Force-Conditioned Imitation Learning from Vision-Only Human Demonstrations via Robot Replay

Sakib Chowdhury, Raymond Huang, Vanessa Chen and Yi Guo

Abstract— **Contact-rich manipulation requires reasoning about force, not just trajectory, yet standard imitation learning predicts pose alone. Admittance control can regulate contact force, but requires a hand-specified reference force, typically constant and poorly suited to tasks where the appropriate force varies over time. Prior work shows force is best learned as part of the action rather than passively observed, but still requires a dedicated force/torque or tactile sensor. We introduce a method for recovering a time-varying reference wrench from a single, uninstrumented human demonstration: a tool is tracked visually, the trajectory is replayed on the robot under position control, and the external wrench is recovered from onboard joint torque sensing via the robot's own dynamics model — requiring no wrist force/torque sensor and no instrumentation on the demonstrator. A visuomotor policy jointly predicts reference pose and wrench, supplied at deployment as a time-varying setpoint to a standard admittance controller. On a pasta scooping-and-transfer task, our method achieves higher success and more closely reproduces demonstrated contact force than a pose-only baseline, and generalizes across diffusion, transformer-based, and flow-matching policies, using only 300 seconds of demonstration.**

I. Introduction

Robots performing contact-rich manipulation tasks — assembly, wiping, assistive cooking/feeding — must reason not only about where to move, but about how much force to apply while doing so. Applying too little force can cause a task to fail outright, while applying too much can damage the tool, the environment, or, in physically assistive settings, pose a safety risk to a person receiving care. A policy that reproduces a demonstrated motion with high spatial fidelity but no notion of appropriate contact force is therefore an incomplete solution to contact-rich manipulation.

Recent work has substantially reduced the cost of collecting manipulation demonstrations by removing the need for a robot or any instrumentation at demonstration time. Tool-as-Interface [1] lets a human demonstrate a task with a handheld tool in front of a camera, tracking its motion visually and later converting it into a robot-executable trajectory, eliminating the teleoperation rig or VR headset otherwise required. This makes large-scale demonstration collection far more accessible, but it recovers only *motion* — the tool's pose trajectory — with no mechanism for capturing how much force the human applied. A policy trained on this motion alone, deployed through ordinary position control, inherits this gap: it has no way to know, or enforce, an appropriate contact force at any point in the task.

One might expect this gap to be closed by existing force-control techniques: impedance and admittance control are textbook techniques [2], already exposed as built-in control modes on modern collaborative arms [3]. Admittance control in particular lets a robot yield to external force via a programmable virtual mechanical response — inertia, damping, and stiffness — rather than rigidly holding a commanded position. However, its classical formulation targets a *constant* reference force, suited to tasks like surface polishing where the desired contact force does not vary. Many demonstration-learned tasks do not fit this mold: scooping food, for instance, requires near-zero force during approach, rising force as the tool engages the food, and further variation with food consistency and fill level. A single hand-tuned reference cannot capture this; what is needed is a reference force that varies with the task, the way a human naturally modulates grip and push force without thinking about it — exactly what a Tool-as-Interface-style demonstration does not currently capture.

Beyond task success, this matters for safety and for the physical integrity of the tools and objects involved. A position-only policy has no mechanism to detect or limit excessive contact force: a small trajectory error, or a scene slightly outside the training distribution, can cause the robot to drive a utensil into a rigid surface with far more force than a human demonstrator ever applied, risking damage to the tool or the environment. A force-conditioned policy, by contrast, has an explicit target for how hard it should be pushing at every point in the task, and can use the error between this target and the sensed force to moderate its own behavior in real time. This paper's contribution is a way to supply such a controller with the time-varying reference it requires, learned directly from a human demonstration. In summary, this paper offers:

1) A method for recovering time-varying target contact force from vision-only human demonstrations, via robot replay and the robot's own onboard joint-torque sensing — requiring no wrist force/torque sensor and no instrumentation on the human demonstrator.
2) A visuomotor policy that jointly predicts a reference pose and a reference wrench at each timestep, trained on this recovered data.

3) A policy deployment scheme in which the predicted wrench serves as a time-varying setpoint for standard admittance control.

II. Related Work

Imitation learning for manipulation: Recent advances in imitation learning have enabled robots to acquire complex manipulation behaviors directly from demonstration, without hand-engineered controllers. Behavior cloning methods based on action chunking and transformers [4], implicit energy-based policies [5], and diffusion-based policies [6] have proven particularly effective at capturing the multimodal, non-deterministic nature of human demonstration data. More recently, flow-matching [7] and generalist policies [8] have extended this paradigm across tasks.

Demonstration collection: A parallel line of work has focused on reducing the cost and instrumentation burden of collecting demonstrations. Tool-as-Interface [1] enable a human to demonstrate a task using a handheld tool, with no robot present during the demonstration; the tool's motion is tracked visually and later used to derive a robot-executable trajectory. The Universal Manipulation Interface (UMI) [9] similarly collects demonstrations with a handheld gripper decoupled from any specific robot embodiment. Low-cost teleoperation platforms [10], [11] have further lowered the barrier to large-scale data collection. These methods substantially reduce the cost of acquiring demonstrations, since no teleoperation rig or robot time is required at the point of demonstration. However, they are concerned exclusively with recovering motion (the tool's or gripper's pose trajectory) and do not consider or recover the contact force from the demonstration.

