General In-Hand Object Rotation with Vision and Touch

Haozhi Qi[1,2], Brent Yi[1], Sudharshan Suresh[2,3], Mike Lambeta[2], Yi Ma[1], Roberto Calandra[4,5], Jitendra Malik[1,2]

***Abstract*— We introduce *RotateIt*, a system that enables fingertip-based object rotation along multiple axes by leveraging multi-modal sensory inputs. Our system is first trained in simulation, where it has access to ground-truth object shapes and physical properties. Then we distill it to operate solely on realistic yet noisy visual, tactile, and proprioceptive sensory inputs. These multimodal inputs are fused via a visuotactile transformer, enabling online inference of object shapes and physical properties during deployment. Our work highlights that incorporating visual and tactile sensing enables the policy to rotate diverse objects over multiple axes, and significantly outperforms previous methods. Website: `https://haozhi.io/rotateit/`.**

I. Introduction

Despite recent progress on in-hand manipulation for a single or a few objects [1], [2], [3], [4], generalizable object manipulation remains a challenge. In this paper, we demonstrate that fingertip-based in-hand object rotation over multiple different axes can be achieved from visual, tactile, and proprioceptive sensory inputs. This task is challenging for robots because of the need to simultaneously maintain stable force closure while rotating objects with diverse geometries.

An overview of our method, *RotateIt*, is shown in Figure 2. Our approach draws inspiration from recent advances in training reinforcement learning policies with privileged information [5], [6], [7], [8], more specifically rapid motor adaptation [6], [7]. We first train an oracle policy that is conditioned on a representation of the privileged information (called extrinsics, denoted as z_t, as shown in Figure 2), which contains ground-truth physical properties and shapes of the objects. With access to this representation, the oracle policy is able to efficiently and stably manipulate diverse objects over multiple axes *in simulation*.

The key challenge for real-world deployment lies in estimating the extrinsics encoding when privileged information is inaccessible. To address this challenge, we take inspiration from the importance of multimodal sensing, particularly vision and touch, during human manipulating objects [9], [10], [11]. We implement this by designing a visuotactile transformer which operates on a history of multimodal proprioceptive, visual, and tactile inputs to infer z_t. Concretely, during training, we rollout the oracle policy in simulation and collect the foreground object depth, contact locations on the fingertips, proprioception, and action history. Then we feed these multimodal streams into a transformer to produce an estimate of the z_t, denoted as $\hat{z}_t$. The visuotactile transformer is trained to minimize the difference between the predicted and estimated encodings of the privileged information.

[1]UC Berkeley,[2]Meta AI,[3]CMU,[4]TU Dresden,[5]The Centre for Tactile Internet with Human-in-the-Loop (CeTI)
hqi@berkeley.edu

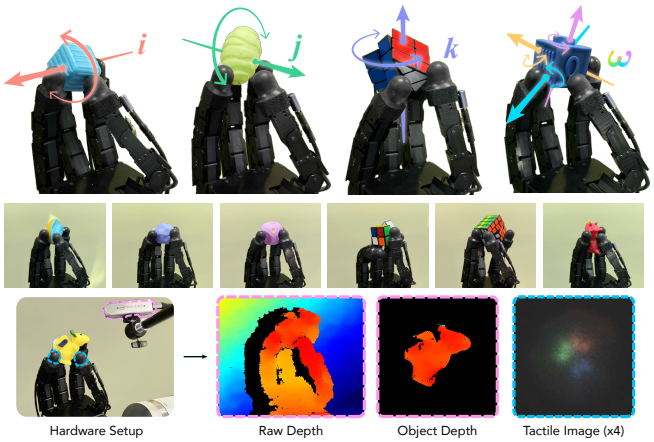


Fig. 1: **Rotation over multiple axes by integrating proprioception, vision, and touch sensing.** *RotateIt* is trained in simulation and deployed directly to the real-world, where it generalizes to diverse test objects without requiring fine-tuning.

We demonstrate *RotateIt* can perform multi-axis object rotation using only its fingertips. In simulation, we quantitatively study the performance of rotating skills over three principal axes in the hand-centric frame and the impact of incorporating vision and touch in various stages (Section V-A and Section V-B). To further understand what is learned by the policy, we investigate how accurately the latent representation of the policy captures the shape of the objects by trying to use it to recover 3D shape (Section V-C). Finally, we deploy the learned policies to rotate multiple different objects over multiple axes in the real world, where it enables rotation of objects that fail without visuotactile sensing (Section V-D). In our website, we show our policy can rotate objects including but not limited to the three canonical axes. Our work highlights the importance of both visual and tactile sensing in manipulation and presenting a step towards general dexterous in-hand manipulation.

