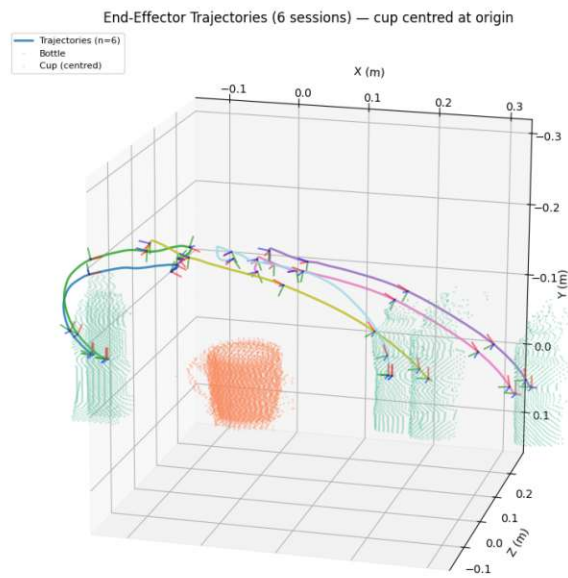


Figure 5.16: Qualitative Comparison of Multiple Executed Trajectories on the Kinova Robot

Chapter 6

Conclusion

This chapter concludes the thesis by summarizing the key findings of the proposed Learning from Observation framework and relating them back to the research questions introduced earlier. Building on the motivation that humans can learn complex motor skills from visual observation alone, we investigated whether similar capabilities can be transferred to robotic systems through object-centric 3D motion representations. We summarize the performance of particle-based trajectory learning from out-of-domain data and its ability to generalize across environments and robot embodiments. Furthermore, we reflect on the role of object-centric representations in improving robustness and enabling the discovery of inter-object relationships without explicit supervision. Finally, we discuss the limitations of the current system and outline promising directions for future work toward more scalable and closed-loop robot learning systems.

State-of-the-art robot learning methods broadly fall into two categories, each with significant drawbacks. VLA models and video-conditioned policies such as RT-2 [6] and Vid2Robot [16] learn an implicit sense of 3D structure through end-effector pose supervision during robot-specific training. Because spatial reasoning in these models is never made explicit, it emerges as a byproduct of fitting training motions tied to particular hardware, these methods are inherently bound to a robot embodiment in training and cannot straightforwardly transfer to new platforms. Approaches that do represent motion explicitly in 3D, such as CPM [8] or SPOT [46], rely on curated datasets with depth sensors, simulation environments, or CAD models. This dependence on specialized acquisition pipelines fundamentally prevents them from exploiting the vast amounts of manipulation-relevant videos freely available on the internet, limiting their scalability and generalization. Crucially, there remains a gap in end-to-end systems that learn explicit, 3D-aware motion representations directly from video data and can be deployed on real robotic hardware without any robot-specific training.

To address these limitations, we propose a Learning from Observation framework built around a particle-based 3D trajectory representation extracted directly from human demonstrations in videos. The framework consists of two distinct pipelines. The training pipeline: task-relevant objects are first identified and segmented from RGB-D videos using open-vocabulary detection (YOLOWorld [48]) and the Segment Anything Model [49], which produce precise binary masks of object surfaces. These masks initialize dense particle tracks that SpatialTracker [9] propagates through time, yielding temporally consistent 3D particle trajectories in metric Cartesian space. The resulting trajectories serve as the training signal for a conditional diffusion model [10], which learns the distribution over object-level motion conditioned on an initial scene observation and a discrete task label. The inference pipeline operates in the robot domain. At deployment, the same detection and segmentation mechanism extracts object point clouds from a live RGB-D camera stream. The trained diffusion model generates a batch of candidate 3D particle trajectories, from which the most likely to succeed candidate is selected via Mahalanobis-distance scoring against the training distribution. The selected trajectory is then converted into a sequence of rigid object poses and executed on the robot through a Cartesian impedance controller. Because the policy operates entirely in object-centric 3D space rather than robot joint space, this execution step is embodiment-agnostic: the same predicted motion can be mapped to different robotic platforms via standard geometric transformations, requiring no retraining.

