MASTER'S THESIS 2026

# Diffusion Policy for Force- and Vision-Guided Robotic Liquid Pouring Task

Wenrui Xie, Jiacheng Zhang


ISSN 1650-2884
LU-CS-EX: 2026-53

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2026-53

# Diffusion Policy for Force- and Vision-Guided Robotic Liquid Pouring Task

**Wenrui Xie, Jiacheng Zhang**

# Diffusion Policy for Force- and Vision-Guided Robotic Liquid Pouring Task

Wenrui Xie
we5415xi-s@student.lu.se

Jiacheng Zhang
ji7517zh-s@student.lu.se

July 6, 2026

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Volker Krueger, volker.krueger@cs.lth.se

Examiner: Maj Stenmark, maj.stenmark@cs.lth.se

**Abstract**

Liquid pouring is a critical, high-precision task in laboratory automation and pharmaceutical manufacturing. Existing imitation learning methods struggle with the high demands for accuracy and robustness in contact-rich pouring tasks. This thesis presents an extension of the diffusion policy framework by integrating force feedback for a dual-arm robot pouring approach. Unlike traditional methods, diffusion-based policy effectively captures the multi-modal action distributions inherent in pouring tasks, where diverse trajectories may lead to successful outcomes. Additionally, a diffusion policy generates temporal action chunks rather than point-wise outputs, which ensures smooth, continuous control suitable for temporally extended pouring motions. To complement these capabilities with physical awareness, we augment the observation space by directly integrating joint torque signals as proprioceptive inputs together with visual observations. This multimodal observation design allows the policy to condition action generation on visual information, robot joint states, and raw joint torque signals, which provide force-related cues about load changes and physical interaction during pouring. For safety during data collection, rice is used as a proxy medium instead of real liquids. The policy, trained on 150 human demonstrations, was evaluated over 60 trials (20 per configuration) to compare the single-view baseline, multi-view baseline, and force-integrated multi-view policy. A trial is recorded as successful if the robot precisely transfers rice into the target container without missing and both arms return to designated positions. Results show that the force-integrated multi-view policy achieved the highest overall success rate and improved task progression compared with policies without force-feedback input. However, the system is evaluated using rice as a granular proxy medium, and its generalization to real liquid pouring with different fluid properties remains an objective for future work.

**Keywords**: Diffusion Policy, Force Feedback, Robotic Pouring, Laboratory Automation, Imitation Learning

# Acknowledgements

We would like to express our sincere gratitude to all those who have supported, guided, and assisted us throughout the course of this thesis project.

First and foremost, we extend our deepest appreciation to our supervisor, Volker Krueger, for his expert guidance, constructive feedback, and continuous support at every stage of this work.

We are also profoundly grateful to Jialong Li for his invaluable assistance with the VR teleoperation system and the KUKA robotic platform. His technical guidance played a crucial role in establishing a solid foundation for our experimental setup.

Our heartfelt thanks further go to Marcus Klang for his essential technical support in hardware and system configuration, which was indispensable for ensuring the stable operation of our research platform.

Finally, we would like to thank Alexander Pisarevskiy for his help in setting up the experimental environment and providing the necessary tools for our work.

# Contents

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Lab automation has become an essential way to improve the efficiency and accuracy of experimental workflows [2]. In traditional laboratory settings, tasks such as liquid handling, sample transfer, and titration are often performed manually. This repetitive work not only consumes a significant amount of time, but also exposes lab personnel to risks from toxic reagents, endangering their health [7]. Among these tasks, precise liquid pouring is critical; errors during the process can lead to failed experiments, damage to devices, and safety hazards.

Automated robotic systems offer a promising solution to these challenges. Recent advancements in artificial intelligence, computer vision, and data-driven methods have enabled robots to perceive their laboratory environment [23]. This allows robotic systems to perform certain tasks with high precision and consistency, reducing the workload on lab workers [27].

Among laboratory automation tasks, liquid pouring is a representative example of contact-rich manipulation. Unlike simple pick-and-place operations, pouring requires the robot to continuously adjust its motion based on the container state, the evolving behavior of the poured material, and the interaction between the robot and the environment [25]. Accurate execution requires precise 6-DoF control under dynamic and contact-rich conditions. Factors such as container geometry, material flow, alignment, and environmental uncertainty make explicit modeling and traditional control approaches difficult to generalize.

While Reinforcement Learning (RL) is theoretically suitable for sequential decision-making, its application to high-frequency and fine-grained pouring tasks is often hindered by sparse rewards and the complexity of reward design. Imitation Learning (IL), in contrast, offers a more practical and data-efficient alternative by learning temporally coordinated manipulation skills directly from expert demonstrations.

## 1.2 Research Context and Gap

Recent imitation learning methods for robotic manipulation can be broadly divided into two trends. One trend is foundation-model-based robot policies, such as Vision-Language-Action (VLA) models including OpenVLA [14] and $\pi_{0.5}$ [13]. These models aim to generalize across diverse tasks, environments, and language instructions. Another trend is task-specific visuomotor policies, such as Diffusion Policy [3], which are trained on demonstrations collected for a specific manipulation task and can model multiple valid action trajectories for the same observation.

Although these approaches have shown strong potential, many policies still rely mainly on visual observations and robot proprioception. In contact-rich tasks such as pouring, visual information alone may be insufficient to detect physical interaction events such as load changes, container contact, or stalling. As a result, a policy may fail to react appropriately when the robot reaches the pouring stage or when the container dynamics change during execution.

Prior work has shown that force feedback can improve contact-rich manipulation by providing physical interaction cues that are difficult to infer from images alone. For example, ForceSight [4], ForceMimic [18], DIPCOM [1], and force-conditioned diffusion policies [26] incorporate force-related information into manipulation policies. However, such approaches often rely on external force/torque sensors, force-motion capture systems, or explicitly modeled force targets, which can increase hardware cost, calibration effort, and system complexity.

This thesis therefore investigates whether raw internal joint torque measurements from the KUKA LBR iiwa can be used as force-related proprioceptive observations in a diffusion-policy-based imitation learning system. The work builds on Diffusion Policy [3] and an existing Quest2ROS2-based teleoperation codebase [15]. Instead of developing a new VR teleoperation framework from scratch, this thesis adapts the existing teleoperation code into a data collection workflow for dual-arm granular pouring and extends the recording pipeline to include synchronized joint torque measurements.

## 1.3 Goal and Research Questions

The goal of this thesis is to evaluate whether raw internal joint torque measurements can improve the task performance of a diffusion-policy-based imitation learning system for dual-arm granular pouring.

This goal is addressed through the following research questions:

- **RQ1:** Does multi-view visual input improve task performance compared with a single-view visual policy?
- **RQ2:** Does incorporating raw joint torque measurements improve task progression and full-task success compared with a vision-only multi-view diffusion policy?
- **RQ3:** What failure modes remain when raw joint torque measurements are used as force-related policy inputs?

## 1.4 Methodology

The methodology follows a task-specific imitation learning approach. First, expert demonstrations are collected on a dual-arm KUKA LBR iiwa platform using a VR-based teleoperation interface built on Quest2ROS2 [15]. The existing teleoperation code is adapted into a complete demonstration-collection workflow for the pouring task. This allows a human operator to demonstrate the temporally coordinated motions required for dual-arm pouring.

Second, the data collection workflow records synchronized multi-view RGB observations, robot joint states, gripper commands, raw joint torque measurements, and estimated end-effector wrench signals. The estimated wrench is used for exploratory force-feedback analysis, while the final policy uses raw joint torque measurements as force-related proprioceptive inputs.

Third, three diffusion-policy variants are trained and compared: a single-view baseline, a multi-view baseline, and a force-integrated multi-view policy. This design allows the effects of visual information and joint torque measurements to be evaluated separately.

Finally, the trained policies are deployed on the real dual-arm robot and evaluated through repeated trials. Performance is measured using subtask progression and full-task success, followed by failure-mode analysis to identify remaining limitations.

## 1.5 Limitations

This thesis has several limitations. First, the experiments use rice as a granular proxy for liquid pouring. Rice preserves important aspects of pouring, such as flow regulation, container alignment, and temporal coordination, but it does not reproduce liquid-specific properties such as viscosity, surface tension, sloshing, or fluid inertia. Therefore, the experiments should be interpreted as validation on granular pouring rather than direct validation of liquid manipulation.

Second, the dataset consists of 150 demonstrations collected for a specific dual-arm setup and task. It should therefore be considered a task-specific demonstration dataset rather than a large-scale dataset.

Third, the force-related input is limited to raw internal joint torque measurements from the KUKA LBR iiwa. These measurements avoid the need for external force/torque sensors, but they are not direct measurements of contact force and include both robot dynamics and interaction effects.

In addition, this thesis does not compare raw joint torque inputs against gravity-compensated or residual torque inputs. Such an ablation would be needed to determine whether the observed improvement is specifically caused by external force-related information, or whether it is partly due to the additional proprioceptive dimensions provided by the raw torque vector.

Finally, the evaluation is limited to the specific robot platform, container setup, camera configuration, and task conditions used in this thesis. Broader generalization to other liquids, containers, robot platforms, and laboratory tasks remains future work.

## 1.6 Contributions

The main contributions of this thesis are as follows:

1. **Task-specific demonstration dataset with joint torque measurements:** We adapted an existing Quest2ROS2-based bi-manual teleoperation codebase into a complete demonstration-collection workflow for a dual-arm granular pouring task. The resulting task-specific dataset contains 150 human demonstrations with synchronized multi-view RGB observations, dual-arm joint states, gripper commands, raw joint torque measurements, and estimated end-effector wrench signals.

2. **Torque-conditioned Diffusion Policy for dual-arm pouring:** We extend Diffusion Policy [3] by incorporating raw internal joint torque measurements as force-related proprioceptive observations together with multi-view visual inputs and robot joint states. This allows the policy to condition action generation on both visual scene information and physical interaction cues without adding external force/torque sensors.

3. **Real-robot evaluation of visual and torque observations:** We evaluate three policy variants on the physical dual-arm robot: a single-view baseline, a multi-view baseline, and a force-integrated multi-view policy. Through ablation studies and failure-mode analysis, we investigate how visual observations and raw joint torque measurements affect task progression and full-task success.

## 1.7 Division of Work

The thesis work was carried out jointly by both authors. Wenrui Xie and Jiacheng Zhang both contributed to system integration, demonstration collection, dataset preparation, policy training, real-robot evaluation, failure-mode analysis, and thesis writing. Design decisions, experimental procedures, and result analysis were discussed and validated jointly throughout the project.

## 1.8 Thesis Outline

The remainder of the thesis is organized as follows. Chapter 2 presents related work on robotic laboratory automation, imitation learning, force-aware manipulation, and diffusion-policy-based approaches. Chapter 3 introduces the technical background of imitation learning, diffusion models, Diffusion Policy, and the neural network components used in the policy. Chapter 4 describes the data collection system, force-related signal processing, and ROS 2-based communication framework. Chapter 5 presents the experimental setup, quantitative results, and failure-mode analysis. Chapter 6 concludes the thesis and discusses future work.

# Chapter 2
# Related Work

This chapter is structured in two key dimensions relevant to the work presented in this thesis. First, we provide an overview of the background of the application and recent advances in the field of laboratory automation. The discussion then shifts to imitation learning (IL), where we examine the inherent challenges IL faces in contact-rich tasks. Following this, we introduce teleoperation approaches used for collecting expert demonstrations, highlighting the VR teleoperation method utilized in our work. Finally, we present the two key components of our framework: the policy design that integrates force-feedback information, and the diffusion model used for robust policy learning.

**Robotic Laboratory Automation:** In the domain of laboratory automation, robotic systems have been successfully deployed to perform basic repetitive tasks such as picking and placing standardized labware, significantly improving experimental throughput and consistency. For example, the LAPP framework [31] leverages digital-twin simulations to enable reliable pick-and-place transportation of standard sample carriers in modular laboratory environments.

Despite these advances, extending robotic automation to contact-rich manipulation tasks remains challenging. Liquid pouring, in particular, requires high precision and continuous adaptation to the dynamic interaction between the container, the liquid, and the environment. Existing laboratory automation solutions still lack sufficiently robust and accurate approaches for such tasks.

The PourIt! framework [16], for instance, proposes a visual closed-loop method for robotic liquid pouring. While effective under controlled conditions, it relies solely on a single RGB image for perception. As a result, its performance is highly sensitive to variations in lighting conditions, liquid appearance, and visual occlusions, which can significantly degrade reliability in real-world laboratory settings. These limitations highlight the need for more robust learning-based approaches that can better handle the complexity and uncertainty inherent in contact-rich laboratory manipulation tasks.

**Imitation Learning in Robotic Manipulation:** Imitation learning (IL)[11] has become a fundamental paradigm for robot skill acquisition, enabling robots to learn complex behaviors

directly from expert demonstrations rather than relying on manually designed controllers or reward functions [5]. Depending on how demonstrations are utilized, existing IL methods can be broadly categorized into several classes.

Behavioral Cloning (BC)[11] formulates imitation learning as a supervised learning problem, where a policy is trained to directly map observed states to actions by minimizing the discrepancy between predicted and demonstrated actions. While simple and effective in many settings, BC often suffers from distribution shift and limited robustness, particularly in tasks that require precise physical interaction.

Inverse Reinforcement Learning (IRL)[20] aims to infer an underlying reward function that explains expert behavior, after which a policy is derived by optimizing the learned reward. Although IRL provides a more principled formulation, it typically incurs high computational cost and complexity, making it less suitable for high-frequency, fine-grained manipulation tasks.