II. Related Work

Sim2Real Methods. OpenAI et al. [1], [2] first transfers dexterous in-hand manipulation policies to the real-world. Similarly, [12], [4] uses a torque-controlled hand for cube rotation and reorientation when hand facing downwards. However, they focus on manipulating one single object. Recently, several works [7], [12], [13], [14] study generalizable in-hand object rotation using reinforcement learning. Our method differs from [7], [12], [14] as it is capable of rotating objects along multiple axes instead of just the z-axis. Compared

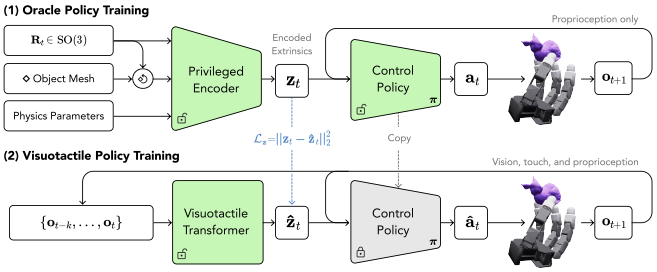


Fig. 2: **An overview of our training pipeline.** Trainable components are highlighted in green. In oracle policy training, we jointly optimize the privileged encoder and control policy using PPO. In the visuotactile policy training, we feed a sequence of visuotactile and proprioception to a transformer to infer $\hat{\boldsymbol{z}}_t$. The visuotactile transformer is trained by minimizing the regression loss between $\boldsymbol{z}_t$ and $\hat{\boldsymbol{z}}_t$.

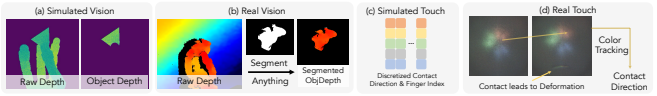


Fig. 3: **Visuotactile information used in simulator and real-world.** In simulation, we use the object's foreground depth as the input. In real-world, to reduce the sim2real gap, we segment out the object's depth map using Segment-Anything. For touch, we use discretized contact location in simulation provided by the simulator. In real-world, we parse the same information from temporal stream of tactile images.

to [13], our task is more challenging as it does not utilize a supporting surface, which allows constant tactile feedback on fingertips and enables a natural finger-gaiting to emerge.

Learning with privileged information is first shown in [15] and is successful in sim2real for legged locomotion [5], [6] and manipulation [7], [8]. [7] shows the policy can infer object position and physical properties online using proprioceptive history, but only works for z-axis rotation. [8] uses depth image as input while we use both vision and touch and show their importance.

Visuotactile Sensing and Learning. Vision-based tactile sensors, such as GelSight [16], TacTip [17], DIGIT [18], and GelTip [19], have been used for numerous applications including grasping [20], playing the piano [21], 3D reconstruction and localization [22], [23], [24], [25]. Previous work explores to use vision and touch for manipulation [26], [27] but they do not study in-hand manipulation with multi-fingered hand. To the best of our knowledge, *RotateIt* is the first work that intersects visuotactile sensing and learning to achieve general in-hand object rotation with a dexterous hand.

III. Method

An overview of the method is shown in Figure 2. Our policy training consists of two stages: First, we train an *oracle policy* with privileged information in simulation. Next, we train a *visuotactile policy* with realistic yet noisy observations.

A. Oracle Policy Training

Privileged Information. For the object's shape information, we sample N_p points from the object's mesh and encode it to a feature vector $\boldsymbol{z}_t^{\text{shape}}$ with c_p dimensions using PointNet [28]. One key difference from previous works [7], [8] is that we explicitly encode object's shape into the oracle policy, which we find to be beneficial especially for complex objects that are harder to manipulate.

