General-Purpose Sim2Real Protocol for Learning Contact-Rich Manipulation With Marker-Based Visuotactile Sensors

Weihang Chen, Jing Xu, *Member, IEEE*, Fanbo Xiang, Xiaodi Yuan, Hao Su, *Member, IEEE*, and Rui Chen, *Member, IEEE*

***Abstract*—Visuotactile sensors can provide rich contact information, having great potential in contact-rich manipulation tasks with reinforcement learning (RL) policies. Sim2Real technique tackles the challenge of RL's reliance on a large amount of interaction data. However, most Sim2Real methods for manipulation tasks with visuotactile sensors rely on rigid-body physics simulation, which fails to simulate the real elastic deformation precisely. Moreover, these methods do not exploit the characteristic of tactile signals for designing the network architecture. In this article, we build a general-purpose Sim2Real protocol for manipulation policy learning with marker-based visuotactile sensors. To improve the simulation fidelity, we employ an FEM-based physics simulator that can simulate the sensor deformation accurately and stably for arbitrary geometries. We further propose a novel tactile feature extraction network that directly processes the set of pixel coordinates of tactile sensor markers and a self-supervised pretraining strategy to improve the efficiency and generalizability of RL policies. We conduct extensive Sim2Real experiments on the peg-in-hole task to validate the effectiveness of our method. And we further show its generalizability on additional tasks including plug adjustment and lock opening. The protocol, including the simulator and the policy learning framework, will be open-sourced for community usage.**

***Index Terms*—Contact-rich manipulation, robot simulation, sim-to-real, tactile sensing.**

I. Introduction

TACTILE sensing can provide rich information about contacts between the robot and the environment, including local geometry and force. This information is crucial for contact-rich robot tasks, and cannot be obtained from visual signals [1]. Therefore, tactile sensing is drawing increasing attention in robot applications, such as object manipulation and assembly [2], [3], [4]. However, tactile sensor signals, particularly those from visuotactile sensors, are high-dimensional, which poses a significant challenge to manually designing rules for robust and generalizable control policies.

To develop control policies that can utilize high-dimensional input data, robotics researchers have employed deep reinforcement learning (RL) to autonomously extract features and learn complex control policies without manual design [5], [6]. However, the integration of tactile sensing into RL policies remains underexplored currently. A major obstacle is the high demand for massive amounts of interaction data, which are costly and time-consuming to collect in reality. Sim2Real addresses this problem by enabling robots to be trained in simulation and transferring the learned policies to real robots with minimal or zero real-world data [7], [8], [9]. Furthermore, the access to ground-truth environment states in simulation accelerates the policy learning process.

Most existing tactile sensor simulation methods are based on rigid-bod physics simulation [10], [11], [12], [13], [14]. While the penalty-based contact model can approximate the deformation of the tactile sensor, it cannot capture the elastic dynamics of the tactile sensor accurately. These methods tradeoff realism for computational efficiency, resulting in a large Sim2Real gap that hinders the transferability of manipulation policies learned in simulation.

In this work, we explore how to build an efficient, general, and effective tactile Sim2Real protocol for robot manipulation with marker-based visuotactile sensors. The protocol comprises four essential components, each of which significantly contributes to the final Sim2Real performance. First, we employ a state-of-the-art finite-element-method (FEM) based physics simulation to model the elastic dynamics of the tactile sensor and the contacts between sensors and objects more precisely than rigid-body-based physics simulation. Second, through experiments, we discover that it is better to use a point-based representation for tactile feature extraction and policy learning, where the input points come from the set of pixel coordinates of tactile sensor markers. Compared with CNNs that process the tactile sensor image, our proposed network can achieve higher learning efficiency and better generalizability to unseen objects. Third, while the raw signal from the marker-based tactile sensor

Manuscript received 2 October 2023; accepted 3 January 2024. Date of publication 11 January 2024; date of current version 31 January 2024. This paper was recommended for publication by Associate Editor W. Yuan and Editor M. Yim upon evaluation of the reviewers' comments. *(Weihang Chen and Jing Xu contributed equally to this work.) (Corresponding authors: Jing Xu; Hao Su; Rui Chen.)*

Weihang Chen, Jing Xu, and Rui Chen are with the Department of Mechanical Engineering, Tsinghua University, Beijing 100084, China (e-mail: chen-wh18@mails.tsinghua.edu.cn; jingxu@tsinghua.edu.cn; callmeray@163.com).

