university of groningen

faculty of science and engineering

Low-Cost Imitation Learning for Dual-Arm Robots: Leveraging a Single Human Demonstration

Jiayun Zhang

University of Groningen

Low-Cost Imitation Learning for Dual-Arm Robots: Leveraging a Single Human Demonstration

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Prof. dr. Hamidreza Kasaei (Artificial Intelligence, University of Groningen)
and
Georgios Tziafas (Artificial Intelligence, University of Groningen)

Second evaluator: Prof. dr. Raffaella Carloni (Artificial Intelligence, University of Groningen)

Jiayun Zhang (s3541312)

Contents

Acknowledgments

First of all, I would like to thank my supervisors, Prof. Dr. Hamidreza Kasaei and Georgios Tziafas, for their support and guidance throughout this thesis. Their suggestions, feedback and encouragement were very helpful to my research process. Additionally, I learned a lot about AI and robotics under their guidance, this has had a significant impact on my career and life.

I would also like to thank my family and friends. Whenever I faced difficulties in study or life, they were always there for me and supported me unconditionally. I truly appreciate their love, patience, and encouragement.

Abstract

Imitation learning through human demonstration has become the mainstream method for current robotic learning field, its impressive superior performance has been shown in a huge amount of research. However, the expensive, complex and time-consuming data collection remains one of the main challenges of imitation learning. In this project, we proposed a framework that can automatically extract the 3D coordinates of hand keypoints from human demonstration videos and convert them into the trajectory of the robot end-effector. Based on this, we further developed a data augmentation method that can generate multiple robot end-effector trajectories with generalization capabilities from a single human demonstration. Finally, we utilized large language models to evaluate the generalization of the augmented data through in-context trajectory prediction, verifying the effectiveness of our method in generating high-quality, generalizable training data.

1 Introduction

1.1 Overview

With the high development of Artificial Intelligence in the field of robotics, people's expectations for robots are no longer limited to simple grasping, palletising and other task. They expect that robots can perform complex and dexterous tasks like humans, to getting closer to the goal of achieving Artificial General Intelligence(AGI) robots. In recent years, humanoid robots have attracted much attention, and many technology companies such as Unitree, Tesla, and Figure have launched humanoid robot products. Their performance is impressive, but how to make these humanoid robots complete tasks as flexibly and efficiently as humans is still a challenging issue.

Imitation learning as one of the best solution, can enable robots to learning and perform complex tasks from human demonstrations. As we all know, the ultimate goal of any deep learning methods is the generalization, and the same is true for deep learning-based robot policy. Robots need to recognize and operate new objects with different positions in a completely new environment and understand task instructions that have never been encountered. The premise for achieving this goal is to build a large-scale and diverse dataset covering a wide range of scenarios. However, the current robotic data collection method commonly uses motion capture or teleoperation, which are costly, complex, and extremely time consuming, so that the cost of building such a large-scale and diverse dataset may be prohibitively high.

Therefore, we are going to propose a low-cost, high-efficiency framework that can automatically extract dual-arm robot executable trajectories from human demonstrations in a single video. The framework is also combined with a generalizable data generation method to automatically construct a variety of task instances covering different object locations and categories, greatly improving the diversity and practicality of the data. Through this approach, we significantly reduce the cost and time of robot data collection required to train deep learning policy, providing a practical new path for promoting general robot skill learning.

1.2 Research Questions

To summarize, this thesis focuses on this following problem:

Question: How can single human demonstrations be utilized within a low-cost framework for data generation and augmentation to enable generalizable dual-arm robot learning?

1.3 Thesis Outline

- **Section 2** reviews some relevant literature, focusing on the development of robotic dexterous manipulation techniques. In particular, imitation learning is discussed, including the data collection methods and current challenges. Also, this section introduce the emerging application of large language models (LLMs) in robot trajectory prediction.

- **Section 3** describe the methodology of the proposed framework. It details how robot trajectories are extracted from human demonstrations through hand keypoint estimation and depth correction, how these are converted into gripper poses, and proposed how a data augmentation method is used to generate a diverse and generalizable set of training samples.

- **Section 4** describes the experimental setup and implementation details. This includes the collection of our self-recorded demonstration data, evaluation metrics for hand keypoint estimation in Arctic dataset, and use a retrieval-based prompting method for LLM-based trajectory prediction. It also shows the test settings used to assess the generalization in the real world.
- **Section 5** reports the experimental results. Quantitative evaluations demonstrate the accuracy of hand keypoints extraction and the effectiveness of depth alignment. The performance of the LLM in generating robot trajectories is analyzed under various augmentation and real-world settings. Key limitations are also identified and discussed.
- **Section 6** concludes the thesis. It summarizes the main contributions, answers the research question, and proposes several directions for future work.

The code for this project is available on my GitHub repository: `https://github.com/Jiayun-Zhang/Hand_keypoint_detector`

2 Background Literature

In this section, we first introduce the evolution of AI robotic manipulation methods, tracing the development from early grasp prediction models and reinforcement learning approaches to imitation learning. We then review current methods for collecting data for robot imitation learning and analyze their advantages and limitations. Following this, we present traditional imitation learning approaches that rely on training or fine-tuning neural networks. Finally, we discuss a recent study that repurposes large language models for robot trajectory prediction, which is greater to demonstrate efficiency in few-shot settings.

2.1 Evolution of Robotics for AI

As robotics enters the era of artificial intelligence, a large number of robotics algorithms based on deep learning have emerged [1]. Early methods mostly used convolutional neural network (CNN) as the architecture, for example Generative Grasping CNN (GG-CNN) [2] and GR-ConvNet [3]. Such models usually take the RGB-D image as input and predict the top-down grasping angle, width and confidence corresponding to each pixel. The subsequently proposed GraspNet [4] [5] has a deeper network structure and integrates 3D point cloud information, allowing the model to predict the grasping posture of six degrees of freedom (6 DoF) from more different perspectives.

However, in recent years, with the continuous advancement of robotics technology, Dual-Arm humanoid robots have gradually replaced traditional single-arm systems and become the focus of research and industry[6]. People's expectations for robots have also shifted from performing a single grasping task to performing more complex tasks with higher dexterity and planning capabilities, like the humans themselves. Previous models based on grasping point prediction are essentially downstream applications of computer vision in robotic field. They lack the ability to understand temporal actions, task semantics, and environmental changes, and are difficult to handle diverse tasks or cope with complex and changing operational requirements in the real world. Therefore, researchers have begun to explore more advanced methods to enable robots to complete tasks in a way that is closer to humans.

In reinforcement learning-based robot policy learning, the robot interacts with the environment and explores the optimal policy through exploration and trial and error, which aims to maximize the cumulative reward, to learn the operating skills with autonomous decision-making ability[7]. Although the application of reinforcement learning in robot policy learning is theoretically feasible, it still faces many challenges in practical applications. To ensure the stability and convergence of the learning process, training often requires a large amount of interaction data and a carefully designed reward system[8]. If training is performed on a real robot, it requires a lot of hardware resources and experimental time, making it difficult to achieve large-scale data collection[9]. If the robot performs in the simulation environment, the use of a simulation environment requires the construction of a simulation system that highly restores the real physical laws, which itself faces problems such as complex modeling and limited accuracy[10][11]. These factors make the application of reinforcement learning in the actual operation tasks of robots face high thresholds and limitations.

2.2 Imitation Learning

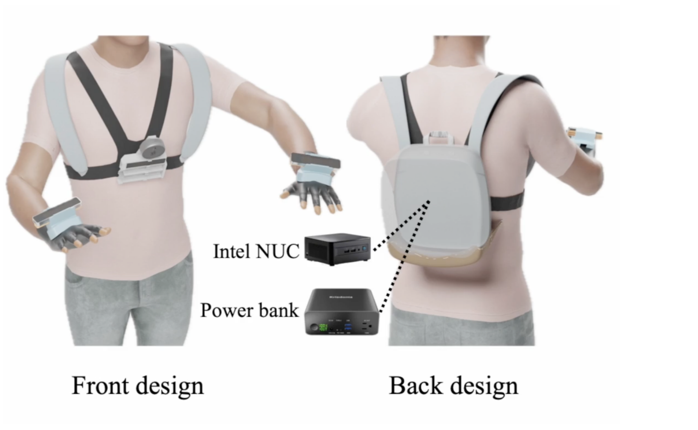
In response to the above limitations, imitation learning provides a more efficient and feasible alternative. Imitation learning refers to robots directly learning behavioral strategies from human demonstrations. In other words, it allows robots to perform tasks similar to human actions[12]. Compared with reinforcement learning, it can significantly reduce the costs of robot interaction and exploration, and there is no need to construct complex world models and reward functions. Therefore, it has gradually become an important research direction for current robot skill learning.

2.2.1 Data Collection

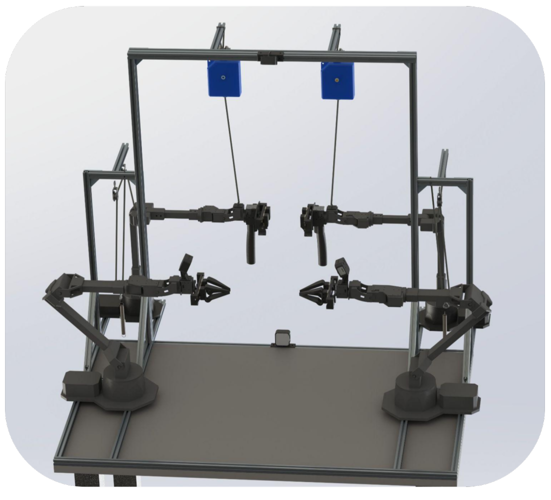
A typical method in imitation learning——Behavioral Cloning enables robots to reproduce expert actions by learning state-action pairs from expert demonstrations. Its training process is essentially similar to supervised learning in deep learning, and data format is like an observation-action pair. Like all deep-supervised learning methods, the performance of imitation learning is highly dependent on the quantity and quality of training data. Therefore, efficiently achieving accurate and generalizable demonstration data has become one of the main challenges that restrict the application of imitation learning to actual robotics tasks.

By collecting such demonstration data, deep learning models can be trained to perform actions similar to those in the demonstration in similar states[15] [16]. There are many types of demonstration collection method for imitation learning. One is to use specially designed wearable devices to accurately capture joint movements. For example, the DexCap system [13] (Figure 1a) uses electromagnetic field (EMF) gloves to track fine finger movements and uses a RGB-D camera based on the SLAM (Simultaneous Localization and Mapping) algorithm placed on the chest to accurately estimate hand posture.

In addition, MoCap is also a popular data collection method. For example, in the Arctic dataset[17], researchers used 54 motion capture cameras [18], which were synchronized with multi-view RGB cameras to achieve precise spatial alignment (see Figure 2). These MoCap cameras accurately es-



(a) DexCap: A Portable Hand Motion Capture System [13].



(b) ALOHA: A low-cost teleoperation system for data collection [14].

Figure 1: Two methods for collecting data for imitation learning manipulations

timate hand movements by capturing tiny markers attached to human experts' hands and objects. While these kind of method provides high accuracy, it is complex and costly due to the complicated algorithms and expensive equipment required.

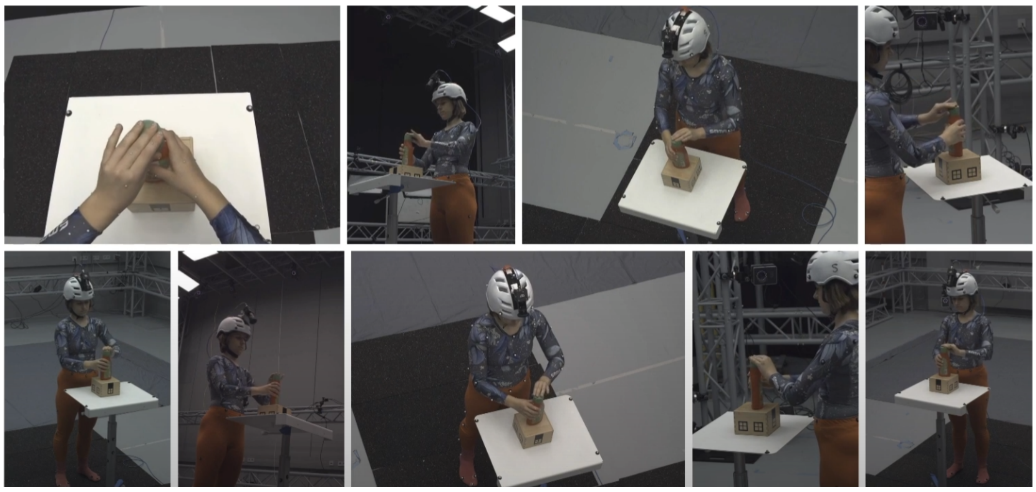


Figure 2: Arctic multi-view, 8 $\times$ Allocentric + 1 $\times$ Egocentric [17].

ALOHA from Stanford University and the subsequent ALOHA-mobile launched a new leader-follower dual-arm teleoperation data acquisition system [14][19]. The experimenter controls the follower arm by operating the leader arm, thereby recording the joint status information of the follower arm. Aloha system greatly reduced the complexity and cost of data collection, enabling robots to quickly learn complex dual-arm operation tasks (such as plugging in wires, folding clothes, opening bottle caps, etc.). However, ALOHA equipment still costs tens of thousands of dollars, which can be a significant burden for some university labs with limited funding.

Another low-cost approach is visual imitation learning, which is based solely on video analysis to perform imitation learning [20]. Compared to using specialised equipment, visual imitation learning does not require expensive professional equipments. Video data is more accessible and can be obtained using existing cameras or smartphones, or even can directly use some online videos. This almost zero hardware cost method of receiving data from video is more suitable for low-budget research.

2.2.2 Imitate from human demonstration video

In order to collect data at low cost, learning robot skills from human video demonstration has become an important direction in the field of imitation learning. Many methods teach robots new skills or generalize the skills to new scenarios by observing humans performing tasks [21][22][23].

The Ditto method [21] segments the operating objects in the video and estimates the 3D motion trajectory of the objects. Then the relative relationship between the hand position and the object is extracted, and the demonstration trajectory is deformed and adapted to the current scene according to the pose of the object in the new scene. This method shows the feasibility of going from demonstration to real robot execution, but the disadvantage is that it cannot perform tasks that do not involve grasping and moving, such as pushing or pressing. SCREWMIMIC [22] converts human bimanual demonstrations into a low-dimensional screw action representation, enabling robots to efficiently learn from a single video and successfully perform complex bimanual tasks through self-supervised exploration. However, it still relies on learning single tasks in isolation with a fixed camera setup.

In contrast, R+X (Retrieval and Execution) [24] is a more flexible approach. By leveraging off-the-shelf 3D hand keypoint detection methods, it maps hand keypoints to gripper poses, eliminating the need for camera calibration or fixed viewpoints—only a video is required to obtain the end-effector trajectory. Furthermore, since it directly extracts the trajectory from hand keypoints, it supports a broader range of tasks, including those that do not involve grasping or moving, such as pushing or pressing. However, R+X still requires 5 to 10 demonstrations per task. Therefore, how to achieve broad generalization with fewer demonstrations remains an open problem in imitation learning.

2.2.3 Embodiment Gap

A key challenge in imitation learning from human demonstrations is the embodied gap, which the mismatch in morphology, kinematics, and sensing between the human body and the robot. This gap is especially significant in dexterous manipulation, where differences in finger joint structures, contact surface geometries, and tactile sensing can make human demonstrations difficult to transfer directly to robotic hardware. This gap manifests itself both in the action space (e.g., kinematics, joint limits) and observation space (e.g., visual appearance of hands, available tactile signals). Bridging this gap requires, as proposed in DexUMI[25]—a combination of wearable exoskeletons and vision-based inpainting to ensure data consistency between human demonstrations and robot execution. However, most current methods that can minimize the embodiment gap rely on specific hardware designs, such as wearable exoskeletons or tactile sensors. In the absence of hardware facilities, how to minimize the embodiment gap as much as possible to perform high-quality imitation learning by relying only on video data remains an open and challenging research direction.