6.1 Answering the Research Questions

Learning motor skills by observing others and subsequently reproducing them in new situations is a fundamental aspect of human learning [3]. In this thesis, we adopt this perspective as motivation for robot learning from observation: the goal is to enable robots to acquire manipulation skills by observing human demonstrations and applying them in novel environments. We implemented and evaluated this concept in this thesis and can now answer the research questions posed in the Introduction of this thesis.

Can we leverage out-of-domain data using particle-based tracking to learn generalizable robot motion policies?

The results show that out-of-domain human demonstration data, when processed through particle-based tracking using SpatialTracker [9], can be transformed into a representation that supports robust and generalizable robot motion learning. Although the system is trained on human demonstrations rather than robot-specific trajectories, the learned policy successfully generalizes across both environmental configurations and robotic embodiments. In the main evaluation on a structured grid of 24

bottle–cup configurations, the proposed policy achieves a success rate of 66.2% over 77 valid real-world trials, despite operating in a fully open-loop setting without any feedback correction during execution. The generated trajectories remain diverse yet feasible, consistently producing smooth, human-like arc motions that reflect the structure of the underlying demonstration data. This indicates that the particle-based representation preserves essential geometric and temporal structure required for manipulation. Generalization beyond the training distribution is further demonstrated in cross-embodiment experiments. Without retraining, the same policy transfers from a Franka Emika Panda robot to a Kinova arm, achieving a success rate of 62.5% on the same task subset. Despite differences in kinematics, grasp geometry, and workspace constraints, the policy maintains stable performance and produces consistent motion patterns. This confirms that the learned representation encodes task-level geometric structure that is transferable across embodiments. Taken together, these results support the hypothesis that particle-based tracking of out-of-domain human demonstrations provides a viable mechanism for learning robot motion policies with both spatial generalization and embodiment independence.

How to leverage object-centric information in learning generalizable robot motion policies?

The second research question investigates whether object-centric representations improve generalization by focusing learning on task-relevant structure while ignoring irrelevant environmental variation. The experimental results clearly indicate that grounding perception in object-level geometry significantly improves robustness compared to a visual diffusion policy baseline operating on raw RGB observations. In cross-object experiments involving unseen cups and a modified bottle, the proposed method retains performance, achieving up to 50% success on unseen object instances, whereas the visual diffusion baseline fails completely with 0% success. This sharp contrast highlights the importance of explicitly representing objects and their spatial relationships, rather than relying on implicit visual features. The improvement is primarily attributed to the use of a local object-centric representation. The modular structure of our approach allows us to leverage mature models such as YOLOWorld [48] and SAM [49] to reliably extract the point cloud representation from visual input, enabling robustness across variations in color, shape, and background clutter. Object detection and segmentation methods generalize significantly better than end-to-end learned visual features, allowing the policy to function under changes in lighting, scene composition, and object appearance. In contrast, the visual diffusion baseline is highly sensitive to environmental changes, with performance degrading even under minor modifications such as background alterations, indicating a strong reliance on global visual context. Beyond robustness to appearance changes, the object-centric representation enables the policy to explicitly learn inter-object relationships. In the pouring task, this is reflected in the learned coordination between the bottle and cup, where trajectories encode relative spatial structure rather than absolute

image features. The policy consistently generates motions that respect the geometric relationship between objects, including approach direction, alignment, and pouring orientation, even when object instances are previously unseen. However, performance degrades as task precision requirements increase, particularly for smaller target geometries. This is consistent with the residual positional and rotational errors observed in the validation metrics (e.g., Chamfer and endpoint errors), which become more critical when object tolerances are reduced. Overall, the results demonstrate that object-centric representations improve generalization by (i) filtering out irrelevant environmental variation, (ii) leveraging robust pre-trained perception modules, and (iii) enabling explicit modeling of inter-object spatial relationships. Compared to visual diffusion policies, which rely on implicit scene encoding and depth perception, our object-centric policy provides substantially stronger generalization to novel objects and environments.