Another line of work focuses on trajectory-level modeling, such as Dynamic Movement Primitives (DMPs)[12] and Probabilistic Movement Primitives (ProMPs)[21], which encode demonstrated motions as structured dynamical systems or probabilistic distributions. These methods enable smooth and stable trajectory reproduction but often struggle to generalize to situations with varying contact dynamics and environmental uncertainty.

Despite their differences, many imitation learning approaches for robotic manipulation rely heavily on visual observations as the primary source of feedback [28]. In contact-rich tasks, such as liquid pouring, visual information alone is often insufficient to capture physical interaction forces and subtle contact events. As a result, policies trained solely on vision may fail to react appropriately to rapid changes in pose, force, or contact state. Even recent Vision-Language-Action (VLA) models, such as OpenVLA [14], still exhibit limited performance in tasks that demand high precision and dynamic force regulation. These limitations motivate the incorporation of additional modalities, such as force-related and proprioceptive information, into imitation learning policies.

To overcome the limitations of vision-dominant imitation learning in contact-rich manipulation tasks, recent research has emphasized incorporating force feedback into policy design. Force information provides a physically grounded representation of robot–environment interaction, allowing policies to adapt to delicate contacts and dynamic forces that are difficult to infer from vision alone. Representative approaches include ForceMimic [18], which learns continuous force and pose trajectories for adaptive contact control, and ForceSight [4], which fuses force, visual, and proprioceptive inputs to predict combined force and pose goals. These studies demonstrate that integrating force feedback can substantially improve robustness, adaptability, and generalization in contact-rich manipulation tasks, motivating further exploration of advanced policy representations such as diffusion-based methods.

**Diffusion Models in Robotics:** Diffusion models initially achieved breakthrough success in generative modeling, demonstrating exceptionally high generation quality and diversity, particularly in image synthesis. This success is primarily due to their natural ability to model complex, multimodal distributions. The process involves gradually adding noise to data through a Markov chain until it converges to a Gaussian distribution, followed by learning the reverse denoising process to reconstruct the original data.

In robotic imitation learning, one core challenge is the multi-modality of action distributions: a single observation may correspond to multiple valid action sequences. Traditional methods, such as Behavioral Cloning (BC), often struggle with such distributions, which can

limit robustness and generalization. Diffusion models, by contrast, are well-suited to capture these complex, multimodal action distributions.

Diffusion Policy [3] was the first to apply diffusion models to robot policy learning. The key idea is to transform the denoising process into a conditional process, where the current observation serves as a condition. This approach enables the generation of high-dimensional, continuous action trajectories while effectively modeling multiple possible action outcomes.

**Integration of Force and Diffusion Policies:** To improve precision in contact-rich tasks, recent work has explored incorporating force feedback into diffusion policies. Two main strategies have been proposed:

- **Force as an Input:** Force feedback is provided as an input to guide the policy's actions. For example, DIPCOM [1] integrates force measurements to predict end-effector pose and arm stiffness.
- **Force as an Output:** Force is included in the predicted action, allowing the policy to directly control contact forces. ForceMimic [18] exemplifies this approach by predicting both continuous force and pose trajectories.

These approaches illustrate complementary ways to combine force information with diffusion-based policies, highlighting the flexibility of diffusion models in handling multimodal action distributions while incorporating physical interaction cues.

# Chapter 3
# Background

The theoretical foundation of our approach consists of imitation learning, Denoising Diffusion Probabilistic Models (DDPMs) [9], Diffusion Policy [3], and the neural network architectures used for visual encoding and action denoising. Following Diffusion Policy, visual observations are encoded using a ResNet-based visual encoder [8], while noisy action trajectories are denoised using a FiLM-conditioned one-dimensional U-Net-style temporal convolutional network. The U-Net architecture was originally introduced for biomedical image segmentation by Ronneberger et al. [24], and its encoder–decoder structure with skip connections has since been widely adopted in diffusion-based generative models.

The term *multimodal* is used in two ways in this thesis. For the observation space, it refers to combining different input modalities, such as RGB images, robot joint states, and raw joint torque measurements. For diffusion policy, it refers to the existence of multiple valid action trajectories for the same observed task state. Thus, our method uses multimodal observations to learn a policy for a task whose demonstrations may also contain multimodal action behavior.

## 3.1 Imitation Learning Problem Formulation

In this dual-arm robotic granular pouring task, we formulate the problem as an imitation learning problem. The goal of imitation learning is to learn a policy $\pi(\mathbf{a}_t \mid \mathbf{o}_t)$ that predicts actions conditioned on the current observation $\mathbf{o}_t$ to imitate expert behavior.

A demonstration trajectory consists of a sequence of observations and actions over time, denoted as $\boldsymbol{\tau} = (\mathbf{o}_0, \mathbf{a}_0, \mathbf{o}_1, \mathbf{a}_1, \dots)$, and we collect $N$ such demonstrations to form a dataset $\mathcal{D} = \{\boldsymbol{\tau}^i\}_{i=1}^{N}$.

In this thesis, the observation vector $\mathbf{o}_t$ represents the real-time observation of the environment, integrating visual information, robot joint states, and raw joint torque measurements. The process and format of $\mathbf{o}_t$ are described in Section 4.2. The action vector $\mathbf{a}_t$, as defined in this work, corresponds to the absolute joint positions of the robot arms.

## 3.2 Diffusion Models for Policy Learning

The diffusion policy framework is well-suited for predicting continuous action sequences, a requirement essential for achieving the high precision needed in our granular pouring task. This section is structured to first introduce the theoretical foundation through the forward noising process and the reverse denoising process, followed by an explanation of how the diffusion model is adapted to predict action trajectories in robot learning.

### 3.2.1 The Forward Diffusion Process

The forward diffusion process [9] is a noise-adding procedure that forms a Markov chain. In the context of diffusion policy, the data sample $\mathbf{A}_t$ corresponds to a clean action trajectory drawn from expert demonstrations. Its core idea is to gradually and systematically destroy the structure present in the original data distribution so that any data sample $\mathbf{A}_t$ is eventually transformed into isotropic Gaussian noise $\mathbf{A}_t^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

This diffusion procedure is defined over $K$ time steps, where in the original DDPM work [9] $T$ was set to 1000, and for diffusion policies in robotics, $K$ is typically chosen smaller depending on the task complexity [3]. The transition from step $k-1$ to $k$ follows the distribution

$$q(\mathbf{A}_t^k \mid \mathbf{A}_t^{k-1}) = \mathcal{N}\left(\mathbf{A}_t^k;\ \sqrt{1-\beta_k}\,\mathbf{A}_t^{k-1},\ \beta_k \mathbf{I}\right), \tag{3.1}$$

where $\beta_k \in (0, 1)$ is a noise-schedule hyperparameter controlling the variance of the Gaussian noise injected at each step. In this formulation, the mean of the Gaussian is given by $\sqrt{1-\beta_k}\,\mathbf{A}_t^{k-1}$, and the covariance is $\beta_k \mathbf{I}$.

Although sampling $\mathbf{A}_t^k$ from $\mathbf{A}_t$ appears to require $k$ sequential transitions, the reparameterization trick [9] yields a closed-form expression for the marginal distribution $q(\mathbf{A}_t^k \mid \mathbf{A}_t)$:

$$q(\mathbf{A}_t^k \mid \mathbf{A}_t) = \mathcal{N}\left(\mathbf{A}_t^k;\ \sqrt{\bar{\alpha}_k}\,\mathbf{A}_t,\ (1-\bar{\alpha}_k)\mathbf{I}\right), \tag{3.2}$$

where $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$.

This closed-form result enables direct sampling at an arbitrary time step during training, and expresses $\mathbf{A}_t^k$ as a weighted combination of the clean data $\mathbf{A}_t$ and pure Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$:

$$\mathbf{A}_t^k = \sqrt{\bar{\alpha}_k}\,\mathbf{A}_t + \sqrt{1-\bar{\alpha}_k}\,\boldsymbol{\epsilon}. \tag{3.3}$$

This formulation is fundamental to the training phase of diffusion policy, as it provides a direct mathematical relationship between clean data and its noisy counterpart.

### 3.2.2 The Reverse Denoising Process

The reverse denoising process is the core of diffusion models. Its goal is to gradually transform pure Gaussian noise back into a sample from the target data distribution $\mathbf{A}_t$ over $K$ time steps.

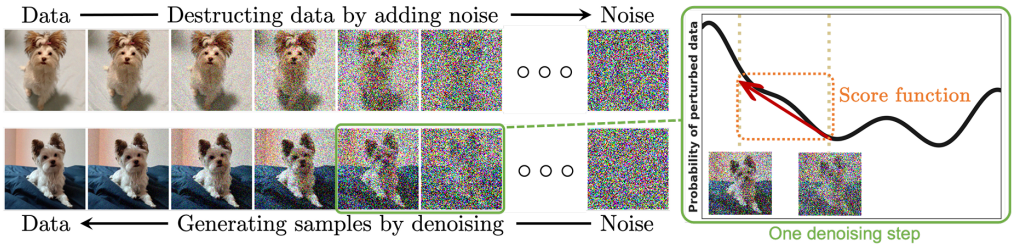



**Figure 3.1:** Forward and reverse process of diffusion, picture from [30]

Core of this procedure is a deep neural network $\boldsymbol{\epsilon}_\theta$, that predicts the noise component that was added to the current noisy action trajectory $\mathbf{A}_t^k$. Given $\mathbf{A}_t^k$ and its corresponding diffusion step $k$, the network outputs an estimate $\boldsymbol{\epsilon}_\theta(\mathbf{A}_t^k, k)$ of the injected noise.

Once this noise estimate is obtained, we can use the known parameters from the forward diffusion process, such as $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k$—along with a covariance term $\Sigma_\theta$ (typically fixed as $\sigma_k^2\mathbf{I}$), to compute a cleaner sample $\mathbf{A}_t^{k-1}$.

The reverse transition distribution $p_\theta(\mathbf{A}_t^{k-1} \mid \mathbf{A}_t^k)$ is modeled as a Gaussian:

$$p_\theta(\mathbf{A}_t^{k-1} \mid \mathbf{A}_t^k) = \mathcal{N}\Big(\mathbf{A}_t^{k-1};\, \mu_\theta(\mathbf{A}_t^k, k),\, \Sigma_\theta(\mathbf{A}_t^k, k)\Big), \tag{3.4}$$

where the mean $\mu_\theta(\mathbf{A}_t^k, k)$ is derived from the predicted noise $\boldsymbol{\epsilon}_\theta$, and the variance $\Sigma_\theta$ is usually a pre-defined hyperparameter.

The final sampling equation for obtaining $\mathbf{A}_{t-1}$ from $\mathbf{A}_t$ is:

$$\mathbf{A}_t^{k-1} = \frac{1}{\sqrt{\alpha_k}}\left(\mathbf{A}_t^k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\boldsymbol{\epsilon}_\theta(\mathbf{A}_t^k, k)\right) + \sigma_k\boldsymbol{\epsilon}, \tag{3.5}$$

where $\boldsymbol{\epsilon}$ is standard Gaussian noise. The first term yields the estimated denoised sample, while the second term injects stochasticity to encourage sample diversity. By iterating this reverse denoising step from $k = K$ down to $k = 1$, the model reconstructs a clean data sample from pure noise.

### 3.2.3 Diffusion Policy

A diffusion policy leverages the intrinsic capability of diffusion models to represent complex multi-modal conditional distributions. In robotic imitation learning, a major challenge lies in the inherently multi-modal nature of human demonstrations [6] – there are often many valid ways to accomplish the same task. A diffusion policy naturally models such complex, multi-modal conditional distributions $p(\mathbf{A}_t \mid \mathbf{O}_t)$, where $\mathbf{A}_t$ denotes the action sequence and $\mathbf{O}_t$ represents the observation, as illustrated in Figure 3.2.

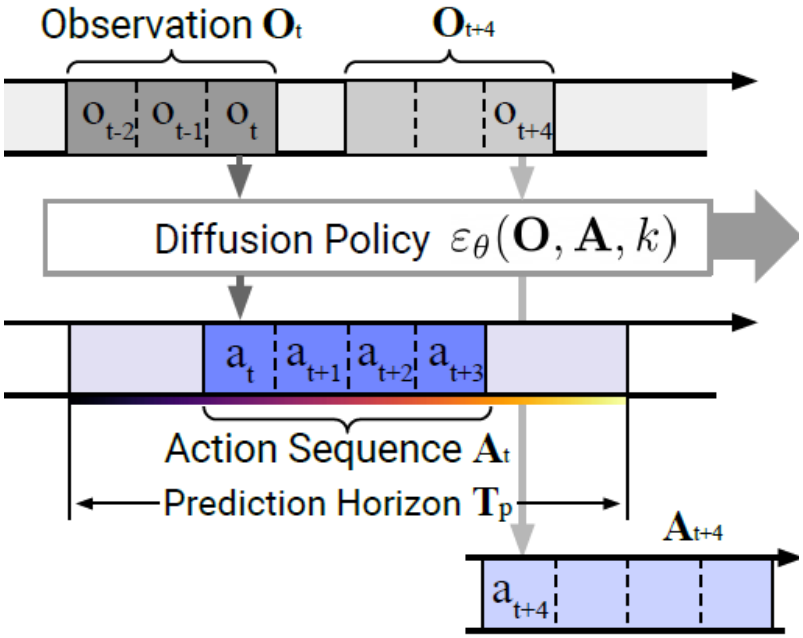



**Figure 3.2:** Architecture of diffusion policy, picture from [3]

The key to implementing a diffusion policy is to incorporate observation information as conditioning throughout the denoising process. To achieve this, we concatenate the robot's current joint state, visual observations, and force feedback to construct the full conditioning vector:

$$\mathbf{o}_t = \text{Concat}(\mathbf{s}_t^{\text{joint}}, \mathbf{s}_t^{\text{visual}}, \mathbf{s}_t^{\text{torque}}), \tag{3.6}$$

where $\mathbf{s}_t^{\text{joint}}$ denotes the robot joint state at time $t$, $\mathbf{s}_t^{\text{visual}}$ represents the visual observation encoded from raw images using a convolutional encoder, and $\mathbf{s}_t^{\text{torque}}$ corresponds to the raw joint torque measurements obtained from the robot joints. These multi-modal features are normalized and adaptively injected into the denoising network via FiLM conditioning, as detailed in Section 3.4.2.