Force-conditioned policy learning: A separate line of work has sought to incorporate force into learned policies. Reactive Diffusion Policy (RDP) [12] and its extension ImplicitRDP [13] both find that naively appending force to the policy's observation is largely ineffective — a diffusion policy given force as a passive observation channel performs comparably to a vision-only baseline. Both works show that force is effective when incorporated into the action space: RDP via a fast, force-reactive decoder operating alongside a slower latent policy, and ImplicitRDP via a virtual-target formulation derived from quasi-static compliance control. Outside the imitation-learning setting, reinforcement learning approaches have also demonstrated learned force control on position-interface robots [14]. Across these literatures, the consistent finding is that force is best incorporated at the action or control level, compared to an incidental observation.

Robot-assisted feeding: Assistive feeding is a long-studied application of manipulation research, encompassing the role of force in fork-based bite acquisition [15], adaptive strategies for food properties and user preferences [16], and general-purpose feeding systems [17]. These tasks are inherently contact-rich and safety-critical: excessive force during bite acquisition can damage utensils, spill food, or, in the physical vicinity of a human user, pose a direct safety concern. This makes robot-assisted feeding a natural and high-stakes testbed for force-conditioned imitation learning, and motivates the pasta scooping-and-transfer task studied in this work.

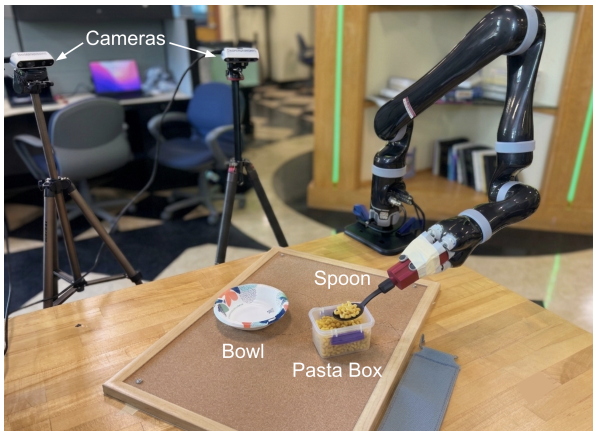

Fig. 1: Experimental setup. Two fixed RGB-D cameras observe the workspace, in which the Kinova Jaco^2 6-DOF arm performs the pasta scooping-and-transfer task: scooping pasta from the pasta box with a spoon and transferring it to the bowl.

These lines of work have progressed largely independently: methods that eliminate instrumented demonstrations do not incorporate force, and methods that incorporate force still require a dedicated wrist force/torque or tactile sensor during both collection and deployment. This leaves a subtler resource unused: many modern robot arms already have onboard joint-torque sensing as standard [3], yet no existing force-conditioned imitation learning method exploits it to recover force from an otherwise instrumentation-free demonstration. This paper sits at that intersection: eliminating instrumentation from the demonstration side while still recovering a time-varying force from the robot's existing onboard torque sensors.

Following this finding, our policy predicts a target wrench (force and moments) alongside a target pose, placing force in the action space as prior work suggests it should be. But we also keep wrench in the observation space to serve as an additional signal of the current contact state so the policy can tell what's happening right now and decide what force to apply next.

III. The System & Problem Formulation

A. The System

We use a Kinova Jaco^2 6-DOF robotic arm with a spherical wrist configuration. Each of the six joints is driven by modular Kinova actuators (K-75, K-75+

and K-58) equipped with integrated torque sensors, in addition to position encoders, enabling joint torque to be read directly at every joint. The scene contains a pasta box, a bowl and the robotic arm as shown in Fig. 1. Two Intel RealSense RGB-D cameras are pointed at the scene. The experiments are conducted with the robot holding a rigid, 3D-printed spoon as its end-effector tool for the pasta scooping-and-transfer task.

B. Problem Formulation

We formulate the task (scooping pasta from a box and transferring to a bowl) as a Markov Decision Process (MDP), and aim to learn a policy π that enables the robot to perform a given task.

Observation space: At each timestep t, the robot's observation $\mathbf{o}_t \in \mathcal{O}$ consists of three modalities: a single-view RGB image $\mathbf{I}_t \in \mathcal{I}$, the robot's proprioceptive pose $\mathbf{X}_t \in SE(3)$, and the sensed external wrench at the end-effector $\mathbf{F}_t^{ext} \in \mathbb{R}^6$, recovered from onboard joint torque sensing. Here $\mathbf{F}_t^{ext} = [F_x, F_y, F_z, M_x, M_y, M_z]^\top$ denotes the full six-dimensional wrench, comprising the three force and three moment components at the end-effector. Each image $\mathbf{I}_t$ is a tensor in $\mathbb{R}^{128\times128\times3}$. As is standard in visuomotor imitation learning, the policy conditions on a short history of these observations rather than a single timestep. We denote the observation horizon T_o, such that the full conditioning input is

$$\mathbf{o}_t = \left\{\left(\mathbf{I}_\tau,\ \mathbf{X}_\tau,\ \mathbf{F}_\tau^{ext}\right)\ :\ \tau \in [t - T_o + 1,\ t]\right\}. \tag{1}$$