Fanbo Xiang, Xiaodi Yuan, and Hao Su are with the Department of Computer Science and Engineering, University of California, San Diego, CA 92122 USA (e-mail: fxiang@eng.ucsd.edu; x9yuan@ucsd.edu; haosu@ucsd.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2024.3352969, provided by the authors.

Digital Object Identifier 10.1109/TRO.2024.3352969

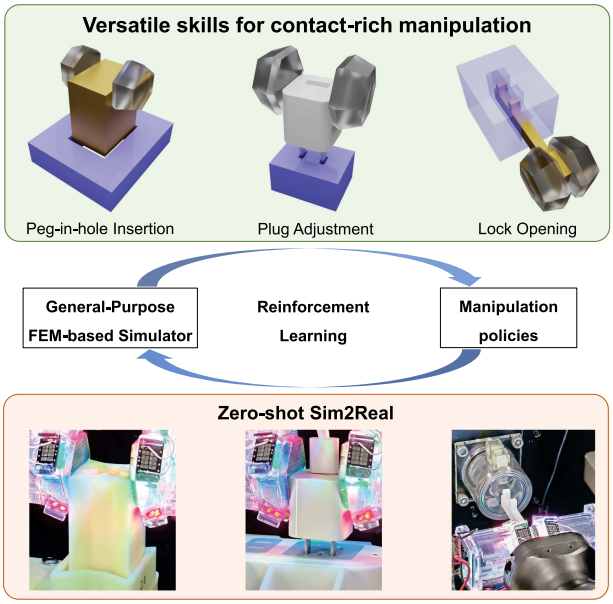


Fig. 1. In this work, we build a general-purpose Sim2Real protocol for marker-based visuotactile sensors, which includes FEM-based physics simulation, policy learning architectures, self-supervised representation learning, and domain randomization techniques. We demonstrate its effectiveness for zero-shot Sim2Real for three high-precision contact-rich manipulation tasks.

is high dimensional, the feature related to the manipulation tasks is usually low dimensional. Therefore, to further improve learning efficiency, we propose to leverage self-supervised representation learning to obtain general tactile representation that can be applied to various manipulation tasks. Finally, after systemically analyzing the sources of Sim2Real discrepancy, we employ domain randomization techniques in the physical, optical, and task domains to improve the Sim2Real transferrability of the learned policy.

While our method is designed for general contact-rich tasks, we take peg-in-hole insertion as an example task and test our system extensively on it, because this task is ubiquitous and fundamental for many advanced tasks. In this task, our simulation achieves a speed two orders of magnitude faster than reality on a desktop CPU. We conduct extensive Sim2Real experiments on the peg-in-hole task and compare our method against state-of-the-art tactile simulation and feature extraction methods. We also propose an arbitrary-shape training scheme for generalizable peg-in-hole policy learning. We further validate the generalizability of our Sim2Real pipeline on two challenging real-life tasks: plug adjustment and lock opening (see Fig. 1).

In summary, the contributions of this work are as follows:

1) We present a general-purpose Sim2Real protocol for marker-based visuotactile sensor that has been validated effective for various real-world high-precision contact-rich manipulation tasks and will be open-sourced for community usage.
2) We introduce point cloud learning architecture for tactile feature extraction that directly processes the set of pixel coordinates of tactile sensor markers.
3) We introduce a self-supervised pretraining framework to improve the sample efficiency of tactile-based robot learning.
4) We demonstrate the effectiveness of our method on various real robot manipulation tasks through extensive quantitative experiments.

The rest of this article is structured as follows. Section II provides a summary of related works. Section III outlines the proposed methodology, including the simulation method, tactile feature extraction, and policy training scheme. The subsequent three sections present the experiments. Section IV focuses on the self-supervised tactile representation learning. Section V discusses the Sim2Real experiments for the peg-in-hole task. Section VI examines the proposed method on two additional contact-rich manipulation tasks that closely resemble real-life scenarios. Finally, Section VII presents the conclusion.

II. Related Works

A. Tactile Sensing

According to working principles, existing tactile sensing approaches can be divided into resistive, capacitive, piezoresistive, piezoelectric, triboelectric, optical and others. We refer to [15] for a systematic survey on tactile sensing techniques. Marker-based visuotactile sensors transform the contact information into marker flow in the captured image by embedding markers in the surface or interior of the sensor's flexible elastomer [16]. It has the following advantages: simple hardware configuration, low cost, simultaneous measurement of normal and shear forces, and it has been successfully applied to various robot tasks, including peg-in-hole insertion, cable manipulation [4], [17], [18]. Therefore, marker-based visuotactile sensors are widely studied, and many type of sensors have been developed, including GelForce [19], TacTip [20], [21], GelStereo [22], Tac3D [23], and the sensor by ETH [24]. In this work, we mainly verified our proposed protocol on our marker-based visuotactile sensors based on GelSight [25]. We note that our approach can be adapted to other types of marker-based visuotactile sensors.