In the context of video-based imitation learning, we should convert the human video demonstrationto robot-executable commands. This process typically involves three key steps(Figure 3)[26]: (1) *pose estimation*, where a sequence of human joint positions x_A is extracted from a demonstration video of a human performer A; (2) *motion retargeting*, where these human joint positions x_A are mapped into a new sequence of joint positions x_B that are feasible for performer B—in our case, a robot with different morphology and actuation; and (3) *robot control*, where the transformed joint sequence x_B is converted into low-level motor commands that control the robot's actuators.

The main difficulty in reducing the embodied gap lies in Step 2: motion retargeting. In this step, we need to convert human joint actions into robot actions, but this is not straightforward because humans and robots have very different body structures, movement ranges, and ways of interacting with objects. Unlike transferring motion between two humans, mapping a human hand or arm trajectory to a robot, especially one with rigid arms and simple grippers, often leads to information loss and inaccuracy. This step is where the embodied gap is most visible and directly affects how well the robot can imitate the human. Designing a method to make this conversion more accurate and flexible, especially without needing special calibration between the human and robot, is still a core challenge in video-based imitation learning.

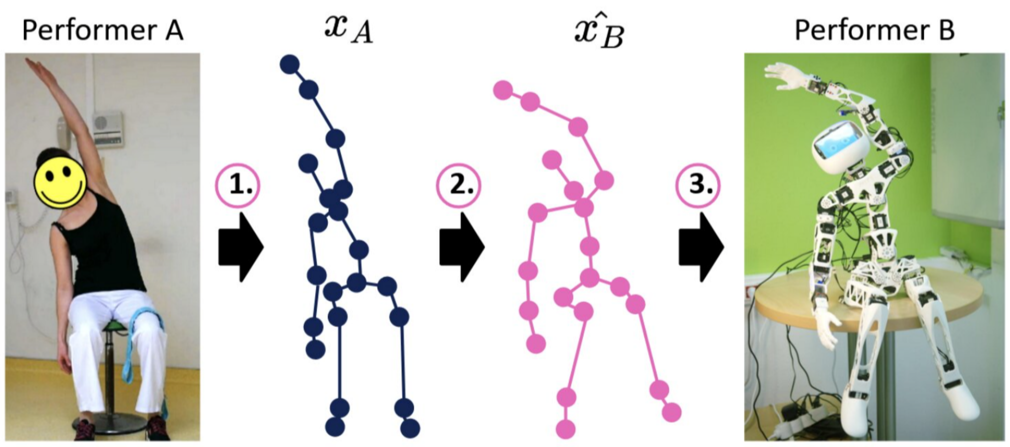


Figure 3: Steps of the human-to-robot imitation process: (1) pose estimation extracts human pose x_A from a video of expert A; (2) motion retargeting maps x_A to robot joint positions x_B; (3) robot control executes x_B via low-level commands.[26]

2.3 Large Language Models for Robot Trajectory Prediction

2.3.1 Traditional Imitation Learning: Training or Fine-tuning a Model

Traditional imitation learning methods typically rely on collecting large datasets of demonstrated trajectories from expert policies. This data is then used to train or fine-tune a neural network model to map perceptual observations (e.g., RGB environment images and current robot joints or end-effector state vectors) to action sequences. The most basic approaches include behavior cloning (BC), which directly trains a model to predict the expert's actions based on observations. In addition, some few-shot imitation learning methods rely on models that are first pretrained on large-scale interaction data, and then fine-tuned using only a small amount of task-specific data. Recently, some approaches that leverage visual-language models(VLMs) [27] for robot control, such as OpenVLA[28], RDT[29], RT-2[30]. Although they generalize well to new environments and new tasks, training them requires millions of records of robot trajectories.

2.3.2 Large Language Models

In recent years, large language models (LLMs) based on the Transformer architecture have become a breakthrough in the field of natural language processing. Represented by GPT, this type of model is pre-trained on a large-scale text corpus and demonstrates excellent language understanding and generation capabilities, especially in dealing with long context dependencies, which greatly surpasses traditional models [31]. Several studies have explored the possibility of using standard LLMs in robotics tasks, and many researchers have used them as high-level planners. Some works use LLMs to generate a series of high-level instructions based on natural language task descriptions[32][33], and some others map natural language into reward functions for trajectory optimizers[34]. Although these methods perform well, they still rely on additional modules to ”land” the output of the LLM into actual actions. Therefore, some studies force on showing LLMs are not only good at language tasks, but also have the ability to act as general pattern learners [35][36]. It can perform in-context learning with a small number of examples, that is, generalize new mapping relationships based on existing input-output examples without fine-turning any parameters. This feature makes it shows great application potential in scenarios where data is scarce but tasks are varied, such as robotics.

Especially when used in combination with reasonable representation and tokenization strategies, it can significantly improve learning efficiency and generalization ability.

2.3.3 Keypoint Action Tokens

A novel approach, Keypoint Action Tokens (KAT) [37], proposes a paradigm that repurposes existing LLMs for imitation learning (Figure 4). The approach transforms robot observations into a set of keypoint and encodes action trajectories as structured token sequences that can be processed by LLMs. These sequences are not natural language, but instead representations of the actions of the robot (such as 3D keypoints and SE(3) poses). Representing objects using keypoints provides a compact encoding of visual information, significantly reducing input dimensionality while preserving essential geometric features. For example, an RGB image may contain millions of pixels, whereas a keypoint-based representation of the same object might consist of ten 3D (xyz) keypoints, which only equivalent to the information from ten RGB pixels. This not only improves data efficiency, but also enhances generalization to novel object instances and visual scenes. Unlike traditional methods, KAT does not rely on retraining or fine-tuning the LLM and can achieve accurate and executable action prediction with only a small number of demonstrations. Experiments show that this imitation learning method can match or even surpass the performance of current state-of-the-art methods, such as Diffusion Policies, in very low-data scenarios. By adapting existing LLMs for trajectory learning tasks, KAT shows the potential to significantly reduce training costs and data requirements while maintaining high performance, heralding a new possibility for the robotic learning paradigm.

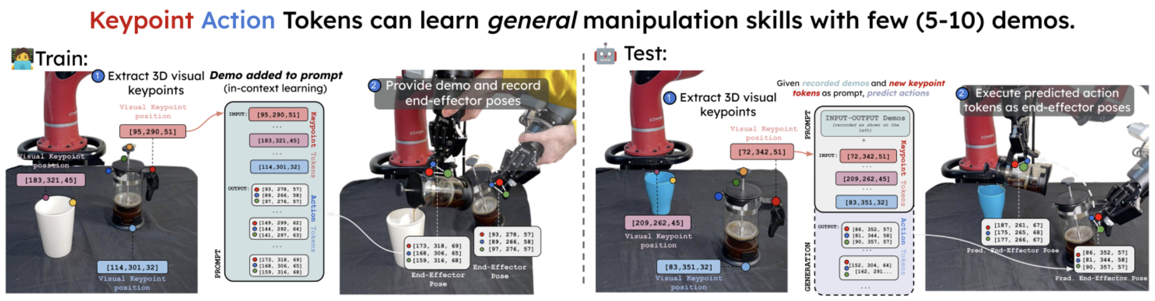

Keypoint Action Tokens can learn *general* manipulation skills with few (5-10) demos.
Train:
1 Extract 3D visual keypoints
Demo added to prompt (in-context learning)
Visual Keypoint position
[95,290,51]
[183,321,45]
[114,301,32]
INPUT:
OUTPUT:
Keypoint Tokens
Action Tokens
PROMPT
2 Provide demo and record end-effector poses
End-Effector Pose
Test:
1 Extract 3D visual keypoints
Given recorded demos and new keypoint tokens as prompt, predict actions
INPUT-OUTPUT Demos
[72,342,51]
[209,262,45]
[83,351,32]
GENERATION
2 Execute predicted action tokens as end-effector poses
Pred. End-Effector Pose

Figure 4: Keypoint Action Tokens (KAT) is an imitation learning method that reuses text-based pre-trained Transformers by representing observation and action sequences as keypoint-based tokens and converting them into textual form to include prompts [37].

3 Methods

In this section, we propose a complete framework for extracting dual-arm robot trajectories from a single human demonstration video. We introduced a simplified HaMeR model for hand keypoint extraction, depth-based correction for improved spatial accuracy, a heuristic mapping from hand keypoints to gripper poses, and an object-centric augmentation method including spatial translation and mirror-flipping. Together, these methods form the core of our low-cost and generalizable imitation learning data generalization method.

3.1 Extract robot trajectory from human demonstration

In the issue of robot learning skills by imitating human demonstration videos, the core challenge is how to accurately convert human actions in the video into actions that the robot can perform and minimize the embodied gap as much as possible. In this conversion stage, the analysis of hand movements is particularly important because hand movements often carry the most critical and complex information in dexterous manipulation tasks. We use a method based on 21 key points to represent hand motion, deconstructing complex hand postures into a set of anatomically meaningful key points, also known as joints [38] As shown in Figure 5, these keypoints correspond to various joint positions of the human hand, including 5 fingertips, 14 interphalangeal joints, and 1 wrist joint. Through the spatial combination of these points in three-dimensional space, various hand actions from simple grasping to dexterous manipulation can be accurately represents.

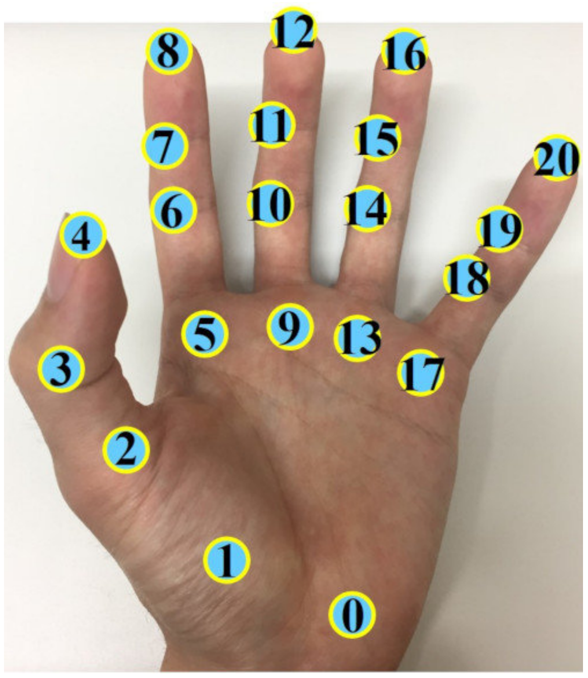


Figure 5: 21 keypoints of hand pose. [38]

In our hand-to-gripper framework (see Figure 6), we use a method called HaMeR (Hand Mesh Recovery) [39], which is a deep learning model based on the Transformer architecture to achieve high-precision 3D hand pose reconstruction. By analyzing RGB image sequences, this model reconstructs the 3D coordinates of 21 keypoints for both hands from human demonstrations. Next, we incorporate the depth information to correct these keypoint coordinates in the world coordinate system. Finally, by analyzing the spatial distribution pattern of the 21 hand keypoints, combined with the common points of human and two-finger gripper operations, we extract the main feature points that represent the grasping intention and establish a heuristic mapping model from human hand keypoints to robot gripper poses.

3.1.1 Extract hand keypoints from human demonstration

First, we adopt and simplify the HaMeR method for extracting hand keypoints from human demonstrations. HaMeR is a deep learning model that aims to reconstruct 3D hand meshes from RGB images or video frames with high accuracy [39][40]. The original HaMeR takes an RGB image as input and outputs camera intrinsics π and MANO pose θ, as well as mesh shape parameters β for each detected hand. Since this framework focuses only on hand keypoints, we omit the mesh-reconstruction stage after keypoint estimation.

Furthermore, the original HaMeR architecture incorporates ViT-Pose [41] to handle the association between hand and person, which is particularly beneficial in multi-person scenes. This module ensures that the system can correctly associate each detected hand with the corresponding person in the frame. However, in our demonstration videos, we always assume that only one single operator is present at all times. As a result, the association module is discarded to reduce computational overhead and retain only the keypoint regression network of the main hand, which predicts θ, π.

The HaMeR core network, denoted as $\mathcal{T}$, is fully transformer-based. It uses a Vision Transformer (ViT) backbone to process the input RGB image $I \in R^{H\times W\times 3}$. Unlike traditional convolutional architectures, this ViT-based design offers strong representational capacity without relying on complex convolutional operations. The input image is first divided into fixed-size patches, which are linearly embedded and augmented with positional encodings to produce a sequence of input tokens. These tokens are processed through multiple Transformer encoder layers, where multi-head self-attention is

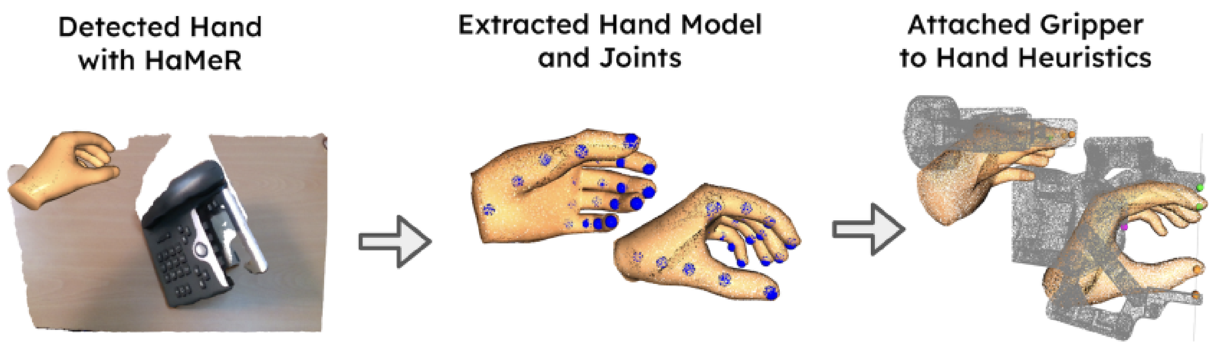


Figure 6: Converting human demonstrations into robot gripper trajectories. First, hand keypoints are extracted from video frames using the HaMeR model; then, a heuristic mapping is applied to convert the keypoints into gripper poses. [24]

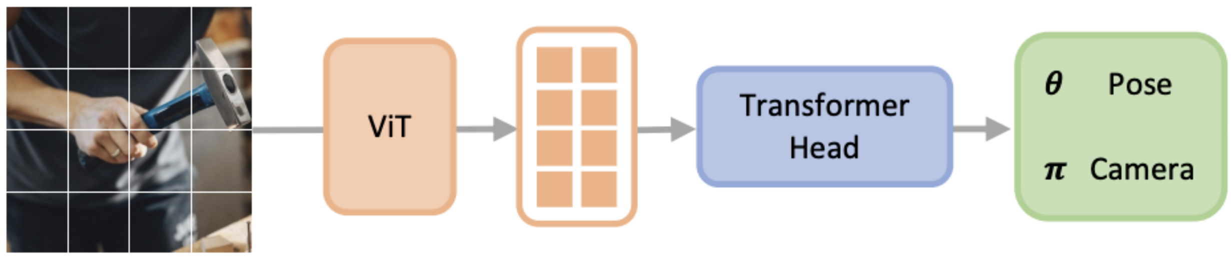


Figure 7: The HaMeR architecture adopts a fully transformer-based design, using a large-scale ViT backbone followed by a transformer decoder to regress hand parameters.[39]

used to capture both local and global visual features.

After the encoder, a Transformer decoder head receives a single query token and performs cross-attention with the encoder outputs to regress the final hand keypoints $\theta \in R^{21\times 3}$ and camera intrinsics π:

$$(\theta, \pi) = \mathcal{T}(I)$$

In the post-processing phase, a confidence-based selection is applied to identify the most reliable hand keypoint prediction. For each candidate prediction k, a confidence score $c_k \in [0,1]$ is also produced, reflecting the reliability of the corresponding keypoints θ_k. The final left and right hand keypoints are determined by selecting the candidates with the highest confidence scores from the sets of all detected left and right hands keypoints candiates, denoted by $\mathcal{H}_{\text{left}}$ and $\mathcal{H}_{\text{right}}$, respectively:

$$\theta^*_{\text{left}} = \arg\max_{\theta_k \in \mathcal{H}_{\text{left}}} c_k$$

$$\theta^*_{\text{right}} = \arg\max_{\theta_k \in \mathcal{H}_{\text{right}}} c_k$$

This process ensures that the system consistently selects the most confident keypoint predictions for both hands, allowing accurate downstream motion analysis and control.

