PTLD: Sim-to-Real Privileged Tactile Latent Distillation for dexterous manipulation

Author Names Omitted for Anonymous Review. Paper-ID [804]

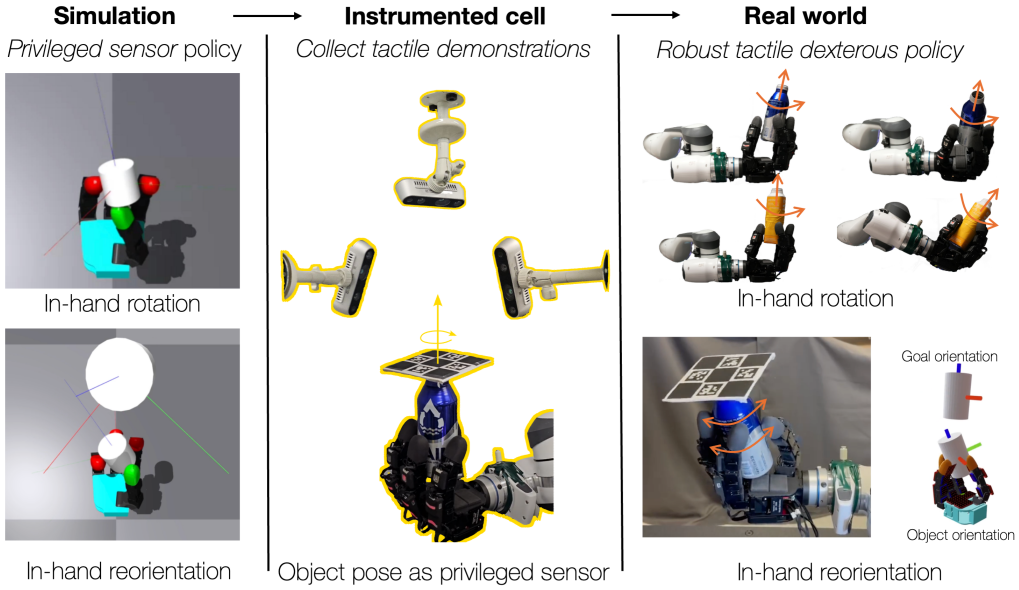

Fig. 1: PTLD: sim-to-real Privileged Tactile Latent Distillation is an approach to learn tactile dexterous policies without simulating tactile sensors. First, *Privileged sensor* policies are trained in simulation using Reinforcement learning which produces strong policies. These policies are deployed in instrumented real-world setups to collect tactile demonstrations. Finally, a tactile state estimator is trained from tactile demonstrations to obtain robust real-world deployable tactile policies. With PTLD, we demonstrate that **in-hand rotation** is robust to object property changes such as slip, mass, and wrist orientation changes, and that performance for the challenging task of **in-hand reorientation** improves significantly by over 57% with tactile sensing, when compared to proprioception only policy.

***Abstract*— Tactile dexterous manipulation is essential to automating complex household tasks, yet learning effective control policies remains a challenge. While recent work has relied on imitation learning, obtaining high quality demonstrations for multi-fingered hands via robot teleoperation or kinesthetic teaching is prohibitive. Alternatively, with reinforcement we can learn skills in simulation, but fast and realistic simulation of tactile observations is challenging. To bridge this gap, we introduce PTLD: sim-to-real Privileged Tactile Latent Distillation, a novel approach to learning tactile manipulation skills without requiring tactile simulation. Instead of simulating tactile sensors or relying purely on proprioceptive policies to transfer zero-shot sim-to-real, our key idea is to leverage *privileged sensors* in the real world to collect real-world tactile policy data. This data is then used to distill a robust state estimator that operates on tactile input. We demonstrate from our experiments that PTLD, can be used to improve proprioceptive manipulation policies trained in simulation significantly by incorporating tactile sensing. On the benchmark in-hand rotation task, PTLD achieves a 182% improvement over a proprioception only policy. We also show that PTLD enables learning the challenging task of tactile in-hand reorientation where we see a 57% improvement in number of goals reached over using proprioception alone.**

I. Introduction

Contact-rich dexterous manipulation with multi-fingered robot hands has remained a grand goal in robotics for several decades. The potential to solve tasks with human-like dexterity and use tools designed for humans paves a path toward physical intelligence in areas such as healthcare and household tasks. Recent work in learning from demonstrations [1]–[3] has provided a scalable recipe for learning new policies by collecting large demonstration datasets via robot teleoperation, hand-held grippers, or kinesthetic teaching. However, such approaches are impractical for multi-fingered dexterous hands due to the difficulty of reliably teleoperating hardware for intricate tasks like using a screwdriver, a wrench, or turning a doorknob. Kinesthetic teaching is equally challenging when more than two fingers are required for tasks such as in-hand object reorientation [4]. While hand-held grippers are promising [3], [5], they require exoskeleton structures that must balance flexibility (degrees of freedom) and stability; consequently, recent successes have largely been limited to simple tasks.

Sim-to-real reinforcement learning (RL) offers an alterna-

tive for learning dexterous tasks and has seen tremendous success in robot locomotion [6], [7]. However, most existing approaches focus on blind, proprioception-only policies for both locomotion [8]–[10] and manipulation [11], [12]. While a few works have succeeded with perceptive policies [13]–[15], training visual policies in simulation is relatively slow due to the overhead of image rendering. Furthermore, these policies are often challenged by a significant sim-to-real gap.

Our focus in this work is on *tactile dexterous manipulation*, encompassing dynamic tasks such as in-hand rotation, reorientation, and pinch-to-power grasp transitions [11], [16]–[19]. The standard approach for these tasks is RL. However, akin to challenges in training visual policies, there are several hurdles to training tactile sensorimotor policies in simulation. First, simulating tactile sensors accurately is difficult; therefore, most existing works [18], [20] resort to simplified models, such as single-point or binary contact. Second, existing tactile simulation packages [21]–[23] are not standardized for all tactile sensors, and primarily rely on rigid-body simulation. Third, even when using approximate soft-body simulation, a large sim-to-real gap often prevents straightforward deployment.

In this paper, we present a new approach to learning tactile manipulation policies without paying the high cost of simulating tactile sensors. Our method takes inspiration from *privileged latent distillation*, where an "oracle" policy is first trained with access to privileged state information (available in simulation) and then imitated by a zero-shot deployable policy using a perceptive state estimator that only has access to partial observations such as vision or proprioception. This concept has appeared in literature under various names, such as learning by cheating [24], data-driven planning via imitation [25], or rapid motor adaptation (RMA) [26], and has achieved significant empirical success.

We extend privileged latent distillation in two distinct ways. First, we extend policy distillation from simulation to the real world. To achieve this, we require an executable oracle policy in the real environment. To this end, second, we treat properties such as object pose and shape as *privileged sensors* and deploy these *privileged sensor* policies in the real world by instrumenting a robot cell. Once the privileged policy is executing in reality, we distill its latent representations into a tactile policy through supervised learning using a paired dataset of tactile observations and privileged latents.

Privileged latent distillation [24]–[26] is typically implemented with a teacher-student setup requiring two stages: oracle training and student distillation. Because our method relaxes student distillation to leverage a few privileged quantities requiring an additional round of real-world distillation, the standard two-stage simulation training process can become laborious. Therefore, we also present architectural advancements demonstrating that the initial two-stage simulation approach can be replaced with an asymmetric actor-critic training step, requiring only a single round of training.

Finally, we demonstrate successful results on the significantly harder task of continuous tactile in-hand reorientation—a task that cannot be accomplished in simulation using proprioceptive history alone.

In summary, our contributions in this paper are threefold:

- We present a novel approach to learn sensorimotor tactile dexterous manipulation policies without paying the cost of simulating tactile sensors. We use privileged sensors as the interface between simulation and reality to perform *privileged latent distillation* using real world data.
- We present architectural advances for training manipulation policies, simplifying the two-stage distillation step in simulation into a single training step.
- Through our experiments we demonstrate that tactile policies trained through our sim-to-real latent distillation approach consistently outperform proprioception policies as well as adaptation based tactile policies in both robustness and performance.