Method	x-axis RotR ↑	x-axis TTF ↑	x-axis RotP ↓	y-axis RotR ↑	y-axis TTF ↑	y-axis RotP ↓	z-axis RotR ↑	z-axis TTF ↑	z-axis RotP ↓
Hora [7]	79.13$\pm$11.22	0.52$\pm$0.02	0.55$\pm$0.03	82.25$\pm$14.21	0.54$\pm$0.04	0.44$\pm$0.01	99.83$\pm$11.72	0.60$\pm$0.03	0.39$\pm$0.04
Oracle	**125.23**$\pm$16.24	**0.79**$\pm$0.03	**0.35**$\pm$0.02	**118.26**$\pm$13.20	**0.79**$\pm$0.05	**0.30**$\pm$0.01	**140.90**$\pm$17.26	**0.82**$\pm$0.02	**0.27**$\pm$0.01
w/o shape	85.10$\pm$12.56	0.56$\pm$0.03	0.39$\pm$0.03	99.92$\pm$10.21	0.62$\pm$0.04	0.41$\pm$0.02	129.38$\pm$10.26	0.75$\pm$0.03	0.29$\pm$0.01

TABLE I: We show the performance improvement over varies baselines on the rotation task over 3 different axis. We add two components to Hora [7]. The first one is object and finger pose, indicating the estimation is important. The second thing is the object shape, which further improves the performance.

The physics property contains object's mass, center of mass, coefficient of friction, scale, and restitution, resulting a 7-dimensional vector. The pose contains object's position, orientation (as a quaternion), and angular velocity, resulting a 10-dimensional vector. These vectors are concatenated together and projected to an 8-dim encoding vector $\boldsymbol{z}_t^{\text{phys}}$. Our final privileged encoding is concatenated from the shape encoding and physical property encoding $\boldsymbol{z}_t = [\boldsymbol{z}_t^{\text{phys}}, \boldsymbol{z}_t^{\text{shape}}]$.

Observations and Outputs. The oracle policy $\boldsymbol{\pi}$ takes the robot's proprioception and the encoded privileged information $\boldsymbol{z}_t$ as input. It outputs the targets of the PD Controller $\boldsymbol{a}_t \in \mathbb{R}^{16}$. The observation $\boldsymbol{p}_t$ contains a small temporal window of joint positions and actions $\boldsymbol{p}_t = [\boldsymbol{q}_{t-2:t}, \boldsymbol{a}_{t-3:t-1}] \in \mathbb{R}^{96}$, where $\boldsymbol{q}_t \in \mathbb{R}^{16}$ stands for the joint positions of the robot. Formally, we have $\boldsymbol{a}_t = \boldsymbol{\pi}(\boldsymbol{p}_t, \boldsymbol{z}_t)$.

Reward Function. Our reward function

$$r \doteq r_{\text{rotr}} + \lambda_{\text{rotp}} r_{\text{rotp}} + \lambda_{\text{pose}}\, r_{\text{pose}} + \lambda_{\text{linvel}}\, r_{\text{linvel}} \quad (1)$$

$$+ \lambda_{\text{work}}\, r_{\text{work}} + \lambda_{\text{torque}}\, r_{\text{torque}} \quad (2)$$

is modified from [7] with an additional penalty on undesired angular velocities component. The object rotation task is defined as $r_{\text{rotr}} \doteq \max(\min(\boldsymbol{\omega} \cdot \boldsymbol{k}, r_{\max}), r_{\min})$ where $\boldsymbol{\omega}$ is the object's angular velocity and $\boldsymbol{k}$ is the desired rotation axis in hand-centric axis. We find that naively applying this reward will result in unstable and oscillating behaviors when rotating over x and y-axis. To alleviate this problem, we add a rotation penalty term $r_{\text{rotp}} \doteq \|\boldsymbol{\omega} \times \boldsymbol{k}\|_1$. To make the policy stable, smooth, and energy efficient, we use a few penalty terms as in [7]. We use PPO [29] to optimize the oracle policy.

B. Visuotactile Policy Training with Transformers.

We find robust and adaptive finger-gaiting emerges from the oracle policy training. However, it is assumed to know full object physical properties, pose, and shape as the input. To deploy it in the real-world, we need to use real-world observations to infer (representations of) these properties. [7] uses proprioceptive states to estimate such information. In this work, we augment the senses with vision and tactile and study their important roles in improving manipulation performance.