B. Tactile Sensor Simulation

The simulation process of tactile sensors is typically divided into two phases. The first phase involves simulation of the sensor's deformation caused by contact. The second phase involves simulation of the transduction of physical quantities from the deformation. For marker-based visuotactile sensors, the first phase is a fundamental step for the second phase. This section primarily summarizes methods related to the first phase. Tactile sensors that operate on other principles, such as BioTac [26], share a similar first phase as marker-based visuotactile sensors and are also included in this section.

In [10] and [11], a Gazebo-based GelSight simulator was proposed, where the deformation was simulated from the contact geometry using Gaussian filtering followed by difference of Gaussian. In [12], [13], and [27], PyBullet was used to simulate the contact between the object and the sensor, and the deformation was approximated by calculating the penetration

of the object into the sensor. A similar method is also reported in [28]. The focus of these works is the simulation of sensors' optical properties. Despite their computational efficiency, these geometry-based methods cannot model the sensor's tangential deformation, which is crucial for many robot applications such as slip detection and peg-in-hole insertion. Xu et al. [14] proposed a penalty-based tactile model upon rigid body dynamics, which is able to simulate both normal and shear tactile force fields at high speed. While the penalty-based simulation can approximate the sensor deformation in manipulation, it cannot simulate the elastic behavior of the elastomer accurately, especially the contact force caused by the tangential deformation. Therefore, its Sim2Real transferrability is limited, which has been demonstrated by our experimental results.

Compared with rigid-body-based simulation, finite element methods (FEM) can model the deformation of the sensor's elastomer more accurately. Bi et al. [29] developed a FEM-based tactile sensor simulator and achieved zero-shot Sim2Real transfer of RL policies for aggressive swing-up manipulation. However, they utilized the cylindrical geometry of the poles to simplify the simulation, and constrained the motion of the pole in the *x*-*y* plane, rendering it inapplicable to objects with various geometries. Si and Yuan [30] proposed a superposition method to approximate the FEM dynamics and successfully simulate the sensor's tangential deformation, but no manipulation tasks were demonstrated. Narang et al. built a linear-FEM-based tactile simulator for BioTac with Isaac Gym [31], achieving faster speeds than the commercial FEM software (ANSYS) [32]. Recently, Lu et al. employed SOFA [33] to build a simulator for large-scale marker-cum-vision-based-tactile sensor [34]. However, both [31] and [34] primarily used their simulators to collect supervised datasets for interpreting tactile signals, leaving the potential of using simulation to train manipulation policies unexplored. In this work, we build a Sim2Real protocol for learning contact-rich manipulation with marker-based tactile sensors that, to the best of our knowledge, is the first one that models the sensor deformation in a physics-grounded manner and has been validated as successful in multiple Sim2Real manipulation tasks. We demonstrate that our protocol is efficient enough to train robust and generalizable manipulation policies for deployment in reality.

C. Sim2Real for Robots

Deep RL has shown great success in robotics, enabling the learning of complex control policies that are beyond manual design [5], [6]. However, deep RL requires a large amount of interaction data, which is costly and time-consuming to collect in reality.

Due to the inherent discrepancy between simulation and reality caused by the imperfect modeling of real-world dynamics, the policies learned in simulation often suffer from poor transferrability to reality. Sim2Real approaches can be broadly classified into two categories: domain randomization and domain adaptation [35]. In [13] and [27], a domain adaptation method based on generative adversarial networks was proposed to translate the real tactile images into the simulation domain. A similar method for a large-scale tactile sensor was adopted in [34]. In these works, the translation was trained in a supervised-learning fashion, which required a dataset of precisely aligned real and simulated image pairs. The generalizability to different sensor instances (of the same type) was not discussed. Some works proposed domain adaptation methods that does not require paired datasets [10], [36], [37], but they mainly focus on the optical properties of the GelSight sensor, which cannot be applied to Sim2Real of manipulation tasks directly. In this work, focusing on dynamic manipulation, we propose a different tactile representation instead of images, and employ domain randomization techniques to enhance the Sim2Real transferrability and generalizability across different real tactile sensor instances.