II. Related work

A. Dexterous In-hand Manipulation

Dexterous in-hand manipulation has been an active area of research for decades [15], [16], [27]–[30]. It features the cooperative use of multiple fingers on a multi-fingered hand to grasp and manipulate objects. While classical approaches need a physical model of the object and robot geometry to plan robot finger motions [31], [32], recent approaches have had success with using RL directly to learn policies in a model free manner [11], [15], [16], [20], [33]. However, RL approaches face the *sim-to-real* gap i.e., it is challenging to reproduce real world sensor observation and physics in simulation. Even for modalities such as vision where it is feasible to simulate the sensor, the simulation model is physically inaccurate and does not describe the real world sensors, therefore extensive visual domain randomization [13], [34] is crucial. Our method, on the other hand trains in simulation with observations such as object poses and object shape that do not suffer a large sim-to-real gap, but requires one to forgo the zero-shot sim-to-real deployment assumption.

B. Privileged distillation

Partial observability is a significant challenge in RL. For dexterous in-hand manipulation of object, estimating precise contact dynamics is quite important as it determines the object motion and subsequently the requisite action. In the absence of sensors to estimate these properties, most existing approaches in the literature have prominently used privileged distillation [11], [24]–[26], [35], [36] to learn dexterous manipulation policies. As described in III, first one trains an oracle policy that has access to privileged information only available in simulation, then one distills it into a deployable policy that produces probabilistic estimates of the privileged state using a small history of sensor observations. Concretely, this is usually a history of proprioceptive observations or visual observations.

C. Tactile sensing and Representation learning

The tactile modality has long been promised to be imperative for such contact-rich dexterous manipulation. The last decade has seen a plethora of tactile sensors introduced for manipulation ranging from vision-based tactile sensors such as the

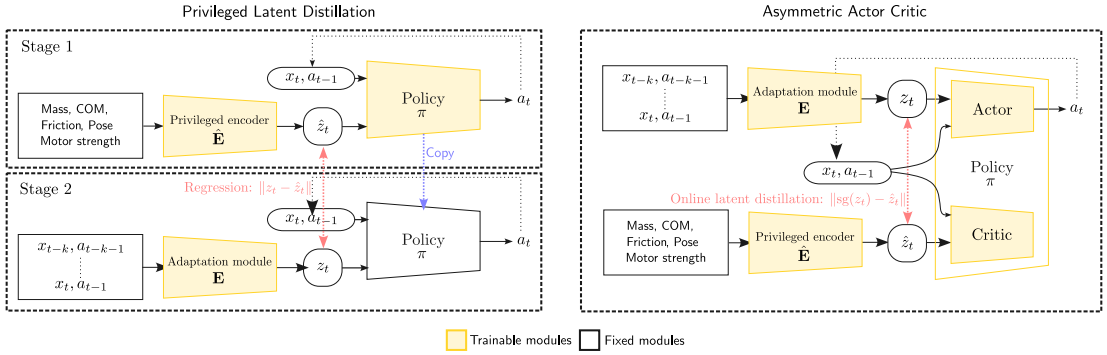

Fig. 2: (left) *Privileged latent distillation* is a two stage approach to training policies in simulation. An *oracle* policy with privileged information is trained in stage 1, then it is distilled into a deployable policy in stage 2 (in simulation). **(right)** Asymmetric Actor Critic is a single stage approach where two networks actor and critic respectively are trained simultaneously. The critic is provided with privileged information and learns the value function, while the actor is only given deployable partial sensor information

GelSight [37] and DIGIT [38], magnetic-skin tactile sensors such as ReSkin [39] and Xela [40], to resistive and capacitive sensing sensors [41]. Consequently, there also exists a rich body of work that leverages tactile sensors for perception tasks and simple manipulation tasks such as peg-insertion [42], [43], cable manipulation [44] and planar pushing [45]. However, these tactile manipulation tasks in the literature are chosen carefully to satisfy the following features: a) the task is often quasi-static and b) the tasks are simple to demonstrate to make them amenable to behavior cloning methods [1], [2]. Recently, self supervised tactile representations [17], [42], [46], [47] address the lack of standardization in tactile sensors in robotics, demonstrating their use in several manipulation tasks, although the tasks chosen are largely quasi-static. Of these, [17] notably proposed a tactile adaptation algorithm to adapt RL trained dexterous manipulation policies in the real world to use tactile sensing, along the lines of policy finetuning [48] using real world data. However, these methods are fundamentally limited(see Section VI-C) and only gain from rejection sampling of successful real-world trajectories, as the performance ceiling of the proprioceptive teacher policies are limited. In contrast, PTLD produces quantitatively more robust policy behaviours.

III. Background

A. Notation

We model the dexterous manipulation tasks we discuss in the paper as finite horizon ($N \in \mathbb{N}$) Partially Observable Markov Decision Processes (POMDPs) $\mathbf{M} \triangleq (\mathcal{S}, \mathcal{A}, \mathcal{X}, \mathcal{P}, \mathcal{R})$ where $(\mathcal{S}, \mathcal{A}, \mathcal{X}) \in \{\mathcal{S}_t, \mathcal{A}_t, \mathcal{X}_t\}_{t=1}^{N}$ denote the state, action and observation spaces over the finite horizon N respectively. $\mathcal{P} = \{\mathcal{P}_t : \mathcal{S}_{t-1} \times \mathcal{A}_{t-1} \to \mathcal{S}_t\}$ denotes the transition dynamics, and $\mathcal{R} = \{\mathcal{R}_t : \mathcal{S}_t \times \mathcal{A}_t \to [0, 1]\}$ denotes the reward function. Our goal is to learn policies $\pi : \mathcal{X}_{t-k+1:t} \times \mathcal{A}_{t-k:t-1} \to \mathcal{A}_t$ via Reinforcement Learning (RL) to maximize the expected return $G(\tau) = \sum_{t=0}^{N} \gamma^t R_t$ over the horizon as $\pi^* = \arg\max_{\tau \sim \mathcal{P}^\pi} \mathbb{E}\left[G(\tau)\right]$.

Specifically, we choose to parameterize the policy with a neural network that is a combination of an encoder $\mathbf{E}$ which encodes the observations $\mathcal{X}$ into a latent space $\mathcal{L}$, which is then consumed by the policy π to produce actions $\mathcal{A}$. Typically, we employ two encoders during training. First, we have $\hat{\mathbf{E}}$ the privileged encoder which has access to privileged observations in simulation, and $\mathbf{E}$ the adaptation encoder which only has access to deployable observations. Then we have the policy as follows:

$$\mathcal{A}_t \sim \pi(\mathbf{E}(\mathcal{X}_{t-k+1:t}), \mathcal{A}_{t-k:t-1}) \tag{1}$$

B. Privileged latent distillation

Privileged latent distillation is a two stage approach to learning deployable policies in the real-world (see Fig. 2). This requires that the simulation environment for the task supports a) privileged state observations $\mathcal{X}^{\text{priv}}$ which are typically low dimensional quantities such as object, robot and contact states and b) sensor observations $\mathcal{X}^{\text{sensor}}$ which can be realized in the real world such as proprioception and rendered camera. First, an oracle policy is trained that is allowed to 'cheat' and observe the full privileged state $\mathcal{X}^{\text{priv}}$ that describes the environment wholly. Then, a deployable student policy is trained to imitate the oracle policy given only the sensor observations $\mathcal{X}^{\text{sensor}}$. Since the sensor observations only observe the state partially, most approaches employ *frame stacking* where a history of sensor observations, and previous actions are used as the observation.