During training, the policy network $\boldsymbol{\epsilon}_\theta$ learns to predict the noise given $\mathbf{A}_t^k$, the timestep $t$, and the conditioning vector $\mathbf{O}_t$. This learned network then drives the reverse denoising process by parameterizing the conditional distribution $p_\theta(\mathbf{A}_t^{k-1} \mid \mathbf{A}_t^k, \mathbf{O}_t)$, enabling the model to iteratively sample cleaner action sequence $\mathbf{A}_t^{k-1}$ from $\mathbf{A}_t^k$.

### 3.2.4 Key Parameters for Diffusion Policy

When applying Diffusion Policy to robotic control, its performance and runtime behavior are largely determined by several key hyperparameters, which are essential for ensuring coherent and real-time action generation. In this section, we define the three core parameters used in the policy for this study:

**Observation Horizon ($T_o$):** This parameter specifies the number of past observation frames the model conditions on when generating an action sequence. The policy receives a historical observation window $\mathbf{O}_t = (\mathbf{o}_{t-T_o+1}, \ldots, \mathbf{o}_t)$ which enables it to capture temporal trends in the environment. A short $T_o$ improves responsiveness and reduces inference latency,

while a longer $T_o$ provides stronger temporal context for stabilizing motion [3]. Incorporating this history helps the policy smooth out artifacts such as brief pauses or hesitations that may exist in the expert demonstrations.

**Action Prediction Horizon ($T_p$):** This defines the length of the future action sequence predicted during a single denoising step. The policy is trained to output a full sequence of future actions $\mathbf{A}_t = (\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+T_p-1})$, allowing it to perform long-horizon reasoning rather than reacting in a single-step manner. In the Diffusion Policy framework, a typical choice is $T_p$=16 steps [3]. A well-chosen horizon is essential for ensuring the smoothness and consistency of the executed motions.

**Inference Denoising Steps (K):** During inference, the model generates a clean action sequence through a reverse denoising process. The parameter $K$ determines how many denoising steps are performed and thus directly influences the real-time performance of the policy. A larger K improves action accuracy but increases latency, while a smaller K speeds up inference at the cost of precision [3].

## 3.3 Visual Feature Extraction: ResNet

In imitation learning, visual observations constitute a primary source of information, making the extraction of robust semantic features from high-dimensional images a central requirement. Following the standard architecture in diffusion policy, we employ a Residual Network (ResNet) [8] as the visual encoder. The encoder transforms camera observations into low-dimensional feature vectors, which provide essential visual conditioning for the denoising process in the diffusion policy.

### 3.3.1 Residual Learning Principle

The core of ResNet lies in its residual block design and the introduction of skip connections. Instead of learning a full mapping $H(x)$ directly, a residual block learns the difference between the desired output and the input, expressed as:

$$H(x) = F(x) + x, \tag{3.7}$$

where $F(x)$ denotes the residual function.

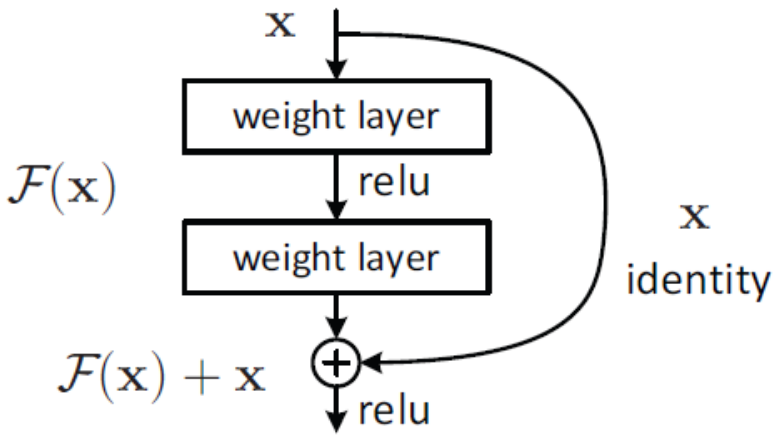



**Figure 3.3:** Architecture of resnet, picture from [8]

These skip connections significantly alleviate the vanishing gradient problem encountered in deep neural networks, enabling the construction of substantially deeper architectures while preserving stable optimization. As a result, ResNet achieves higher-quality feature extraction compared with conventional feedforward networks.

## 3.4 Denoising Network: UNet

The UNet [24] architecture was originally designed as a fully convolutional neural network for image semantic segmentation. Its distinctive design enables effective capture of multi-scale features while preserving spatial information, which is crucial in image processing tasks. In diffusion policy, we adopt the conceptual design of UNet to build a temporal convolutional network for one-dimensional action sequences. This network is used to extract features from noisy action trajectories and perform denoising predictions.

### 3.4.1 UNet Architecture

The UNet architecture consists of a symmetric encoder and decoder. The encoder extracts high-level semantic features through a series of convolutional layers and max-pooling down-sampling operations. The decoder then progressively restores the original data dimensions through up-sampling. UNet is primarily used for image semantic segmentation and outputs a segmentation mask.

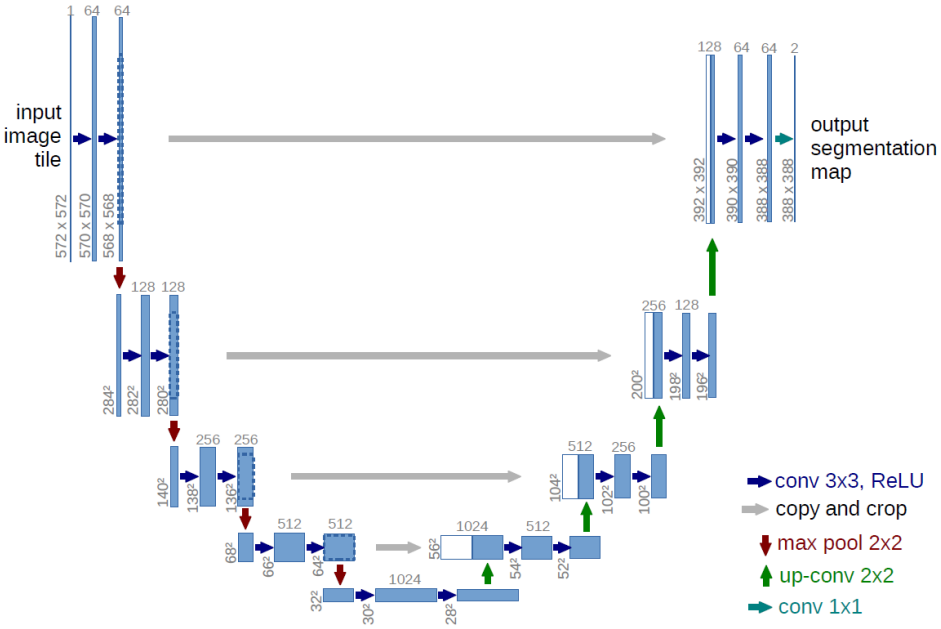



**Figure 3.4:** Architecture of UNet, picture from [24]

A key design feature of UNet is the skip connections, which directly feed features from different encoder layers into their corresponding decoder layers. This mechanism addresses the loss of fine details, such as edges and textures, that can occur during the down-sampling process of the encoder.

### 3.4.2 Application in Diffusion Policy

In the diffusion policy framework, the conditional noise prediction network used during the denoising phase is constructed based on the UNet architecture. Specifically, diffusion policy employs a 1D Temporal CNN with a UNet-like structure as the backbone. This design allows the network to inherit UNet's ability to capture long-range dependencies and preserve fine structural details. As a result, the generated action trajectories exhibit smoother and more coherent temporal patterns.

A key distinction between diffusion policy's denoising phase and that of the original diffusion models lies in the incorporation of multi-modal observation information as conditioning. This conditioning is injected into the denoising process through Feature-wise Linear Modulation (FiLM) [22].

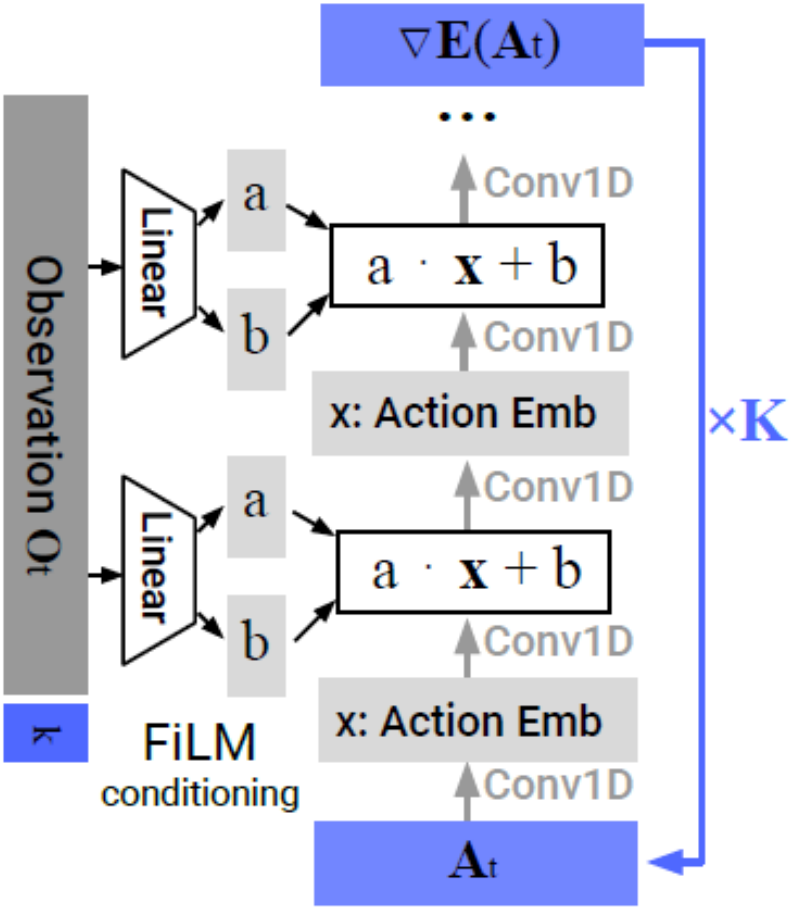



**Figure 3.5:** Architecture of CNN-based Diffusion Policy, picture from [3]

Figure 3.5 provides a detailed illustration of the FiLM-conditioned denoising process used in diffusion policy during inference. At each reverse diffusion step, the network takes as input a noisy action sequence $\mathbf{A_t}$, which is first mapped to an action embedding and processed through a series of one-dimensional convolutional layers.

As shown in the figure, the intermediate action features $x$ at each UNet layer are modulated using Feature-wise Linear Modulation (FiLM). The FiLM parameters, consisting of a scale $a$ and a shift $b$, are generated by applying linear projections to the current observation $\mathbf{O}_t$. The resulting affine transformation $a \cdot x + b$ enables the denoising network to adapt its intermediate representations based on the current multi-modal observation.

The UNet ultimately outputs a prediction of the noise, denoted as $\nabla E(\mathbf{A}_t)$, which is used to update the action sequence for the next denoising step. This FiLM-conditioned UNet denoising procedure is repeated for $K$ inference denoising steps, as indicated by the $\times K$ notation in the figure, progressively refining the action sequence from an initial noisy trajectory into a clean and executable action plan.

In practice, the FiLM-conditioned UNet described above follows the standard inference implementation of diffusion policy as provided in the official open-source implementation.

The number of inference denoising steps $K$ is treated as a hyperparameter and is chosen to match the settings used in the original diffusion policy formulation [3].

## 3.5 Real Robot Application

### 3.5.1 System-Level Execution Framework

Executing diffusion-based policies on physical robots requires a system-level framework that can coordinate policy inference, sensor feedback, and actuator commands in real time. In particular, the action trajectories generated by the diffusion policy must be transmitted reliably to the robot while maintaining synchronization with high-frequency sensor observations.

ROS2 is commonly used in robotics to support such requirements, as it provides a distributed communication infrastructure for integrating perception, learning, and control components within a unified system. ROS 2 is built on DDS and provides native real-time support with deterministic execution APIs, enabling bounded latency and reliable communication for closed-loop robotic control [19]. In this work, ROS2 serves as the execution backbone that connects the diffusion policy inference module with the robot control stack.

### 3.5.2 ROS2 Control for Policy Execution

While ROS2 enables communication between software components, the execution of continuous action trajectories on physical hardware requires a structured control framework that enforces hardware abstraction and real-time safety constraints. ROS2 Control provides such a framework by standardizing the interface between high-level control policies and low-level robot drivers [19].