Action space: The action space $\mathcal{A}$ consists of a future chunk of actions over an action horizon T_a. Each action $\mathbf{a}_t \in \mathcal{A}$ consists of a target end-effector pose $\mathbf{X}^r \in SE(3)$ and a target contact wrench $\mathbf{F}^r \in \mathbb{R}^6$:

$$\mathbf{a}_t = \{(\mathbf{X}_\tau^r,\ \mathbf{F}_\tau^r)\ :\ \tau \in [t,\ t + T_a - 1]\}. \tag{2}$$

We refer to $\mathbf{X}^r$ and $\mathbf{F}^r$ as *reference pose* and *reference wrench*, since they are not commands sent directly to the robot's low-level controller, but are instead supplied as the time-varying reference to a downstream admittance controller. Admittance control imposes a relationship between motion and force via a virtual mechanical model:

$$\mathbf{M}_d\ddot{\mathbf{e}} + \mathbf{D}_d\dot{\mathbf{e}} + \mathbf{K}_d\mathbf{e} = \mathbf{F}^{ext} - \mathbf{F}^r, \qquad \mathbf{e} = \mathbf{X} - \mathbf{X}^r, \tag{3}$$

where $\mathbf{M}_d, \mathbf{D}_d, \mathbf{K}_d \in \mathbb{R}^{6\times6}$ are the desired (virtual) inertia, damping, and stiffness matrices, and $\mathbf{e} \in \mathbb{R}^6$ is the deviation $\mathbf{X}$ and $\mathbf{X}^r$.

Objective: Our goal is for the robotic arm to autonomously scoop pasta from the pasta box and place it in the nearby bowl, applying an appropriate amount of contact force throughout the scooping, transferring, and dropping phases of the task—rather than following a trajectory blind to how hard it is pressing—and to learn this behavior entirely from human demonstration, without any hand-specified force target or control law. A human demonstration alone provides no direct measurement of contact force, however, so recovering a reference wrench trajectory to train on is itself part of our objective. Formally, given a dataset $\mathcal{D}$ of recovered reference pose and wrench trajectories, and the observation space $\mathcal{O}$ and action space $\mathcal{A}$ defined above, we aim to learn a policy $\pi_\theta(\mathbf{a}_t \mid \mathbf{o}_t)$, trained by imitation on $\mathcal{D}$, that, given an observation $\mathbf{o}_t \in \mathcal{O}$, predicts an action $\mathbf{a}_t \in \mathcal{A}$, so that both the motion and the physical force applied throughout the task match what a human demonstrator would apply at each phase.

IV. Main Method

A. Overview

Our method begins with a human demonstrating the task in front of a camera using a tool, whose motion we track. This motion is then replayed on the robot, as shown in Fig. 2. While replaying, the robot records its joint torques and, using an estimate of its own dynamics, extracts the external force acting on the tool. We then generate novel views of the scene using 3D Gaussian Splatting, so that the resulting policy is robust to camera viewpoint. Finally, we train a diffusion policy on the recovered pose and force trajectories and deploy it in closed loop, with its predicted wrench supplied as a time-varying setpoint to an admittance controller.

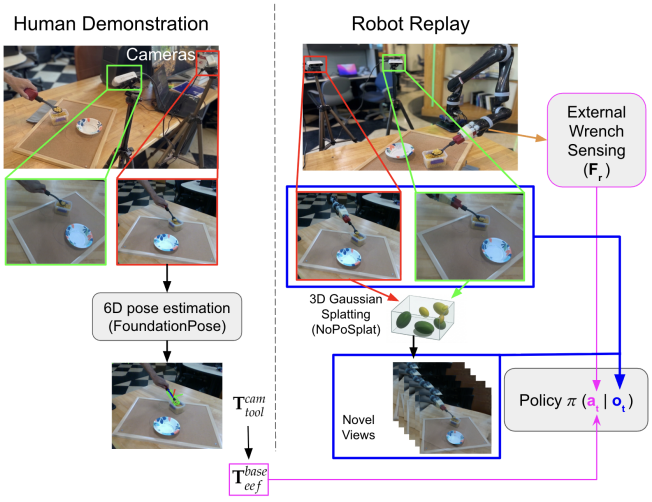

Fig. 2: System overview: human demonstration and robot replay.

B. Human Demonstration Collection

Human demonstrations are collected without the robot present, and without any instrumentation on the human demonstrator. A human performs the pasta scooping-and-transfer task using a standard spoon, in front of one of the two fixed cameras described in Section III-A. We use FoundationPose [18] to track the 6D pose of the spoon at every frame, yielding the tool's pose relative to the camera frame, $\mathbf{T}_{tool}^{cam}(t) \in SE(3)$.

To express this tracked motion in a frame shared with the robot, we introduce a fixed ChArUco board as a common task frame. The transform from the camera

frame to the task frame, $\mathbf{T}_{cam}^{task}$, is obtained by detecting the ChArUco board in the camera image and solving the corresponding PnP problem; since the board does not move during demonstration collection, this transform is computed once and held fixed. The tool's pose in the task frame is then obtained by composing the two transforms:

$$\mathbf{T}_{tool}^{task}(t) = \mathbf{T}_{cam}^{task}\,\mathbf{T}_{tool}^{cam}(t). \tag{4}$$

This yields $\mathbf{T}_{tool}^{task}(t)$, the demonstrated tool trajectory expressed in a frame common to both the human demonstration and the robot, without the robot present during data collection.