The oracle policy is trained in simulation using an RL algorithm such as PPO [49], while distillation is usually implemented as supervised learning. Implementations typically use a) action imitation where the actions between the oracle and student policies are matched or b) latent imitation where an encoded latent between the oracle and student policies is supervised. Specifically, we have

$$\mathcal{L}_{\text{action}} = \left\| \hat{\pi}(\hat{\mathbf{E}}(\mathcal{X}^{\text{priv}}), \mathcal{A}) - \pi(\mathbf{E}(\mathcal{X}^{\text{sensor}}), \mathcal{A}) \right\| \tag{2}$$

$$\mathcal{L}_{\text{latent}} = \left\| \hat{\mathbf{E}}(\mathcal{X}^{\text{priv}}) - \mathbf{E}(\mathcal{X}^{\text{sensor}}) \right\|. \tag{3}$$

It must be noted that experience or observations collected for distillation is collected by the deployable (student) policy, while the supervision signal comes from the oracle encoder. This is importantly distinct from traditional offline imitation learning as student distillation implements an on policy variant of DAgger [50]. Specifically, since the student policy is supervised

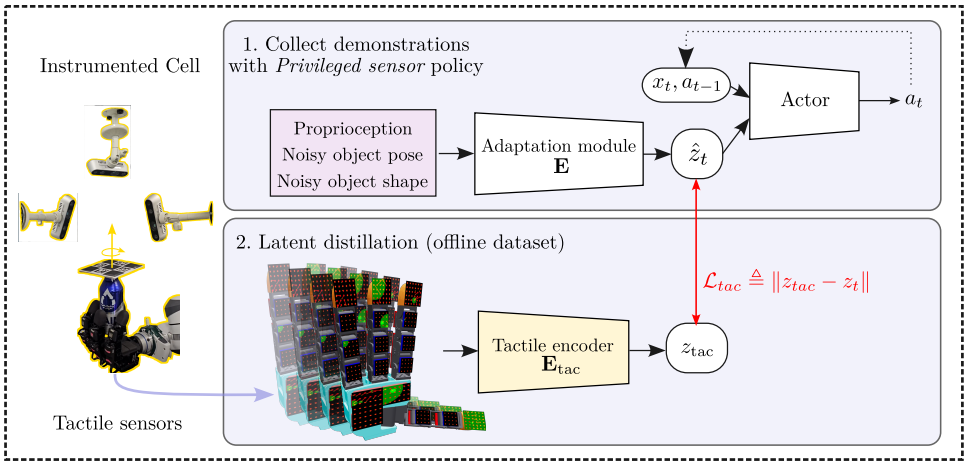

Fig. 3: A simplified illustration of PTLD. Once we have a *privileged sensor* policy trained in simulation using AAC, first we collect demonstrations in the real world by deploying the policy, and additionally collect deployment sensor observations. Then, we train a deployment encoder (tactile encoder in this case) to recover the latents from the *privileged sensor* policy using an offline dataset.

by the teacher on experience collected by the student, the student observes a wider observation space during training, resulting in a robust policy.

C. Asymmetric Actor Critic

Asymmetric Actor Critic (AAC) [51] is another approach which aims to learn robust deployable policies by taking advantage of full-state observability in simulation. Specifically, it employs an actor-critic framework, where the critic is provided with the privileged state $\mathcal{X}^{\text{priv}}$, while the actor is provided with $\mathcal{X}^{\text{sensor}}$ sensor observations (see Fig. 2). In our approach, we employ learning policies with AAC, as opposed to RMA [26] as it simplifies policy learning in simulation into a single training step.

IV. PTLD: Privileged Tactile Latent Distillation

We now describe our method to train tactile manipulation policies using sim-to-real tactile distillation along with architectural improvements that simplify training in simulation.

A. Online distillation with Asymmetric Actor Critic

We employ an asymmetric actor-critic framework for training manipulation policies in our work as opposed to privileged latent distillation. As alluded to before, this simplifies training into a single stage in simulation. Crucially, we find that parameterizing the actor as separated observation encoder $\mathbf{E}$ and policy π is beneficial, as a separate encoder allows one to learn general state representations. Therefore inspired by self-distillation in representation learning [17], [42], [52], [53] approaches, we employ a self-distillation representation loss between the latent representations learnt by the critic (privileged) encoder and the actor (student) encoder:

$$\mathcal{L}_{\text{latent}} \triangleq \left\| \mathbf{E}(\mathcal{X}^{\text{sensor}}) - \text{sg}(\hat{\mathbf{E}}(\mathcal{X}^{\text{priv}})) \right\| \tag{4}$$

where sg denotes stop gradient. This loss encourages the actor encoder to recover privleged object information available to the critic while operating on partial observations. In our experiments (see Section VI-B), we find that this online latent distillation loss improves training reward achieved by the policy. Furthermore, in simulation evaluation, this simple distillation loss results in similar performance to privileged latent distillation, motivating the simplification from two-stage training to single-stage training in simulation.

As shown in Fig. 2, these networks are trained simultaneously, and we use the clip variant of proximal policy optimization (PPO) [49] augmented with the online latent distillation loss:

$$\mathcal{L}_{\text{PPO}} \triangleq \mathcal{L}_{\pi}^{\text{CLIP}}(\mathbf{E}, \pi) + c_V \mathcal{L}_V(\hat{\mathbf{E}}, V) + \mathcal{L}_{\text{entropy}}(\mathbf{E}, \pi) \tag{5}$$

$$\mathcal{L} \triangleq \mathcal{L}_{\text{PPO}} + c_{\text{latent}} \mathcal{L}_{\text{latent}} \tag{6}$$

where c_{latent} is a weighting factor. We optimize the total loss $\mathcal{L}$ via backpropagation.

B. Privileged sensors for Sim-to-Real Tactile Distillation

The essence of our method relies on the observation that allowing the actor access to *privileged sensors* that provide higher observability into the state, such as object pose and object shape, significantly improves policy performance in simulation compared to policies that take as input only partial observations $\mathcal{X}^{\text{sensor}}$. Despite this gap, most prior work still uses the suboptimal policy with simplified sensor inputs that are available at deployment time to achieve zero-shot sim-to-real transfer, sacrificing policy performance for simple input sensor modalities. In contrast, the *privileged sensor* policy cannot be directly deployed in the real world as these observations are naturally inaccessible at deployment time. To overcome this limitation while still benefiting from the privileged policy, we instrument a real-world cell with multiple cameras and object markers to provide (noisy) object poses $\mathbf{T}_t^W \in \mathbf{SE}(3)$ as the real world *privileged sensor*, and additionally also *sensorize* the multi-fingered robot hand with tactile sensors. Then we deploy the *privileged sensor* policy in the real-world cell and collect an offline dataset of *on policy demonstrations*, recording both the latent representations produced by the policy and the associated tactile sensor observations (Fig. 3). Using this data, we train an observation encoder that takes as input both tactile sensor data and proprioception data to match the latents from

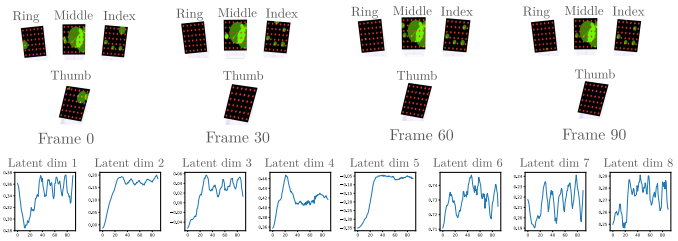

Fig. 4: Visualization of tactile observations and the latents changing over the first 1 second of *privileged sensor* policy deployment. Here we visualize only the tactile data at the robot fingertip for simplicity, however the tactile encoder takes as input all observations from the hand.

the privileged sensor policy. Formally, as illustrated in Fig. 3 we distill the *privileged sensor* policy into a tactile policy and use the MSE loss for supervision.

Since the tactile encoder does not have access to a simulator for interactive training, naïve offline distillation can suffer from distribution shift. To address this, we employ DAgger [50], where we iteratively train the tactile encoder with an aggregated dataset where experience is collected by the policy using intermediate trained tactile encoders.