In summary, the experimental evaluation confirms both research hypotheses. First, particle-based tracking in out-of-domain data (human demonstrations) enables the learning of motion policies that generalize across spatial configurations and robotic embodiments, while maintaining diverse and feasible trajectory generation. Second, object-centric representations significantly improve robustness to environmental variation and unseen objects by focusing learning on task-relevant geometry and implicitly learning inter-object relationships, outperforming the visual-only baseline in generalization scenarios.

6.2 Limitations

Despite these promising results, several limitations remain. First, the overall system performance is constrained by the quality of the particle tracking stage. In particular, the current SpatialTracker [9] implementation exhibits inaccuracies that propagate through the pipeline and affect policy learning and execution. Improving tracking robustness and density is therefore critical for future progress. There are several more recent models [29, 30] that have emerged during the course of this thesis that may offer improved performance. Second, the policy does not account for environmental context beyond the tracked objects. Obstacles and scene constraints are not explicitly modeled, limiting the applicability of the approach in more complex environments. Additionally, the current formulation does not consider task-specific factors such as the fill level of the bottle, which significantly influences pouring dynamics. While such information could be incorporated as additional conditioning in the diffusion model, it is not yet explored in this work. Finally, although we achieve good performance with the open-loop system, it could still be improved using closed-loop control. The absence of feedback leads to failure modes such as overshooting and undershooting, which cannot be corrected during execution. This limitation becomes

particularly evident for previously unseen geometries, as observed in the cross-object experiments, where smaller cups and different bottle openings result in noticeable drops in performance.

6.3 Outlook

The proposed framework opens several promising directions for future research. A key extension is the integration of closed-loop control, where real-time perception is used to continuously update and refine predicted trajectories. This would significantly improve robustness and enable adaptation to dynamic environments. Another important direction is the extension of particle-based representations to more complex object categories. In particular, particle-centric approaches are naturally suited to modeling deformable objects, where explicit geometric representations are essential. Increasing the density and quality of tracked particles would further enhance the expressiveness of the representation. Beyond object-centric tracking, another promising avenue is the tracking of hand and wrist motion as learning data, capturing demonstrations directly from human manipulation and translating them into executable robot motion plans. This could significantly expand the range of learnable behaviors while reducing reliance on robot-collected demonstration data. Future work could also incorporate richer conditioning signals into the diffusion model, such as object affordances, physical properties, or task constraints (fill level of the bottle). This would enable more flexible and context-aware motion generation. Finally, scaling the approach using transformer-based diffusion architectures and training on large-scale internet video data could unlock significantly broader generalization capabilities.

Bibliography

- [1] A. Correia and L. A. Alexandre, "A survey of demonstration learning," *Robotics and Autonomous Systems*, vol. 182, p. 104812, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0921889024001969>
- [2] M. Point, "Diffusion models in machine learning: From chaos to creation," <https://medium.com/data-and-beyond/diffusion-models-in-machine-learning-from-chaos-to-creation-97853f229a0d>, September 2025, medium, Data And Beyond. Accessed: 2026-03-29.
- [3] A. Bandura and R. H. Walters, *Social learning theory*. Prentice-hall Englewood Cliffs, NJ, 1977, vol. 1.
- [4] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei, J. Fu, J. Gong, and X. Qiu, "Libero-plus: In-depth robustness analysis of vision-language-action models," 2025. [Online]. Available: <https://arxiv.org/abs/2510.13626>
- [5] E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frueh, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne,