3.1.2 Use depth to correct predicted hand pose

To reduce the prediction error of HaMeR, we further correct the hand key points output by HaMeR with the help of depth images. First, assume that the hand gesture, which is the relative position of the key points of the HaMeR output is accurate, but the problem is mainly the depth prediction.
Our approach is to first project the predicted 3D hand keypoints $\mathbf{X} = [X, Y, Z]^T$ onto a 2D image plane using the intrinsic matrix of the camera $\mathbf{K}$, and (u, v) is the 2D projection of the HaMeR prediction on the image plane:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \cdot \frac{1}{Z} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{Z} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Then, we manually select some keypoints that are always visible in the demonstration, such as points 1 or 2 in Figure 5, rather than points like 18 or 19 on the pinky that are easily hidden by objects when we are manipulating something in the demonstration. The specific points depend on the demonstration itself. For these selected points, we get the depth value $Z^{'}(u, v)$ from the depth map at their projected locations and convert it back to 3D coordinates $\mathbf{X}'$ as:

$$\mathbf{X}' = Z^{'}(u, v) \cdot \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

After that, we calculate the translation vector between the 3D coordinate of this selected always-visible point $\mathbf{X}'$ and the corresponding keypoint predicted $\mathbf{X}_{\text{HaMeR}}$ by HaMeR.

$$\Delta \mathbf{t} = \mathbf{X}' - \mathbf{X}_{\text{HaMeR}}$$

Then, we apply this translation vector to all 21 hand keypoints predicted by HaMeR to make it aligned with the ground truth depth:

$$\mathbf{x}^{(j)}_{\text{corrected}} = \mathbf{x}^{(j)}_{\text{HaMeR}} + \Delta\mathbf{t}, \quad \text{for } j = 1, \dots, 21$$

Moreover, the input of HaMeR is a single image, which means that each frame is inferred independently. This may lead to a lack of temporal continuity, resulting in large jumps in the prediction results between frames. Therefore, we introduced a simple outlier detection and smoothing strategy to ensure the stability of the prediction. Specifically, we calculate the MSE (Mean Squared Error) between the keypoints of the left and right hands of each frame and the previous frame to evaluate the degree of change of keypoints between the two frames. If the MSE of a frame exceeds the set threshold, we consider the prediction of that frame to be an outlier. At this time, the prediction of the current frame will be ignored, and the keypoint coordinates of the previous frame will be used directly. The hand keypoints will not be updated until a frame with an MSE lower than the threshold appears.

3.1.3 A heuristic to convert hand keypoints into Gripper pose

The last step of converting human demonstrations into robot actions is analyzing the spatial layout of the human hand's keypoints, combined with the common points of human and two-finger gripper operations, to extract representative grasp-related features, and then constructing a heuristic mapping model from these hand keypoints to the robot gripper's pose. Specifically, we compute the gripper's pose by aligning its tips with the tips of the human thumb and index finger; see Figure 8.
Frist, the coordinates of gripper pose xyz can be determined by the midpoint of the index finger tip $\mathbf{p}_{\text{index\_tip}} \in R^3$ and the thumb finger tip $\mathbf{p}_{\text{thumb\_tip}} \in R^3$:

$$\mathbf{p}_{\text{gripper}} = \frac{\mathbf{p}_{\text{index\_tip}} + \mathbf{p}_{\text{thumb\_tip}}}{2}$$

Then, the direction or rotation of the gripper can be determined by the line between the point **a**(midpoint of index finger tip and thumb finger tip) and the point **b** (midpoint between keypoint with point 0 and 1):

$$\mathbf{R}_{gripper} = \text{Rodrigues}(\hat{\mathbf{a}}, \hat{\mathbf{b}}) = \mathbf{I} + [\mathbf{v}]_\times + [\mathbf{v}]^2_\times \cdot \frac{1 - \hat{\mathbf{a}} \cdot \hat{\mathbf{b}}}{\|\mathbf{v}\|^2}$$

$$\text{with } \mathbf{v} = \hat{\mathbf{a}} \times \hat{\mathbf{b}}, \quad [\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

Combining the rotation and the coordinates xyz, the homogeneous transformation matrix of the gripper pose T_{gripper} can be determined:

$$T_{\text{gripper}} = \begin{bmatrix} \mathbf{R}_{\text{gripper}} & \mathbf{p}_{\text{gripper}} \\ 0 & 1 \end{bmatrix}$$

In addition, to find out whether the gripper should execute a closing or opening motion, we apply a heuristic method based on the Euclidean distance between the thumb and index finger tips. This distance is compared with an open-close threshold w_{gripper}, resulting in the action decision:

$$d_{\text{fingertips}} = \left\| \mathbf{p}_{\text{index\_tip}} - \mathbf{p}_{\text{thumb\_tip}} \right\| .$$

$$\text{Action}_{\text{gripper}} = \begin{cases} \text{Close}, & \text{if } d_{\text{fingertips}} < w_{\text{gripper}} \\ \text{Open}, & \text{otherwise.} \end{cases}$$

3.2 Trajectory Transformation and Augmentation

Current deep learning-based robotic manipulation policies face a fundamental bottleneck during training: they rely heavily on large amounts of human-collected expert demonstrations. For complex tasks such as dexterous hand manipulation or dual-arm coordination, it typically takes dozens or even hundreds of demonstrations to achieve basic competence [14]. Training more powerful and generalizable models, such as vision-language-action (VLA) policies, may require millions of demonstrations [28][29]. Collecting such data often demands significant time and effort from skilled operators using teleoperation or similar methods.

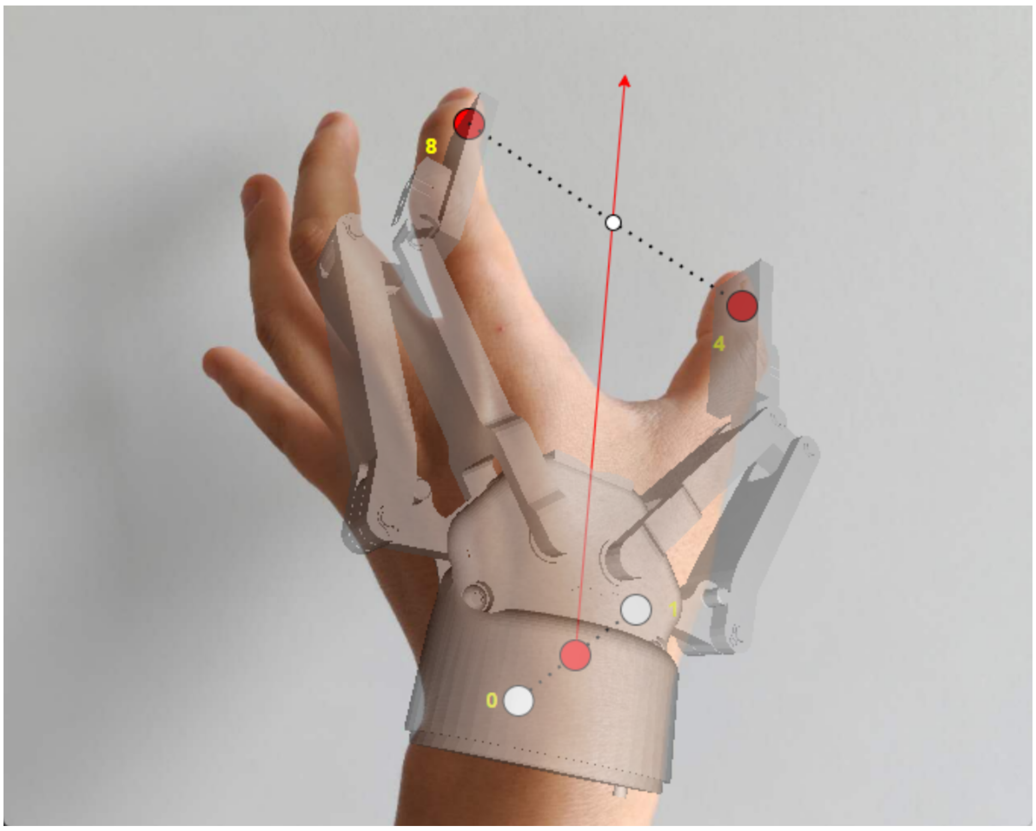


Figure 8: Heuristic mapping from human hand keypoints to the robot gripper pose. The thumb and index fingertip positions (keypoints 4 and 8) are used to compute the gripper center, while the direction from their midpoint to the wrist base (keypoints 0 and 1) defines the gripper's orientation. This provides a approximation of a grasp pose compatible with a two-finger gripper.

More critically, these training datasets usually cover only a limited range of initial object configurations (e.g., specific positions and orientations). As a result, when the robot encounters out-of-distribution scenarios—such as objects placed in unseen workspace regions—policy performance degrades significantly. This lack of spatial generalization necessitates additional human effort to densely collect demonstrations that span the entire configuration space, further increasing the data collection burden.

To address this issue, we propose an improved version of DemoGen [42] by developing a keypoint-based dual-arm demonstration augmentation framework. Our method enables the generation of diverse and spatially adaptive training data from a single human demonstration video, allowing the policy to generalize to arbitrary object poses and positions without requiring extensive additional demonstrations(see Figure 9).

To enable demonstration augmentation, we apply an object-centric transformation based on the 3D keypoints of the manipulated object. We begin by transforming the original end-effector (EE) trajectory, defined in the world coordinate frame, into the object coordinate frame. Let $\mathbf{T}^{\text{obj}}$ be the original pose of the object in the world frame, and $\mathbf{T}_t^{\text{EE}}$ the end-effector pose at time step t. The trajectory in the object frame is given by:

$$\mathbf{T}_t^{\text{EE,obj}} = (\mathbf{T}^{\text{obj}})^{-1} \cdot \mathbf{T}_t^{\text{EE}}.$$

Next, we simulate spatial variations of the object by applying the same translation to all of its keypoints. Let $\Delta\mathbf{x} \in R^3$ be the sampled translation offset. Each keypoint $\mathbf{k}_i \in R^3$ is updated as:

$$\mathbf{k}_i' = \mathbf{k}_i + \Delta\mathbf{x}, \quad \forall i \in \{1, \dots, N\}.$$

Based on this translation, we construct a new object pose:

$$\mathbf{T}^{\text{obj}'} = \mathbf{T}_{\text{move}} \cdot \mathbf{T}^{\text{obj}},$$

where $\mathbf{T}_{\text{move}}$ is a homogeneous transformation matrix derived from $\Delta\mathbf{x}$. Using this new pose, we reproject the relative end-effector trajectory back into the world frame:

$$\mathbf{T}_t^{\text{EE,new}} = \mathbf{T}^{\text{obj}'} \cdot \mathbf{T}_t^{\text{EE,obj}}.$$

To support structured augmentation, we decompose the original trajectory into alternating *motion segments* and *skill segments*. When the end-effector moves in free space without object interaction, the segment is labelled as a motion segment. When the end-effector interacts with the object—such as during grasping or manipulation—it is labelled as a skill segment.

Example: Grasp Flower to Vase (Single Arm)
As a concrete example, consider the left arm performing a flower arrangement task. The trajectory sequence is decomposed into four segments:

- **Motion Segment 1:** The end-effector moves from its starting position to approach a flower placed on the table.
- **Skill Segment 1:** The gripper closes to grasp and lift the flower.
- **Motion Segment 2:** The end-effector, now holding the flower, moves above the target location — the opening of the vase.

- **Skill Segment 2:** The end-effector inserts the flower into the vase and releases it.

After completing the trajectory transformation, we apply cubic spline interpolation to generate a smooth transition for each end-effector from its current pose to the starting point of the first motion segment. This target pose, denoted as $\mathbf{T}^{\text{EE,new}}_{\text{grasp}}$, is obtained by transforming the original grasp pose $\mathbf{T}^{\text{EE}}_{\text{grasp}}$ through the object-relative frame:

$$\mathbf{T}^{\text{EE,new}}_{\text{grasp}} = \mathbf{T}^{\text{obj}'} \cdot (\mathbf{T}^{\text{obj}})^{-1} \cdot \mathbf{T}^{\text{EE}}_{\text{grasp}}.$$

Execution begins only after both arms complete this interpolated motion. The arms then proceed to execute the First Motion Segment, which moves the end-effectors toward the new object position in the transformed coordinate frame.
Once both First Motion Segments are completed, the First Skill Segment is executed simultaneously. For each timestep t, the original skill trajectory $\mathbf{T}^{\text{EE,skill}}_t$ is reprojected using:

$$\mathbf{T}^{\text{EE,skill,new}}_t = \mathbf{T}^{\text{obj}'} \cdot (\mathbf{T}^{\text{obj}})^{-1} \cdot \mathbf{T}^{\text{EE,skill}}_t.$$

This ensures the skill is executed in the new scene while maintaining the interaction logic.
After the First Skill Segment ends, each arm generates a return trajectory from its current pose $\mathbf{T}^{\text{EE}}_{\text{end}}$ to the starting pose of the Second Motion Segment in the original world frame, $\mathbf{T}^{\text{EE}}_{\text{motion2-start}}$, using cubic spline interpolation.
The robot continues by executing the original Second Motion Segment $\{\mathbf{T}^{\text{EE,motion2}}_t\}$, followed by the original Second Skill Segment $\{\mathbf{T}^{\text{EE,skill2}}_t\}$, which remain unchanged as they operate in the world frame. Once both arms finish this final phase, the full augmented demonstration is complete.

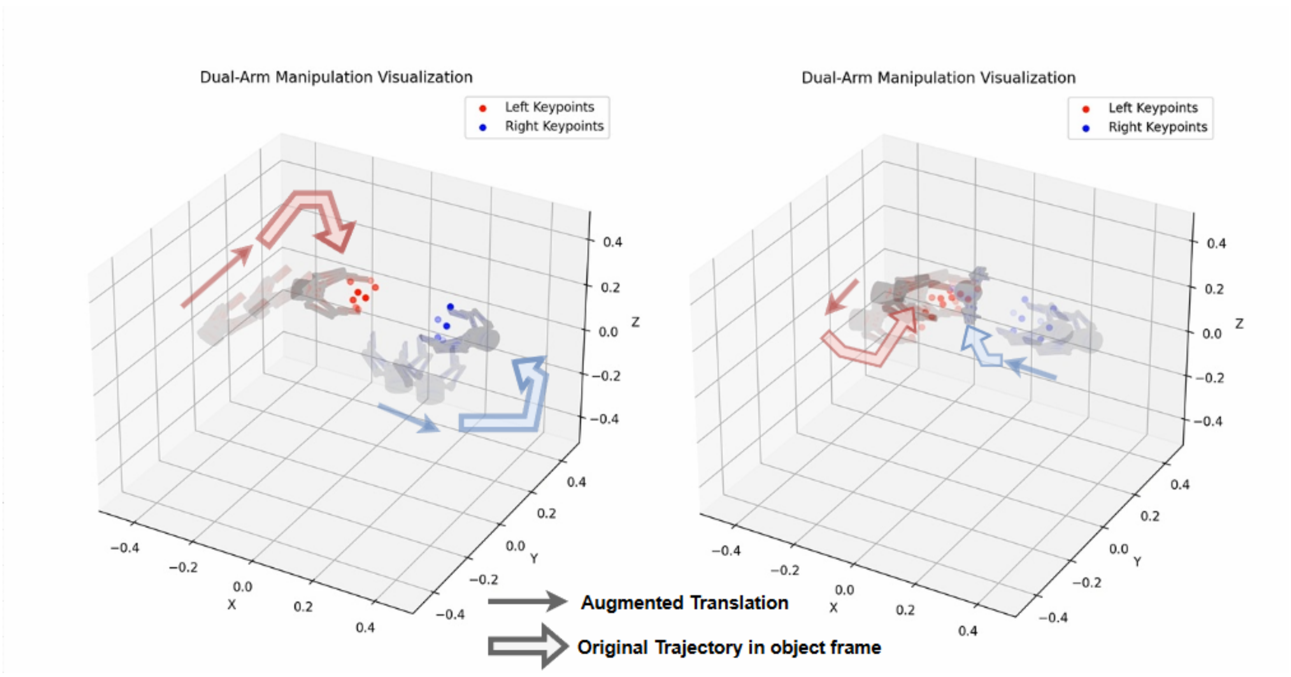


Figure 9: Left figure begins by transforming the original end-effector trajectory, defined in the world frame, into the start point of the object coordinate frame. Then replay the first skill segment original trajectory in the object frame. Right figure first reproject the relative end-effector trajectory back into the world frame, then replays the first skill segment original trajectory in the world frame.