V. The Privileged Tactile Manipulation System

A. Real world privileged sensor cell

a) Robot Setup: For all experiments, we use the Allegro hand sensorized with Xela uSkin [40], attached to a Franka Panda robot arm. There are a total of 18 Xela uSkin sensing pads on the Allegro hand amounting to a total of 368 individual sensors. We use the continuous 3-axis raw tactile sensor measurements from the Xela sensor instead of the processed force measurements as we find that those measurements contain significant hysteresis and lag. A baseline signal with no contact is additionally collected (over 2 minutes) and subtracted from the raw tactile measurements before data collection each time.

b) Privileged Sensors Setup: To deploy the privileged sensor policies for latent dataset collection, we instrument the real world robot cell with 4 Realsense D435i/D435 RGBD cameras which view the in-hand manipulation area. These cameras are calibrated jointly and track an Aruco marker attached to the object being manipulated to produce reliable multi-view pose estimate that is refined via Pose Graph Optimization (PGO) [54]. We assume known shapes of the objects that are being deployed.

B. Manipulation task I: In-hand rotation

We choose the task of in-hand rotation [11], [18], [55] about the z-axis for our experiments. In this task the robot hand is required to rotate an object along a specified axis, ensuring that the object does not drop and remains held by the robot fingers. We demonstrate that task performance on this task can be vastly improved by incorporating rich tactile information as part of the policy observation. For state observation and reward details for this task, we refer the reader to [11].

a) Tactile encoder: For the tactile encoder (Fig. 5 (a)), we concatenate the tactile observations $\mathcal{X}^{\text{tactile}} \in \mathbb{R}^{368\times 3}$, and the tactile sensor positions computed from the Allegro joint states using forward kinematics $\mathcal{X}^{\text{sensor\_pos}} \in \mathbb{R}^{368\times 3}$ as

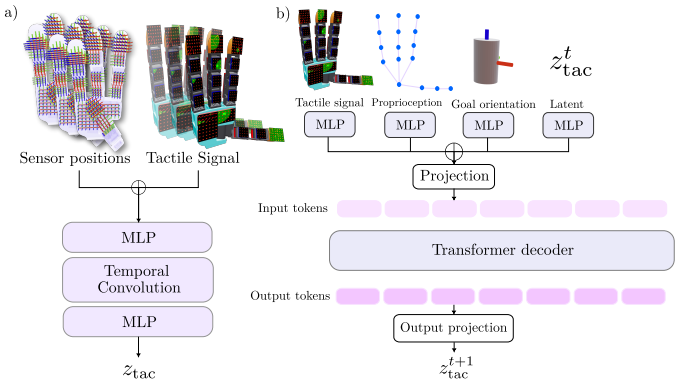

Fig. 5: Tactile encoders for manipulation tasks: a) For in-hand rotation, we concatenate a history of tactile signals and sensor positions, and encode them using a 1D temporal convolution network to predict the tactile latents. b) For in-hand reorientation, we concatenate, tactile signals, proprioception, goal orientations and past latents and embed them with a causal transformer to produce future tactile latents.

the input $\mathcal{X} = \text{cat}(\mathcal{X}^{\text{tactile}}, \mathcal{X}^{\text{sensor\_pos}}) \in \mathbb{R}^{368\times 6}$. The tactile observations are produced at 100Hz, and we use a history of 0.5s of tactile data as input to the encoder. Specifically, our tactile encoder uses a combination of MLP and 1D temporal convolution as follows: ($\mathcal{X} \rightarrow$ MLP $\rightarrow$ Temporal conv $\rightarrow$ MLP $\rightarrow z$), where z is the predicted latent.

C. Manipulation task II: In-hand reorientation

We also develop an in-hand general purpose re-orientation task designed as a goal orientation reaching problem. We wish to demonstrate that PTLD can not only be used to improve existing policies that can be deployed in the real world with proprioception, but that we can also learn more difficult policies which require additional information to be encoded. In this task, we randomly sample goal orientations $\mathbf{R}_t^{\text{goal}} \in \mathbf{SO}(3)$ of an object, and the robot is tasked to manipulate the object held by the fingers, such that it's pose $\mathbf{R}_t^{\text{object}}$ reaches within a set threshold angular distance ($\delta \triangleq 0.25$ rad $\simeq 14.5°$) of the goal orientation (see Fig. 1). However, due to limitations in our hardware in the real robot cell (i.e., the four cameras are placed above the manipulation area), we sample goal orientations in the upper hemisphere within $\theta = 40°$ about the z-axis. We use the asymmetric actor critic framework trained with PPO to solve this task as well.

a) State: The state observation input to the adaptation module $\mathbf{E}$ in simulation includes the Allegro joint states $q_t \in \mathbb{R}^{16}$, previous joint targets $\tilde{q}_t \in \mathbb{R}^{16}$, noisy object position $p_t \in \mathbb{R}^3$, noisy object orientation $\mathbf{R}_t \in \mathbf{SO}(3)$, and goal orientation $\mathbf{R}_t^{\text{goal}} \in \mathbf{SO}(3)$. All orientations use the the 6-D rotation representation [56]. We also concatenate the relative difference between the goal and current object pose to the goal pose, to facilitate learning. Here, we note that, without current object orientation input to the actor, we are unable to learn a successful policy, even in simulation. Further, since a single observation maybe insufficient, we employ *frame stacking* and provide the actor with a 30-step history ($\sim 1.5s$ for a control rate of 20Hz) of these observations. On the other hand, the privileged encoder $\hat{\mathbf{E}}$ is also provided with object linear and angular velocity, fingertip states (position, orientation and

velocities.) in addition to the inputs of $\mathbf{E}$.

b) Reward: We train the policy with a mixture of rewards to ensure that the robot not only succeeds in reaching goal poses, but also employs natural realizable finger gaits. The reward (Table V) contains three main terms $(r_{\text{goal}}, r_{\text{success}}, r_{\text{streak}})$, which includes the rotational distance reward, success bonus and a streak bonus to encourage the agent to reach multiple goals one after another. We also encourage the policy to maintain contact with the fingertips and keep the object in the center using $(r_{\text{contact}}, r_{\text{position}})$. In addition to the positive rewards, we also use motion penalties and energy penalties to obtain smooth motions. Crucially, we find that penalizing $r_{\text{finger\_pose}}$ the robot finger motion from the initial finger pose is useful in generating a finger gait. Without this penalty, the policy learns to curl the fingers along with the object, resulting in unrecoverable hand configurations. We detail other motion and energy penalties for reward shaping in the Appendix.

c) Reset: We start the episode with a stable grasp sampled from a grasp set as commonly implemented, and reset the goal orientations multiple times within an episode when the object orientation reaches the goal orientation ($\leq \delta$). Additionally, we use a z-height threshold to reset the episode when the robot hand drops the object from its fingers or the object slips to unrecoverable states. Finally, we also use the standard episode reset after a fixed number of simulation steps.

Reward	**Scale**
$r_{\text{goal}} \triangleq \frac{1}{d(\mathbf{R}_t^{\text{object}}, \mathbf{R}_t^{\text{goal}}) + \epsilon}$	2.0
$r_{\text{success}} \triangleq (1 \text{ if } d(\mathbf{R}_t^{\text{object}}, \mathbf{R}_t^{\text{goal}}) \leq \delta \text{ else } 0)$	5.0
$r_{\text{streak}} \triangleq \frac{N_{\text{success}}}{N_{\text{max\_success}}}$	2.0
$r_{\text{contact}} \triangleq \sum_i (C_i > \delta_{\text{contact}})$	0.1
$r_{\text{position}} \triangleq \|p_t - p_0\|$	0.05
$r_{\text{finger\_pose}} \triangleq \|q_t - q_0\|$	-1.0
$r_{\text{fingertip\_object}} \triangleq \sum_i \|p_{\text{fingertip}_i} - p_t\|$	-0.2

TABLE I: Reward function for any target in-hand reorientation

d) Tactile encoder: Since the in-hand reorientation task is significantly more complex, and requires the policy to change gait according to the relative difference between current object pose $\mathbf{R}_t^{\text{object}}$ and the goal pose $\mathbf{R}_t^{\text{goal}}$, for this task we use a *recursive state estimator* as the tactile encoder. This estimator reasons about the full sequence of tactile observations, proprioception, and goal orientations of the object. To this end, we use a Transformer network (see Fig. 5 (b)) to auto-regressively predict the current latent given the history of observations and latents. Specifically, we embed each input, tactile observations $\mathcal{X}_t^{\text{tactile}} \in \mathbb{R}^{368\times 3}$, proprioception input $\mathcal{X}_t^{\text{proprio}} \in \mathbb{R}^{32}$, the goal orientations $\mathbf{R}_t^{\text{goal}} \in \mathbf{SO}(3)$ and the previous latent $z_t \in \mathbb{R}^8$, individually using 2-layer MLP networks. We subsequently concatenate all the embeddings, linearly project it and add positional encodings to produce the input embedding to the transformer. Finally, the embeddings are processed by a decoder (causal) transformer, from which we select the next predicted latent z_{t+1} during inference.