- K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, M. Z. Irshad, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, V. Guizilini, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin, “Open x-embodiment: Robotic learning datasets and rt-x models,” 2025. [Online]. Available: <https://arxiv.org/abs/2310.08864>
- [6] A. Brohan, N. Brown, and et. al, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” 2023. [Online]. Available: <https://arxiv.org/abs/2307.15818>
- [7] T. Yuan, Y. Liu, C. Lu, Z. Chen, T. Jiang, and H. Zhao, “Depthvla: Enhancing vision-language-action models with depth-aware spatial reasoning,” 2025. [Online]. Available: <https://arxiv.org/abs/2510.13375>
- [8] W. Liu, J. Mao, J. Hsu, T. Hermans, A. Garg, and J. Wu, “Composable part-based manipulation,” 2024. [Online]. Available: <https://arxiv.org/abs/2405.05876>
- [9] Y. Xiao, Q. Wang, S. Zhang, N. Xue, S. Peng, Y. Shen, and X. Zhou, “Spatialtracker: Tracking any 2d pixels in 3d space,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [10] J. Ho and T. Salimans, “Classifier-free diffusion guidance,” 2022. [Online]. Available: <https://arxiv.org/abs/2207.12598>

- [11] G. Li, R. Wang, P. Xu, Q. Ye, and J. Chen, "The developments and challenges toward dexterous and embodied robotic manipulation: A survey," *IEEE Robotics Automation Magazine*, vol. 33, no. 1, pp. 24–38, 2026.
- [12] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, Apr. 2017. [Online]. Available: <https://doi.org/10.1145/3054912>
- [13] S. Ye, J. Jang, B. Jeon, S. Joo, J. Yang, B. Peng, A. Mandlekar, R. Tan, Y.-W. Chao, B. Y. Lin, L. Liden, K. Lee, J. Gao, L. Zettlemoyer, D. Fox, and M. Seo, "Latent action pretraining from videos," 2025. [Online]. Available: <https://arxiv.org/abs/2410.11758>
- [14] NVIDIA, :, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu, "Gr00t n1: An open foundation model for generalist humanoid robots," 2025. [Online]. Available: <https://arxiv.org/abs/2503.14734>
- [15] Y. Yan, E. V. Mascaro, and D. Lee, "Imitationnet: Unsupervised human-to-robot motion retargeting via shared latent space," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023, pp. 1–8.
- [16] V. Jain, M. Attarian, N. J. Joshi, A. Wahid, D. Driess, Q. Vuong, P. R. Sanketi, P. Sermanet, S. Welker, C. Chan, I. Gilitschenski, Y. Bisk, and D. Dwibedi, "Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers," 2024. [Online]. Available: <https://arxiv.org/abs/2403.12943>
- [17] H. Bharadhwaj, D. Dwibedi, A. Gupta, S. Tulsiani, C. Doersch, T. Xiao, D. Shah, F. Xia, D. Sadigh, and S. Kirmani, "Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation," 2024. [Online]. Available: <https://arxiv.org/abs/2409.16283>
- [18] Y. Du, M. Yang, B. Dai, H. Dai, O. Nachum, J. B. Tenenbaum, D. Schuurmans, and P. Abbeel, "Learning universal policies via text-guided video generation," 2023. [Online]. Available: <https://arxiv.org/abs/2302.00111>
- [19] S. Zhou, Y. Du, J. Chen, Y. Li, D.-Y. Yeung, and C. Gan, "Robodreamer: Learning compositional world models for robot imagination," 2024. [Online]. Available: <https://arxiv.org/abs/2404.12377>