Within this architecture, the diffusion policy outputs joint-space action trajectories, which are consumed by trajectory-based controllers implemented in ROS2 Control [19]. The hardware abstraction layer exposes a unified read–write interface, allowing controllers to access sensor states and issue actuator commands independently of the underlying robot hardware. A centralized controller manager coordinates controller lifecycles and resource allocation [19], ensuring safe and deterministic execution of learned policies.

# Chapter 4

# Method

## 4.1 Existing Setup and Thesis-Specific Contributions

Before describing the data collection and policy pipeline, we distinguish between the existing laboratory infrastructure, previously developed teleoperation software, and the work carried out in this thesis. The physical dual-arm KUKA LBR iiwa platform and the basic ROS 2 communication infrastructure were already available in the laboratory. In addition, the thesis builds on the existing Quest2ROS2 teleoperation codebase, which provides a ROS 2 framework for bi-manual VR teleoperation [15]. We did not develop the full VR teleoperation framework from scratch.

The thesis-specific work consisted of adapting the existing Quest2ROS2-based teleoperation code to a complete demonstration-collection workflow for the granular pouring task. This included designing and integrating the camera setup, implementing the camera acquisition and synchronization code, designing the gripper setup for holding the pouring containers, extending the data recording pipeline, and collecting synchronized demonstrations. The recorded dataset includes multi-view RGB observations, dual-arm joint states, gripper commands, raw joint torque measurements, and estimated end-effector wrench signals. These data were then used to train and evaluate the diffusion-policy variants studied in this thesis.

Therefore, the system contribution of this thesis is best described as the development of a task-specific data collection and evaluation pipeline built on top of existing laboratory hardware and Quest2ROS2-based teleoperation software, rather than the development of a new VR teleoperation framework from scratch.

## 4.2 Data Collection Method

To enable the diffusion policy to effectively model the multi-modal action distribution required for the granular pouring task, we design a VR-teleoperated dual-arm robotic system for expert demonstration collection. The VR teleoperation component builds on the existing Quest2ROS2 codebase [15], which extends earlier Quest2ROS-style VR teleoperation ideas to bi-manual teleoperation in ROS 2. In this thesis, we adapted this codebase into a task-specific data collection workflow for dual-arm granular pouring.

### 4.2.1 VR-based Teleoperation System

Demonstrations are collected using a virtual reality (VR)-based teleoperation system built upon an Oculus Quest 2 headset and two handheld controllers. The teleoperation software builds on the existing Quest2ROS2 codebase [15]. We did not implement the full VR-to-ROS 2 teleoperation stack from scratch. Instead, our work adapted the existing teleoperation code to the dual-arm granular pouring task and integrated it into a complete data collection workflow.

The human operator controls the dual-arm robot to perform the pouring task through intuitive hand motions. When the operator presses a designated button on the controller, a local virtual coordinate frame is initialized. Subsequent controller motions are recorded as relative poses with respect to this frame and are mapped in real time to the robot end-effectors. This relative pose control strategy enables stable and precise manipulation while reducing operator fatigue.

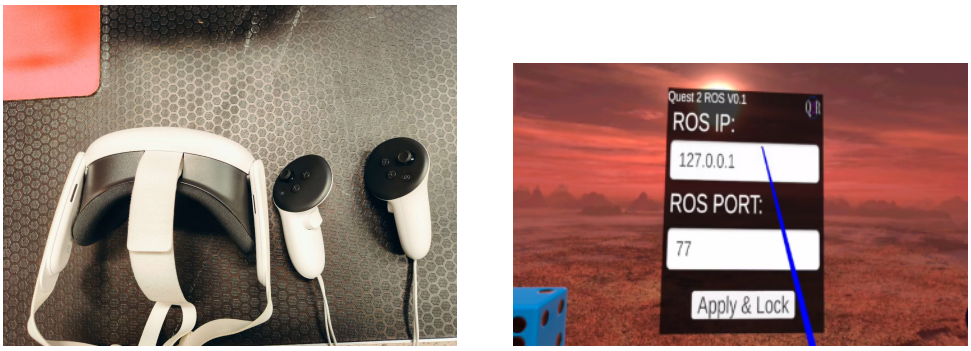



**Figure 4.1:** Left: Oculus Quest 2 headset and two controllers used for VR-based teleoperation. Right: User interface of the Quest2ROS software.

### 4.2.2 Robotic Platform and End-Effectors

The demonstrations are executed on a KUKA LBR iiwa 7 R800 dual-arm platform, with each 7-DoF arm providing a 7 kg payload capacity. Although all seven joints are equipped with integrated torque sensors, the raw joint torques are processed via the KUKA software into a 6-axis Cartesian end-effector wrench, consisting of 3D force and 3D torque components expressed relative to the robot base frame. Although KUKA software provides an estimated

6-axis end-effector wrench derived from joint torque measurements, raw joint torque signals are used as the primary force-related input for the policy in our implementation.

Each arm is equipped with a Robotiq Hand-E adaptive gripper. To accommodate the large laboratory beakers, we designed custom 3D-printed fingers (see Figure 4.3) that extend the maximum opening range, ensuring stable and secure grasps throughout the demonstrations.

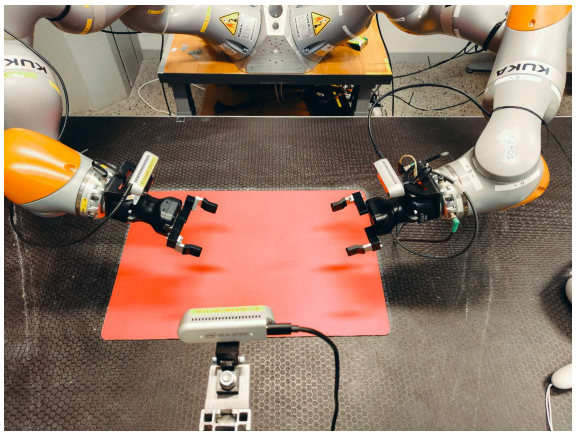



**Figure 4.2:** Dual-arm robotic workspace based on the KUKA LBR iiwa platform.

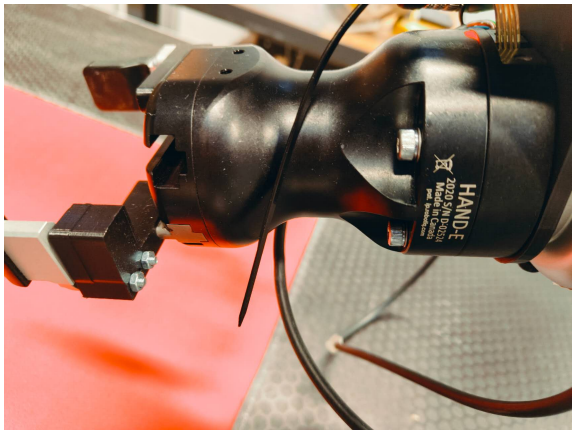



**Figure 4.3:** Robotiq Hand-E gripper with custom 3D-printed fingers.

### 4.2.3 Vision System

Visual observations are captured using three Intel RealSense cameras. Two RealSense D435 cameras are mounted on the wrists of the dual-arm robot, providing egocentric views of the manipulation. An additional RealSense D435i camera is fixed in the workspace to provide a global view of the task environment.

All cameras operate at a resolution of $640 \times 480$ pixels and a frame rate of approximately 30 FPS. Although the RealSense cameras can provide depth measurements, the policies in

this thesis use RGB observations only as visual input. This keeps the experimental comparison focused on camera-view configuration and torque feedback, while RGB-D fusion is left as a direction for future work. In this work, the visual input to the policy consists of multi-view RGB images captured from a fixed top-view camera positioned above the workspace and two eye-in-hand cameras mounted on the respective robotic end-effectors. This configuration provides both a global context of the pouring scene and localized, high-resolution visual feedback for precise contact manipulation as shown in Figure 4.4.

**(a)** Top-view camera

**(b)** Right arm camera

**(c)** Left arm camera

**Figure 4.4:** Multi-view observations used as policy input, including a global top-view and two eye-in-hand perspectives.

### 4.2.4 State and Action Representation

During data collection, the system records synchronized state–action pairs at each control timestep. The recorded data define both the action space and the conditioning inputs used by the diffusion policy.

**Action State.** The action is represented by the joint angles of the two 7-DoF robotic arms together with the gripper commands, resulting in a 16-dimensional action vector.

**Force Feedback.** To capture interaction-related information during pouring, we record joint torque measurements from all 14 joints (7 joints per arm). The joint torque signals are obtained from the integrated sensors of the KUKA LBR iiwa robot.

**Visual State.** Visual observations are captured using the camera system described in Section 4.2.3. Depending on the policy variant, the visual input consists of either a single top-view RGB image or three synchronized RGB images from the top-view and two wrist-mounted cameras.

### 4.2.5 Pouring Medium Selection

Directly collecting demonstrations with real liquids such as water introduces potential safety risks to the robotic platform and the surrounding environment, especially during early-stage data collection. To address this challenge, we adopt a proxy medium strategy, drawing on methodologies that utilize diverse materials to validate pouring tasks [10].

In the initial stage, rice is used as a surrogate pouring medium for collecting expert demonstrations and evaluating baseline policies. Rather than treating rice as a full physical substitute for liquid, this thesis uses it as a safe and repeatable granular proxy for studying

selected pouring-related aspects, such as material transfer, container alignment, flow initiation, and temporal coordination. Although rice does not reproduce liquid-specific properties such as viscosity, surface tension, sloshing, or fluid inertia, it still provides a controlled setting for evaluating whether force-related observations can help the policy reason about task progression during pouring.

When transitioning from rice to water, the system encounters different dynamics, such as increased flow momentum and surface tension effects. However, the underlying control logic that maps force-torque changes to motions is expected to remain consistent across different media. This consistency between granular and liquid materials has been observed in the cross-material experiments of prior research [10]. By leveraging this documented similarity, we validate our force-feedback framework's core logic on rice before evaluating its potential generalization to real-world liquid conditions in future stages.

## 4.3 Force Feedback Processing

During data collection, joint torque measurements from all 14 joints (7 joints per arm) are recorded. The KUKA LBR iiwa internally provides an estimated end-effector wrench based on the measured joint torque signals through its built-in dynamics and control software. The estimated end-effector wrench is then published to ROS 2 via the Force Torque Sensor Broadcaster interface, a standard ROS 2 component that exposes force and torque information as ROS messages.

The calculation process is achieved in two primary steps: first, the raw joint torque ($\boldsymbol{\tau}_{raw}$) is compensated for the gravity torque ($\boldsymbol{\tau}_{grav}$) using a calibrated load data procedure ($\boldsymbol{\tau}_{\text{compensated}} \approx \boldsymbol{\tau}_{raw} - \boldsymbol{\tau}_{grav}$). Second, the resulting compensated torque is then transformed into the end-effector wrench ($\mathbf{F}$) in the Cartesian space by utilizing the inverse mapping of the robot's Jacobian matrix ($\mathbf{J}^T$) [17].

Our initial motivation for investigating the estimated wrench information was to explore whether a physically grounded signal correlated with pouring progress could be derived from the robot's internal force-related measurements. The most intuitive signal is the combined mass of the cup and the remaining liquid. Therefore, our initial approach was to estimate the change in liquid mass by computing variations in the estimated end-effector wrench.

If the estimated granular material mass variation proved to have a significant error compared to the actual change, we would then attempt the secondary strategy: directly feeding the raw joint torque measurements as one of the inputs to the diffusion policy, expecting the policy to implicitly learn the correlation between the raw joint torque measurements and the pouring progress.

The following subsections describe these two approaches in detail.

### 4.3.1 Weight Estimation from the Estimated End-Effector Wrench

The estimated end-effector wrench used in this section was not implemented from scratch in this thesis. It is provided by the KUKA robot software based on the robot's internal sensing and dynamic model. Since the proprietary implementation is not fully accessible, we treat

this signal as a black-box estimate rather than as a ground-truth force/torque measurement. Our contribution in this section is therefore not the implementation of a new wrench estimator, but an empirical evaluation of whether the available estimated wrench signal can be used to infer pouring progress through mass-change estimation.

Because the pouring demonstrations were executed with deliberately slow teleoperated motions, we assume that the quasi-static approximation is reasonable for this exploratory analysis. Nevertheless, the approximation is not exact, especially during the active rotation of the container, and residual dynamic effects may still contribute to the estimation error.

In our initial approach, we attempted to estimate the change in mass of the liquid in the container based on variations in the estimated end-effector wrench.

The idea was to use changes in the estimated end-effector wrench to compute the amount of liquid poured over time. However, accurate calculation requires knowledge of the liquid's center of mass to correctly determine the lever arm in the wrench-based estimation process. Because the center of mass of the liquid is generally unknown and constantly changing during pouring, the resulting weight estimation proved to be unreliable.

Consequently, while this approach provided a rough indication of the liquid mass change, it was not sufficiently accurate for use in the final deployed diffusion policy.