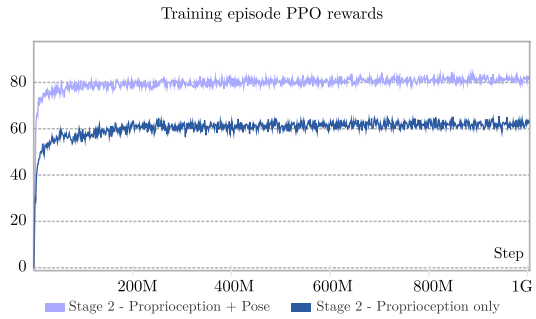

Fig. 6: Policy performance for stage 2 distillation step in simulation improves significantly when object pose information is provided in addition to proprioception input

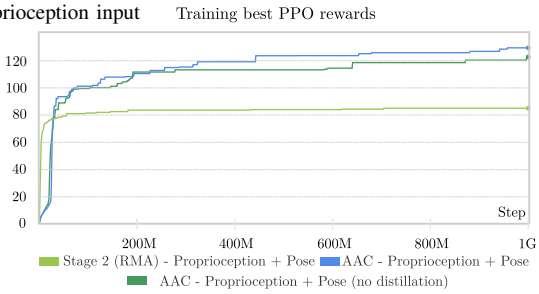

Fig. 7: Asymmetric actor critic (blue) trained in a single stage in simulation outperforms the RMA distillation approach which requires two stage training in simulation

VI. Experiments

A. Implementation details

We train the policies using PPO [49], and use IsaacGym [57] as our simulator. Once an offline dataset is collected in the real-world cell, we train the tactile encoder via supervised learning. For optimizing the tactile encoder, we use AdamW optimizer, with a learning rate of $1\text{e}{-4}$. We use ROS2 for communication between the different robot processes, and achieve a real-time policy deployment rate of $\sim$ 20Hz for both in-hand rotation and in-hand reorientation.

B. Privileged sensors improve performance in simulation

We first verify that access to privileged sensors substantially improves policy learning in simulation. In Fig. 6, we demonstrate that policies trained with object pose in addition to proprioception achieve significantly higher rewards during the distillation stage than baseline trained with proprioception alone. We further evaluate simulation performance under randomized object properties in Table II, using the quality metrics established in [11]: (1) *Rotation reward* defined as $\omega \cdot k$ where k denotes the z-axis, (2) *Time to Fall* – defined as the fraction of the episode completed before a drop – and (3) *Undesired rotation penalty*, which penalizes any off-axis rotation. As expected, providing object pose information to the Stage 2 policy yields improvements across all metrics.

We also compare our single-stage AAC approach with the two-stage RMA distillation approach. In Fig. 7, we observe that while the RMA stage 2 distillation step learns much faster initially, the AAC approach, and more specifically the AAC variant which uses the online latent distillation (see Section IV-A) eventually outperforms RMA and leads to

Method	Input modalities	z-axis RotR ↑	z-axis TTF ↑	z-axis RotP ↓
Oracle		**159.4**	**0.89**	**31.86**
Latent distillation (RMA [11])	Proprioception	139.0	0.79	28.53
Latent distillation (RMA [11])	+ Pose	153.1	0.86	31.09
AAC	Proprioception	141.91	0.80	27.83
AAC	+ Pose	168.58	0.88	26.51
AAC (no distillation)	+ Pose	161.48	0.85	29.74

TABLE II: We compare the performance improvement over various baselines on z-axis in hand rotation, under the same training setting *in simulation*. Specifically, compared to [11], we first demonstrate that Asymmetric Actor critic with latent supervision (AAC) improves performance in simulation significantly. Further, we demonstrate that additional input modalities such as object shape and pose produce significant improvements in simulation, which is a pre-requisite for tactile distillation to outperform the baselines.

policies with better real-world behaviors. Finally, in Table II we also see that the AAC policy with pose as the privileged sensor input performs similarly to the RMA policy. This motivates our choice to simplify training in simulation to a single AAC training stage.

C. PTLD *improves policy robustness in the real world*

We evaluate PTLD on hardware against several proprioceptive and tactile baselines, including RMA [11], the two-stage distillation approach in simulation, and AAC, our single-stage asymmetric actor-critic variant. To isolate the specific contribution of tactile feedback, we also implement a real-world proprioceptive distillation baseline, which verifies that tactile signals actively recover privileged information rather than simply providing additional data for encoder finetuning. Furthermore, we compare against a Tactile Adaptation baseline [17] adapted for Xela sensors using Sparsh-Skin [42] representations, which involves distillation from a proprioception-only policy. We conducted 10 trials per method across two cylindrical objects (see Appendix), quantifying performance through Total Rotation (angular displacement), Time to Fall (TTF) (manipulation duration), and Vertical Drift (indicator of instability).

As illustrated in Figure 8, PTLD outperforms all baselines by a significant margin, in terms of total rotation and TTF. We attribute this improvement to the fact that PTLD overcomes a core limitation of traditional tactile adaptation by sourcing teacher data from a high-fidelity privileged sensor policy in the real world, allowing the student policy to reach a much higher performance ceiling. Additionally, compared to proprioception-only distillation, the inclusion of tactile feedback enables the model to capture contact dynamics that are otherwise unobservable; without these sensors, real-world distillation often regresses into modest policy finetuning. Qualitatively, the tactile policy demonstrates sophisticated recovery behaviors, such as adaptive finger-gaiting adjustments in response to object slippage, inherited directly from the privileged teacher (see Fig. 8 (bottom)). In the counter-clockwise rotation policy metrics, although RMA proprioception outperforms PTLD in terms of vertical drift, qualitatively we observe that the policy tilts the object resulting in high undesired rotation off the z-axis.

Pose parameterization	Distillation modality	Avg. Rotation error over 100 steps Rad (↓)
Absolute object pose	Proprioception	$0.43_{\pm 0.11}$
Absolute object pose	Proprioception + Tactile	$\mathbf{0.21}_{\pm \mathbf{0.03}}$
Relative object pose	Proprioception	$0.95_{\pm 0.09}$
Relative object pose	Proprioception + Tactile	$\mathbf{0.26}_{\pm \mathbf{0.05}}$

TABLE III: Average cumulative rotation error over 4.5 seconds (30 inference steps) for a decoder trained to recover object orientation from latents learned after real-world distillation. We freeze the proprioception / tactile encoder and only train the decoder.

D. Tactile information enhances object state estimation

While previous sections established that PTLD improves policy robustness, here we analyze the information overlap between the privileged sensor policy and the tactile policy i.e., how much object orientation information is encoded in the tactile latent.

We trained a 6D rotation decoder (supervised via MSE loss) on tactile latents produced by student tactile encoder using tactile policy rollouts. Table III compares the cumulative orientation error over 30 prediction steps for encoders using proprioception alone versus combined proprioception and tactile feedback. We evaluated two pose parameterizations:

1) Absolute Pose ($\mathbf{R}_t^W$): The object's orientation relative to the hand coordinate frame.
2) Relative Pose ($\mathbf{R}_t^{t-H}$): The orientation relative to the start of a temporal window H.