- [20] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick, “Dreamitate: Real-world visuomotor policy learning via video generation,” 2024. [Online]. Available: <https://arxiv.org/abs/2406.16862>
- [21] P.-C. Ko, J. Mao, Y. Du, S.-H. Sun, and J. B. Tenenbaum, “Learning to act from actionless videos through dense correspondences,” 2023. [Online]. Available: <https://arxiv.org/abs/2310.08576>
- [22] C. Doersch, Y. Yang, M. Vecerik, D. Gokay, A. Gupta, Y. Aytar, J. Carreira, and A. Zisserman, “Tapir: Tracking any point with per-frame initialization and temporal refinement,” 2023. [Online]. Available: <https://arxiv.org/abs/2306.08637>
- [23] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht, “Cotracker: It is better to track together,” 2024. [Online]. Available: <https://arxiv.org/abs/2307.07635>
- [24] S. Cho, J. Huang, J. Nam, H. An, S. Kim, and J.-Y. Lee, “Local all-pair correspondence for point tracking,” 2024. [Online]. Available: <https://arxiv.org/abs/2407.15420>
- [25] A. W. Harley, Z. Fang, and K. Fragkiadaki, “Particle video revisited: Tracking through occlusions using point trajectories,” 2022. [Online]. Available: <https://arxiv.org/abs/2204.04153>
- [26] N. Karaev, I. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht, “Cotracker3: Simpler and better point tracking by pseudo-labelling real videos,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.11831>
- [27] S. Koo, D. Lee, and D.-S. Kwon, “Incremental object learning and robust tracking of multiple objects from rgb-d point set data,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 108–121, 2014, visual Understanding and Applications with RGB-D Cameras. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1047320313000606>
- [28] Y. Xiao, Q. Wang, S. Zhang, N. Xue, S. Peng, Y. Shen, and X. Zhou, “Spatialtracker: Tracking any 2d pixels in 3d space,” 2024. [Online]. Available: <https://arxiv.org/abs/2404.04319>
- [29] T. D. Ngo, P. Zhuang, C. Gan, E. Kalogerakis, S. Tulyakov, H.-Y. Lee, and C. Wang, “Delta: Dense efficient long-range 3d tracking for any video,” 2025. [Online]. Available: <https://arxiv.org/abs/2410.24211>
- [30] B. Zhang, L. Ke, A. W. Harley, and K. Fragkiadaki, “Tapip3d: Tracking any point in persistent 3d geometry,” 2025. [Online]. Available: <https://arxiv.org/abs/2504.14717>

- [31] D. Lee and Y. Nakamura, "Motion capturing from monocular vision by statistical inference based on motion database: Vector field approach," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007*, pp. 617–623.
- [32] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," 2023. [Online]. Available: <https://arxiv.org/abs/2302.12288>
- [33] A. Masoumian, H. A. Rashwan, J. Cristiano, M. S. Asif, and D. Puig, "Monocular depth estimation using deep learning: A review," *Sensors*, vol. 22, no. 14, 2022. [Online]. Available: <https://www.mdpi.com/1424-8220/22/14/5353>
- [34] M. Vecerik, C. Doersch, Y. Yang, T. Davchev, Y. Aytar, G. Zhou, R. Hadsell, L. Agapito, and J. Scholz, "Robotap: Tracking arbitrary points for few-shot visual imitation," 2023. [Online]. Available: <https://arxiv.org/abs/2308.15975>
- [35] C. Wen, X. Lin, J. So, K. Chen, Q. Dou, Y. Gao, and P. Abbeel, "Any-point trajectory modeling for policy learning," 2024. [Online]. Available: <https://arxiv.org/abs/2401.00025>
- [36] H. Bharadhwaj, R. Mottaghi, A. Gupta, and S. Tulsiani, "Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation," 2024. [Online]. Available: <https://arxiv.org/abs/2405.01527>
- [37] H. Li, Q. Feng, Z. Zheng, J. Feng, Z. Chen, and A. Knoll, "Language-guided object-centric diffusion policy for generalizable and collision-aware robotic manipulation," 2025. [Online]. Available: <https://arxiv.org/abs/2407.00451>
- [38] H. Ahn, E. V. Mascaro, and D. Lee, "Can we use diffusion probabilistic models for 3d motion prediction?" 2023. [Online]. Available: <https://arxiv.org/abs/2302.14503>
- [39] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," 2015. [Online]. Available: <https://arxiv.org/abs/1503.03585>
- [40] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: <https://arxiv.org/abs/2006.11239>
- [41] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022. [Online]. Available: <https://arxiv.org/abs/2010.02502>
- [42] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: <https://arxiv.org/abs/1505.04597>