### 4.3.2 Normalized Pouring Ratio Derived from the Estimated End-Effector Wrench

The method described above produces a time-varying estimate of the total weight of the object $W_{\text{total}}$. To derive a more interpretable quantity from the estimated end-effector wrench, we compute a Normalized Pouring Progress Ratio ($\rho$) as an exploratory pouring-progress indicator:

$$\rho = \frac{W_{\text{total}}}{W_{\text{container}} + W_{\text{liquid, initial}}} \tag{4.1}$$

where $W_{\text{container}}$ is the known container mass, and $W_{\text{liquid, initial}}$ is the mass of the liquid measured right after successful grasping (the initial maximum liquid mass for the current run). This ratio $\rho$ is designed to explicitly quantify the task's progress:

- **Grasp Confirmation ($\rho \approx 1$):** If the robot successfully grasps the container, $\rho$ is close to 1. If grasping fails, $\rho$ approaches 0, potentially allowing the policy to correct Failure Mode 1.
- **Progress Tracking ($\rho \in [\rho_{\text{min}}, 1]$):** During pouring, $\rho$ smoothly decreases, providing an unambiguous physical cue to track the volume reduction.
- **Completion Signal ($\rho \approx \rho_{\text{min}}$):** When $\rho$ stabilizes near its minimum value $\rho_{\text{min}} = \frac{W_{\text{container}}}{W_{\text{container}}+W_{\text{liquid, initial}}}$, it could potentially signal to the policy that the container is empty. This prevents the robot from stalling in the pouring posture (Failure Mode 2).

Although the normalized pouring ratio $\rho$ derived from the estimated end-effector wrench provided an interpretable indication of pouring progress, the instability of the underlying wrench estimation limited its reliability during dynamic pouring motions. Consequently, the estimated end-effector wrench and its derived quantities were not adopted in the final deployed policy.

### 4.3.3 Final Approach: Raw Joint Torque Measurements as Policy Input

The use of force-related information as policy input is related to recent work on force-aware imitation learning and force-conditioned diffusion policies. Force-conditioned diffusion policies have been used for compliant bimanual manipulation tasks [26], and DIPCOM incorporates force measurements into a diffusion-policy-based framework for compliant contact-rich manipulation [1]. ForceSight combines visual, force, and proprioceptive information to predict force-aware manipulation goals [4], while ForceMimic learns force-centric imitation policies using a force-motion capture system for contact-rich manipulation [18]. These works show that force-related signals can improve contact-rich manipulation, but they commonly rely on external force/torque sensing, force-motion capture, or explicitly modeled force targets.

In contrast, this thesis uses raw internal joint torque measurements from the KUKA LBR iiwa as force-related proprioceptive observations. This avoids adding external force/torque sensors, but the signal is noisier and contains both robot dynamics and interaction effects. Therefore, we do not attempt to reconstruct exact contact forces from the joint torques. Instead, we provide the raw joint torque vector directly to the diffusion policy and evaluate whether the learned policy can use this signal to improve task progression.

A gravity-compensated or residual torque representation could in principle provide a more direct estimate of external interaction effects, because it attempts to remove part of the torque contribution caused by robot gravity and internal dynamics. Such a representation was considered as a possible alternative during the design of the force-related input. However, we did not train an additional diffusion policy using gravity-compensated torques in this thesis. The available compensated wrench signal was treated as a black-box estimate from the robot software and showed limited reliability in the mass-estimation analysis described in Section 4.3.1. Therefore, the final deployed policy uses raw joint torque measurements as a minimally processed proprioceptive signal.

This choice avoids relying on an additional model-based preprocessing step whose assumptions and proprietary implementation are not fully accessible. At the same time, it also means that the torque input contains both task-relevant interaction information and internal dynamic effects. Consequently, the results should be interpreted as evidence that raw joint torque observations can improve task progression compared with the vision-only baseline, rather than as evidence that raw torques are superior to gravity-compensated or residual torque representations.

In the final approach, rather than relying on the estimated end-effector wrench and its derived quantities, the final deployed policy directly incorporates raw joint torque measurements as an additional input modality.

The motivation behind this design is to provide the policy with direct access to the raw joint torque measurements generated during robot interaction and manipulation. Raw joint torque measurements inherently reflect both the robot's internal dynamics (e.g., gravity and inertia) and external influences, such as variations in load during grasping and pouring. By leveraging these signals, the policy can learn correlations between raw joint torque measurements and task progression, without relying on explicit physical modeling.

The raw joint torque measurements, denoted as $\mathbf{s}_t^{\text{torque}}$, correspond to the raw joint torque measurements recorded at time step $t$. Specifically, they are defined directly from the raw

joint torque measurement vector $\boldsymbol{\tau}_{\text{raw},t} \in \mathbb{R}^{14}$ as:

$$\mathbf{s}_t^{\text{torque}} = \boldsymbol{\tau}_{\text{raw},t} = [\tau_1, \tau_2, \ldots, \tau_{14}], \tag{4.2}$$

where $\tau_i$ denotes the $i$-th component of $\boldsymbol{\tau}_{\text{raw},t}$. For a dual-arm system with two 7-DoF manipulators, $\tau_1$–$\tau_7$ correspond to the joint torques of the first arm, and $\tau_8$–$\tau_{14}$ correspond to those of the second arm.

These raw joint torque measurements capture the instantaneous actuator torques required to maintain or change joint configurations. As such, they contain information related to both internal dynamic effects and external interactions, including load variations that arise during manipulation tasks such as grasping and pouring. The raw joint torque measurements from all 14 joints constitute the final force-related input representation used in the deployed diffusion policy.

## 4.4 Torque-Conditioned Diffusion Policy Implementation

This section connects the diffusion-policy background described in Chapter 3 to the task-specific implementation used in this thesis. All policy variants follow the same conditional diffusion-policy framework described in Section 3.4.2. The differences between the variants are limited to the observation modalities provided to the policy.

For the force-integrated multi-view policy, the observation at time step $t$ consists of visual observations, low-dimensional robot state, and raw joint torque measurements:

$$\mathbf{o}_t = \text{Concat}\left(\mathbf{s}_t^{\text{visual}}, \mathbf{s}_t^{\text{robot}}, \mathbf{s}_t^{\text{torque}}\right). \tag{4.3}$$

Here, $\mathbf{s}_t^{\text{visual}}$ denotes the visual feature representation extracted from the RGB camera observations, $\mathbf{s}_t^{\text{robot}}$ denotes the low-dimensional robot state including the dual-arm joint positions and gripper commands, and $\mathbf{s}_t^{\text{torque}} \in \mathbb{R}^{14}$ denotes the raw joint torque vector from the two 7-DoF KUKA arms.

The single-view baseline uses only the fixed top-view RGB image together with the robot state. The multi-view baseline uses three RGB camera views together with the robot state. The force-integrated multi-view policy uses the same multi-view visual input and robot state as the multi-view baseline, but additionally appends the 14-dimensional raw joint torque vector to the low-dimensional observation.

The torque input is not processed by a separate force encoder. Instead, it is treated as an additional proprioceptive modality and concatenated with the other low-dimensional state variables before being used as conditioning information for the FiLM-conditioned temporal UNet denoising network. This design keeps the architecture close to the standard diffusion-policy implementation while allowing the ablation study to isolate the effect of adding raw joint torque observations.

To avoid numerical scale imbalance between joint positions, gripper commands, and torque measurements, the low-dimensional observation components are normalized using statistics computed from the training demonstrations. The same normalization parameters are used during real-robot deployment. This is particularly important for the torque dimensions, whose numerical range differs from the range of joint angles and gripper commands.

No torque-specific network module was introduced, and the visual encoder, temporal UNet denoising backbone, observation horizon, action prediction horizon, and inference denoising steps were kept the same across the compared policy variants. Therefore, the main implementation difference between the multi-view baseline and the force-integrated multi-view policy is the inclusion of the raw 14-dimensional joint torque vector in the conditioning observation. This controlled design allows the experiment to evaluate whether adding torque observations improves task progression and full-task success under otherwise comparable training and evaluation settings.

## 4.5 ROS 2-Based Communication Framework

The distributed ROS 2 communication architecture used in this thesis builds on previous laboratory work carried out by Wenrui Xie and Jialong Li. Although this earlier design has not been published as a separate paper, it formed the basis for the system architecture used in this thesis. In particular, the separation between robot control, visual sensing, and GPU-based policy inference follows this existing internal design. In this thesis, we reused and adapted the architecture for torque-aware demonstration collection, synchronized data recording, and diffusion-policy execution. To support real-time teleoperation, synchronized data collection, and closed-loop policy execution, we design a distributed communication architecture based on ROS 2, as illustrated in Fig. 4.5. The system consists of three computing units: two robot-side modules (Robot Mind 1 and Robot Mind 2) and an upper-level workstation (Bokertov) responsible for neural policy inference.

Robot Mind 1 serves as the real-time control interface between the ROS 2 ecosystem and the proprietary controller of the KUKA LBR iiwa robot, which does not provide native open-source communication support. A high-frequency ROS 2 controller running on Robot Mind 1 sends joint position commands to the robot and receives joint position, velocity, and torque feedback. All joint-related states are published as ROS 2 topics, enabling downstream modules to access synchronized robot state information.

In addition to robot control, Robot Mind 1 integrates human teleoperation and gripper actuation. Cartesian pose data from the Oculus Quest 2 VR controllers are streamed through the Quest2ROS2 interface and converted into joint-space targets for teleoperated demonstrations. A ROS 2-based gripper controller is also deployed on Robot Mind 1, allowing execution of open and close commands during both demonstration and policy execution phases.

Robot Mind 2 is responsible for visual sensing and multimodal data synchronization. The Intel RealSense cameras publish RGB images at a resolution of $640 \times 480$ via ROS 2 topics. During the data collection phase, Robot Mind 2 subscribes to joint state information from Robot Mind 1, performs preprocessing, and stores temporally aligned image–state pairs as demonstration data.

During policy inference, Robot Mind 2 aggregates all preprocessed sensory inputs, including visual observations, joint states, and force-related signals, and transmits them to the upper-level workstation. The Bokertov workstation hosts the diffusion policy model and performs GPU-accelerated inference. Predicted joint position commands are published back to the ROS 2 network and subscribed to by Robot Mind 1, which applies them to the robot controller in real time, thereby closing the control loop.

Compared with an alternative architecture in which a single GPU-equipped computer is directly connected to the robot, the distributed architecture may introduce additional latency through ROS 2 message passing, image transmission, synchronization, and network communication between machines. This latency can affect real-time policy execution, especially for diffusion policies where inference already introduces computational delay through iterative denoising. However, the distributed design also provides practical advantages: it separates real-time robot control from GPU inference, allows the vision system and policy inference to run on dedicated machines, and reuses existing laboratory infrastructure. ROS 2 is designed for distributed robotic systems and provides a communication framework suitable for integrating perception, learning, and control components [19]. In this thesis, the communication latency of the architecture was not systematically measured. Therefore, we treat latency analysis as a limitation and rely on previous use of the architecture in the laboratory to support its practical suitability for real-robot teleoperation and learning experiments.

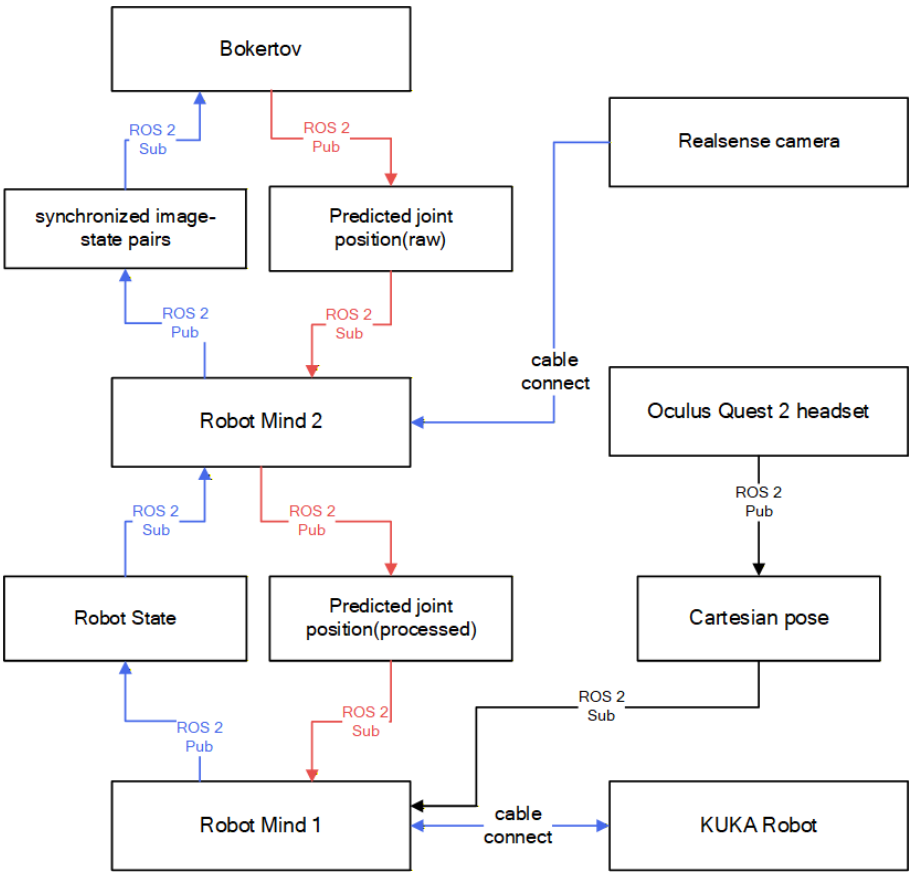



**Figure 4.5:** Overview of the distributed ROS 2 communication architecture used in our experimental setup. Black lines indicate connections present only during data collection, red lines indicate connections present only during policy inference, and blue lines indicate connections active in both phases.

# Chapter 5

# Experiments

The experimental evaluations aim to validate the diffusion policy's performance on the highly precise liquid pouring task, focusing specifically on how the integration of force feedback enhances the policy's robustness.