In both cases, incorporating tactile information significantly reduced rotation prediction errors. Notably, proprioception-only encoders performed poorly with relative pose inputs. We attribute this to the multi-modal nature of the task i.e., similar proprioceptive finger-gait patterns can correspond to vastly different object motions due to unobserved slippage or sliding. Tactile sensors disambiguate these cases by providing direct contact signals. Conversely, with absolute pose, the decoder likely learns a fixed transform since the hand remains stationary during deployment in this dataset. Qualitative results for absolute pose estimation are visualized in Figure 9.

E. Tactile object In-hand reorientation

As detailed in Section V-C, we trained an in-hand reorientation policy with privileged access to object and goal poses. This task serves as a benchmark for PTLD, demonstrating its ability to learn complex policies that demand high-precision object state estimation. We observed that a temporal-convolutional encoder was insufficient for this task, often collapsing into a single finger-gaiting mode, rotating the object in only one direction. To address this, we implemented an Autoregressive Transformer-based encoder, which captures the long-range dependencies required for multi-directional reorientation. We evaluate performance using two metrics: the number of goals reached (N_{goals}) and the Time to Fall (TTF). As shown in Table IV, removing tactile information from the Transformer

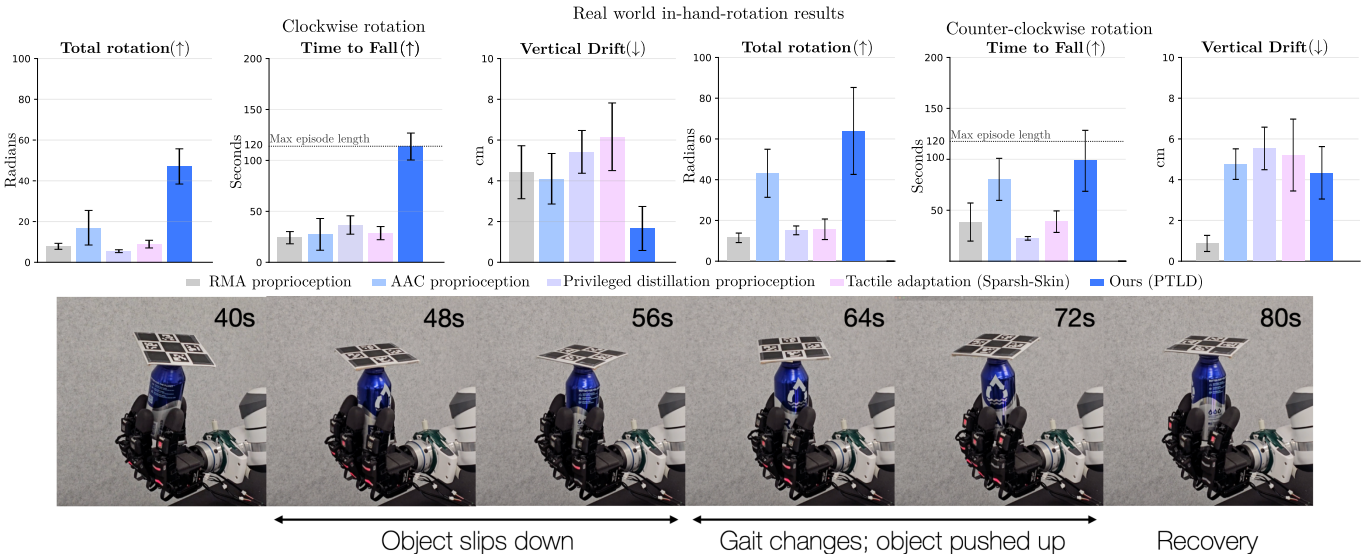

Fig. 8: (top) Real world comparison of in-hand rotation policy performance over 10 trials with three cylinder like objects. PTLD consistently outperforms all baselines by a large margin **(bottom)** We observe that with PTLD, the tactile policies show recovery behavior where finger gaiting patterns change to keep the object pose in a

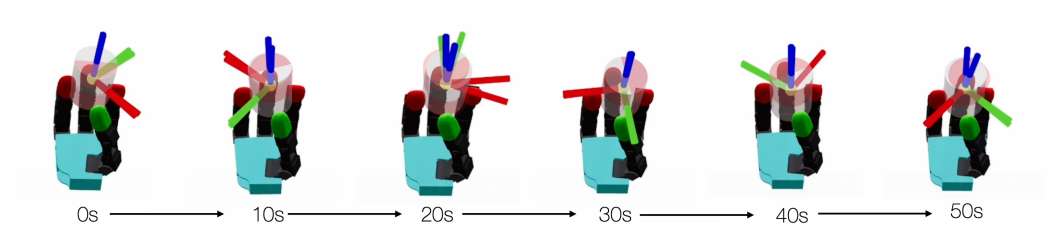

Fig. 9: Visualization of object pose reconstruction from tactile latent decoder: The red transparent cylinder denotes the predicted object pose prediction, while the gray translucent cylinder denotes the true object pose recorded from the instrumented cell during deployment. Specifically, we compose the predictions over each second to visualize the cumulative tactile object pose reconstruction over time.

	In-hand reorientation	
Method	$N_{\text{goals reached}}$ (↑)	TTF (s) (↑)
Autoregressive Transformer **(proprio)**	$2.1_{\pm 2.14}$	$10.99_{\pm 10.15}$
Autoregressive Transformer **(tactile+proprio)**	$\mathbf{3.3}_{\pm 1.55}$	$\mathbf{13.42}_{\pm 10.18}$

TABLE IV: In-hand reorientation performance significantly drops in the absence of tactile information. Metrics are computed over 10 trials, and each episode is run until the object is dropped out of the robot hand.

encoder leads to a significant drop in success rate. Qualitatively, tactile feedback proves essential for robustness, allowing the policy to detect and compensate for object slippage in real-time.

VII. Conclusion

In this paper, we presented PTLD, a framework for learning dexterous manipulation policies that leverage tactile sensing without the need for accurate tactile simulation. Our core insight is to move beyond the traditional zero-shot sim-to-real constraint. Instead, we execute policies in real-world instrumented setups using privileged sensors, then distill these rollouts into tactile policies by aligning the implicit state latents of the privileged sensor policy with those of the tactile policy. We demonstrated the efficacy of PTLD on complex tasks, including in-hand rotation and reorientation, achieving significant performance gains in both. While this work focuses on tactile sensing, our methodology is general-purpose and provides a blueprint for learning perceptive policies across other modalities, such as vision.

Limitations: While PTLD offers a novel streamlined approach to training policies with sensing modalities that are difficult to simulate, we identify the following limitations:

- **Information Overlap & Asymmetry:** PTLD relies on distillation from a privileged sensor policy. Success depends heavily on the information overlap between the privileged and deployment sensors. For example, a privileged sensor providing noisy object poses primarily recovers kinematic data. In contrast, a tactile sensor can capture rich dynamic information, such as contact forces and vectors. Selecting a privileged sensor that minimizes this informational gap is critical for high-fidelity distillation.
- **Privileged Sensor Noise Floors:** All real-world sensors possess inherent noise. Because our distillation source is a policy trained in simulation, it must be robust to the specific noise profiles of the real-world privileged sensors. High noise in object pose estimation, for instance, sets a performance ceiling on the deployed policy. While high-precision systems like motion capture can mitigate this, the dependence on such instrumentation remains a constraint.
- **Instrumented Setup Requirements:** The requirement for an instrumented setup (e.g., external cameras or trackers) to provide privileged state information during distillation may limit rapid application of this method to completely unstructured or "in-the-wild" environments.

REFERENCES

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/02783649241273668?journalCode=ijra

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

[3] 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," *arXiv preprint arXiv:2402.10329*, 2024.

[4] C. Chen, Z. Yu, H. Choi, M. Cutkosky, and J. Bohg, "Dexforce: Extracting force-informed actions from kinesthetic demonstrations for dexterous manipulation," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 6416–6423, 2025.