- [43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” 2022. [Online]. Available: <https://arxiv.org/abs/2112.10752>
- [44] W. Peebles and S. Xie, “Scalable diffusion models with transformers,” 2023. [Online]. Available: <https://arxiv.org/abs/2212.09748>
- [45] P. Dhariwal and A. Nichol, “Diffusion models beat gans on image synthesis,” 2021. [Online]. Available: <https://arxiv.org/abs/2105.05233>
- [46] C.-C. Hsu, B. Wen, J. Xu, Y. Narang, X. Wang, Y. Zhu, J. Biswas, and S. Birchfield, “Spot: Se(3) pose trajectory diffusion for object-centric manipulation,” 2025. [Online]. Available: <https://arxiv.org/abs/2411.00965>
- [47] B. Wen, W. Yang, J. Kautz, and S. Birchfield, “Foundationpose: Unified 6d pose estimation and tracking of novel objects,” 2024. [Online]. Available: <https://arxiv.org/abs/2312.08344>
- [48] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, “Yolo-world: Real-time open-vocabulary object detection,” *arXiv preprint arXiv*, 2024.
- [49] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, “Segment anything,” 2023. [Online]. Available: <https://arxiv.org/abs/2304.02643>
- [50] S. Haddadin, “The franka emika robot: A standard platform in robotics research,” *IEEE Robotics Automation Magazine*, vol. PP, pp. 2–14, 12 2024.
- [51] D. Sliwowski and D. Lee, “Conditionnet: Learning preconditions and effects for execution monitoring,” *IEEE Robotics and Automation Letters*, vol. 10, no. 2, p. 1337–1344, Feb. 2025. [Online]. Available: <http://dx.doi.org/10.1109/LRA.2024.3520916>
- [52] Microsoft, “Azure kinect dk,” <https://azure.microsoft.com/de-de/products/kinect-dk>, accessed: 2026-04-01.
- [53] P. Von Platen, S. Patil, A. Lozhkov, P. Cuenca, N. Lambert, K. Rasul, M. Davaadorj, and T. Wolf, “Diffusers: State-of-the-art diffusion models,” 2022.
- [54] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” 2021. [Online]. Available: <https://arxiv.org/abs/2103.00020>

- [55] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, Sep 1976. [Online]. Available: <https://doi.org/10.1107/S0567739476001873>
- [56] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, ser. Open-Source Software workshop, April 2013, pp. 1–6.
- [57] D. Sliwowski, S. Jadav, S. Stanovic, J. Orbik, J. Heidersberger, and D. Lee, "Reassemble: A multimodal dataset for contact-rich robotic assembly and disassembly," 2025. [Online]. Available: <https://arxiv.org/abs/2502.05086>
- [58] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.
- [59] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," 2021. [Online]. Available: <https://arxiv.org/abs/2103.14127>
- [60] Kinova Robotics, "Gen3 robots," <https://www.kinovarobotics.com/product/gen3-robots>, 2026, accessed: 2026-04-01.

Appendix A

Model Parameters

Parameter	Value
down block types	CrossAttnDownBlock2D CrossAttnDownBlock2D CrossAttnDownBlock2D DownBlock2D
up block types	UpBlock2D CrossAttnUpBlock2D CrossAttnUpBlock2D CrossAttnUpBlock2D
in channels	3
out channels	3
layers per block	2
block output channels	[64, 128, 256, 256]
dropout rate	0.1
Noise Scheduler	
Scheduler	diffusers.DDIMScheduler
Num train timesteps	1000
beta start	0.0001
beta end	0.02
beta schedule	scaled_linear
prediction type	epsilon
num_inference_steps	50

Table A.1: Diffusion Model Configuration

Data Parameters	Value
Points per Point Cloud	25
Timesteps per Action	100
Training Parameters	
Num of training epochs	30000
Optimizer	Adam
Learning Rate	0.0001
Learning Rate Scheduler	CosineAnnealingWarmupScheduler
Batch Size	16
Noise Scheduler	
Num train noise timesteps	1000

Table A.2: Training Configuration

Diameter				
Original Cup	10 cm			
White Cup	8.5 cm			
Green Cup	8.5 cm			
Yellow Cup	7.2 cm			
(a) Cup Diameters				
		Opening Diameter		Height
		Bottle	3.5 cm	20 cm
		Second Bottle	2.8 cm	24.5 cm
		(b) Bottle Dimensions		

Table A.3: Object Dimensions