Vision. We use object depth as the vision representation since 1) it is a general representation and does not require human labeling in the real-world and 2) it is hard to realistically simulate RGB images whereas depth is a good abstraction of object shape [30], [31]. In real-world deployment, instead of using the raw depth from a RGBD camera, we use Segment-Anything [32], [33] to segment out the objects to reduce the sim2real gap. Formally, given an object depth image $\boldsymbol{o}_t^{\text{depth}}$, we encode it 3-layer ConvNet to output $\boldsymbol{f}_t^{\text{depth}}$. An overview of the vision pipeline is shown in Figure 3. We also randomize the camera position and orientation during training, to make the policy robust to minor viewpoint changes.

Touch. To reduce the sim2real gap for tactile sensors, we choose to use the discretized contact location projected on 2D plane as the proxy of tactile information. In simulation, we directly parse the contact position provided by the simulator, project it onto a 2D plane in fingertip frame, and discretize it to 8 locations. The touch observation $\boldsymbol{o}_t^{\text{touch}}$ is a 9 dimensional vector with a binary indicator on 8 locations and one for finger index indicator. During training, since the number of contact points across episodes are not the same, we use an MLP to each contact information and take an average of different contact point features. In the real-world, we use four omnidirectional vision-based touch sensors at the fingertips. We track the deformation of the highest intensity pixel on each sensor, which serves as a proxy for contact position (Figure 3). This 2D keypoint from vision-based touch, similar in spirit to [34], is directly fed into the policy. The touch pipeline is shown in Figure 3.

Visuotactile Transformer. The goal of our visuotactile policy is to accurately infer the learned representation of privileged information. To tackle these challenges, we use a transformer ϕ architecture to model these multimodal sensory stream. We concatenate the encoded depth image $\boldsymbol{f}_t^{\text{depth}}$, encoded tactile contact points $\boldsymbol{f}_t^{\text{touch}}$, joint positions $\boldsymbol{q}_t$, and action at the previous timestep $\boldsymbol{a}_{t-1}$ to form the feature vector $\boldsymbol{f}_t$. We feed a sequence of features $\boldsymbol{f}_T = \{\boldsymbol{f}_{t-k}, \ldots, \boldsymbol{f}_{t-1}, \boldsymbol{f}_t\}$ as input to the transformer. The transformer outputs $\hat{\boldsymbol{z}}_t$ as the predicted extrinsic vector.

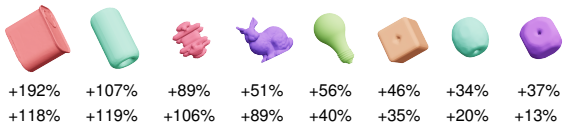
Training. Similar to previous work [5], [6], [7], we roll out the *oracle policy* with the *predicted extrinsic vectors* $\boldsymbol{a}_t = \boldsymbol{\pi}(\boldsymbol{p}_t, \hat{\boldsymbol{z}}_t)$ where $\hat{\boldsymbol{z}}_t = \boldsymbol{\phi}(\boldsymbol{f}_T)$. Meanwhile we also store the ground-truth extrinsic vector $\boldsymbol{z}_t$ and construct a training set $\mathcal{B} = \{(\boldsymbol{f}_T^{(i)}, \boldsymbol{z}_t^{(i)}, \hat{\boldsymbol{z}}_t^{(i)})\}_{i=1}^N$. Then we optimize $\boldsymbol{\phi}$ by minimizing the ℓ_2 distance between $\boldsymbol{z}_t$ and $\hat{\boldsymbol{z}}_t$ using Adam [35]. The process is iterated until the loss converges. We apply the same object initialization and dynamics randomization setting as the above section.

IV. Evaluation Setup

Hardware Setup. We use an AllegroHand from Wonik Robotics [36] for our experiments. The Allegro hand is a dexterous anthropomorphic robot hand with four fingers, with four degrees of freedom per finger. Position commands are sent to these 16 joints at 20 Hz. The target position commands are converted to torque using a PD Controller at 300 Hz. We use an Intel RealSense D435 placed at approximately 36cm from the Allegro base. We use an omnidirectional vision-based touch sensor at the distal end of each finger.