3.3 Mirror-Flipping for Spatial Generalization

In dual-arm manipulation tasks, the robot needs to grasp an object with each hand to complete the task. Usually, it will selects the object closest to each arm to grasp. For example, in a Pouring task, when the bottle is on the left and the cup is on the right, the robot will grasp the bottle with its left hand and the cup with its right hand to perform the pouring motion.

However, if the system relies on a single demonstration trajectory, problems arise when the positions of the objects are reversed. In such cases, the robot may still attempt to reproduce the original motion plan, resulting in incorrect hand-object assignments—for example, trying to grasp the bottle on the left with the right hand, or the cup on the right with the left hand. This mismatch can easily violate the robot's kinematic constraints or the affordance of the objects, leading to failure in task execution or even potential collisions which may cause damage.

To address this issue, we introduce a mirror-flipping strategy that flip both the object keypoints and the robot's trajectory across the YZ plane. This operation simulates a mirrored scene configuration and enables the system to generate action sequences that are semantically equivalent to the original demonstration, even when the spatial arrangement of the objects is altered (e.g., left-right flipped). This significantly improves the model's generalization to changes in spatial configuration.
Specifically, given a 3D point $\mathbf{p} \in R^3$ and a unit normal vector of the reflection plane $\mathbf{n} \in R^3$, the reflected point is computed as:

$$\mathbf{p}_{\text{reflected}} = (\mathbf{I} - 2\mathbf{n}\mathbf{n}^\top)\,\mathbf{p} \tag{1}$$

A rotation matrix $\mathbf{R} \in R^{3\times3}$ is similarly reflected column-wise:

$$\mathbf{R}_{\text{reflected}} = (\mathbf{I} - 2\mathbf{n}\mathbf{n}^\top)\,\mathbf{R} \tag{2}$$

To ensure the resulting matrix remains a valid right-handed rotation (i.e., $\det(\mathbf{R}) = +1$), one column is negated based on the reflection axis. The full pose matrix is then reconstructed as:

$$\mathbf{T}_{\text{reflected}} = \begin{bmatrix} \mathbf{R}_{\text{corrected}} & \mathbf{t}_{\text{reflected}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3}$$

This mirror-flipping augmentation expands the dataset with spatially diverse configurations and improves the system's robustness and adaptability to varied object layouts.

3.4 LLM-based Trajectory Prediction

To generate dual-arm robot trajectories for new scenes, we employ a large language model (LLM) guided by structured prompts. For each query scene, we first extract the 3D object keypoints and initial gripper poses. Then, we retrieve the top-k most similar demonstrations from a pre-generated dataset using a mean squared error (MSE) metric computed over object keypoints.
Given a pair of keypoint sets $\mathbf{K}_{\text{query}}, \mathbf{K}_{\text{db}} \in R^{N\times3}$, where N is the number of keypoints, MSE is computed as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} \left|\left|\mathbf{K}_{\text{query},i} - \mathbf{K}_{\text{db},i}\right|\right|^2$$

The top-3 retrieved demonstrations are then formatted into structured prompts, each containing the object keypoints, initial gripper poses, and the corresponding action sequences. The current query input is appended in the same format. These prompts are passed to an off-the-shelf LLM, which generates the predicted action sequence for both arms. The full pipeline is illustrated in Figure 10, and an example of a prompt format is provided in Appendix A.

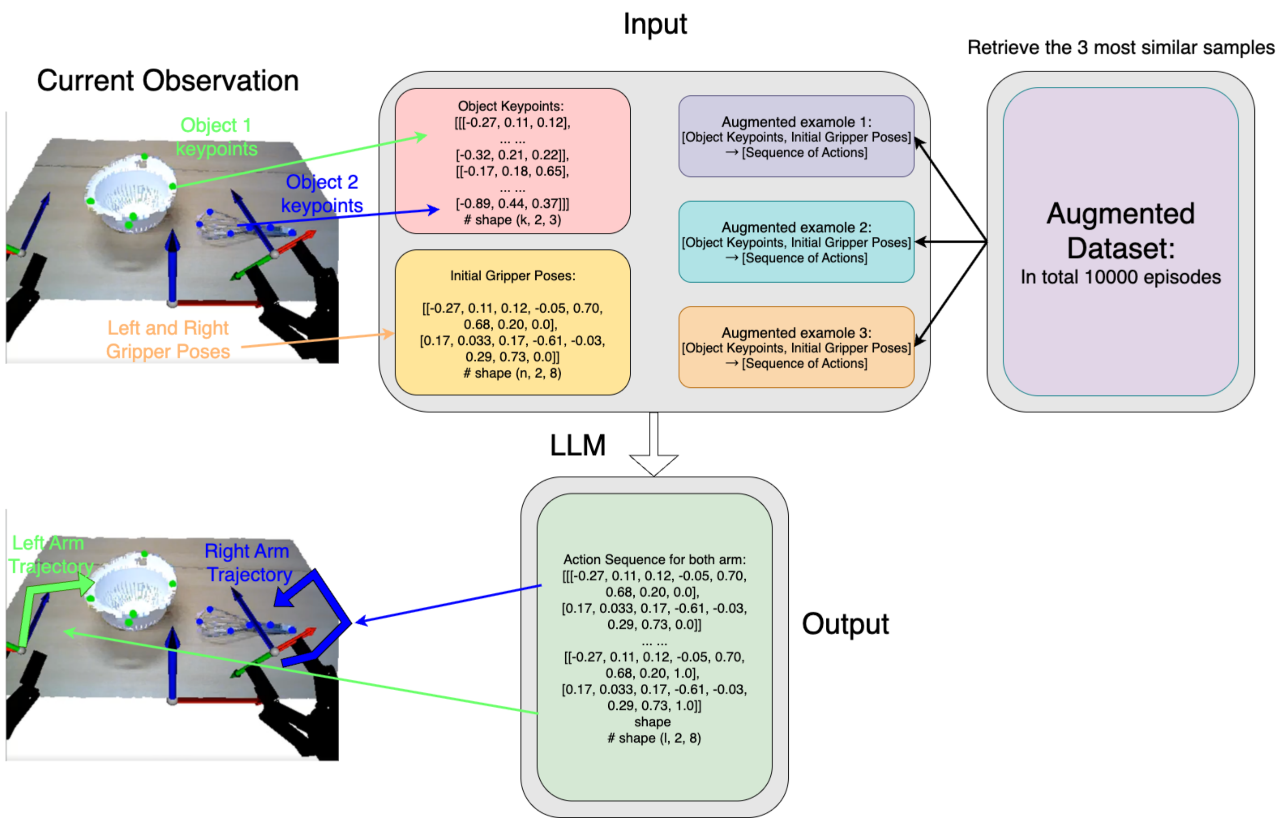


Figure 10: Overview of the LLM-based trajectory prediction pipeline. Given a current observation consisting of object keypoints and initial gripper poses, the system retrieves the three most similar demonstrations from a dataset of 10,000 augmented episodes based on object keypoint similarity. These retrieved examples are formatted as input prompts to a large language model (LLM), which predicts the output action sequence for both arms. The predicted trajectory is then mapped back onto the scene to guide robot execution.

4 Experimental Setup

This section presented our experimental setup, including how we collected real demonstration data, evaluated keypoint prediction accuracy using the ARCTIC dataset, and tested our system in diverse real-world settings. We also described the use of a dummy simulator for visualizing trajectories, and how LLMs were employed for trajectory prediction using retrieval-based prompts. These experiments validate the effectiveness of each component in our framework.

4.1 Data Collection

We collect human demonstration data using the following setup. A tripod-mounted RGB-D camera is placed at a fixed position above and angled downward toward the workspace, mimicking the viewpoint of a dual-arm robot's head camera in our lab setting (Figure 11). Due to the short distance between the camera and the table, it is not feasible for a single operator to stand in front of the setup without blocking the view. To address this, we adopt a two-person collaborative demonstration strategy—each participant controls one hand and stands on opposite sides of the table, ensuring both hands remain visible and centered in the camera's field of view throughout the task (see Figure 12).

We record a total of six different dual-hand manipulation tasks. For each task, we also record a mirrored version by swapping the roles of the left and right hands. The tasks are as follows (see Figure 13):

- **Pouring:** One hand picks up a cup and pours its contents into another cup held by the other hand.
- **Trash-in-bin:** One hand picks up a can and drops it into a bin held by the other hand.
- **Paper-in-case:** One hand picks up a waste paper ball and drops it into a case held by the other hand.
- **Mix-bowl:** One hand holds a whisk and inserts it into a bowl held by the other hand, performing a mixing motion.
- **Open-container:** One hand lifts the lid off a container held steady by the other hand.
- **Empty-vase:** One hand removes flowers from a vase held by the other hand.

4.2 Hand keypoints Evaluation

In order to evaluate the accuracy of the robot end-effector trajectory mapped from human demonstration videos, we first need to ensure the accuracy of the predicted hand keypoints. However, our self-recorded demonstration videos do not contain the ground-truth labels for the hand keypoints. Therefore, in order to verify the accuracy of the hand keypoints of the HaMeR method, we use the ARCTIC dataset as an evaluation benchmark.

The ARCTIC dataset [17] (Figure 14) is collected by a high-precision MoCap system, by using 54 high-resolution cameras to capture the markers attached to the hands and objects. These markers are small and hidden, and do not affect the interaction between the hands and objects, and they also hardly appear in the image. Each task in the dataset contains an RGB video captured from a first-person perspective, which closely resembles the viewpoint of our own recorded demonstrations. Additionally, it provides the ground truth 3D coordinates of 21 hand keypoints relative to the camera frame, which precisely the information required for our evaluation. Therefore, this dataset is very suitable for our task and provides an ideal evaluation benchmark for us to recover accurate hand posture estimation from videos.

In the experiment, we selected 18 task takes from the ARCTIC dataset as evaluation samples. These takes cover the operation of six types of objects, including three large objects: boxes, capsule coffee machines, and espresso machines, and three small objects: ketchup bottles, mobile phones, and scissors. These objects are of varying sizes and can be used to thoroughly evaluate the 3D hand keypoint detection method on manipulation tasks in a variety of different scenarios.

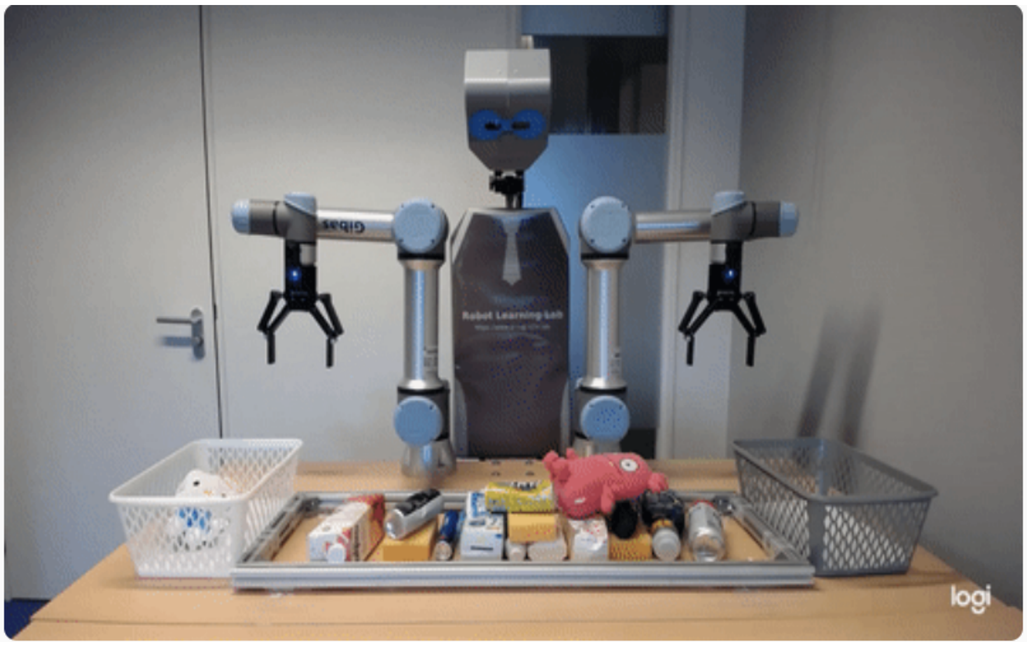


Figure 11: The dual-arm robot used in the robotics lab at the University of Groningen consists of two 6-DOF arms equipped with parallel-finger grippers. There is a RGB-D camera in the middle, placed above the workspace and angled downward to capture the interaction area.

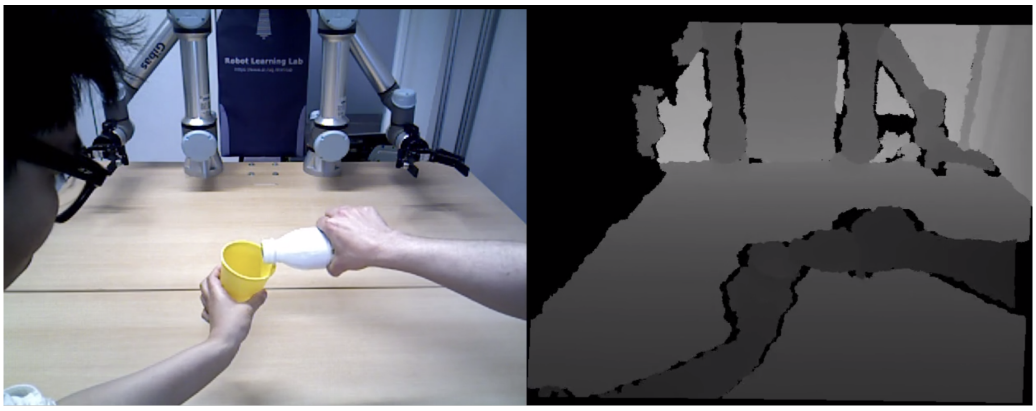
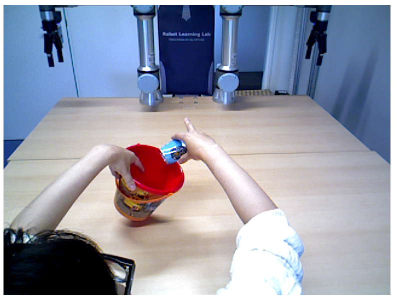
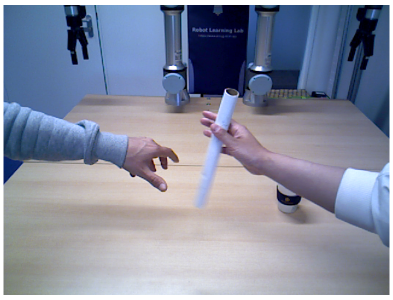
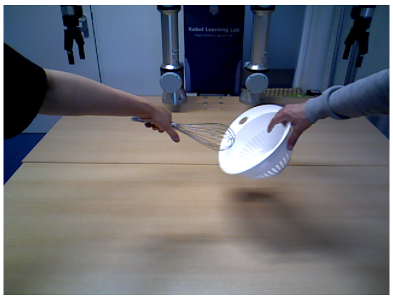


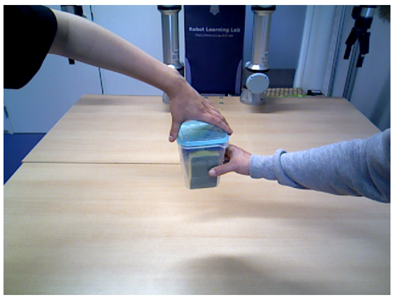
Figure 12: An example of a human demonstration of a pouring task recorded by us, showing both RGB and depth images