C. Robot Replay & Force Recovery

In this step, the robot is separately calibrated with respect to the same ChArUco board, giving a fixed transform $\mathbf{T}_{task}^{base}$ from the task frame to the robot's base frame as shown in Fig. 3. Composing this with the $\mathbf{T}_{tool}^{task}(t)$, we get the tool motion directly in the robot's base frame:

$$\mathbf{T}_{tool}^{base}(t) = \mathbf{T}_{task}^{base}\,\mathbf{T}_{tool}^{task}(t). \tag{5}$$

To determine the corresponding target pose for the robot's end-effector, we account for the fixed rigid transform $\mathbf{T}_{tool}^{eef}$ between the robot's end-effector and the tool it holds — identical in geometry to how the human holds the tool — such that:

$$\mathbf{T}_{eef}^{base}(t) = \mathbf{T}_{tool}^{base}(t)\left(\mathbf{T}_{tool}^{eef}\right)^{-1}. \tag{6}$$

This yields $\mathbf{T}_{eef}^{base}(t)$, the desired end-effector pose trajectory the robot must follow to replicate the human-demonstrated tool motion. We solve for the corresponding joint-space trajectory via inverse kinematics:

$$\mathbf{q}_r(t) = \mathrm{IK}\left(\mathbf{T}_{eef}^{base}(t)\right). \tag{7}$$

The robot executes $\mathbf{q}_r(t)$ using position control, so that it strictly retraces the demonstrated trajectory regardless of any contact force encountered along the way. As the robot moves, we record the camera frames ($\mathbf{I}_t^{cam0}$ and $\mathbf{I}_t^{cam1}$), as well as joint torques $\boldsymbol{\tau}(t)$ at every timestep using the onboard torque sensor in each actuator. This step also differs from tool-as-Interface [1] in how training images are obtained. They train on the human demonstration images themselves, masking out the human hand with a segmentation model, and similarly mask the robot arm at deployment so that the training and deployment image distributions remain aligned. We find that this segmentation-based masking is not reliably accurate across all environments, and inaccurate masks at deployment time can noticeably degrade policy performance. Because our training images come directly from the robot replaying the task — rather than from the human demonstration — no human hand or robot-arm masking is required at any stage: the robot is simply present, performing the task, in both training and deployment images alike.

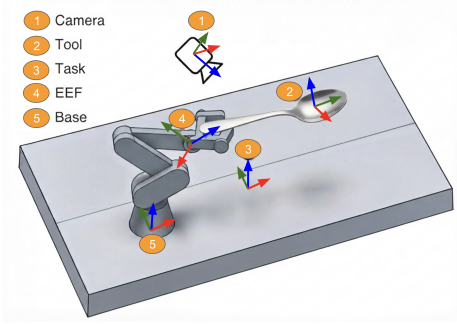

Fig. 3: Coordinate frames used in this work: the camera frame, the tool frame, the task frame, the robot end-effector (EEF) frame, and the robot base frame.

The measured torque $\boldsymbol{\tau}(t)$ reflects both the torque required to move the robot's own body and any torque arising from external contact:

$$\boldsymbol{\tau}(t) = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{J}(\mathbf{q})^{\top}\mathbf{F}^{ext}, \tag{8}$$

where $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}$, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$, and $\mathbf{G}(\mathbf{q})$ are the inertial, Coriolis/centrifugal, and gravitational torque contributions from the robot's own dynamics, and $\mathbf{J}(\mathbf{q})^{\top}\mathbf{F}^{ext}$ is the contribution of the external wrench $\mathbf{F}^{ext}$ at the end-effector, mapped into joint torques via the manipulator Jacobian $\mathbf{J}(\mathbf{q})$.

We compute the gravitational, Coriolis, and inertial torque terms from the measured joint trajectory (details in appendix), and subtract them from the measured torque $\boldsymbol{\tau}(t)$. The residual is attributed entirely to external contact, and is converted into the corresponding end-effector wrench via the Jacobian pseudoinverse:

$$\mathbf{F}^{r}(t) = (\mathbf{J}(\mathbf{q})^{\top})^{+}\left(\boldsymbol{\tau}(t) - \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})\right). \tag{9}$$

This yields $\mathbf{F}^{r}(t)$, the recovered reference wrench, without requiring any wrist force/torque sensor or instrumentation on the human demonstrator. Together, $\left(\mathbf{q}_r(t), \mathbf{F}^{r}(t)\right)$ — or equivalently $\left(\mathbf{X}^{r}(t), \mathbf{F}^{r}(t)\right)$ after forward kinematics — constitute the training labels used to supervise the policy described in Section IV-F.

D. Novel View Synthesis

The two camera images used for this augmentation are collected during *robot replay*, since it is only then that both fixed cameras observe the robot executing the task, providing the paired stereo images $\mathbf{I}_t^{cam0}$ and $\mathbf{I}_t^{cam1}$ used below.

Our setup provides only two fixed camera viewpoints per episode, while the policy is deployed with access to only a single camera. To improve robustness to camera placement without collecting additional physical demonstrations, we augment the training set with synthetically rendered viewpoints of each recorded replay episode using NoPoSplat [19], a feed-forward 3D Gaussian Splatting model that reconstructs

a 3D scene from a pair of unposed images in a single forward pass. Novel viewpoints are sampled by interpolating between the cam0 and cam1 poses. Each view is cropped to the largest hole-free rectangle, and rescaled to native resolution, yielding novel views $\{\mathbf{I}_{t,k}^{novel}\}_{k=1}^{K}$ per replay timestep.