D. Pretraining for Robot Learning

The FEM-based simulation is more realistic than rigid-body-based simulation, but suffers from slow simulation speed that hinders the Sim2Real learning efficiency. Self-supervised pretraining has been demonstrated to be able to enhance data efficiency for robot learning by recent works [38], [39], [40], [41]. However, most existing methods focus on visual and language inputs and employ transformer-based networks to extract feature representations. Pretraining for tactile signals is still underexplored. For marker-based visuotactile sensors used in our work, the tactile signals only contain local contact information within a relatively small area. The information content is considerably lower compared to that in common visual signals. Therefore, we propose a light-weight autoencoder to pretrain task-agnostic latent representations from tactile signals.

III. Methodology

This section presents the simulation and policy learning pipeline for marker-based visuotactile sensors. The tactile sensing principle is first introduced in Section III-A. The FEM-based physics simulation of the tactile sensor's elastomer is described in Section III-B. The pixel coordinates of markers are used as the tactile signal and the synthesis method is shown in Section III-C. To efficiently process raw tactile observations in policy training, we propose a tactile feature extractor network and the corresponding pretraining method in Section III-D. The tactile feature extractor is further utilized in RL policies as introduced in Section III-E. Finally, Section III-F summarizes the domain randomization techniques that facilitate Sim2Real transfer.

A. Tactile Sensing Principle

Fig. 2(a) shows the schematic of the marker-based visuotactile sensor used in our work, which consists of an RGB camera, LEDs, an acrylic support, and a transparent elastomer covered with a reflective surface. A 2-D array of markers is distributed on the elastomer surface. When the elastomer deforms due to contacts, the normal deformation can be computed from the image captured by the camera using photometric stereo, and the

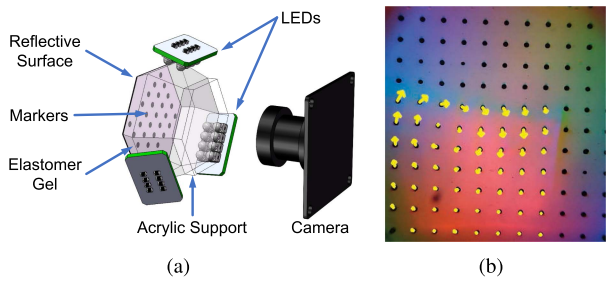


Fig. 2. (a) Schematic of the marker-based visuotactile sensor used in our work, which is based on GelSight [25]. (b) One example of real tactile image with marker tracking. The marker flow in contact area is shown by yellow arrows.

tangential deformation can be measured by tracking the markers as shown in Fig. 2(b).

For most contact-rich manipulation tasks, the tangential deformation of the elastomer is more informative than the normal deformation as it corresponds to the object displacement during manipulation and can indicate the contact force and torque. It is pointed out in [25] that using marker flow, a sensor can achieve equal or better performance in terms of force sensing, compared with surface map measurement. Therefore, the marker flow is usually used as the tactile signal in manipulation tasks and we aim to simulate the marker flow accurately in this work. We believe that the proposed pipeline can be applied to other marker-based visuotactile sensors, as introduced in Section II-A.

B. General-Purpose FEM-Based Physics Simulation

Marker-based visuotactile sensors typically employ silicone rubber as the elastomer, which exhibits hyperelasticity and allows for large deformation. In order to achieve successful transfer of RL policies trained in simulation for versatile manipulation skills, the physics simulation needs to satisfy the following three requirements:

1) it can support physics simulation of arbitrary 3-D geometries;
2) it can accurately capture the dynamic and elastic properties of the elastomer;
3) it can simulate the large deformation of the elastomer during RL training stably.

Therefore, we use incremental potential contact (IPC) [42] as the physics simulation in this work. IPC is based on FEM and supports elastomers with hyperelastic material models such as the widely adopted Neo–Hookean model. The major difference between IPC and other FEM simulators such as SOFA [33] is the contact solver. Traditional contact solvers can introduce intersections and put the simulator into an invalid state. Such artifact is typically mitigated by using very small time steps, which significantly degrades the simulation speed. On the other hand, IPC models contact with barrier energy and integrates continuous collision detection into its solver to guarantee that the simulation is intersection free and inversion free. Thus, IPC enables robust FEM simulation at large time steps and improves the simulation speed, which is desired for the RL training process.