(a) Pouring


(b) Trash-in-bin


(c) Paper-in-case


(d) Mix-bowl


(e) Open-container

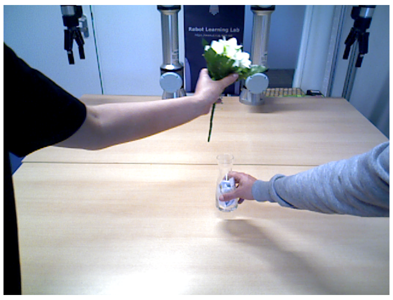
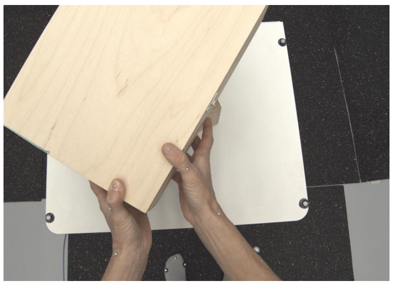
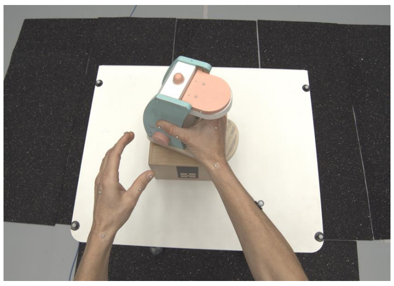
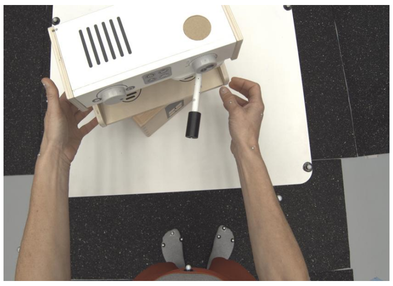
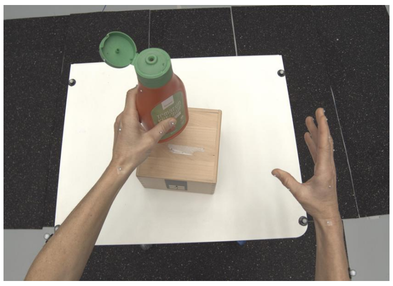

(f) Empty-vase

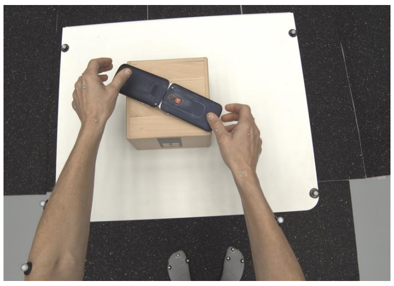
Figure 13: Six distinct dual-hand manipulation tasks in our dataset. For each task, we also record a mirrored version by swapping the roles of the left and right hands.


(a) Box


(b) Capsule Coffee Machine


(c) Espresso Coffee Machine


(d) Ketchup


(e) Mobile Phone

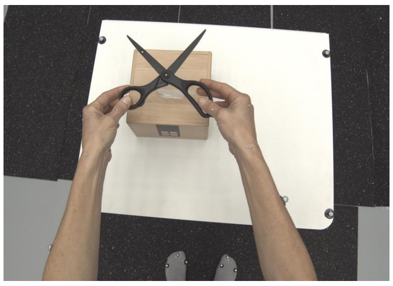

(f) Scissors

Figure 14: Six object categories used in the manipulation tasks.

We pre-processed the video frames by first removing frames where the hands were not visible at the beginning and end of the task, and then selected the image sequences where the hands were always clearly visible from the remaining frames, so that this process met the criteria when we recorded the demonstration image: that is, the hands always appeared in the frame. Then, we input the RGB image from the first-person perspective into the HaMeR model, predicted the coordinates of the key points of the hand relative to the camera coordinate system, and compared them with the ground truth provided in the dataset to evaluate the model's posture reconstruction ability on real task data.

4.2.1 3D Evaluation Metrics

To evaluate the accuracy of the predicted 3D hand keypoints, we adopt two standard metrics widely used in 3D pose estimation: Mean Per Joint Position Error (MPJPE) and Procrustes Aligned Mean Per Joint Position Error (PA-MPJPE).

MPJPE MPJPE measures the average Euclidean distance between the predicted keypoints and the ground truth keypoints, and is defined as:

$$\text{MPJPE} = \frac{1}{N \times J} \sum_{i=1}^{N} \sum_{j=1}^{J} \left\| \hat{\mathbf{p}}_{i,j} - \mathbf{p}_{i,j} \right\|_2 , \tag{4}$$

where $\hat{\mathbf{p}}_{i,j} \in R^3$ is the predicted 3D position of joint j in sample i, $\mathbf{p}_{i,j}$ is the corresponding ground truth, N is the number of samples, and J is the number of keypoints per sample.

PA-MPJPE PA-MPJPE applies a rigid alignment (translation, rotation, and scaling) to the predicted keypoints before computing the MPJPE. This alignment is achieved via Procrustes analysis:

$$\text{PA-MPJPE} = \frac{1}{N \times J} \sum_{i=1}^{N} \sum_{j=1}^{J} \left\| \mathcal{P}(\hat{\mathbf{p}}_i)_j - \mathbf{p}_{i,j} \right\|_2 , \tag{5}$$

where $\mathcal{P}(\hat{\mathbf{p}}_i)$ denotes the aligned version of the predicted keypoints for sample i.

MPJPE reflects the absolute accuracy of the predicted keypoints in 3D space. It captures all sources of error, including relative pose, global position, and scale. PA-MPJPE evaluates the structural similarity between the predicted pose and the ground truth after rigid alignment. It is invariant to global translation and scale, and thus primarily measures how accurately the hand gesture or relative keypoint configuration is captured.

By using both MPJPE and PA-MPJPE, we can more comprehensively evaluate the performance of the hand pose estimation system. If PA-MPJPE is low but MPJPE is high, it means that the gesture structure predicted by the model is correct, but there is a large error in global position or scale. The model may perform well in structure learning but lack depth or position estimation. If PA-MPJPE is high, even after rigid alignment, the predicted result is still far from the true pose, indicating that the model has weak prediction ability on gesture structure. If both indicators are low, it indicates that the model performs well in both gesture recognition and 3D positioning.

4.2.2 2D Evaluation Metrics

To further evaluate the quality of the predicted 3D hand keypoints, we project them on the image plane by using the camera intrinsics and evaluate their accuracy in 2D space. Projecting the key points into 2D can provide better visualization, and 2D hands keypoints evaluaction metric can also serve as complementary evidence on the performance of 3D evaluation. We adopt two commonly used metrics for 2D keypoint evaluation: **2D Mean Per Joint Position Error (2D MPJPE)** and **Percentage of Correct Keypoints (PCK)**.

$$\text{2D MPJPE} = \frac{1}{N \times J} \sum_{i=1}^{N} \sum_{j=1}^{J} \left\| \hat{\mathbf{u}}_{i,j} - \mathbf{u}_{i,j} \right\|_2, \tag{6}$$

where $\hat{\mathbf{u}}_{i,j} \in R^2$ is the predicted 2D location of joint j in image i, and $\mathbf{u}_{i,j}$ is the corresponding ground truth 2D location. This metric directly measures the average pixel-level error between the projected keypoints and their ground truth locations on the image plane.
In addition, we are also going to report PCK@0.05 and PCK@0.15, which compute the percentage of keypoints whose prediction error falls within a normalized distance threshold. The normalization is based on the size of the hand bounding box:

$$\text{PCK@}\tau = \frac{1}{N \times J} \sum_{i=1}^{N} \sum_{j=1}^{J} 1 \left[\left\| \hat{\mathbf{u}}_{i,j} - \mathbf{u}_{i,j} \right\|_2 < \tau \cdot s_i \right], \tag{7}$$

where τ is the threshold (e.g., 0.05 or 0.15), and s_i is the size of the hand bounding box in image i. PCK captures the proportion of accurately predicted keypoints under a defined tolerance, complementing 2D MPJPE with a thresholded accuracy measure.

4.3 Evaluation in Dummy Simulator

A dummy simulator is used to replay the original and the augmented end-effectot trajectory in a virtual environment to verify its performance in the task (Figure 15). The system first reads the observation information of the specific task from our dataset, including the first RGB-D frame, camera internsic and externsic, initial hand pose, and world coordinate transformation matrix, and uses the depth map and camera parameters to restore the pixel coordinates to a 3D point cloud to construct the 3D scene under the current task. Then, the 2D key point positions of the object are obtained from the annotation file, and its position in 3D space is obtained by back-projecting the depth information and converted to a unified coordinate system as a reference. If the data augmentation function is enabled, the system will identify the key action segments in the original demonstration and apply perturbations to them to generate new enhanced trajectories, while synchronously transforming the point cloud representation of the object to maintain scene consistency. Then, the dummy simulator will initialize the environment with the current point cloud, key points, and initial gripper pose, and input the predicted hand pose and opening and closing state into the simulator when executing each step. During the trajectory playback process, the system will render the current state in real time and record the change trajectory of the key points of the target object in each frame for subsequent analysis and visualization.

All experiments presented in the following sections were conducted using this dummy simulator. It serves as the primary evaluation platform, allowing us to validate the predicted trajectories under diverse conditions in a consistent and controlled environment.

4.3.1 Evaluation of Augmentation + Retrieval Prompting Strategies

To evaluate the effectiveness of our Augmentation + Retrieval prompting strategy for LLM-based trajectory generation, we design a set of comparative experiments under three different configurations. For each manipulation task, we first generate a dataset of 10,000 augmented scenarios using the data augmentation techniques described earlier: 5,000 scenes are created by randomly placing objects at different positions within the workspace, and another 5,000 are produced using a mirror-flipping strategy to enhance spatial diversity. All augmented samples are stored in a task-specific demonstration database.

In all experiments, the LLM is prompted with a fixed system instruction and a structured input format. However, the content of the demonstration examples within each prompt varies according to the experimental setting. Task success is defined as the successful execution of the manipulation task without error, and the success rate is used as the primary evaluation metric. The three prompting strategies are defined as follows:

- **No Augmentation (Baseline):** This setting uses only a single original demonstration for prompting. The demonstration includes the object keypoints, initial gripper poses, and the full action sequence. No additional samples or augmentations are used. This serves as the baseline to reflect the model's performance without external guidance.
- **Random Augmentation:** In this setting, we randomly sample 3 episodes from the same task-specific database used in our retrieval method. These episodes are generated via data augmentation but are not selected based on similarity to the query scene. The goal is to test whether augmenting the prompt with additional data, regardless of relevance, improves performance over the baseline.
- **Retrieval-based Augmentation:** This is our proposed method. We first compute MSE between the object keypoints in the query scene and each candidate episode in the database. We then select the top-3 demonstrations with the lowest MSE values as the most relevant samples.

For each setting, we run 10 independent tests per task, each with a different randomly generated scenario and example trajectory (depending on the method). We then execute the generated action sequences in the dummy simulator environment and record the overall success rate, defined by the completion of the intended manipulation task without failure. This allows us to compare the impact of each prompting strategy on the generalization and reliability of the task.

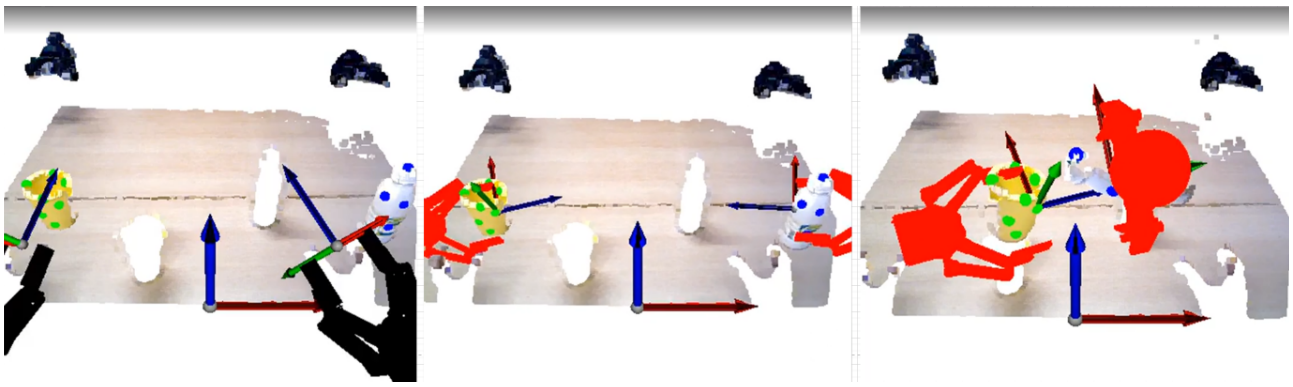
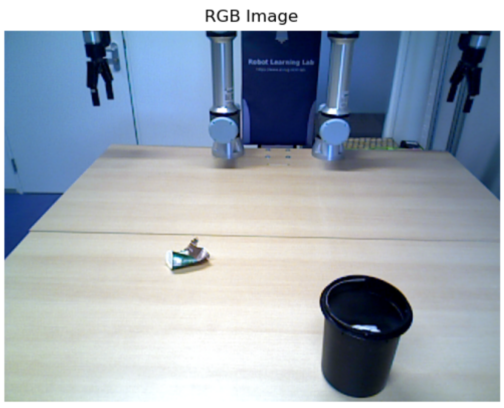
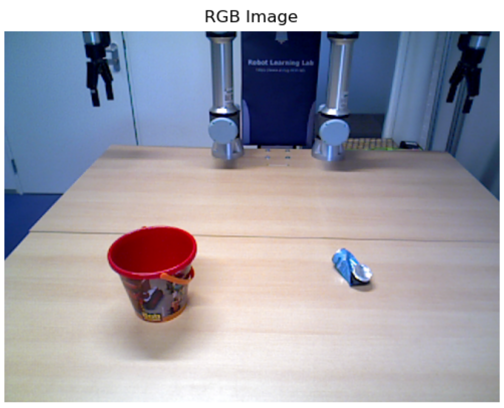


Figure 15: An example of the pouring task display in our dummy simulator.

(a) Background clutter



(b) Different Objects



(c) Different Spatial Configurations

Figure 16: Examples of testing scenarios for evaluating real-world generalization. Top: (a) visual variations with different clutter, and (b) object variations using different instances. Bottom: (c) spatial variations with the same object placed in diverse poses and positions.

4.3.2 Testing in Real and Diverse Scenarios

The previous experiments focused on scenarios generated through our data augmentation method, it is mainly used to verify the reliability of our Augmentation+Retrieval method and its spatial generalization ability. However, these types of generated scenario are all based on the same scenario augmentation, which can not fully cover the diversity of real-world environments. To further assess the generalizability of our proposed framework, we conducted additional tests under a variety of challenging and diverse real-world conditions that extend beyond the default simulation setup. These scenarios were carefully designed to mimic realistic deployment contexts and evaluate the robustness of the system under varying visual and physical conditions (Figure 16):

- **Background clutter:** We modify the appearance of the scenarios by introducing various irrelevant objects in the clutter, thereby increasing the level of visual clutter. This setting changes the visual characteristics of the environment.

- **Different Objects:** We replaced the original manipulated objects with new object instances of similar categories (e.g., different cups, containers) to verify whether the model can generalize across intra-class variations.
- **Different Spatial Configurations:** We placed the same object in different positions and orientations throughout the workspace to challenge the spatial generalization of the system. This is different from the previous test of spatial generalization in that the scenes here are all real recordings instead of being generated using random key point positions. This setting ensures that the method can adapt to arbitrary object poses and still generate correct end-effector trajectories.

In our experiments, we selected three representative dual arm manipulation tasks, Poring, Mix Bowl, and Trash-in-Bin, to evaluate the robustness of our framework. For each task, we conducted trials under the three real-world variation settings described above. By testing on multiple tasks and under different environmental conditions, we aim to provide a more comprehensive evaluation of the generalizability and robustness of our framework in the real world.

4.4 Third-party utilities used for experiments

The utilities described in this section were developed by my supervisor, Georgios Tziafas, and were used under his permission. These components played an essential role in supporting the experimental setup of this thesis. The following descriptions aim to document how these tools were utilized in the context of the research.