This augmentation is valid because our action labels $\left(\mathbf{X}^r(t), \mathbf{F}^r(t)\right)$ are defined in the robot's base/task frame (Section IV-C), not in any camera's frame: the real images and all novel-view renders at a given timestep therefore share an identical action label, letting each timestep be paired with several alternate-viewpoint images at no additional data collection cost. During training, the observation image is drawn at random from this set,

$$\mathbf{I}_t \sim \left\{\mathbf{I}_t^{cam0},\ \mathbf{I}_t^{cam1},\ \mathbf{I}_{t,1}^{novel},\ \ldots,\ \mathbf{I}_{t,N}^{novel}\right\}. \tag{10}$$

This synthesis is performed once, offline; at deployment, the policy receives only the live feed from the single deployed camera, with no rendering performed online.

E. Neural Network Architecture

Having recovered the pose and wrench trajectories through robot replay and augmented the corresponding images via novel view synthesis, we now turn to how this data is used to train our policy. We adopt a diffusion-based policy architecture, which models an action as a sample from a learned distribution, generated by iteratively denoising Gaussian noise conditioned on the current observation — a formulation well suited to capturing the multimodal, non-deterministic nature of human demonstration data, where similar observations may correspond to several equally valid actions. Fig. 4 illustrates the resulting policy architecture, which follows the visuomotor diffusion policy design of Chi et al. [6], extended to accommodate the wrench modality in both the observation and action space.

Observation encoding: At each of the T_o observation steps, the camera image is passed through a shared ResNet-18 encoder [20] (ImageNet-pretrained); the resulting visual feature is concatenated with the proprioceptive pose and the sensed external wrench for that step. This concatenated vector is passed through an MLP (Multi Layer Perceptron) to produce a per-step observation embedding. The embeddings from all T_o steps are then concatenated to form the full observation embedding used for conditioning.

Conditioning signal: In parallel, the current diffusion timestep is encoded via a sinusoidal positional encoding and passed through an MLP to produce a time embedding. The observation embedding and time embedding are concatenated to form a conditioning signal, which is broadcast to every stage of the denoising network via FiLM [21].

Algorithm 1 Policy Training

Require: Dataset $\mathcal{D}$ from real/novel-view images, recovered pose $\mathbf{X}^r$, and wrench $\mathbf{F}^r$ per timestep, noise schedule $\{\bar{\alpha}_k\}_{k=1}^{K}$, learning rate η
Ensure: Trained network parameters θ

Initialize θ randomly
repeat
  Sample a minibatch of timesteps t from $\mathcal{D}$
  for each sampled t **do**
    for each $\tau \in [t - T_o + 1, t]$ **do**
      Draw $\mathbf{I}_\tau \sim \{\mathbf{I}_\tau^{cam0}, \mathbf{I}_\tau^{cam1}, \mathbf{I}_{\tau,1:N}^{novel}\}$
    end for
    $\mathbf{o}_t \leftarrow \{(\mathbf{I}_\tau, \mathbf{X}_\tau, \mathbf{F}_\tau^{ext}) : \tau \in [t - T_o + 1, t]\}$
    $\mathbf{a}_t^0 \leftarrow \{(\mathbf{X}_\tau^r, \mathbf{F}_\tau^r) : \tau \in [t, t + T_a - 1]\}$ ▷ recovered labels
    Sample $k \sim \text{Uniform}\{1, \ldots, K\}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
    $\mathbf{a}_t^k \leftarrow \sqrt{\bar{\alpha}_k}\,\mathbf{a}_t^0 + \sqrt{1 - \bar{\alpha}_k}\,\boldsymbol{\epsilon}$
    $\hat{\boldsymbol{\epsilon}}_t \leftarrow \epsilon_\theta(\mathbf{a}_t^k, k, \mathbf{o}_t)$
  end for
  $\theta \leftarrow \theta - \eta \nabla_\theta \dfrac{1}{|\mathcal{B}|} \displaystyle\sum_{t \in \mathcal{B}} \|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_t\|^2$ ▷ AdamW update
until convergence
return θ

Denoising network: The noise-prediction network is a Conditional U-Net 1D [22] operating on the action chunk. It consists of an input projection (Conv1D), two downsampling stages (Down A, Down B; each two residual Conv1D blocks), a bottleneck (two residual Conv1D blocks), two upsampling stages (Up A, Up B; each two residual Conv1D blocks), and an output projection (Conv1D). Skip connections link Down A to Up A and Down B to Up B, following the standard U-Net pairing of corresponding resolution levels. Every stage — input projection through both upsampling stages — is modulated by the same FiLM conditioning signal.

The network takes a noisy action as input and predicts the noise as output. This is applied iteratively for k number of steps following the reverse diffusion process, and the final denoised representation is mapped to $(\mathbf{X}^r, \mathbf{F}^r)$, over the action horizon T_a, used for deployment.