Method	Modality		x-axis			y-axis			z-axis		
	Vision	Touch	RotR ↑	TTF ↑	RotP ↓	RotR ↑	TTF ↑	RotP ↓	RotR ↑	TTF ↑	RotP ↓
Oracle	N/A	N/A	125.23$_{\pm 16.24}$	0.79$_{\pm 0.03}$	0.35$_{\pm 0.02}$	118.26$_{\pm 13.20}$	0.79$_{\pm 0.05}$	0.30$_{\pm 0.01}$	140.90$_{\pm 19.26}$	0.82$_{\pm 0.02}$	0.27$_{\pm 0.01}$
Conv			66.23$_{\pm 8.72}$	0.41$_{\pm 0.04}$	0.64$_{\pm 0.01}$	54.19$_{\pm 9.27}$	0.38$_{\pm 0.02}$	0.69$_{\pm 0.02}$	89.21$_{\pm 12.37}$	0.56$_{\pm 0.03}$	0.47$_{\pm 0.03}$
	✓		87.21$_{\pm 12.11}$	0.59$_{\pm 0.02}$	0.59$_{\pm 0.02}$	72.51$_{\pm 9.10}$	0.57$_{\pm 0.03}$	0.62$_{\pm 0.03}$	102.35$_{\pm 10.74}$	0.68$_{\pm 0.02}$	0.42$_{\pm 0.01}$
		✓	82.19$_{\pm 9.21}$	0.60$_{\pm 0.03}$	0.57$_{\pm 0.01}$	69.99$_{\pm 7.26}$	0.58$_{\pm 0.01}$	0.61$_{\pm 0.02}$	107.73$_{\pm 9.83}$	0.63$_{\pm 0.02}$	0.46$_{\pm 0.02}$
	✓	✓	98.20$_{\pm 10.18}$	0.70$_{\pm 0.03}$	0.45$_{\pm 0.03}$	89.82$_{\pm 9.22}$	0.67$_{\pm 0.03}$	0.47$_{\pm 0.01}$	113.26$_{\pm 13.98}$	0.70$_{\pm 0.04}$	0.40$_{\pm 0.01}$
Transformer			79.37$_{\pm 8.72}$	0.46$_{\pm 0.03}$	0.55$_{\pm 0.02}$	67.21$_{\pm 7.25}$	0.48$_{\pm 0.02}$	0.55$_{\pm 0.03}$	108.25$_{\pm 10.92}$	0.62$_{\pm 0.01}$	0.43$_{\pm 0.02}$
	✓		102.36$_{\pm 9.82}$	0.65$_{\pm 0.04}$	0.41$_{\pm 0.04}$	92.22$_{\pm 7.69}$	0.64$_{\pm 0.01}$	0.36$_{\pm 0.03}$	122.60$_{\pm 10.39}$	0.73$_{\pm 0.02}$	0.35$_{\pm 0.01}$
		✓	99.29$_{\pm 5.79}$	0.62$_{\pm 0.05}$	0.43$_{\pm 0.03}$	91.47$_{\pm 7.26}$	0.60$_{\pm 0.02}$	0.37$_{\pm 0.02}$	125.24$_{\pm 9.32}$	0.72$_{\pm 0.03}$	0.39$_{\pm 0.04}$
	✓	✓	**118.42**$_{\pm 9.46}$	**0.75**$_{\pm 0.03}$	**0.37**$_{\pm 0.02}$	**109.31**$_{\pm 12.29}$	**0.73**$_{\pm 0.02}$	**0.31**$_{\pm 0.04}$	**136.25**$_{\pm 11.12}$	**0.80**$_{\pm 0.04}$	**0.29**$_{\pm 0.02}$

TABLE II: **The importance of vision and touch.** We show the performance improvement of using vision, touch, and the transformer architecture. Each of the three components significantly improves the performance of rotating over x/y/z axis.



Stage 1	+192%	+107%	+89%	+51%	+56%	+46%	+34%	+37%
Stage 2	+118%	+119%	+106%	+89%	+40%	+35%	+20%	+13%

Fig. 4: **Relative rotation reward improvements before and after shape or visuotactile information.** For stage 1 training (oracle policy), we our oracle policy and our policy without point-cloud as input. For stage 2 training (visuotactile policy), we compare the improvement of *RotateIt* and the policy with only proprioceptive input. In both cases, having vision and touch information significantly improve the performance.