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

[6] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[7] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[8] T. He, W. Xiao, T. Lin, Z. Luo, Z. Xu, Z. Jiang, J. Kautz, C. Liu, G. Shi, X. Wang *et al.*, "Hover: Versatile neural whole-body controller for humanoid robots," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 9989–9996.

[9] Y. Li, Z. Luo, T. Zhang, C. Dai, A. Kanervisto, A. Tirinzoni, H. Weng, K. Kitani, M. Guzek, A. Touati *et al.*, "Bfm-zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning," *arXiv preprint arXiv:2511.04131*, 2025.

[10] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs *et al.*, "Mujoco playground," *arXiv preprint arXiv:2502.08844*, 2025.

[11] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-Hand Object Rotation via Rapid Motor Adaptation," in *Conference on Robot Learning (CoRL)*, 2022.

[12] J. Pitz, L. Röstel, L. Sievers, and B. Bäuml, "Dextrous tactile in-hand manipulation using a modular reinforcement learning architecture," *arXiv preprint arXiv:2303.04705*, 2023.

[13] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam *et al.*, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5977–5984.

[14] R. Singh, A. Allshire, A. Handa, N. Ratliff, and K. Van Wyk, "Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands," *arXiv preprint arXiv:2412.01791*, 2024.

[15] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[16] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[17] C. Higuera, A. Sharma, T. Fan, C. K. Bodduluri, B. Boots, M. Kaess, M. Lambeta, T. Wu, Z. Liu, F. R. Hogan, and M. Mukadam, "Tactile beyond pixels: Multisensory touch representations for robot manipulation," in *9th Annual Conference on Robot Learning*, 2025. [Online]. Available: https://openreview.net/forum?id=sMs4pJYhWi

[18] M. Yang, C. Lu, A. Church, Y. Lin, C. Ford, H. Li, E. Psomopoulou, D. A. Barton, and N. F. Lepora, "Anyrotate: Gravity-invariant in-hand object rotation with sim-to-real touch," *arXiv preprint arXiv:2405.07391*, 2024.

[19] Z.-H. Yin, C. Wang, L. Pineda, F. Hogan, K. Bodduluri, A. Sharma, P. Lancaster, I. Prasad, M. Kalakrishnan, J. Malik *et al.*, "Dexteritygen: Foundation controller for unprecedented dexterity," *arXiv preprint arXiv:2502.04307*, 2025.

[20] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang, "Rotating without seeing: Towards in-hand dexterity through touch," *arXiv preprint arXiv:2303.10880*, 2023.

[21] S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra, "Tacto: A fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3930–3937, 2022.

[22] I. Akinola, J. Xu, J. Carius, D. Fox, and Y. Narang, "Tacsl: A library for visuotactile sensor simulation and learning," *IEEE Transactions on Robotics*, 2025.

[23] E. Su, C. Jia, Y. Qin, W. Zhou, A. Macaluso, B. Huang, and X. Wang, "Sim2real manipulation on unknown objects with tactile-based reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9234–9241.

[24] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on robot learning*. PMLR, 2020, pp. 66–75.

[25] S. Choudhury, M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer, and D. Dey, "Data-driven planning via imitation learning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1632–1672, 2018.

[26] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[27] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 255–262.

[28] D. Rus, "In-hand dexterous manipulation of piecewise-smooth 3-d objects," *The International Journal of Robotics Research*, vol. 18, no. 4, pp. 355–381, 1999.

[29] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on robot learning*. PMLR, 2020, pp. 1101–1112.

[30] L. Han and J. C. Trinkle, "Dextrous manipulation by rolling and finger gaiting," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 1. IEEE, 1998, pp. 730–735.

[31] R. Fearing, "Implementing a force strategy for object re-orientation," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 1986, pp. 96–102.

[32] A. S. Morgan, K. Hang, B. Wen, K. Bekris, and A. M. Dollar, "Complex in-hand manipulation via compliance-enabled finger gaiting and multi-modal planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4821–4828, 2022.

[33] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, vol. 8, no. 84, p. eadc9244, 2023.

[34] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[35] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," *arXiv preprint arXiv:2309.05665*, 2023.

[36] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," *arXiv preprint arXiv:2210.01841*, 2022.

[37] W. Yuan, S. Dong, and E. Adelson, "GelSight: High-resolution Robot Tactile Sensors for Estimating Geometry and Force," *Sensors: Special Issue on Tactile Sensors and Sensing*, vol. 17, no. 12, pp. 2762 – 2782, November 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/12/2762

[38] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, D. Jayaraman, and R. Calandra, "DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9018215

[39] R. Bhirangi, T. Hellebrekers, C. Majidi, and A. Gupta, "Reskin: versatile, replaceable, lasting tactile skins," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://proceedings.mlr.press/v164/bhirangi22a/bhirangi22a.pdf

[40] T. P. Tomo, A. Schmitz, W. K. Wong, H. Kristanto, S. Somlor, J. Hwang, L. Jamone, and S. Sugano, "Covering a Robot Fingertip

With uSkin: A Soft Electronic Skin With Distributed 3-Axis Force Sensitive Elements for Robot Hands," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 124–131, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8000399

[41] B. Huang, Y. Wang, X. Yang, Y. Luo, and Y. Li, "3d-vitac: Learning fine-grained manipulation with visuo-tactile sensing," in *8th Annual Conference on Robot Learning*, 2024.

[42] A. Sharma, C. Higuera, C. K. Bodduluri, Z. Liu, T. Fan, T. Hellebrekers, M. Lambeta, B. Boots, M. Kaess, T. Wu, F. R. Hogan, and M. Mukadam, "Self-supervised perception for tactile skin covered dexterous hands," in *9th Annual Conference on Robot Learning*, 2025. [Online]. Available: https://openreview.net/forum?id=eLeCrM5PEO

[43] S. Dong, D. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez, "Tactile-rl for insertion: Generalization to objects of unknown geometry," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021. [Online]. Available: https://arxiv.org/pdf/2104.01167.pdf

[44] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, "Cable manipulation with a tactile-reactive gripper," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1385–1401, 2021.

[45] S. Suresh, M. Bauza, K.-T. Yu, J. G. Mangelson, A. Rodriguez, and M. Kaess, "Tactile slam: Real-time inference of shape and pose from planar pushing," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 11 322–11 328.

[46] C. Higuera, A. Sharma, C. K. Bodduluri, T. Fan, P. Lancaster, M. Kalakrishnan, M. Kaess, B. Boots, M. Lambeta, T. Wu, and M. Mukadam, "Sparsh: Self-supervised touch representations for vision-based tactile sensing," 2024. [Online]. Available: https://openreview.net/forum?id=xYJn2e1uu8

[47] J. Zhao, Y. Ma, L. Wang, and E. H. Adelson, "Transferable tactile transformers for representation learning across diverse sensors and tasks," 2024.

[48] J. Wang, Y. Yuan, H. Che, H. Qi, Y. Ma, J. Malik, and X. Wang, "Lessons from learning to spin "pens"," in *CoRL*, 2024.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[50] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[51] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *arXiv preprint arXiv:1710.06542*, 2017.

[52] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[53] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *International conference on machine learning*. PMLR, 2022, pp. 1298–1312.

[54] F. Dellaert and G. Contributors, "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam)

[55] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, "General In-Hand Object Rotation with Vision and Touch," in *Conference on Robot Learning (CoRL)*, 2023.

[56] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5745–5753.

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

APPENDIX A
GRASP GENERATION

For both the policies we consider in this paper, we generate a grasp cache tuned for the Xela hand, which provides initial stable grasps as starting points for the episode. Inspired from [11], to generate the stable grasp poses, first, we initialize the robot hand (Xela hand) to a canonical pose and randomize the joint configuration within set limits, while the object is initialized slightly above the robot hand. Once the simulation proceeds, we retain the grasps if the object is held stably after 50 simulation steps.

APPENDIX B
IN-HAND ROTATION

A. Object diversity

Figure 10 shows the set of objects used for data collection and policy evaluation. We use four cylindrical objects with radii of 30 mm, 31.75 mm (×2), and 33.4 mm; heights of 70 mm, 178 mm (×2), and 200 mm; and varying surface frictions. In addition, we use two square bottles with side lengths of 54 mm and 60 mm, and heights of 187 mm and 194 mm.