F. Policy Training

We train the policy using the standard denoising diffusion objective for visuomotor policies [6] as shown in Algorithm 1. A ground-truth action chunk from the demonstration dataset, $\mathbf{a}_t^0 = \left(\mathbf{X}_{t:t+T_a-1}^r, \mathbf{F}_{t:t+T_a-1}^r\right) \sim \mathcal{D}$, is corrupted by a fixed forward noising process: at diffusion step k, Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ is added to $\mathbf{a}_t^0$ to produce a noisy action $\mathbf{a}_t^k$. The network is trained to reverse this process: given the noisy action $\mathbf{a}_t^k$, the diffusion step k, and the current observation $\mathbf{o}_t$, it predicts the noise that was added, and its parameters θ are optimized to minimize the mean-squared error

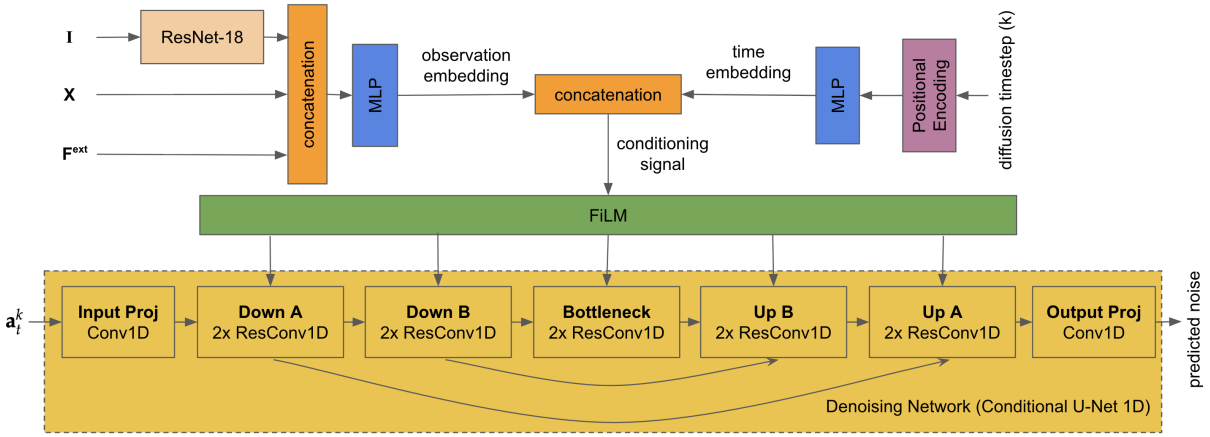

Fig. 4: Neural network architecture with the observation encoding, FiLM conditioning, and the denoising U-Net. The U-Net predicts noise at each step and is applied iteratively over K denoising steps, ultimately yielding the denoised action $(\mathbf{X}^r, \mathbf{F}^r)$.

between the predicted and true noise:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{a}_t^0, k, \epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\mathbf{a}_t^k,\, k,\, \mathbf{o}_t\right)\right\|^2\right]. \tag{11}$$

We use an observation horizon of $T_o = 2$ and an action horizon of $T_a = 16$.

G. Policy Deployment

At deployment, the trained policy is executed in closed loop on the robot. At each control step, the policy consumes the current observation $\mathbf{o}_t$ and produces a reference pose and reference wrench chunk, $\left(\mathbf{X}_{r,t:t+T_a}, \mathbf{F}_{r,t:t+T_a}\right)$, by iteratively denoising from Gaussian noise through the trained network ϵ_θ over k diffusion steps as shown in Algorithm 2.

Live force sensing: At every control step, we read the current joint torques from the onboard torque sensors and recover the live external wrench $\mathbf{F}^{ext}(t)$ using the same dynamics-based subtraction and Jacobian mapping described in Section IV-C, applied online rather than offline. This yields a live estimate of the wrench currently experienced at the end-effector, computed consistently with the reference wrench recovered during robot replay.

Admittance correction: The predicted reference pose $\mathbf{X}^r(t)$ and reference wrench $\mathbf{F}^r(t)$, together with the live sensed wrench $\mathbf{F}^{ext}(t)$, are supplied to the admittance controller introduced in Eq. (3). The resulting corrected pose $\mathbf{X}^{cmd}(t)$ is the command actually sent to the robot's low-level position controller.

V. Results

We evaluate the performance based on success rate on various conditions evaluated over 40 trials per condition. We define a trial as successful if the robot both scoops pasta from the pasta box and transfers it into the bowl without any spilling. All conditions are trained on 30 episodes of demonstration data, each averaging 10 seconds in duration, totaling approximately 300 seconds (5 minutes) of demonstration data.

A. Ablation Study

Table I reports task success rate across variants that ablate the data collection method, the use of force, the deployment controller, and the policy architecture, all trained and evaluated under otherwise identical conditions.

Three patterns emerge. First, comparing the first row against ours isolates the contribution of our replay-based wrench recovery specifically. Both rows use the same pose and wrench formulation and admittance deployment, differing only in how the training data was collected; replacing human demonstration and replay with direct joystick teleoperation (recording pose and force live, with no replay stage) collapses success to zero. This is likely driven by a combination of degraded demonstration quality inherent to joystick teleoperation indicating that our pipeline's benefit comes from the combination of naturalistic human demonstration and replay-based recovery together, not from access to live force alone.

Second, comparing the second row against ours isolates the contribution of force itself: with pose only (position control), and no wrench prediction, success rate drops to 82.5%, confirming that force contributes to task performance beyond pose alone.

Third, the last three rows show that our pose and wrench formulation is not limited to the diffusion policy used elsewhere in this paper: ACT [4] and flow-matching [7] also achieve comparable success (92.5% each) when trained on the same recovered data, with diffusion slightly ahead (97.5%).