4.4.1 Dummy Simulator

The dummy simulator used for visualizing and evaluating robot trajectories was implemented by the supervisor. It enabled the replay of both original and augmented end-effector trajectories in a virtual environment.

4.4.2 Data augmentation

The data augmentation framework described in Section 3.2 was developed and implemented as part of this thesis, but the version used in the final experimental setup was provided by the supervisor. His implementation differs in structure and integration details, although the core augmentation concepts remain aligned. The experiments presented in this section were conducted using his version to ensure seamless interoperability with the dummy simulation environment.

5 Results

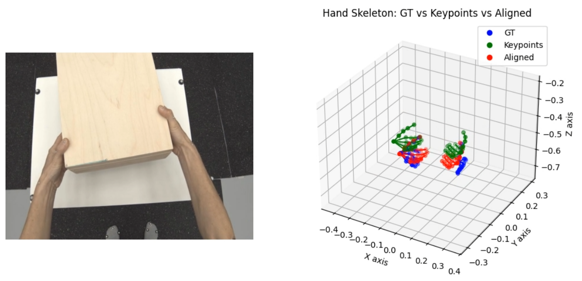
This section summarized the quantitative and qualitative results of our framework. We demonstrated strong keypoint estimation accuracy, reliable gripper pose prediction, and high task success rates in both augmented and real-world scenarios. Our results confirmed the feasibility of using single demonstrations combined with augmentation to generate generalizable robot behaviors, even under varied environmental conditions.

5.1 Results for Hands Keypoint Extraction

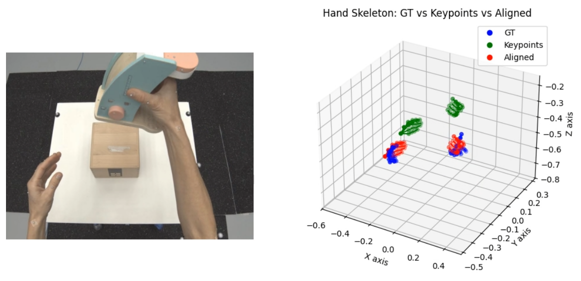
5.1.1 Arctic dataset

We applied the previously implemented HaMeR hand keypoint detection method to three takes of each of the six selected tasks from the ARCTIC dataset. For the quantitative metric, PA-MPJPE is significantly lower than MPJPE for both hands (see Table 1, the full version can be found in Appendix B). This result suggests that HaMeR has high accuracy in predicting the structure of the hand's gesture but has a certain bias in reproducing the absolute position of the hand in 3D space. This is mainly because inferring depth information from monocular images is a big challenge. Although 3D gestures can be inferred from visual clues such as hand models, shadows, and occlusion patterns, there is a lack of strong visual evidence for the absolute depth of the hand or specific joints. In addition, the focal length of the first-person camera used in the ARCTIC dataset is very large (approximately 2431.5). However, the input image is downsampled before being fed into HaMeR. This mismatch between the high focal length and the reduced image resolution further exacerbates depth prediction errors. Under such a large focal length, even small pixel deviations in the image space can result in significant errors in 3D space, particularly along the depth axis.
This also demonstrates the necessity of introducing the depth alignment method based on a single stable keypoint, as described in the Methods section. We select a keypoint that is always visible in all frames (e.g. point 2) and use the deviation between its true 3D position in the depth image and the position predicted by HaMeR as the global translation bias for a single hand. This offset is then applied to the entire set of predicted keypoints, thereby aligning the entire set of hand postures to true depth.

This can also be observed through a visualization. In Figure 17a and Figure 17b, the left image



(a) Box grab



(b) Capsule machine grab

Figure 17: Comparison of predicted hand keypoints before and after depth-based alignment. Left: input scene from ARCTIC dataset; Right: Ground truth (blue), HaMeR prediction (green), and depth-aligned prediction (red).

Table 1: Quantitative evaluation of hand keypoint prediction accuracy. We report Mean Per Joint Position Error (MPJPE) and Procrustes-Aligned MPJPE (PA-MPJPE) in meters. Lower is better.

Method	**MPJPE** $\downarrow$	**PA-MPJPE** $\downarrow$
Left Hand	1.301	0.023
Left Hand (Aligned)	0.024	0.023
Right Hand	1.305	0.019
Right Hand (Aligned)	0.023	0.019

shows a frame of two example tasks in the tasks we tested in the ARCTIC dataset. The right side shows the visualization results of the connected key points in the form of a skeleton to represent the posture structure of the hand in the frame image. Skeletons of different colors represent key points from different sources:

- The blue skeleton represents the ground truth.
- The green skeleton represents the keypoints directly predicted by the HaMeR method.
- The red skeleton represents the result of offset correction based on the original prediction of HaMeR using the keypoint position in the depth map.

It can be clearly seen from the visualization that the red skeleton after depth correction is closer to the true blue label in absolute position, indicating that our alignment strategy plays a significant role in improving the prediction accuracy. In addition, it can be seen from Table 1 that, for quantitative metrics, the MPJPE of the aligned keypoints is already very close to the PA-MPJPE. Since PA-MPJPE measures the error after removing global scale, rotation, and translation via Procrustes alignment, while MPJPE reflects the raw positional error in the global coordinate system, the small gap between the two suggests that our alignment method has effectively corrected most of the global depth translation error.

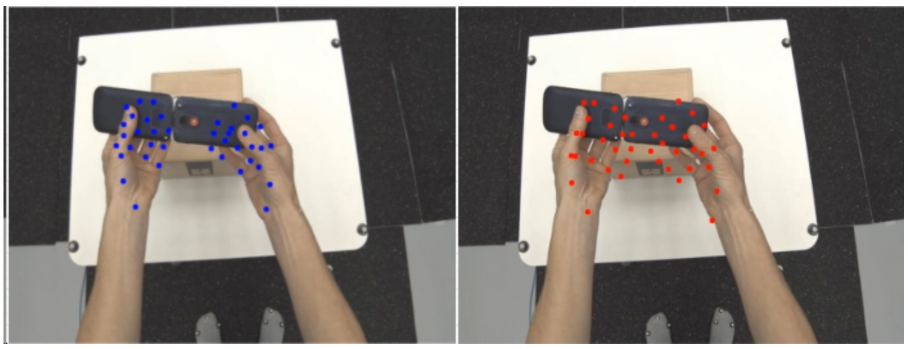
We also evaluated the accuracy of the 2D hand keypoint predictions by projecting the estimated 3D keypoints onto the image plane using the camera intrinsics. As shown in Table 2, we report the mean per joint position error (MPJPE) in the image pixels, as well as the Percentage of Correct Keypoints (PCK) at two distance thresholds: 0.05 and 0.15 (normalized). The MPJPE reflects the average Euclidean distance between predicted and ground-truth 2D keypoints, where lower values indicate better accuracy. The PCK metrics measure the proportion of keypoints falling within a specified distance of the ground truth, where higher values are preferred. The results show that the model achieves an average 2D MPJPE of under 30 pixels for both hands, with over 72% and 74% of keypoints correctly predicted under the 0.05 threshold for the left and right hands, respectively. PCK@0.15 is nearly perfect for both hands. Some visualizations of the estimated 3D keypoints projected onto the image plane can be found in figure 18.
To further evaluate the robustness of our hand keypoint prediction system across diverse object manipulation tasks, we conduct a detailed analysis on six tasks: box, capsule machine, espresso machine, ketchup bottle, phone, and scissors. For each task, we report two 3D metrics—PA-MPJPE and alignment MPJPE—averaged over both the left and right hands (see Figure 19). Additionally, three 2D evaluation metrics are provided in Figure 20, namely PCK@0.15, PCK@0.05, and 2D MPJPE. The prediction errors vary depending on the object. Generally, tasks involving larger objects (e.g., box,

Table 2: Quantitative evaluation of 2D hand keypoint prediction accuracy. We report Mean Per Joint Position Error (MPJPE) in pixels and Percentage of Correct Keypoints (PCK) at thresholds 0.05 and 0.15. For MPJPE, lower is better; for PCK, higher is better.

Hand	**MPJPE (px)** $\downarrow$	**PCK@0.05** $\uparrow$	**PCK@0.15** $\uparrow$
Left Hand	25.46	0.726	0.997
Right Hand	26.21	0.743	0.996

espresso machine) yield slightly higher PA-MPJPE and aligned MPJPE values compared to those involving smaller objects (e.g., scissor, phone).

To further investigate this pattern, we grouped the objects into two categories, large and small, and plotted the average error within the group for each metric. As shown in Figure 23, large objects consistently show a higher MPJPE, while PA-MPJPE is slightly worse. This can be attributed to the more frequent and severe occlusions between the hand and the object when manipulating large objects, causing HaMeR to have to predict the invisible hand parts from the visible hand parts, resulting in decreased accuracy. When manipulating small objects, on the other hand, the small object rarely occludes the hand, so for a fully visible hand, HaMeR's prediction accuracy for its keypoints is higher. This phenomenon can also be observed intuitively from the visualization. Taking the 2D visualiza-


(a) Phone grab task

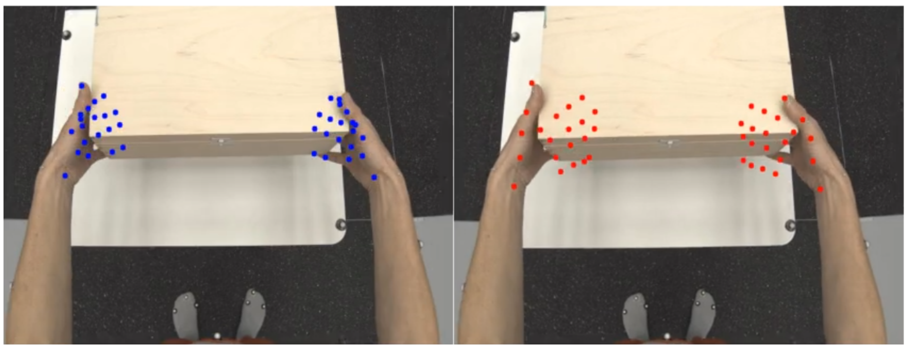

(b) Box grab task

Figure 18: Hand keypoint visualizations for two manipulation tasks involving objects of different sizes. (a) shows the manipulation of a phone(relatively small object), while (b) shows the manipulation of a box(relatively large object). In each subfigure, the left image displays the ground truth keypoints (blue), and the right image displays the keypoints predicted by the HaMeR model (red).

tion Figure 18 as an example, in the box manipulation task (Figure 18b), except for the thumb, most fingers are invisible during the operation, which makes it difficult to predict key points. In contrast, in the phone manipulation task(Figure 18a), most areas of the hand are clearly visible due to the small size of the phone, so the model does not need to infer the finger position from the invisible area, thereby achieving higher prediction accuracy.

5.1.2 Self-recorded dataset

For the demonstrations recorded by ourselves, we cannot provide quantitative evaluation metrics due to the lack of ground truth of the hand key points. So in this section we will use visualization and discuss the problems encountered in trajectory extraction.

From the results, HaMeR performs well in most frames. However, there are still some failure cases.

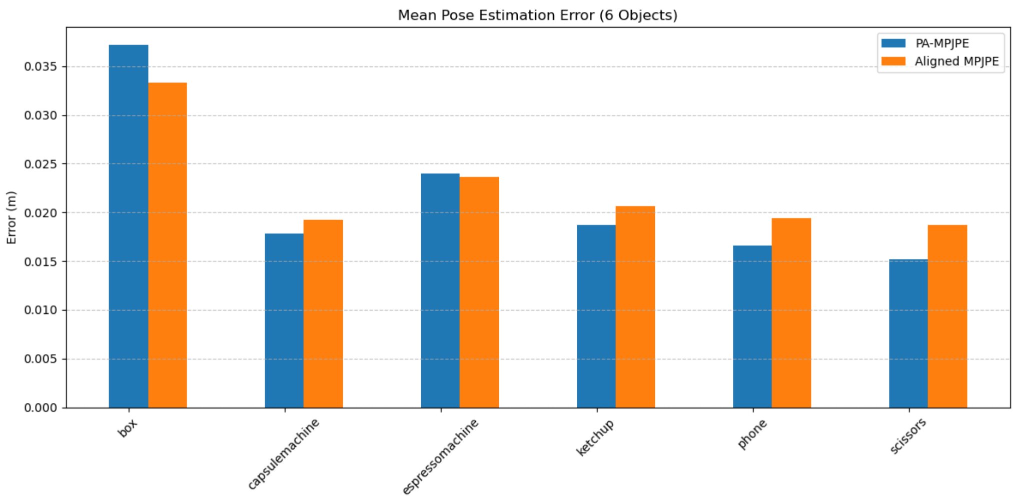
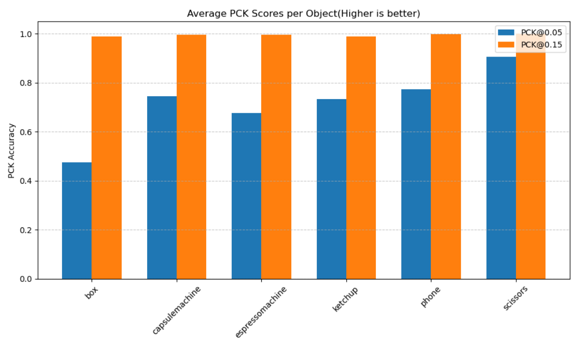
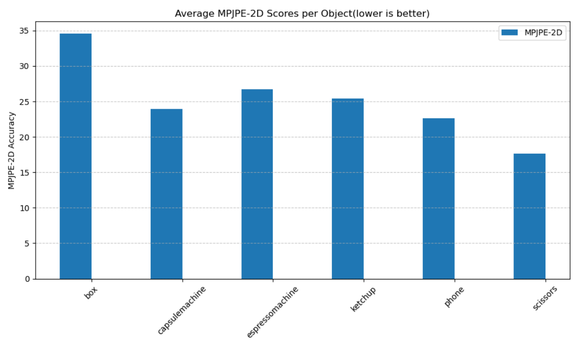


Figure 19: Comparison of pose estimation errors across six manipulated objects task. The metrics include MPJPE after alignment and PA-MPJPE. Metrics are averaged over left and right hands for each object task.



(a) 2D pose estimation error measured by PCK@0.05 and PCK@0.15 (higher is better).



(b) 2D pose estimation error measured by MPJPE after alignment(lower is better)

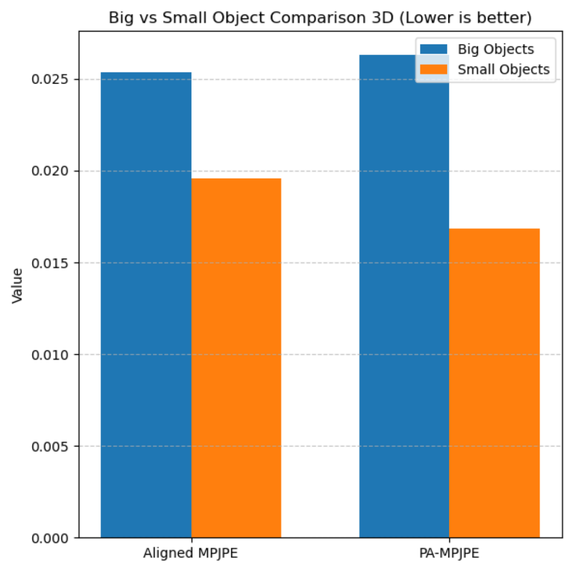
Figure 20: Comparison of 2D pose estimation errors across six manipulated objects task. The metrics include MPJPE after alignment(lower is better), PCK@0.05 and PCK@0.15(Higher is better). Metrics are averaged over left and right hands for each object task.

For example, HaMeR sometimes mistakenly identifies the left hand as the right hand (as shown in Figure 22), causing the position of the left hand key point to suddenly appear at the position of the right hand, resulting in obvious outliers in the generated gripper trajectory. This error usually only occurs in a small number of consecutive frames and is not frequent.