## 5.1 Data Collection and Analysis

Before starting the experiment, we manually collected 150 sets of trajectories for training. Each trajectory set included moving the left arm to the target position, grasping a cup, and moving it to the right arm's cup position for pouring. The data collected during this process were stored in Robot Mind 2, compressed into a dataset format, and sent to Bokertov for training.

The collected data was further analyzed. While joint positions and gripper commands are commonly utilized in imitation learning, this study focuses on joint torque measurements, which provides additional insight into physical interaction during manipulation. The following figure shows the variation of all joint torques over time.

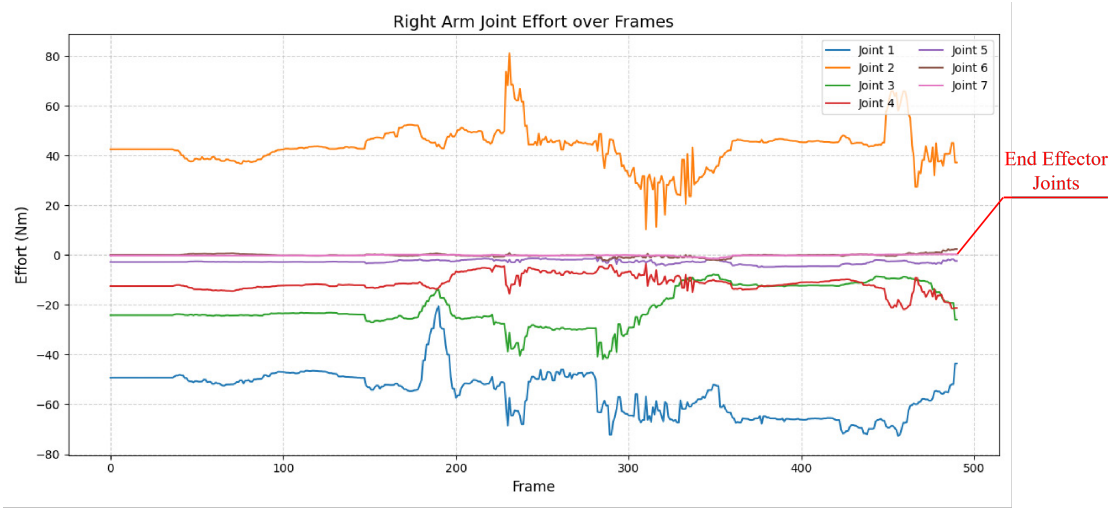



**Figure 5.1:** Right Arm Seven Joint torques Over Time

From Fig. 5.1, we observe that multiple joints reach their local extrema simultaneously at several time instances. These synchronized extrema provide useful *temporal cues* and can be exploited as key time labels for segmenting the task, regardless of the absolute magnitude of the joint torques.

The raw joint torque measurements of the robot arm are obtained from the integrated joint torque sensors, where the raw joint torque measurements consist of two components: (i) torques induced by the robot's own weight and internal dynamics, and (ii) torques resulting from external forces applied during task execution. For joints closer to the robot base, the torque contribution caused by the arm's self-weight dominates the measurement and is typically orders of magnitude larger than the torque induced by task-related external forces. Consequently, although joints closer to the robot base can capture some global temporal events and major task transitions, their torque signals are dominated by large background components, which makes them less sensitive to subtle, task-relevant force variations and may cause some important segmentation points to be missed.

In contrast, joints closer to the end-effector are less affected by gravity-induced torques and are physically closer to the point of interaction. Although their absolute torque magnitudes are significantly smaller—often by two orders of magnitude compared to the base joint—their torque signals contain richer task-related information. Therefore, different joints contribute differently to the task-relevant interaction information, with joints closer to the end-effector exhibiting more informative torque variations during pouring.

Based on this observation, we analyze the torque signal of the seventh joint separately. As shown in Fig. 5.2, three prominent peaks can be identified, corresponding to the onset of arm motion, the beginning of the pouring action, and the completion of the pouring task, respectively.

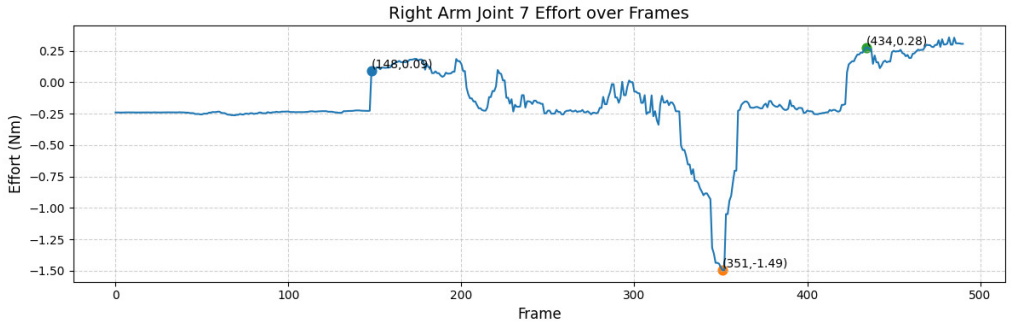
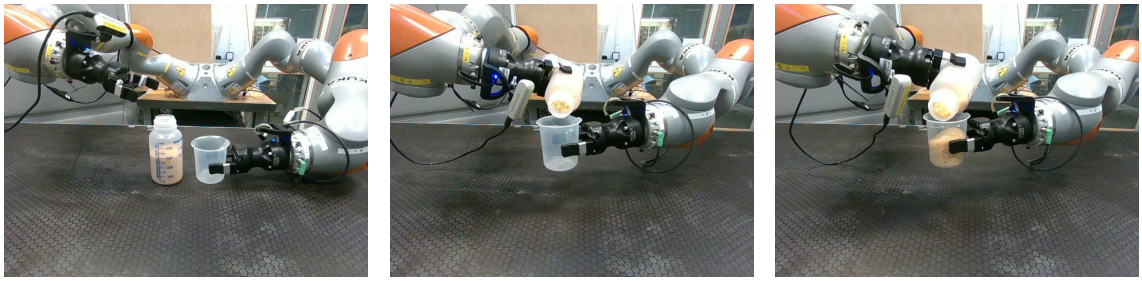



**Figure 5.2:** Torque signal of the seventh joint of the right arm over time. Three prominent peaks corresponding to key segmentation points are highlighted.



**(a)** Frame 148

**(b)** Frame 352

**(c)** Frame 434

The three peaks identified in Fig. 5.2 correspond to three key task events and are therefore treated as key segmentation points. Specifically, these points indicate the start of the task, the onset of the pouring action, and the completion of the pouring task, respectively.

An inspection of torque signals from other joints reveals that, at the third key point, the torque signal of the first joint also reaches a clear local maximum, suggesting that this global event is reflected across multiple joints. However, for the first and second key points, the torque variations in other joints are much less pronounced, and no consistent extrema can be reliably observed. This indicates that while some task boundaries may be captured by proximal joints, certain critical task transitions are primarily manifested in the torque signals of joints closer to the end-effector. These observations motivate further investigation of end-effector wrench estimation as an exploratory signal for task-state analysis.

## 5.2 Evaluation of Estimated End-Effector Wrench-Based Mass Estimation

In this experiment, we focus on evaluating the estimated end-effector wrench provided by the KUKA robot software. The software outputs a twelve-dimensional vector consisting of estimated forces and torques along the $x$, $y$, and $z$ directions for both end effectors. Due to the closed-source nature of the KUKA software, this estimated end-effector wrench module

functions as a black-box model from our perspective. Therefore, the estimated values cannot be treated as ground-truth data.

In practice, accurately updating gravity compensation models under frequently changing payloads remains a challenging problem [29]. This is particularly challenging in pouring tasks, where the liquid's mass and its center of mass shift dynamically, introducing unpredictable lever arms that render wrench-based estimation unreliable. Consequently, instead of attempting to calculate the exact torque contributions, we focus on the incremental change in the estimated force magnitude, $|\Delta F|$, to estimate added mass. In Table 5.1, we define the incremental force change as $\Delta F_k = F_{i,k} - F_{i-1,k}$ for each axis $k \in \{x, y, z\}$, where the net force magnitude $|\Delta F| = \sqrt{\Delta F_x^2 + \Delta F_y^2 + \Delta F_z^2}$ is used to calculate the estimated mass change $\Delta m = |\Delta F|/g$.

**Table 5.1:** Incremental mass estimation using net force magnitude (lever arm ignored).

| Step | $\Delta F_x$ [N] | $\Delta F_y$ [N] | $\Delta F_z$ [N] | $\lvert\Delta F\rvert$ [N] | $\Delta m$ [g] | Ground Truth [g] | Error [%] |
|---|---|---|---|---|---|---|---|
| 2 | 0.640 | 0.580 | 0.210 | 0.889 | 90.6 | 110.0 | 17.6 |
| 3 | 0.440 | 0.390 | 0.040 | 0.589 | 60.1 | 110.0 | 45.4 |
| 4 | 0.620 | 0.540 | 0.050 | 0.824 | 84.0 | 110.0 | 23.6 |
| 5 | 0.650 | 0.550 | 0.310 | 0.906 | 92.4 | 110.0 | 16.0 |
| 6 | 0.610 | 0.540 | 0.160 | 0.830 | 84.6 | 110.0 | 23.1 |
| 7 | 0.620 | 0.570 | 0.220 | 0.871 | 88.7 | 110.0 | 19.4 |
| 8 | 0.650 | 0.520 | 0.230 | 0.864 | 88.0 | 110.0 | 20.0 |
| 9 | 0.700 | 0.650 | 0.180 | 0.972 | 99.1 | 110.0 | 9.9 |

Table 5.1 summarizes the results from a representative trial. The calculated mass changes $\Delta m$ range from 60.1 g to 99.1 g, with an average of approximately 85.9 g. Compared to the 110.0 g ground truth, the mean relative error is approximately 21.9%, with the maximum error reaching 45.4% in Step 3. This demonstrates that this method is not reliable for precise mass estimation.

This large estimation error is primarily attributed to the inherent limitations of the KUKA iiwa 7 R800 robotic arm. According to the manufacturer, the robot has a nominal payload capacity of 7 kg, and the accuracy of force and torque estimation degrades noticeably when the external load falls below approximately 30% of this rated capacity (i.e., around 2.1 kg).

In our experiments, the combined mass of the beaker, rice, and gripper is approximately 1.3 kg, which is well below this threshold. To further examine this effect, we conducted additional experiments by increasing the external payload to approximately 2.1 kg during the pouring task. Under this condition, overall end-effector wrench estimation error could be controlled within 1%. However, even at this reduced error level, a 1% deviation still corresponds to noise on the order of tens of grams. Such a magnitude of noise is non-negligible for continuous and smooth pouring, where precise mass tracking is required. More importantly, this level of uncertainty propagates directly into the feedback signal used for policy training, introducing stochasticity that can degrade learning stability and lead to suboptimal policies.

Therefore, the estimated end-effector wrench provided by the KUKA software was not

sufficiently reliable to be adopted as the final force-related input for policy learning in this work.

## 5.3 Evaluation Setup

We evaluate three policy variants to systematically study the contribution of visual perception and force feedback in the liquid pouring task:

1. **Single-view Baseline Policy:** The policy takes as input a single RGB observation from the fixed top-view camera and dual-arm joint states. This setting represents the most minimal configuration for visuomotor control.

2. **Multi-view Baseline Policy:** The policy uses three RGB camera views together with dual-arm joint states. This variant is designed to evaluate the impact of richer visual perception on task performance.

3. **Force-Integrated Multi-view Policy:** In addition to the inputs in the multi-view baseline, this policy incorporates dual-arm joint torque signals as force feedback, enabling the model to perceive contact and load variation during manipulation.

## 5.4 Ablation on Visual Input

This section conducts a controlled ablation study based on the experimental setup defined in Section 5.3, focusing on the performance difference between the Single-view Baseline Policy and the Multi-view Baseline Policy. The only variable between the two policies is the visual input configuration, strictly consistent with the definition in Section 5.3: the former takes as input a single RGB observation from the fixed top-view camera, while the latter uses three RGB camera views. All other configurations, including the training dataset, model hyperparameters, dual-arm joint state input, and evaluation environment, are kept completely identical, with no joint torque measurements incorporated in either policy. We analyze the dominant failure modes of the single-view policy below.

### 5.4.1 Failure Mode of the Single-view Baseline Policy

The single-view visual input provides only limited geometric observability of the task scene, leading to severe performance degradation across the full task workflow. We identify two dominant failure modes along the sequential task flow, listed in the order of task execution:

#### Failure Mode 1: Failure to Grasp the Empty Target Container

The dominant failure mode of the single-view baseline is the failure to complete the initial right-arm grasp of the empty target container, leading to immediate task termination at the first subtask. This failure accounts for 85% of all test trials (17 out of 20 total trials).

A likely reason for this failure is the limited geometric observability of the single-view RGB setting used in this baseline. Since the policy receives only one fixed top-view RGB

image and no explicit depth input, it has less information for inferring the 3D relationship between the gripper and the target container. In our setup, this limited observability was insufficient for reliable grasping in many trials. As a result, the gripper often failed to reach a valid grasping pose, leading to early task termination.

A representative snapshot of this failure mode is shown in Fig. 5.4.