Simulation Setup. We use the IsaacGym [37] simulator. Each environment contains a simulated AllegroHand and a sampled object from our curated object datasets. Each object is of different physical properties and a random initial pose. For depth and viewpoint consistency between the real and simulated cameras, we measure the camera-robot extrinsics with an ArUco tag [38] placed on the palm of the real-world Allegro. In IsaacGym, we use this $SE(3)$ transformation augmented with random pose noise, and further apply realistic depth noise on the resultant images [39].

Object Set. We create a curated dataset for objects used in our experiments from EGAD [40], Google Scanned Objects [41], YCB [42], and ContactDB [43]. We select objects with width/depth/height (w/d/h) aspect ratio less than 2.0.

Evaluation Metric. We use the metrics defined in [7] to evaluate our method both in simulation and in the real-world. In addition, we also evaluate undesired rotation penalties in simulation. We find this metric is particularly important for rotation over x and y axis.

V. Results and Analysis

A. Object Shape helps Policy Training

The performance is shown in Table I. We compare *RotateIt* with previous work [7] and our method without the usage of point cloud while still using the quaternion. Experiments show that using point-cloud significantly improves the performance on all of the metrics and for all rotation axis. To get

Touch	x-axis RotR ↑	x-axis TTF ↑	x-axis RotP ↓	y-axis RotR ↑	y-axis TTF ↑	y-axis RotP ↓	z-axis RotR ↑	z-axis TTF ↑	z-axis RotP ↓
Full	104.29$_{\pm 10.29}$	0.68$_{\pm 0.04}$	0.41$_{\pm 0.02}$	93.05$_{\pm 9.28}$	0.65$_{\pm 0.01}$	0.34$_{\pm 0.03}$	126.73$_{\pm 10.11}$	0.72$_{\pm 0.03}$	0.32$_{\pm 0.03}$
NoTouch	79.37$_{\pm 8.72}$	0.46$_{\pm 0.03}$	0.55$_{\pm 0.02}$	67.21$_{\pm 7.25}$	0.48$_{\pm 0.02}$	0.55$_{\pm 0.03}$	108.25$_{\pm 10.92}$	0.62$_{\pm 0.01}$	0.43$_{\pm 0.02}$
Binary	80.14$_{\pm 7.25}$	0.47$_{\pm 0.02}$	0.53$_{\pm 0.03}$	66.29$_{\pm 8.53}$	0.49$_{\pm 0.01}$	0.56$_{\pm 0.04}$	110.24$_{\pm 9.48}$	0.63$_{\pm 0.03}$	0.42$_{\pm 0.02}$
ContactLoc	102.36$_{\pm 9.82}$	0.65$_{\pm 0.04}$	0.41$_{\pm 0.04}$	92.22$_{\pm 7.69}$	0.64$_{\pm 0.01}$	0.36$_{\pm 0.03}$	122.60$_{\pm 10.39}$	0.73$_{\pm 0.02}$	0.35$_{\pm 0.01}$

TABLE III: **The importance of using a finer tactile information.** We compare *RotateIt* which use contact location (ContactLoc) and its variant of using binary contact (Binary) or full contact (position, normal, and scale) information. All methods are without vision information. We find that binary contact does not provide additional value compared to NoTouch. We find using discretized contact locations already match the performance of using full contact.

Method	Cocoon	Squishy	Baseball	Puzzle	Box	Stego
Hora [7]	0.54$_{\pm 0.39}$	0.50$_{\pm 0.47}$	0.26$_{\pm 0.19}$	0.48$_{\pm 0.25}$	0.52$_{\pm 0.18}$	0.46$_{\pm 0.23}$
RotateIt	**12.71**$_{\pm 1.29}$	**8.29**$_{\pm 1.73}$	**6.72**$_{\pm 0.91}$	**6.12**$_{\pm 0.79}$	**5.05**$_{\pm 0.87}$	**5.01**$_{\pm 0.79}$

Fig. 5: **Rotations rotated (↑) for *RotateIt* and [7] in Real-world Evaluation.** We test *RotateIt* and Hora [7] on six different objects for x-axis rotation. Hora is not able to finish this task and does not learn finger-gaiting to rotate the object while *RotateIt* can.

more insights, we further plot the relative improvements on varies objects shape for x-axis rotation, shown in Figure 4 (the "stage1" row). We find that point-cloud gives the largest improvement on objects with non-uniform w/d/h (width/depth/height) ratios and objects with irregular shapes such as the bunny and light bulb. The improvements on regular objects are smaller but still over 40%. Point-cloud as an input is also used in [44] and [45] but they do not explore how to use it for in-hand manipulation. Note that our design is different from [8], which uses only pose for the oracle policy and uses object shape information only in the student policy. In our setting using object pose is not sufficient to achieve good enough performance.