The object masses range from 22 g to 90 g (22 g, 24 g, 30 g, 46 g, 87 g, and 90 g). To further vary the mass during data collection and evaluation, we insert additional weights of 18 g and 36 g into the bottles. Each weight piece (black nut) has a mass of 9 g.

B. Results of in-hand rotation policy with comparison against baseline policies

Figure 11 shows screen shots comparing policy performance on hardware. The top row shows screenshots from executing our tactile policy, where frequent finger gait adjustments are observed. Between 15–17 s, the object is pushed upward and tilts to the right. The tactile sensing detects this orientation change and the controller slightly loosens the grip, allowing the object to settle back to a stable height for the next 100 s. Later, at 124 s, the object drops below the stable region again. The fingers exhibit recovery behavior and push the object back upward. The full trajectory lasts for 220 seconds.

The bottom row shows screenshots from the real-world proprioceptive distillation baseline, where the object extrinsic encoder relies only on proprioception at deployment time. As the object is pushed upward, the fingers exhibit no recovery behavior. The object continues to slip up and falls out of the grasp after 13 s.

APPENDIX C
IN-HAND REORIENTATION

A. Additional reward details

In addition to the rewards described in Section V-C, we also use several additional reward shaping terms to ensure that the policy results in natural gaits. We detail the rewards below:

Fig. 10: objects used in data collection and policy evaluation

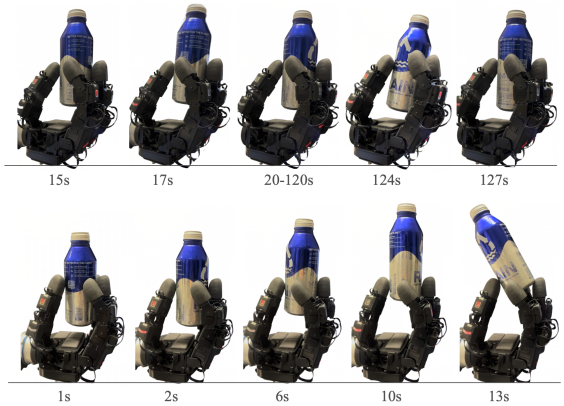

Fig. 11: Screenshots comparing ours (top) and proprioception baseline (bottom).

Reward	Scale	
$r_{\text{angular\_velocity}}$	$\triangleq \lVert \frac{(\mathbf{R}_t \cdot \mathbf{R}_{t-1}^{\top})}{\Delta t} \rVert$	-0.05
$r_{\text{acceleration}}$	$\triangleq \lVert \dot{q_t} - \dot{q_{t-1}} \rVert$	-0.005
r_{action}	$\triangleq \lVert a_t \rVert$	-0.005
$r_{\text{action\_rate}}$	$\triangleq \lVert a_t - a_{t-1} \rVert$	-0.2
$r_{\text{joint\_limit}}$	$\triangleq \max(q_{\text{lower}} - q_t, 0) + \max(q_t - q_{\text{upper}}, 0)$	-0.1
$r_{\text{object\_velocity}}$	$\triangleq \lVert \frac{p_t - p_{t-1}}{\Delta t} \rVert$	-1.0
r_{torque}	$\triangleq \lVert \tau \rVert$	-0.5
r_{work}	$\triangleq \lVert \tau \cdot \Delta q \rVert$	-4.0
r_{timeout}	$\triangleq t \geq T$	-1.0
r_{alive}	$\triangleq t - t_0$	-0.01

TABLE V: Reward function for any target in-hand reorientation

B. Additional results of in-hand reorientation policy

In Fig. 12 we visualize a roll out of the AAC trained policy in simulation showing continuous goal reaching behaviors. In simulation the policy is provided access to current object orientation as well as goal orientation in addition to proprioception. Furthermore in simulation the policies are trained for a wider distribution of goal poses ($\sim 40^\circ$ about the z-axis), and the

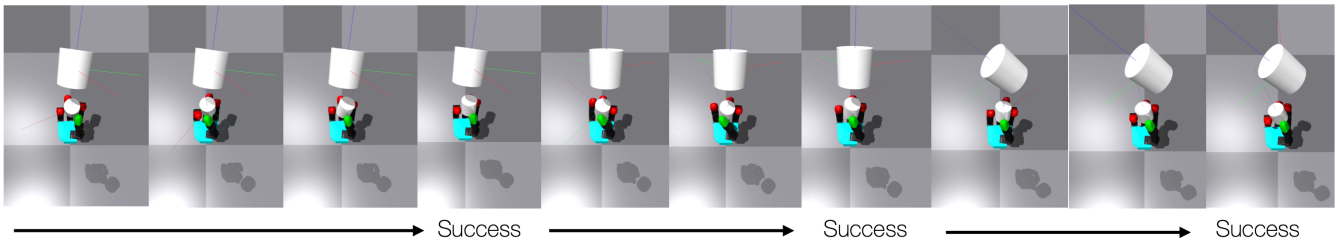

Fig. 12: Visualization of simulation in-hand reorientation

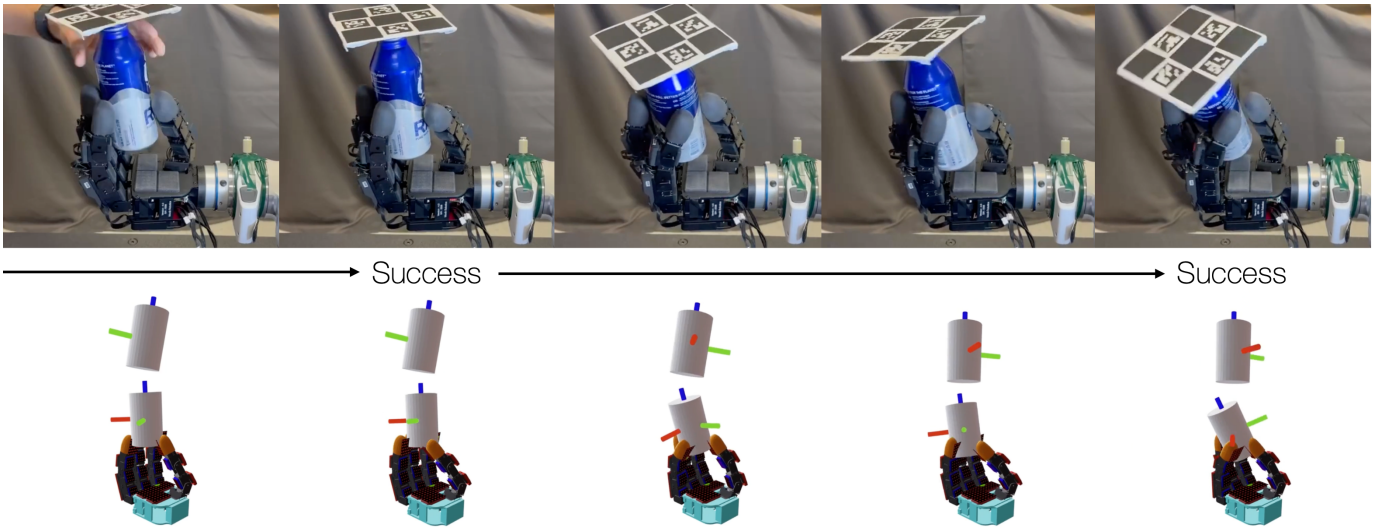

Fig. 13: Visualization of real world tactile in-hand reorientation

success threshold is tighter ($= 20°$). Similarly in Fig. 13 we visualize a real world roll out of the tactile in-hand-reorientation. We find that the tactile policy is able to manipulate objects reaching multiple goal poses within a single deployment. A demo is also additionally provided in the video submission. It must be noted that the marker is only used to evaluate whether a goal pose has been reached, and the policy does not take current object pose as input. In the real world we relax the success threshold to $\sim 30°$.