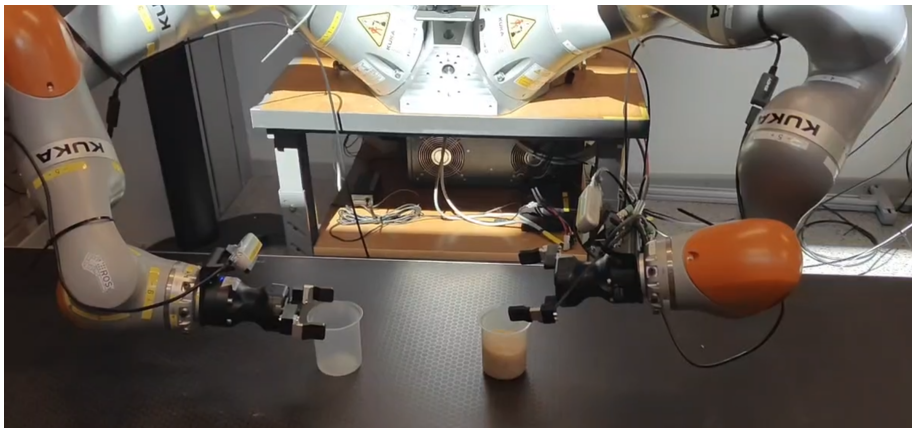



**Figure 5.4:** Representative snapshot of the Single-view Baseline Policy grasping failure: under the single fixed top-view RGB setting, the policy has limited geometric information for estimating a precise grasping pose.

## Failure Mode 2: Incomplete Pouring Execution

In the very limited number of trials where both initial grasping subtasks are successfully completed, the policy is able to initiate the pouring action, but may fail to complete the full pouring process, resulting in incomplete rice transfer and task failure.

The root cause is that the single fixed top-view camera cannot provide a clear, unoccluded view of the rice flow and residual volume in the container during pouring. Unlike the wrist-mounted cameras in the multi-view setup, the fixed top-view camera cannot reliably track the real-time progress of pouring, leading to premature termination of the pouring action before all rice is transferred.

A representative snapshot of this failure mode is shown in Fig. 5.5.

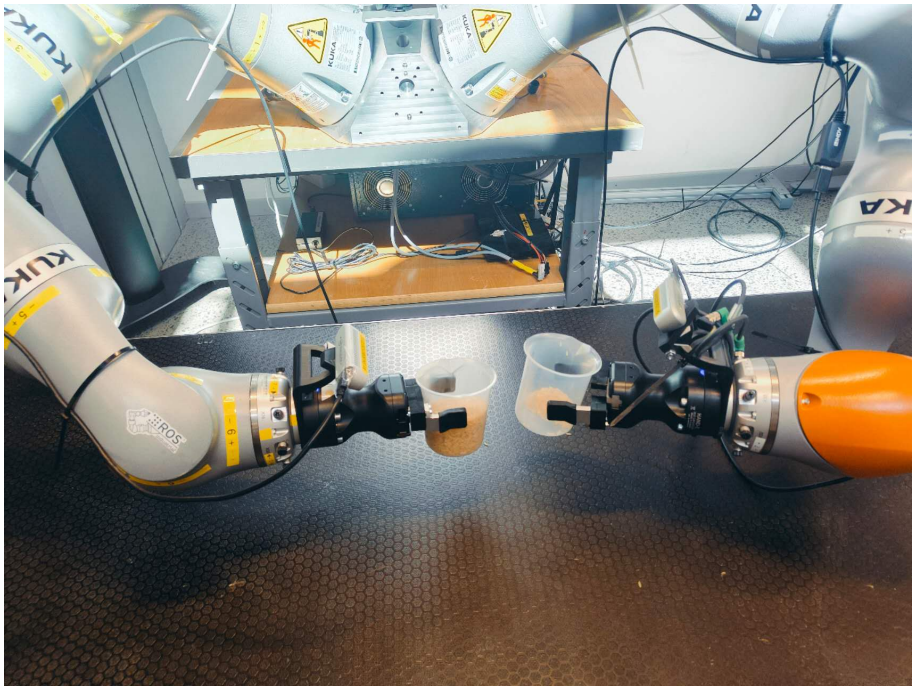


**Figure 5.5:** Representative snapshot of the Single-view Baseline Policy incomplete pouring failure: the policy terminates the pouring action prematurely due to the inability to track pouring progress from a single fixed view.

### 5.4.2 Summary of Visual Input Ablation

The visual-input ablation indicates that, in the current experimental setup, multi-view RGB observations substantially improve task execution compared with the single-view baseline. The fixed top-view RGB input provides global scene context but limited local geometric information around the grippers. By adding two wrist-mounted camera views, the multi-view policy receives additional local visual cues for grasping and alignment, which helps reduce early task failures, especially during the initial grasping stage.

These results suggest that the single-view RGB baseline used in this thesis provides insufficient observability for reliable execution under the tested conditions, while the multi-view setup offers a more informative visual representation for this task. Nevertheless, the remaining failures of the multi-view baseline show that richer visual coverage alone does not fully resolve task-state ambiguity during pouring. This motivates the use of force-related input in the following ablation study.

## 5.5 Ablation on Joint Torque Measurements

This section conducts a controlled ablation study based on the experimental setup defined in Section 5.3, focusing on the behavioral difference between the Multi-view Baseline Policy and the Force-Integrated Multi-view Policy. The only difference is that the latter additionally incorporates raw 14-dimensional dual-arm joint torque signals (7 joints per arm) as input,

while all other training and evaluation configurations remain identical. We analyze the distinct dominant failure modes of the two policies below.

### 5.5.1 Failure Mode of the Multi-view Baseline Policy

The multi-view visual input significantly improves the policy's performance and enhances geometric observability, especially in the early grasping subtask. The wrist-mounted cameras provide detailed local views and enable accurate estimation of the object pose relative to the manipulator, allowing the policy to consistently complete the initial right-arm grasp of the empty container. However, failures still occur in subsequent task stages due to the lack of interactive feedback and explicit task progress signals. We identify three dominant failure modes along the sequential task flow, listed in the order of task execution:

#### Failure Mode 1: Failure to Grasp the Filled Container

After the right gripper successfully grasps the empty target container, the left arm remains inactive and fails to initiate or complete the grasping of the rice-filled container, leading to immediate task termination before the pouring phase. A representative snapshot of this failure mode is shown in Fig. 5.6.

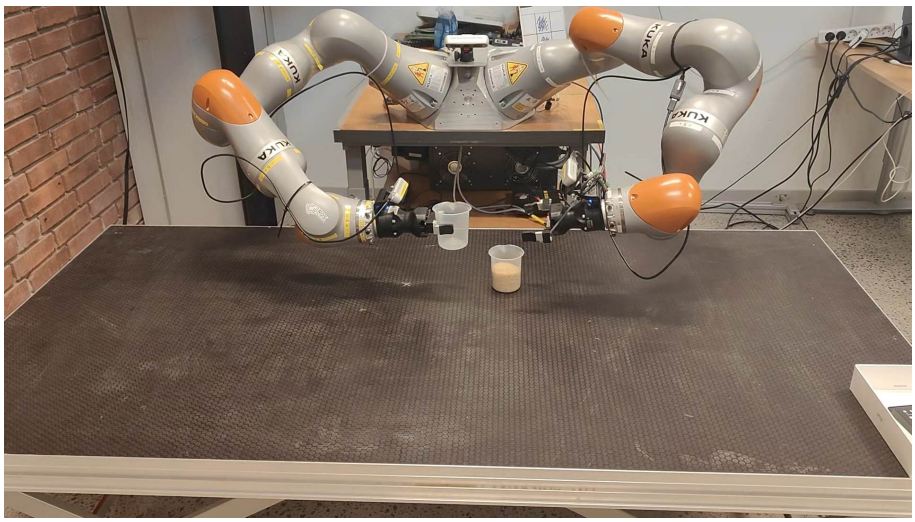



**Figure 5.6:** Representative snapshot of the Multi-view Baseline Policy grasping failure: the left arm remains inactive and fails to grasp the rice-filled container after the right arm completes the empty container grasp.

#### Failure Mode 2: Sustained Task Stagnation Before Pouring

When both containers are successfully grasped, the robot halts at an intermediate pose during the pouring phase and fails to execute the formal pouring action.

The root cause is the inherent limitation of visual-only perception: the policy cannot reliably distinguish between a rice-filled container (to be poured) and an empty container (post-pouring) from RGB images alone. Without an explicit task progress signal, the policy

falls into mode collapse, unable to determine whether to initiate pouring or enter the post-pouring reset phase, eventually resulting in persistent task stagnation.

A representative snapshot of this failure mode is shown in Fig. 5.7.

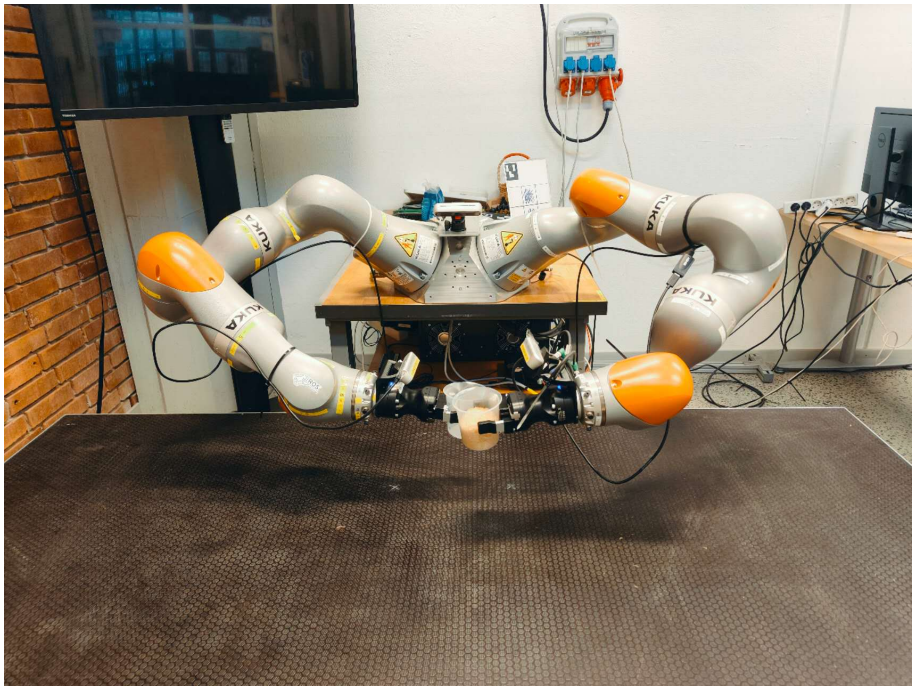



**Figure 5.7:** Representative snapshot of the Multi-view Baseline Policy stagnation failure: the robot halts at the intermediate pouring pose and does not execute the pouring action.

## Failure Mode 3: Rice Spillage During Pouring

The robot successfully initiates the pouring action, but rice is spilled outside the target container during the pouring process, resulting in task failure.

This failure occurs because the model cannot stabilize the dynamic interaction during pouring without force or contact information. The vision-only policy cannot perceive the real-time load change of rice in the container, nor can it adjust the pouring speed and end-effector pose according to the actual task progress, leading to excessive tilting of the container or misalignment between the two containers, and eventually causing incomplete pouring or rice spillage.

A representative snapshot of this failure mode is shown in Fig. 5.8.

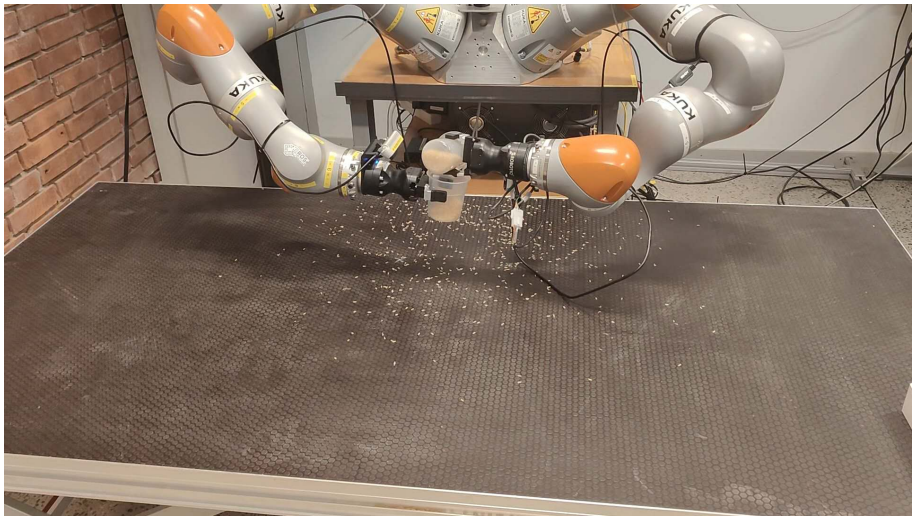


**Figure 5.8:** Representative snapshot of the Multi-view Baseline Policy spillage failure: rice is spilled outside the target container during the pouring process due to unregulated end-effector motion.

### 5.5.2 Failure Mode of the Force-Integrated Multi-view Policy

The Force-Integrated Multi-view Policy nearly eliminates all three failure modes observed in the multi-view baseline and can reliably enter the pouring execution phase in nearly all valid trials.

However, this policy exhibits a distinct failure mode: when the two containers collide during the pouring process, the joint torque signals generate large instantaneous noise and become out-of-distribution (OOD) relative to the training demonstration data. This corrupted OOD input disrupts the policy's inference, leading to unexpected end-effector motions and eventual pouring failure.

The typical scenario of this collision-induced failure is shown in Fig. 5.9.