B. Visuotactile Transformer

The oracle policy evaluated in Section V-A cannot be transferred to the real-world because it needs access to a manipulated object's ground-truth shape and physical properties. We instead learn to infer this representation during execution from proprioceptive, visual, and tactile history. In Table II, we show the impact of using vision and touch, and using transformer architecture to further improve the performance. Specifically, we show that using either vision or touch alone gives a huge performance improvement compared to proprioceptive only regardless of the architecture and rotation axes. For example, it increases RotR by 20 points in each of the x / y / z axes. Secondly, we find using a combination of vision and touch sensing can further improve the performance. Lastly, transformer has better sequence modeling ability compared to temporal convolutions used in previous work [7], [5]. By integrating visuotactile sensing and temporal transformer, our method can match the performance of the oracle policy.

C. Representation Learned in the Latent Space

Next, we study the information that is encoded into $\boldsymbol{z}_t$ and $\hat{\boldsymbol{z}}_t$. After we finish training policies, we perform roll outs on a collection of 16 objects in our dataset and record the estimated extrinsics vectors. Then we train a mesh prediction network in the training set and apply it on the test set.

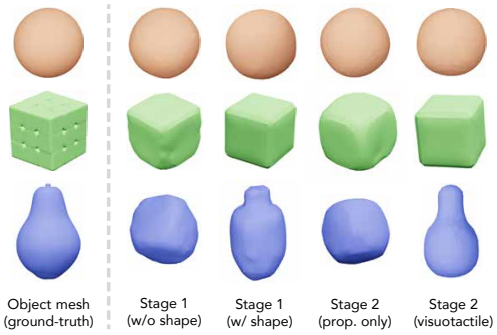


Fig. 6: **Inverting encoded extrinsics.** We predict 3D shapes on novel objects from learned $\boldsymbol{z}_t$ and $\hat{\boldsymbol{z}}_t$. Stage 1 results are provided with / without shape conditioning, stage 2 results are provided with / without visual and tactile sensors.

In Figure 6, we visualize predicted shapes averaged over 100 randomly selected subsequences from rollouts on novel test objects for four policies: the stage 1 oracle policy with and without shape (mesh) conditioning, and the stage 2 policy with and without visuotactile sensory inputs. The results suggest that shape information is preserved and useful for our oracle policy even though the only learning signal is the reward function. We also find policies without object shape will consider all the objects as spherical or cuboid objects, which explains the huge improvement on objects with large w/d/h ratios Figure 4. Next, our results also highlight both the capabilities and limits of proprioception, which we see can robustly distinguish between spherical (beige) and cuboidal (green) objects. Shape understanding for more irregular objects like the pear (blue), however, requires additional sensors. This supports the increased benefit of vision and touch for more complex objects that we observe in Section V-A.

D. Real-world Evaluations

Finally, we quantitatively compare *RotateIt* and *Hora* [7] in the real-world on rotating different objects over the x-axis. We find that without vision and touch, *Hora* cannot finish this task. It only learns in-grasp movement with thumb slowly moving to the bottom of the object. It is also not able to maintain stability; the object quickly falls down. In contrast, *RotateIt* can successfully manipulate multiple objects with different geometries such as cubes, spheres, or cylinders by $\sim 2\pi$ radians within 20 seconds. We show qualitative results on rotation around and beyond the three canonical axes on our website.

VI. Conclusion

In this paper, we show the feasibility of training policies that can rotate many objects over multiple axes. We view this capability as an important step towards general-purpose in-hand manipulation.

REFERENCES

[1] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *IJRR*, 2019.

[2] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *arXiv:1910.07113*, 2019.

[3] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, Y. Narang, J.-F. Lafleche, D. Fox, and G. State, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," in *ICRA*, 2023.

[4] J. Pitz, L. Röstel, L. Sievers, and B. Bäuml, "Dextrous tactile in-hand manipulation using a modular reinforcement learning architecture," in *ICRA*, 2023.

[5] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, 2020.

[6] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *RSS*, 2021.

[7] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-hand object rotation via rapid motor adaptation," in *CoRL*, 2022.