B. Comparison with Baseline

We additionally compare our method against Tool-as-Interface [1] as a baseline. This baseline uses only

TABLE I: Ablation study: task success rate across data collection method, force usage, and policy architecture (40 trials each).

Data Collection	Observation/Action		Architecture			Success Rate
	Pose	Wrench	Diffusion [6]	ACT [4]	Flow Matching [7]	
Joystick	✓	✓	✓	×	×	0%
Human Demo + Replay	✓	×	✓	×	×	82.5%
Human Demo + Replay	✓	✓	×	×	✓	92.5%
Human Demo + Replay	✓	✓	×	✓	×	92.5%
Human Demo + Replay (ours)	✓	✓	✓	×	×	**97.5%**

Algorithm 2 Closed-Loop Force-Aware Policy Deployment

Require: Trained network ϵ_θ, live camera, robot with onboard torque sensing, execution horizon $T_{exec} \leq T_a$
Ensure: Executed motion at every control tick

Initialize $\mathbf{X}^{cmd}$, trajectory buffer, and step counter $c \leftarrow T_{exec}$
loop
  Acquire observation $\mathbf{o}_t$ (Eq. **??**)
  if $c \geq T_{exec}$ **then**
    Sample $\mathbf{a}_t^K \sim \mathcal{N}(0, \mathbf{I})$
    for $k = K, K-1, \ldots, 1$ **do**
      $\hat{\boldsymbol{\epsilon}} \leftarrow \epsilon_\theta(\mathbf{a}_t^k, k, \mathbf{o}_t)$
      $\mathbf{a}_t^{k-1} \leftarrow \text{Denoise}\left(\mathbf{a}_t^k, \hat{\boldsymbol{\epsilon}}\right)$
    end for
    unpack $(\mathbf{X}_{r,\,t:t+T_a}, \mathbf{F}_{r,\,t:t+T_a}) \leftarrow \mathbf{a}_t^0$
    place $(\mathbf{X}_{r,\,t:t+T_a}, \mathbf{F}_{r,\,t:t+T_a})$ into trajectory buffer
    $c \leftarrow 0$
  end if
  $\mathbf{X}^r(t), \mathbf{F}^r(t) \leftarrow$ interpolate trajectory buffer
  Compute $\mathbf{F}^{ext}(t)$
  Compute $\mathbf{e}(t) = \mathbf{F}^{ext}(t) - \mathbf{F}^r(t)$
  $\mathbf{X}^{cmd}(t) \leftarrow \mathbf{X}^r(t) + \mathbf{e}(t)$
  $\mathbf{q}_{cmd}(t) \leftarrow \text{IK}\left(\mathbf{X}^{cmd}(t)\right)$
  Send $\mathbf{q}_{cmd}(t)$ to low-level position controller
  $c \leftarrow c + 1$
end loop

human demonstration, with no robot replay and no force: the policy is trained directly on human demonstration images, observing and predicting pose only. To keep the training and deployment image distributions aligned despite the human hand being present during demonstration but absent at deployment, the original method masks the human hand out of training images and masks the robot arm out of deployment images using a segmentation model. We reimplement this baseline by training a UNet-based segmentation model [22] for this purpose, following the same masking strategy.

Table II reports success rate for the baseline and our method, following the same evaluation protocol described above. Our method outperforms the baseline by a wide margin. Beyond the benefit of force discussed in Section V-A, we attribute part of this gap to the masking strategy itself: segmentation-based masking is not reliably accurate across all environments, and inaccurate masks on deployment time can degrade policy performance. Because our training images come directly from robot replay rather than human demonstration, our method requires no masking of any kind at either training or deployment time, avoiding this failure mode entirely.

TABLE II: Comparison with the Tool-as-Interface baseline (40 trials each).

Method	Success rate
Baseline [1]	80%
Ours	**97.5%**

VI. Discussion & Conclusion

We presented a method for incorporating contact force into imitation learning without instrumenting the human demonstrator with any dedicated force/torque sensor. By tracking a tool during an uninstrumented human demonstration and replaying the resulting trajectory on the robot under position control, we recover a reference wrench directly from onboard joint torque sensing, using the robot's own dynamics model to isolate the external contact force. This recovered wrench, together with the demonstrated pose, forms the supervision for a visuomotor policy that jointly predicts a reference pose and reference wrench at deployment, which are supplied as a time-varying setpoint to an admittance controller. On a pasta scooping-and-transfer task, this approach achieves a higher success rate than other test cases and remains effective across diffusion, transformer-based (ACT), and flow-matching policy architectures alike — all from just 5 minutes of human demonstration in total.

A key limitation of this approach is that the accuracy of the recovered force is fundamentally bounded by the accuracy of the underlying tool pose tracking. Since the entire force-recovery pipeline is driven by replaying a tracked trajectory and attributing the resulting torque residual to external contact, any error in the tracked tool pose propagates directly into the replayed trajectory, and from there into the recovered wrench itself. This makes our approach well suited to tasks

where approximate, task-appropriate contact force is sufficient to improve behavior — as in the scooping task studied here — but less suited to applications demanding high-precision force tracking, such as delicate assembly, where tracking error on the order of our current pose-estimation accuracy would translate into force errors too large to be acceptable. Improving the precision of this pipeline for such applications would likely require higher-fidelity tool tracking than the vision-based estimation used in this work, which we leave to future work.