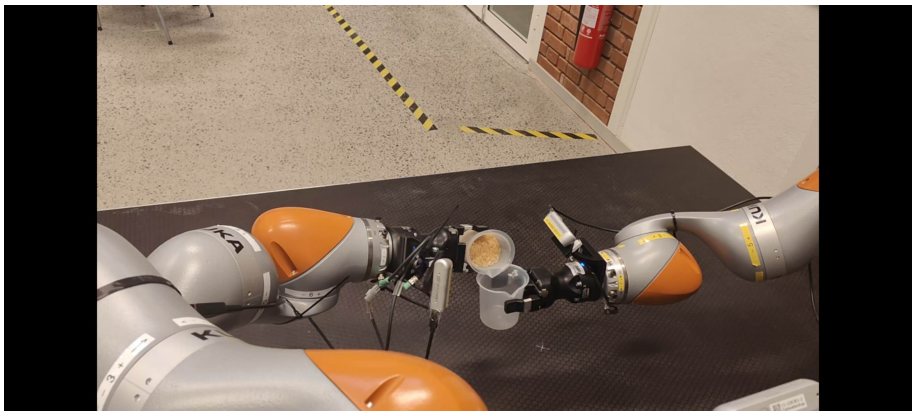


**Figure 5.9:** Typical failure scenario of the Force-Integrated Multi-view Policy: container collision induces OOD noise in joint torque signals, leading to unexpected robot motions.

### 5.5.3 Summary of Joint Torque Ablation

The integration of joint torque measurements effectively mitigates all three core systematic failure modes of the vision-only policy across the full task workflow, but introduces a new scenario-specific failure mode caused by collision-induced OOD joint torque signals. The two policies have clearly distinct dominant failure patterns, which supports the quantitative performance analysis in the subsequent discussion section.

## 5.6 Discussion

To provide a more fine-grained evaluation of policy performance, we decompose the overall liquid pouring task into four sequential subtasks: (1) grasping the empty target container with the right gripper, (2) grasping the filled container with the left gripper, (3) initiating the pouring action, and (4) successfully completing the pouring process.

The success rate of each subtask is evaluated independently based on task-specific criteria. The Right Grasp (Empty) subtask is considered successful if the target container is stably grasped by the right arm. The Left Grasp (Filled) subtask is considered successful if the robot securely grasps the container containing rice without significant misalignment. The Start Pouring subtask is considered successful if the robot successfully transitions into a stable pouring posture and initiates the pouring motion. The Pouring Success subtask is considered successful only if all the rice is transferred into the target container, with no grains spilled outside.

Table 5.2 reports the success rates of each subtask across the three policy variants. All results are computed over the same set of 20 evaluation trials for each policy.

**Table 5.2:** Subtask success rate comparison across different policy variants.

| Policy | Right Grasp (Empty) | Left Grasp (Filled) | Start Pouring | Success |
|---|---|---|---|---|
| Single-view Baseline | 3 | 3 | 3 | 1 |
| Multi-view Baseline | 18 | 12 | 11 | 7 |
| Force-Integrated Multi-view | 19 | 19 | 19 | 9 |

The results in Table 5.2 show clear differences among the evaluated policy variants. The single-view baseline suffers from severe performance degradation in the early subtasks, with only 3 out of 20 trials completing the initial grasping steps. With the multi-view baseline, the policy consistently succeeds in the Right Grasp (Empty) subtask, but failures occur in subsequent stages. Three typical failure modes are observed after placing the empty container: (1) the left arm remains inactive and does not initiate grasping, (2) the left arm successfully grasps the filled container but remains stationary at the pouring position, and (3) rice is spilled during the pouring process, as illustrated in Fig. 5.8. When raw joint torques are incorporated, the policy achieves higher success rates in the first three subtasks and improves the overall pouring success rate, and the previously observed failure modes are no longer present. However, a new scenario-specific failure mode is observed, where collisions between the two

containers occur during execution and lead to unstable pouring and task failure, as shown in Fig. 5.9.

Overall, these results suggest that the main limitation of the vision-only policies is not only insufficient visual coverage, but also incomplete task-state information. Multi-view visual input improves geometric perception and reduces early grasping failures. However, visual observations alone are still insufficient to reliably infer whether the filled container has been securely grasped, whether pouring has started, and how far the pouring process has progressed. The joint-torque input helps reduce this ambiguity by providing an additional signal related to physical interaction and task progression. At the same time, the collision-induced failures observed in the force-integrated policy show that force-related inputs can also make the policy sensitive to contact events not well represented in the demonstrations. Therefore, the benefit of integrating joint torque measurements should be understood as improving task progression rather than guaranteeing robust full-task completion.

# Chapter 6
# Conclusion

This thesis presents a diffusion-based policy integrated with raw joint torque measurements for dual-arm liquid pouring tasks. Our experiments show that policies relying only on visual observations and joint positions suffer from key limitations, such as failing to confirm successful grasping or track task progress. To address these limitations, we initially explored estimating changes in mass from end-effector wrench measurements. However, due to the relatively high nominal payload of the KUKA iiwa 7 R800 robot, this approach produced substantial estimation errors. Consequently, we shifted to directly incorporating raw joint torque signals as part of the policy's conditioning input.

Integrating raw joint torque measurements as a conditional input effectively alleviates some of these limitations. The joint-torques-enhanced policy demonstrates improved task progression compared with the vision-only baselines, particularly in reaching and initiating the pouring stage. Our experimental evaluation, conducted over 60 trials using policies trained on 150 human demonstrations, shows that the force-integrated multi-view policy achieves the highest full-task success rate among the evaluated variants. Specifically, it reaches the pouring stage in 95% of trials and achieves a full-task success rate of 45%, compared with 55% and 35%, respectively, for the multi-view vision-only baseline. These results support the effectiveness of incorporating joint torque measurements into diffusion policy for the studied dual-arm granular pouring task.

Despite these advances, this study has several limitations. First, the hardware-induced noise from the KUKA iiwa 7 R800's internal sensors limits the precision of fine-grained force regulation at low payloads. Second, the dataset scale remains relatively small, which may constrain the policy's zero-shot generalization. Third, the temporal hyperparameters of the diffusion policy, including the observation horizon, action prediction horizon, and number of inference denoising steps, were kept fixed rather than systematically evaluated. These parameters may influence temporal consistency, execution latency, and task success. Finally, using rice instead of real liquids means that the experiments evaluate granular pouring rather than full liquid pouring, thereby bypassing complex fluid dynamics such as sloshing, viscosity, surface tension, and fluid inertia.

Future work will focus on addressing these challenges through four primary directions. To overcome the hardware sensitivity threshold, we aim to integrate high-precision external 6-axis force-torque sensors at the wrist. Furthermore, we plan to scale the dataset to encompass a broader variety of containers and initial conditions to enhance robustness. We will also conduct a systematic analysis of key diffusion policy hyperparameters, including the observation horizon, action prediction horizon, and inference denoising steps, to better understand their effects on temporal consistency, execution latency, and task performance. Lastly, we will extend the policy to handle real liquids with varying viscosities, exploring the diffusion model's capacity to learn compensations for dynamic fluid effects. Overall, this research shows that incorporating raw joint torque measurements is a promising approach for contact-rich pouring tasks, laying the groundwork for more force-sensitive robots in the future.

# References


[1] Malek Aburub, Cristian C. Beltran-Hernandez, Tatsuya Kamijo, and Masashi Hamaya. Learning diffusion policies from demonstrations for compliant contact-rich manipulation, 2024.

[2] Angelos Angelopoulos, James F. Cahoon, and Ron Alterovitz. Transforming science labs into automated factories of discovery. *Science Robotics*, 9(95):eadm6991, 2024.

[3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[4] Jeremy A. Collins, Cody Houff, You Liang Tan, and Charles C. Kemp. Forcesight: Text-guided mobile manipulation with visual-force goals, 2023.

[5] Bin Fang, Shi-Dong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362 – 369, 2019.

[6] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning, 2021.

[7] Xue Li Guan, Dorothy Pei Shan Chang, Zhen Xuan Mok, and Bernett Lee. Assessing variations in manual pipetting: An under-investigated requirement of good laboratory practice. *Journal of Mass Spectrometry and Advances in the Clinical Lab*, 30:25–29, 2023.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[9] Jonathan Ho, Ajay Jain, and P. Abbeel. Denoising diffusion probabilistic models. *ArXiv*, abs/2006.11239, 2020.

[10] Yongqiang Huang, Juan Wilches, and Yu Sun. Robot gaining accurate pouring skills through self-supervised learning and generalization, 2020.

[11] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017.

[12] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[13] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

[14] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024.

[15] Jialong Li, Zhenguo Wang, Tianci Wang, Maj Stenmark, and Volker Krueger. Quest2ros2: A ros 2 framework for bi-manual vr teleoperation. In *Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction*, HRI '26, page 1288–1292, New York, NY, USA, 2026. Association for Computing Machinery.

[16] Haitao Lin, Yanwei Fu, and Xiangyang Xue. Pourit!: Weakly-supervised liquid perception from a single image for visual closed-loop robotic pouring. In *International Conference on Computer Vision (ICCV)*, 2023.

[17] Magnus Linderoth, Andreas Stolt, Anders Robertsson, and Rolf Johansson. Robotic force estimation using motor torques and modeling of low velocity friction disturbances. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3550–3556, 2013.

[18] Wenhai Liu, Junbo Wang, Yiming Wang, Weiming Wang, and Cewu Lu. Forcemimic: Force-centric imitation learning with force-motion capture system for contact-rich manipulation, 2025.

[19] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66), May 2022.

[20] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[21] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, 2009.

[22] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017.

[23] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(Volume 3, 2020):297–330, 2020.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[25] Daniel Schober, Ronja Güldenring, James Love, and Lazaros Nalpantidis. Vision-based robot manipulation of transparent liquid containers in a laboratory setting. In *2025 IEEE/SICE International Symposium on System Integration (SII)*, pages 1193–1200, 2025.

[26] Rishabh Shukla, Raj Talan, Samrudh Moode, Neel Dhanaraj, Jeon Ho Kang, and Satyandra K. Gupta. Force-conditioned diffusion policies for compliant sheet separation tasks in bimanual robotic cells. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7960–7966, 2025.

[27] Mario A. Torres-Acosta, Gary J. Lye, and Duygu Dikicioglu. Automated liquid-handling operations for robust, resilient, and efficient bio-based laboratory practices. *Biochemical Engineering Journal*, 188:108713, 2022.

[28] Toshiaki Tsuji, Yasuhiro Kato, Gokhan Solak, Heng Zhang, Tadej Petrič, Francesco Nori, and Arash Ajoudani. A survey on imitation learning for contact-rich tasks in robotics, 06 2025.

[29] Tian Xu, Hua Tuo, Qianqian Fang, Jie Chen, Jizhuang Fan, Debin Shan, and Jie Zhao. An online payload identification method based on parameter difference for industrial robots. *Robotica*, 42(8):2690–2712, 2024.

[30] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2025.

[31] Ádám Wolf, Stefan Romeder-Finger, Károly Széll, and Péter Galambos. Towards robotic laboratory automation plug & play: Survey and concept proposal on teaching-free robot integration with the lapp digital twin. *SLAS Technology*, 28(2):82–88, 2023.

# Appendix A: Training and Inference Parameters

This appendix summarizes the main training and inference parameters used for the diffusion-policy models.

**Training Parameters**

**Table 1:** Main training parameters used for the diffusion-policy models.

| Category | Parameter | Value |
|---|---|---|
| Task | Task name | `kuka_pouring` |
| Policy | Policy type | `DiffusionUnetHybridImagePolicy` |
| Observation | Image shape | $3 \times 240 \times 320$ |
| Observation | Robot state dimension | 16 |
| Observation | Joint torque dimension | 14 |
| Action | Action dimension | 16 |
| Temporal setup | Prediction horizon | 8 |
| Temporal setup | Observation steps | 3 |
| Temporal setup | Action steps | 6 |
| Temporal setup | Latency steps | 0 |
| Diffusion | Training diffusion steps | 100 |
| Diffusion | Inference denoising steps | 100 |
| Diffusion | Beta schedule | `squaredcos_cap_v2` |
| Diffusion | Prediction type | `epsilon` |
| Network | Diffusion step embedding dimension | 128 |
| Network | Downsampling dimensions | [256, 512, 1024] |
| Network | Kernel size | 5 |
| Network | Number of groups | 8 |
| Network | Freeze encoder | `True` |
| Optimization | Optimizer | AdamW |
| Optimization | Learning rate | $1.0 \times 10^{-4}$ |
| Optimization | Weight decay | $1.0 \times 10^{-6}$ |
| Optimization | Learning-rate scheduler | Cosine |
| Optimization | Warmup steps | 500 |
| Optimization | Batch size | 64 |
| Optimization | Number of epochs | 3000 |

**Inference Parameters**

**Table 2:** Main inference parameters used during real-robot policy execution.

| Category | Parameter | Value |
|---|---|---|
| Policy | Policy type | `DiffusionUnetHybridImagePolicy` |
| Input | Observation steps | 3 |
| Input | Image input | RGB observation |
| Input | Robot state dimension | 16 |
| Input | Joint torque dimension | 14 |
| Output | Action dimension | 16 |
| Execution | Prediction horizon | 8 |
| Execution | Executed action steps | 6 |
| Execution | Latency steps | 0 |
| Diffusion | Inference denoising steps | 100 |
| Diffusion | Prediction type | `epsilon` |

Across the compared policy variants, the main training and inference parameters were kept the same. The difference between the vision-only multi-view baseline and the force-integrated multi-view policy was the inclusion of the 14-dimensional raw joint torque vector as an additional low-dimensional observation.