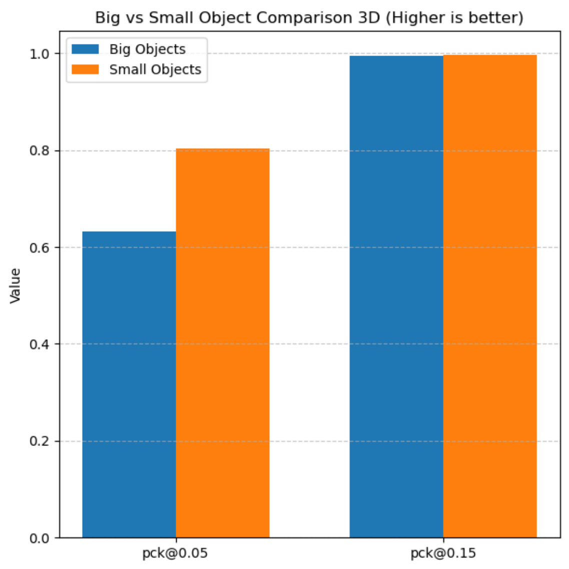
To address this issue, we compute the Euclidean distance between the gripper positions of consecutive frames. Let $\mathbf{p}_t \in R^3$ denote the 3D position of the gripper at frame t, then the inter-frame displacement is calculated as:

$$d_t = \|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2$$

Since the real operation trajectory should be temporally smooth, the value of d_t is expected to remain within a reasonable range. We define a threshold δ. If $d_t > \delta$, we classify frame t as an outlier and freeze the gripper pose (i.e., set $\mathbf{p}_t = \mathbf{p}_{t-1}$) until the distance to a subsequent frame falls back below



(a) Big vs Small Object Comparison 3D (Lower is better)



(b) Big vs Small Object Comparison 2D (Higher is better)

Figure 21: Comparison of 3D and 2D pose estimation errors across six manipulated objects task.

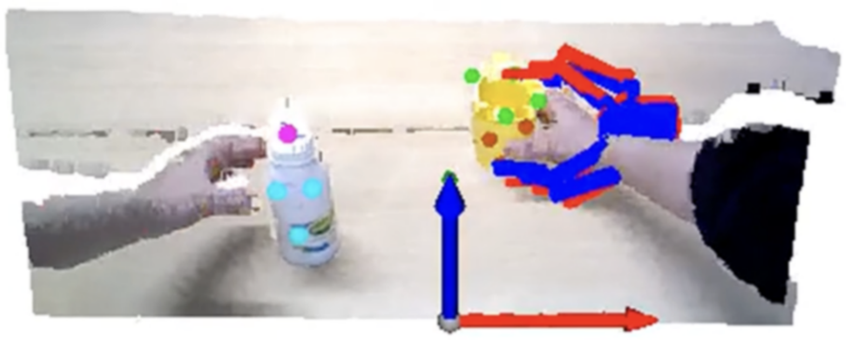


Figure 22: HaMeR occasionally misidentifies the left and right hands, leading to brief but noticeable trajectory outliers

the threshold, i.e., $d_{t+k} \leq \delta$, indicating that the trajectory has returned to normal. This approach effectively addresses most of the outliers in the demonstrations. However, in rare cases where a demonstration contained too many outliers to be reliably corrected using the above strategy, we chose to re-record the demonstration for that specific take.

5.2 Results and Improvements for Hand Keypoints to Gripper Heuristic

As shown in Section 3.1.3, we introduced a heuristic to provide an estimation of the gripper's position and orientation from hand keypoints. In addition, we also proposed a method that uses a threshold to determine the open/close state of the gripper. However, by observing the results, we found that this is not reliable because different tasks and objects require different grasp widths, and the threshold between an open and closed state can be very different across tasks and objects. For example, when grasping objects such as bottles or cups, the required gripper width tends to be relatively large due to their wider bodies. In contrast, objects like a mixer in the Mix-bowl task typically require grasping by the narrow handle, which results in a smaller grasp width. Similarly, tasks involving bowls or bins often require the gripper to pinch the edge or wall of the object, leading to an even narrower grasp width. These variations make it difficult to establish a single, fixed threshold for distinguishing between the open and closed gripper states across different tasks and object geometries. Therefore, in the current implementation, we temporarily manually label the gripper state. First, we observe each demonstration and identify and label specific frames where the human hand is open and closed. This information is then used to determine the gripper state for each frame during trajectory generation.

In addition, we found that another problem often arises when the grasp width becomes very small, for example, when grasping thin structures such as the edge of a bin or bowl. In our heuristic method, the gripper pose is determined based on two fingertip points and a direction vector. When the distance between the two keypoints is very small, even small errors in the hand keypoint estimation (e.g., errors in the hand keypoint prediction by HaMeR) can lead to large fluctuations in the computed gripper pose, especially in the orientation.

In practice, when recording the demonstrations, we will avoid grasping the object solely using the thumb and index finger. Instead, we use the ring and little fingers to clamp the thin object while keeping the thumb and index finger open and framing the object from either side. The grasp point is still positioned between the thumb and index fingertips to simulate the configuration of a two-finger gripper. This strategy allows us to demonstrate grasps on thin structures while maintaining a sufficient distance between the thumb and index finger, which reduces pose instability and prevents abrupt changes in orientation caused by small estimation errors. Furthermore, our manual labeling strategy for identifying open and closed gripper states in the demonstration ensures that the grasping intent is correctly recorded—even when the thumb-index distance remains constant throughout the action.

5.3 LLM Trajectory Prediction

5.3.1 Augmentation + Retrieval in Generated Scenarios

To evaluate the generalization ability of our trajectory augmentation system, we performed an experiment which was introduced in Section 4.3 by using a set of new scenarios with object keypoints generated by using the same data augmentation method described in Section 3.3. In this setting, we selected multiple new takes of the "pouring" task and placed the object keypoints in new random locations to simulate unseen scenarios. The result can be seen in Table 3:

Table 3: Success rate comparison under different augmentation and retrieval strategies on the pouring task. Each configuration was evaluated over 5 different takes, each with 10 trials.

Take	**One Demo (No Aug)**	**3 Random Aug (No Retrieval)**	**3 Retrieved Aug**
Take1	0/10	3/10	**10/10**
Take3	1/10	2/10	**9/10**
Take4	1/10	3/10	**10/10**
Take6	0/10	2/10	**9/10**
Take7	1/10	2/10	**10/10**
Average Success Rate	6%	24%	**96%**

From the results, we can see that the success rates of different prompt inputs vary significantly. When using only a single original demonstration and without data augmentation, the average success rate is only 6%, indicating very limited generalization to unseen scenarios. Introducing three randomly selected augmented demonstrations significantly improves performance, reaching an average success rate of 24%. This shows that our augmentation method increases the diversity of examples in the prompt, which can enhance generalization ability. The best performance is achieved by using our proposed retrieval-based augmentation strategy, which selects the three most closely matched demonstrations based on object keypoint MSE. This method achieves success rates as high as 96% and demonstrates consistent success in almost all trials. These results clearly highlight the effectiveness of combining data augmentation with similarity-based retrieval to guide LLM-driven trajectory generation in new scenes.

We then looked at failure cases in this experiment. When using a single demonstration as a prompt, in most cases, the trajectory generated by LLM either simply replayed the original demonstration (ignoring the new object position) or generated invalid trajectories, such as producing a straight-line action that exited the workspace. In these cases, successful completion was rare and only occurred when the randomized keypoint positions happened to be very close to the original object positions.

Similarly, when using three randomly sampled augmentation scenes (without retrieval), the LLM occasionally produced valid outputs, but the success rate was still limited. The increased diversity of object positions improved generalization to some extent, but the lack of correlation between the prompt scene and the query scene led to inconsistent or misaligned trajectory predictions, limiting the success rate to 24%.

In contrast, the ”augmentation + retrieval” strategy apply in 50 trials but only have two failures. In both failure cases, the LLM generated invalid trajectories. The first error was that the LLM failed to close the gripper during the grasping phase, resulting in a failed attempt. The second error is that the LLM omits the grasp motion entirely and generates a straight line motion out of the workspace.

5.3.2 Generalization to Real-World Scenarios

Table 4 summarizes the success rates of our method when tested under real-world scenario variations, including changes in clutter, object instances, and spatial configurations, which mentioned in

Section 4.4. The average success rates across these three variation scenario types are relatively balanced, ranging from 62.5% to 75%, which suggests that the system is robust across different types of generalization challenges.
However, we observe a significant variation in performance across different tasks. Specifically, the success rate of Mix-bowl task is only 33.4%, which is much lower compared to Pouring (88.9%) and *Trash_in_bin* (77.8%).

Table 4: Success rate under real-world scenario variations for different tasks. Each variation tests generalization across clutter, object instance, and spatial configuration.

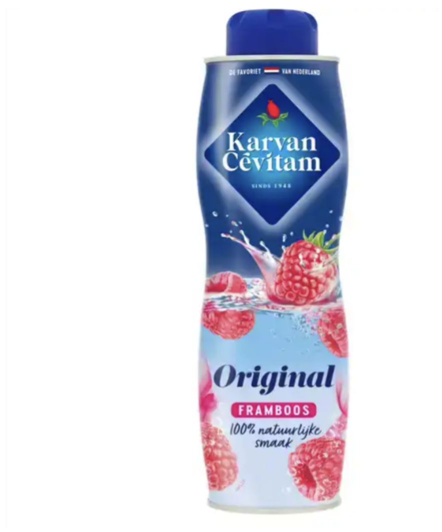
Task	**Clutter**	**Object**	**Spatial**	**Avg. Success Rate**
Pouring	2/3	3/3	3/3	88.9%
Mix_bowl	1/2	0/2	1/2	33.4%
Trash_in_bin	2/3	3/3	2/3	77.8%
Success Rate	62.5%	75%	75%	**70.8%**

For the two tasks with relatively high success rates, Pouring and Trash-in-bin, some failures were not due to completely invalid trajectories, but rather fine-grained execution mismatches. In several cases, the robot failed to grasp the object correctly. Additionally, in the Pouring task, one specific failure occurred in the scenario involving a new bottle with a different geometry(see Figure 23a). Unlike the bottle in original demonstration, which had a long, narrow neck, the new bottle have a shorter bottle neck and the whole shape closer to a cylinder. Although the robot was able to grasp the object, it grasped too close to the mouth of the bottle because the LLM predicted the grasping point based on the structure of the key points of the object and the gripper position in previous demonstrations. However, the LLM did not consider the change in the geometry of its other functional structures. The grasping position was able to pick it up, but because the new bottleneck was too short, the position of the bottle mouth could not be properly aligned with the receiving cup, causing the pouring attempt to fail. From Figure 23b we can see that, when pouring occurs, the bottle mouth still have a long distance away from the cup.

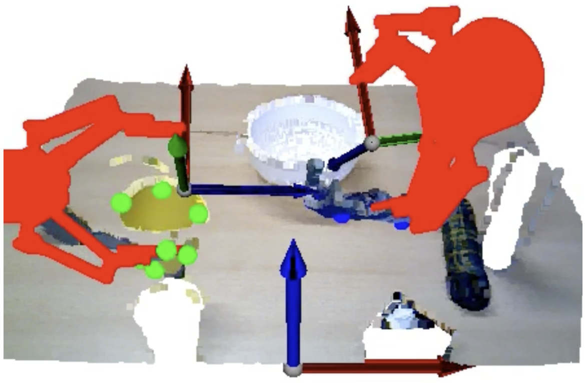
We also analyzed the failed trials in the Mix-bowl task and found several factors that may have led to a significant decrease in its success rate. For the other two tasks with high success rate, Pouring and Trash-in-bin, involves objects with large graspable areas, and the objects are have cylindrical or symmetrical shapes (such as bottles or cans), while the Mix-bowl task requires the robot to grasp a thin mixer handle. This poses two additional challenges. The first is that the graspable area is smaller and more precise, with almost no positional error. In addition, if the handle rotates due to the change in the object's position, the robot must not only translate, but also accurately rotate the gripper to match the new orientation - a task that requires high precision and sophisticated motion planning. These complexities make this task more demanding on the accuracy of the generated trajectory.

5.4 Limitations for LLM Trajectory Prediction

From these two experiments, it can be seen that the trajectory of the robot end effector generated by the LLMs has obvious uncertainty and a lack of interpretability. First, in many tests, the LLM can only complete the task by almost replaying the trajectory in the example when the observation



(a) The bottle with smaller bottle neck.



(b) The failure of the pouring task

Figure 23: One failure case in the Pouring, the robot grasped the bottle too close to the opening, causing misalignment during the pouring motion and leading to task failure.

in the example trajectories in the prompt is close enough to the current observation. However, the LLM sometimes also generates some invalid or unexecutable trajectories, especially when the current input observations are very different from the example observations. For example, sometimes the switching between the opening and closing of the gripper is ignored, or a straight trajectory in the wrong direction is directly generated, causing the end effector to deviate from the workspace. This shows that the generalization performance of trajectory prediction using LLMs for new observations is relatively low. Second, in tasks that require fine manipulation and high spatial accuracy, the LLM has limited generalization ability. Taking the grasping of the handle of a mixer at different angles as an example, the model often cannot accurately adjust the direction and pose of the gripper, which has a great impact on the execution of the task. Since such tasks have high requirements for posture accuracy, even a small error can lead to failure to grasp.

The possible reason for this limitation is the lack of the LLM's own capabilities. The LLM is essentially a large-scale language model trained on text, and its understanding of geometric concepts such as "pose," "direction," and "distance" is indirect and symbolic. It cannot truly perceive the physical layout of objects in space, but as a pattern-generating machine, it relies on symbols or structural representations (such as keypoint coordinates) provided in the prompt. Therefore, when faced with a scene that is very different from the example, the model cannot establish an effective spatial mapping, resulting in unreasonable trajectory generation.

6 Conclusion

This section mainly answers the research questions, summarizes our work in this thesis, and discusses the possible future work.

6.1 Summary of Main Contributions

To answer the core research question:

How can single human demonstrations be utilized within a low-cost framework for data generation and augmentation to enable generalizable dual-arm robot learning?

This paper proposes a complete framework that can convert human video demonstrations into a robot-executable trajectory, and uses a data augmentation method to generate generalizable trajectory data for this task. Specifically, we develop a vision-based framework where we first record a demonstration video for each task and then use a simplified version of the HaMeR model to extract 3D hand keypoints from the recorded RGB demonstration video. These keypoints are then corrected by using the corresponding depth information to improve the spatial accuracy. Afterwards, the corrected hand poses are converted to robot gripper poses through a heuristic mapping method, effectively bridging the embodied gap without relying on motion capture systems or wearable sensors. Quantitative evaluation on the ARCTIC dataset shows that the average 3D MPJPE of the hand keypoints predicted and aligned by our framework is only 0.023 meters, which shows the effectiveness of our method for extracting hand keypoints.

Based on this trajectory extraction, we introduce a trajectory augmentation strategy that is able to generate diverse trajectory samples from a single task demonstration. We generated thousands of diverse trajectories by first transforming the original trajectory under the object coordinate frame, and then applying spatial translations and mirror-flipping to simulate varied task configurations. This allows to simulate object motion in different spatial configurations, significantly increasing the robot's generalization ability. The effectiveness of this augmentation method is verified by predicting the end-effector trajectory using LLMs. We use the augmented data as few-shot prompts to enable the LLM to perform contextual trajectory prediction without any fine-tuning.

In our evaluation on different takes of pouring task—the LLM achieved a success rate of 96% in unseen augmented scenes when using our augmentation + retrieval prompting method, compared to 6% when using only the original single demonstration without augmentation. To further evaluate the generalization ability of our framework, we designed real-world scenarios involving three types of variations: different backgrounds, object replacements, and spatial configurations. We selected three representative dual-arm tasks, which are Pouring, Mix-Bowl, and Trash-in-Bin. Each variation was designed to mimic practical deployment challenges, such as visual clutter, intra-category object differences, and unseen object poses. Despite these complexities, our framework maintained strong performance, achieving an average success rate of 70.8% across all variation types and tasks. These results confirm the robustness and practical applicability of our system in diverse real-world settings.