[8] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," *arXiv:2211.11744*, 2022.

[9] G. Westling and R. S. Johansson, "Factors influencing the force control during precision grip," *Experimental Brain Research*, 1984.

[10] J. R. Flanagan, M. C. Bowman, and R. S. Johansson, "Control strategies in object manipulation tasks," *Current Opinion in Neurobiology*, 2006.

[11] R. S. Johansson and J. R. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nature Reviews Neuroscience*, 2009.

[12] L. Sievers, J. Pitz, and B. Bäuml, "Learning purely tactile in-hand manipulation with a torque-controlled hand," *ICRA*, 2022.

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

[14] G. Khandate, S. Shang, E. T. Chang, T. L. Saidi, J. Adams, and M. Ciocarlie, "Sampling-based exploration for reinforcement learning of dexterous manipulation," in *RSS*, 2023.

[15] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *CoRL*, 2020.

[16] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, 2017.

[17] B. Ward-Cherrier, N. Pestell, L. Cramphorn, B. Winstone, M. E. Giannaccini, J. Rossiter, and N. F. Lepora, "The tactip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies," *Soft robotics*, 2018.

[18] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer *et al.*, "Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation," *RA-L*, 2020.

[19] D. F. Gomes, Z. Lin, and S. Luo, "Geltip: A finger-shaped optical tactile sensor for robotic manipulation," in *IROS*, 2020.

[20] R. Calandra, A. Owens, D. Jayaraman, W. Yuan, J. Lin, J. Malik, E. H. Adelson, and S. Levine, "More than a feeling: Learning to grasp and regrasp using vision and touch," *RA-L*, 2018.

[21] H. Xu, Y. Luo, S. Wang, T. Darrell, and R. Calandra, "Towards learning to play piano with dexterous hands and touch," in *IROS*, 2022.

[22] E. Smith, D. Meger, L. Pineda, R. Calandra, J. Malik, A. Romero Soriano, and M. Drozdzal, "Active 3d shape reconstruction from vision and touch," *NeurIPS*, 2021.

[23] E. Smith, R. Calandra, A. Romero, G. Gkioxari, D. Meger, J. Malik, and M. Drozdzal, "3d shape reconstruction from vision and touch," *NeurIPS*, 2020.

[24] S. Suresh, Z. Si, J. G. Mangelson, W. Yuan, and M. Kaess, "Shapemap 3-d: Efficient shape mapping through dense touch and vision," in *ICRA*, 2022.

[25] S. Suresh, Z. Si, S. Anderson, M. Kaess, and M. Mukadam, "Midastouch: Monte-carlo inference over distributions across sliding touch," in *CoRL*, 2022.

[26] N. Sunil, S. Wang, Y. She, E. Adelson, and A. R. Garcia, "Visuotactile affordances for cloth manipulation with local control," in *CoRL*, 2022.

[27] J. Hansen, F. Hogan, D. Rivkin, D. Meger, M. Jenkin, and G. Dudek, "Visuotactile-rl: Learning multimodal manipulation policies with deep reinforcement learning," in *ICRA*, 2022.

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

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

[30] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, 2021.

[31] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *CoRL*, 2022.

[32] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *ICCV*, 2023.

[33] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," *arXiv:2306.14289*, 2023.

[34] P. Sodhi, M. Kaess, M. Mukadam, and S. Anderson, "Learning tactile models for factor graph-based estimation," in *ICRA*, 2021.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[36] WonikRobotics, "Allegrohand," https://www.wonikrobotics.com/, 2013.

[37] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," in *NeurIPS Datasets and Benchmarks*, 2021.

[38] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and vision Computing*, 2018.

[39] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *CVPR*, 2015.

[40] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *RA-L*, 2020.

[41] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," in *ICRA*, 2022.

[42] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *International Conference on Advanced Robotics (ICAR)*, 2015.

[43] S. Brahmbhatt, C. Ham, C. C. Kemp, and J. Hays, "Contactdb: Analyzing and predicting grasp contact via thermal imaging," in *CVPR*, 2019.

[44] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang, "Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation," in *CoRL*, 2022.

[45] C. Bao, H. Xu, Y. Qin, and X. Wang, "Dexart: Benchmarking generalizable dexterous manipulation with articulated objects," in *CVPR*, 2023.