References

[1] H. Chen, C. Zhu, S. Liu, Y. Li, and K. R. Driggs-Campbell, "Tool-as-interface: Learning robot policies from observing human tool use," in *Conference on Robot Learning*, 2025.

[2] N. Hogan, "Impedance control: An approach to manipulation," in *American Control Conference*, 1984.

[3] F. Boucher, H. Lamontagne, A. Campeau-Lecours, S. Latour, P. Fauteux, V. Maheu, C. Deguire, and L.-J. Caron L'Ecuyer, "Kinova modular robot arms for service robotics applications," 2017.

[4] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Robotics: Science and Systems (RSS)*, 2023.

[5] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on robot learning*, 2022.

[6] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2025.

[7] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint*, 2024.

[8] K. Black, O. Mees, H. Walke, K. Pertsch, D. Ghosh, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," in *Robotics: Science and Systems (RSS)*, 2024.

[9] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," in *Robotics: Science and Systems (RSS)*, 2024.

[10] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation using low-cost whole-body teleoperation," in *Conference on Robot Learning*, 2024.

[11] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, "Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[12] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu, "Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation," in *Robotics: Science and Systems (RSS)*, 2025.

[13] W. Chen, H. Xue, Y. Wang, F. Zhou, J. Lv, Y. Jin, S. Tang, C. Wen, and C. Lu, "Implicitrdp: An end-to-end visual-force diffusion policy with structural slow-fast learning," *IEEE Robotics and Automation Letters*, 2026.

[14] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning force control for contact-rich manipulation tasks with rigid position-controlled robots," *IEEE Robotics and Automation Letters*, 2020.

[15] T. Bhattacharjee, G. Lee, H. Song, and S. S. Srinivasa, "Towards robotic feeding: Role of haptics in fork-based food manipulation," *IEEE Robotics and Automation Letters*, 2019.

[16] E. K. Gordon, X. Meng, M. Barnes, T. Bhattacharjee, and S. S. Srinivasa, "Adaptive robot-assisted feeding: An online learning framework for acquiring previously unseen food items," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[17] R. Feng, Y. Kim, G. Lee, E. K. Gordon, M. Schmittle, S. Kumar, T. Bhattacharjee, and S. S. Srinivasa, "Robot-assisted feeding: Generalizing skewering strategies across food items on a plate," in *International Symposium of Robotics Research (ISRR)*. Springer International Publishing, 2022.

[18] B. Wen, W. Yang, J. Kautz, and S. Birchfield, " FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects ," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[19] B. Ye, S. Liu, H. Xu, X. Li, M. Pollefeys, M.-H. Yang, and S. Peng, "No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 54 009–54 033.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016.

[21] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI conference on artificial intelligence*, 2018.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[23] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *The International Journal of Robotics Research*, 1986.

[24] W. Khalil and E. Dombre, *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

Appendix I
Dynamics Parameter Estimation

Recovering the external wrench $\mathbf{F}^{ext}$ from measured joint torque (Eq. 8) requires estimates of $\mathbf{G}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$, and $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}$, so they can be subtracted and the remainder attributed to external contact. We estimate these using the same replay trajectories, executed in two different modes.

Gravity. Gravity torque is linear in a per-link barycentric parameter set $\boldsymbol{\phi} \in \mathbb{R}^{24}$ (mass and mass-weighted center-of-mass, four per link) [23], [24], giving an analytically-derived regressor $\mathbf{Y}_g(\mathbf{q}) \in \mathbb{R}^{6\times 24}$, computed directly from the robot's forward kinematics following the standard construction in [23], [24], such that $\mathbf{G}(\mathbf{q}) = \mathbf{Y}_g(\mathbf{q})\boldsymbol{\phi}$. Since gravity torque depends only on pose $\mathbf{q}$, not on velocity or acceleration, we calibrate $\boldsymbol{\phi}$ by executing the replay trajectories while repeatedly stopping the robot at many points along each trajectory, ensuring it is fully static, and reading the joint torque at each stop. We fit $\boldsymbol{\phi}$ via ridge-regularized least squares over these static readings; using poses drawn from the task's own trajectories, rather than generic randomized poses spread across the whole workspace, produced a substantially better fit exactly where it is needed. A residual gravity error remains after this fit; we correct it with a small neural network $\mathbf{G}_{res}(\mathbf{q})$ (two hidden layers, width 32, ReLU), taking only $[\sin(\mathbf{q}), \cos(\mathbf{q})]$ as input, trained exclusively on this same static data, keeping it decoupled from the velocity/acceleration-dependent terms below.

Mass and Coriolis. The same parametrization extends to $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ [24], now with a 72-dimensional parameter vector $\boldsymbol{\pi}$ (ten inertial parameters per link, two friction parameters per joint) and a corresponding regressor $\mathbf{Y}_{full}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \mathbf{Y}_{full}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\,\boldsymbol{\pi}. \tag{12}$$

Unlike gravity, these terms depend on velocity and acceleration, so we instead execute the same trajectories continuously, without stopping, and record the joint torque throughout the motion. Since this torque already includes the (now-modeled) gravity contribution, we subtract the gravity estimate described above first; the remainder is attributed to mass and Coriolis effects and used to fit $\boldsymbol{\pi}$ via the same ridge-regularized least squares. We use this linear regressor in our final pipeline, as it is simple, numerically stable, and requires no additional training beyond the least-squares fit itself.