In conclusion, this thesis addresses the core research question by proposing a low-cost and efficient data generation framework for dual-arm imitation learning. By leveraging a single human demonstration and applying a series of transformation and augmentation strategies, the framework is able to generate diverse and generalizable training data. As a result, we significantly reduce the cost and complexity of robot data collection while achieving strong generalization capabilities in the presence of varied objects and environments.

6.2 Future Work

While the proposed framework has demonstrated promising results in low-cost dual-arm imitation learning, several directions remain open for further improvement and exploration.

Currently, we mainly rely on a self-built dummy simulator for the visualization and evaluation of generated trajectories. Although this simulator can be used to verify the grasp angle, affordances, and whether the task was successful for the trajectory, it cannot simulate the real physical interaction process between the robot and the object, such as contact, friction, or collision, due to the lack of real physics engine support. This limits a comprehensive evaluation of task success rate and trajectory executability. Therefore, future work could consider deploying this framework on a more realistic physics simulation platform, such as NVIDIA Isaac Sim [43], to enable evaluation under physically accurate conditions. In addition, deploying the system on a real dual-arm robot platform will further verify the robustness and practical effectiveness of our method in real-world settings, which is a crucial step toward real deployment.

Another promising direction is to replace the current LLM-based trajectory generation with a keypoint-based policy network. In our experiments, results show that although LLMs are capable of few-shot action prediction, they often suffer from instability, and lack of controllability and interpretability. Given that our framework already represents object and end-effector states using 3D keypoints, future work could explore training a Transformer-based or diffusion-based policy model [44][45][29] that maps keypoint observations to continuous robot trajectories[46]. Such a model could offer higher reliability, faster inference, and better scalability to new scenarios.

Finally, our current framework considers each demonstration video as a single-task instance. In future work, we can extend this by decomposing each task trajectory into a sequence of low-level primitive actions such as *approach*, *grasp*, *move*, or *place*. These primitives could be learned or extracted automatically, allowing the system to build a library of reusable and composable skills [33]. By leveraging the task planning capabilities of LLMs, we could use natural language task descriptions to select, sequence, and adapt these primitive actions to generate new task trajectories [47][48]. This would effectively transition the system from pure imitation learning toward high-level planning and skill composition. In the long term, such an approach could potentially enable the framework to handle **zero-shot tasks** [49]—that is, to execute novel tasks that have not been demonstrated before—by recombining existing primitives based on abstract task goals.

Bibliography

[1] R. R. Murphy, *Introduction to AI robotics*. MIT press, 2019.

[2] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *arXiv preprint arXiv:1804.05172*, 2018.

[3] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9626–9633, IEEE, 2020.

[4] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2901–2910, 2019.

[5] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11444–11453, 2020.

[6] Y. Tong, H. Liu, and Z. Zhang, "Advancements in humanoid robots: A comprehensive review and future prospects," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 2, pp. 301–328, 2024.

[7] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[8] J. Hu, R. Hendrix, A. Farhadi, A. Kembhavi, R. Martín-Martín, P. Stone, K.-H. Zeng, and K. Ehsani, "Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning," *arXiv preprint arXiv:2409.16578*, 2024.

[9] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on robot learning*, pp. 651–673, PMLR, 2018.

[10] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, pp. 28694–28698, 2025.

[11] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, *et al.*, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," *arXiv preprint arXiv:2502.01143*, 2025.

[12] S. An, Z. Meng, C. Tang, Y. Zhou, T. Liu, F. Ding, S. Zhang, Y. Mu, R. Song, W. Zhang, *et al.*, "Dexterous manipulation through imitation learning: A survey," *arXiv preprint arXiv:2504.03515*, 2025.

[13] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu, "Dexcap: Scalable and portable mocap data collection system for dexterous manipulation," *arXiv preprint arXiv:2403.07788*, 2024.

[14] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” *arXiv preprint arXiv:2304.13705*, 2023.

[15] S. Schaal, “Is imitation learning the route to humanoid robots?,” *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[16] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, “Deep imitation learning for complex manipulation tasks from virtual reality teleoperation,” in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 5628–5635, IEEE, 2018.

[17] Z. Fan, O. Taheri, D. Tzionas, M. Kocabas, M. Kaufmann, M. J. Black, and O. Hilliges, “Arctic: A dataset for dexterous bimanual hand-object manipulation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12943–12954, 2023.

[18] V. Vantage, “Cutting edge, flagship camera with intelligent feedback and resolution.”

[19] Z. Fu, T. Z. Zhao, and C. Finn, “Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation,” *arXiv preprint arXiv:2401.02117*, 2024.

[20] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, “Imitation from observation: Learning to imitate behaviors from raw video via context translation,” in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1118–1125, IEEE, 2018.

[21] N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada, “Ditto: Demonstration imitation by trajectory transformation,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7565–7572, IEEE, 2024.

[22] A. Bahety, P. Mandikal, B. Abbatematteo, and R. Martín-Martín, “Screwmimic: Bimanual imitation from human videos with screw space projection,” *arXiv preprint arXiv:2405.03666*, 2024.

[23] Y. Zhu, A. Lim, P. Stone, and Y. Zhu, “Vision-based manipulation from single human video with open-world object graphs,” *arXiv preprint arXiv:2405.20321*, 2024.

[24] G. Papagiannis, N. Di Palo, P. Vitiello, and E. Johns, “R+ x: Retrieval and execution from everyday human videos,” *arXiv preprint arXiv:2407.12957*, 2024.

[25] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song, “Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation,” *arXiv preprint arXiv:2505.21864*, 2025.

[26] L. Annabi, Z. Ma, and S. M. Nguyen, “Unsupervised motion retargeting for human-robot imitation,” in *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 204–208, 2024.

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, “Learning transferable visual models from natural language supervision,” in *International conference on machine learning*, pp. 8748–8763, PmLR, 2021.

[28] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, *et al.*, “Openvla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.

[29] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, "Rdt-1b: a diffusion foundation model for bimanual manipulation," *arXiv preprint arXiv:2410.07864*, 2024.

[30] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[31] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[32] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[33] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi, "Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration," *arXiv preprint arXiv:2311.12015*, 2023.

[34] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[35] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[36] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, *et al.*, "In-context learning and induction heads," *arXiv preprint arXiv:2209.11895*, 2022.

[37] N. Di Palo and E. Johns, "Keypoint action tokens enable in-context imitation learning in robotics," *arXiv preprint arXiv:2403.19578*, 2024.

[38] N. Ienaga, B. W. Scotney, H. Saito, A. Cravotta, M. G. Busà, *et al.*, "Natural gesture extraction based on hand trajectory," in *Irish machine vision and image processing conference*, pp. 81–88, 2018.

[39] G. Pavlakos, D. Shan, I. Radosavovic, A. Kanazawa, D. Fouhey, and J. Malik, "Reconstructing hands in 3d with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9826–9836, 2024.

[40] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *arXiv preprint arXiv:2201.02610*, 2022.

[41] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, "Vitpose: Simple vision transformer baselines for human pose estimation," *Advances in neural information processing systems*, vol. 35, pp. 38571–38584, 2022.

[42] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu, "Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning," *arXiv preprint arXiv:2502.16932*, 2025.

[43] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, “Isaac gym: High performance gpu-based physics simulation for robot learning,” *arXiv preprint arXiv:2108.10470*, 2021.

[44] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, “Octo: An open-source generalist robot policy,” *arXiv preprint arXiv:2405.12213*, 2024.

[45] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[46] S. Haldar and L. Pinto, “Point policy: Unifying observations and actions with key points for robot manipulation,” *arXiv preprint arXiv:2502.20391*, 2025.

[47] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, “Vima: General robot manipulation with multimodal prompts,” *arXiv preprint arXiv:2210.03094*, vol. 2, no. 3, p. 6, 2022.

[48] J. Jeon, H.-r. Jung, F. Yumbla, T. A. Luong, and H. Moon, “Primitive action based combined task and motion planning for the service robot,” *Frontiers in Robotics and AI*, vol. 9, p. 713470, 2022.

[49] J. Tang, Z. Ye, Y. Yan, Z. Zheng, T. Gao, and Y. Jin, “Zero-shot robotic manipulation with language-guided instruction and formal task planning,” *arXiv preprint arXiv:2501.15214*, 2025.

Appendices

This appendix provides additional information to support and extend the main content of the thesis.

A Prompt Template for Trajectory Prediction

Below is an example of the structured prompt used to guide the large language model (LLM) in trajectory prediction. The prompt is formatted to include object keypoints, initial gripper pose, and the corresponding action trajectory for a set of demonstrations. The LLM receives this format and generates the trajectory for a new, unseen scene based on the provided input-output mappings.

```
SYSTEM_PROMPT = """
You are a dual-arm robot controller.

You will be given demonstration pairs of:

[Object Keypoints, Initial Gripper Poses] → [Sequence of Actions]

Each keypoint input is a set of 11 points in 3D space representing object geometry.

Each gripper pose is a 7D vector (position + quaternion) plus 1D gripper state
(open=1, closed=0), i.e., 8 values per gripper.

The output is a sequence of actions of shape (n, 2, 8): n steps of
[left_arm_action, right_arm_action].

You will then receive a new keypoint input and gripper initial pose and must
output the best matching action sequence. Only output the action list.
No extra explanations.
"""
FORMAT_PROMPT = """

** Demo 0 **

Input Keypoints:
[[[ 0.35672957,  0.14748058,  0.17977694],
[ 0.3214035 ,  0.11660732,  0.13142524],
[ 0.35583171,  0.10572507,  0.12295601],

... ...

[-0.07754012,  0.27673301,  0.1166117 ],
[-0.07615347,  0.20267133,  0.03556716],
[-0.09157936,  0.20060903,  0.06849983],
[-0.05039972,  0.20129231,  0.09501093]]]
# shape:(k, 2, 8)
```

```
Initial Gripper Poses:
[[-0.2709437551308525, 0.11958405324855437, 0.12078331822700039,
-0.057411362821149316, 0.7014662495854279, 0.6812815490969106,
0.20125726563701, 0.0], [0.1730597776646704, 0.03363280265819868,
0.1780268205083202, -0.6161047186076197, -0.030622304693960458,
0.2926275220131287, 0.7306479203591234, 0.0]]
# shape:(2,8)

Action Sequence:
[[[-0.2709437551308525, 0.11958405324855437, 0.12078331822700039,
-0.057411362821149316,
0.7014662495854279, 0.6812815490969106, 0.20125726563701, 0.0], [0.1730597776646704,
0.03363280265819868, 0.1780268205083202, -0.6161047186076197,
-0.030622304693960458, 0.2926275220131287, 0.7306479203591234, 0.0]],

... ...

[[-0.2343635803417889, 0.09153460895070725, 0.12238684169647099,
-0.042954353488016864, 0.7005556085330175, 0.6737774899467769,
0.2310858215260754, 1.0], [0.1329770275734641, 0.05606903617358758,
0.1752557170139616, -0.6083820354163534, -0.08859450583334977,
0.32189607258806047, 0.7200036326097123, 1.0]]]
# shape:(n, 2, 8)

** Demo 1 **

... ...

** Demo 2 **

... ...

** New Input **
Input Keypoints:
[[[ 0.35672957,  0.14748058,  0.17977694],
[ 0.3214035 ,  0.11660732,  0.13142524],
[ 0.35583171,  0.10572507,  0.12295601],

... ...

[-0.07615347,  0.20267133,  0.03556716],
[-0.09157936,  0.20060903,  0.06849983],
[-0.05039972,  0.20129231,  0.09501093]]]
# shape:(k, 2, 8)

Initial Gripper Poses:
```

```
[[-0.2709437551308525, 0.11958405324855437, 0.12078331822700039,
-0.057411362821149316, 0.7014662495854279, 0.6812815490969106,
0.20125726563701, 0.0],
[0.1730597776646704, 0.03363280265819868, 0.1780268205083202,
-0.6161047186076197, -0.030622304693960458, 0.2926275220131287,
0.7306479203591234, 0.0]]
# shape:(2,8)
"""
```

B Full version for the Evaluation Metrics result

The tables in this section present the full version of our evaluation metrics results across all task demonstrations we tested in the Arctic dataset.

Take	mpjpe_left	mpjpe_left_aligned	mpjpe_right	mpjpe_right_aligned
scissors_use_02	1.420	0.018	1.317	0.021
box_grab_01	1.401	0.030	1.417	0.039
box_use_01	1.381	0.034	1.252	0.027
box_use_02	1.393	0.039	1.343	0.031
capsulemachine_use_01	1.231	0.018	1.143	0.019
capsulemachine_use_02	1.023	0.016	1.025	0.020
capsulemachine_grab_01	1.374	0.020	1.391	0.022
espressomachine_grab_01	1.333	0.026	1.530	0.023
espressomachine_use_01	1.176	0.022	1.551	0.022
espressomachine_use_02	1.199	0.028	1.299	0.021
ketchup_grab_01	1.306	0.023	1.292	0.024
ketchup_use_01	1.156	0.020	0.995	0.019
ketchup_use_02	1.226	0.019	1.054	0.019
phone_grab_01	1.474	0.022	1.504	0.021
phone_use_01	1.138	0.017	1.147	0.018
phone_use_02	1.189	0.017	1.193	0.020
scissors_use_01	1.441	0.017	1.270	0.020
scissors_grab_01	1.464	0.018	1.501	0.017

Take	pa-mpjpe_left	pa-mpjpe_right_aligned	pa-mpjpe_right	pa-mpjpe_right_aligned
scissors_use_02	0.014	0.014	0.017	0.017
box_grab_01	0.036	0.036	0.043	0.043
box_use_01	0.039	0.039	0.028	0.028
box_use_02	0.045	0.045	0.032	0.032
capsulemachine_use_01	0.018	0.018	0.015	0.015
capsulemachine_use_02	0.017	0.017	0.019	0.019
capsulemachine_grab_01	0.020	0.020	0.019	0.019
espressomachine_grab_01	0.029	0.029	0.021	0.021
espressomachine_use_01	0.024	0.024	0.017	0.017
espressomachine_use_02	0.032	0.032	0.021	0.021
ketchup_grab_01	0.021	0.021	0.020	0.020
ketchup_use_01	0.019	0.019	0.017	0.017
ketchup_use_02	0.018	0.018	0.017	0.017
phone_grab_01	0.018	0.018	0.016	0.016

Take	**pa_mpjpe_l**	**pa_mpjpe_la**	**pa_mpjpe_r**	**pa_mpjpe_ra**
phone_use_01	0.015	0.015	0.015	0.015
phone_use_02	0.018	0.018	0.017	0.017
scissors_use_01	0.017	0.017	0.015	0.015
scissors_grab_01	0.015	0.015	0.012	0.012

Take	**2D_mpjpe_r**	**2D_mpjpe_l**	**pck05_r**	**pck15_r**	**pck05_l**	**pck15_l**
scissors_use_02	19.196	17.588	0.867	1.000	0.923	1.000
box_grab_01	38.454	29.654	0.369	0.999	0.520	1.000
box_use_01	32.557	33.799	0.537	0.985	0.472	0.998
box_use_02	33.464	39.288	0.531	0.992	0.430	0.969
capsulemachine_use_01	21.445	20.059	0.801	0.997	0.824	0.999
capsulemachine_use_02	28.974	24.731	0.675	0.986	0.715	0.999
capsulemachine_grab_01	24.256	24.090	0.734	0.995	0.727	0.999
espressomachine_grab_01	25.800	29.868	0.678	1.000	0.595	1.000
espressomachine_use_01	21.640	27.399	0.831	0.990	0.641	1.000
espressomachine_use_02	24.734	30.864	0.740	0.991	0.574	0.995
ketchup_grab_01	25.299	26.768	0.767	0.976	0.720	0.980
ketchup_use_01	27.334	27.775	0.652	0.995	0.673	0.991
ketchup_use_02	24.639	20.717	0.750	0.996	0.837	0.999
phone_grab_01	22.611	25.399	0.778	0.999	0.693	0.995
phone_use_01	23.557	19.804	0.756	0.997	0.851	1.000
phone_use_02	23.869	20.521	0.744	1.000	0.814	0.999
scissors_use_01	16.616	17.413	0.927	1.000	0.941	1.000
scissors_grab_01	16.806	18.310	0.912	1.000	0.871	1.000