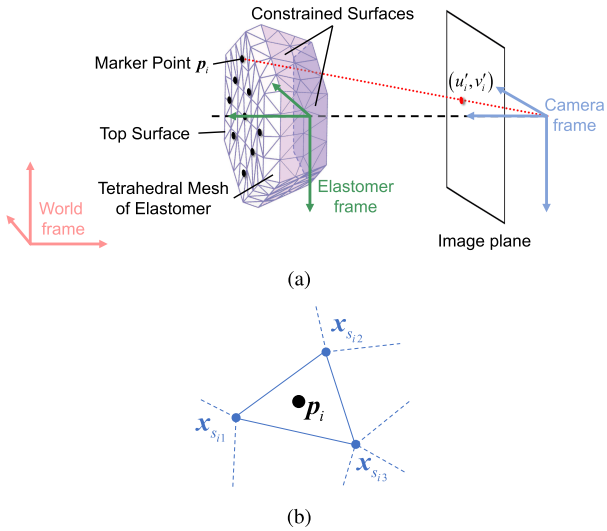


Fig. 3. (a) In our tactile sensor, the markers are distributed on the top surface. The surfaces in purple are constrained by the shell. The elastomer frame is fixed on the constrained surfaces of the elastomer. After transforming the marker point into the camera frame, one can calculate its pixel coordinate using the pinhole model. (b) Illustration of interpolation using adjacent FEM vertices. $\boldsymbol{p}_i$ denotes the ith marker, and the facet it belongs to has three vertices: $\boldsymbol{x}_{s_{i1}}$, $\boldsymbol{x}_{s_{i2}}$, and $\boldsymbol{x}_{s_{i3}}$.

For tactile-based manipulation tasks, the motion of the elastomer is induced by the action executed on the robot end effector. We model the robot action as Dirichlet boundary conditions on the elastomer mesh by setting the position and velocity of boundary vertices. The shape of the tactile sensor's elastomer is shown in Fig. 3(a). The markers are distributed on the top surface of the elastomer, which can deform freely due to contact with the object during manipulation, while the surfaces in purple are constrained by the sensor's shell. In robotic manipulation tasks, the constrained surfaces of the sensor's elastomer are actuated by the robot which the sensor is attached to.

In IPC, the sensor's elastomer is discretized as a tetrahedral mesh. We denote the vertices of the mesh as $\{\boldsymbol{x}_i\}$; for vertex coordinates in the world frame, we add a superscript to distinguish, i.e., $\{\boldsymbol{x}_i^W\}$. The vertices on the aforementioned constrained surfaces form a set $\mathbb{S}$. In the simulation, we enforce the positions and velocities of the vertices in $\mathbb{S}$ as the boundary constraint.

For one step in the simulation, suppose the rigid shell to which the sensor is attached is moving at a linear velocity $\boldsymbol{v}$ and an angular speed ω around an instantaneous axis, defined by its direction $\boldsymbol{d}$ and a point $\boldsymbol{x}_{\text{pivot}}$, then the position at the next step $\boldsymbol{x}_i'$, and the velocity $\boldsymbol{v}_i$ of each boundary point is given by

$$\begin{aligned} \boldsymbol{x}_i' &= \boldsymbol{v}\Delta t + \mathbf{R}(\boldsymbol{d}, \omega\Delta t)(\boldsymbol{x}_i - \boldsymbol{x}_{\text{pivot}}) + \boldsymbol{x}_{\text{pivot}} \\ \boldsymbol{v}_i &= \frac{\boldsymbol{x}_i' - \boldsymbol{x}_i}{\Delta t} \\ i &\in \{i | \boldsymbol{x}_i \in \mathbb{S}\} \end{aligned} \tag{1}$$

where Δt is the timestep of the simulation and $\mathbf{R}(\boldsymbol{d}, \omega\Delta t)$ is the rotation matrix defined by axis $\boldsymbol{d}$ and rotation angle $\omega\Delta t$.

At each step, the simulated robot motion is converted to the boundary condition given in (1). Then, IPC's FEM solver

will solve all the vertex positions at the next step. In this way, the deformation of the sensor's elastomer and the interactions between the sensor and objects are obtained.

C. Synthesis of Tactile Sensor Signals

In this work, we use the marker flow as the tactile sensor signals. Given the vertices of the elastomer mesh in the elastomer frame $\{\boldsymbol{x}_i^E\}$ and the marker coordinates in the elastomer frame $\{\boldsymbol{p}_i^E\}$, we first determine the facet that each marker belongs to, and compute the weight of each vertex

$$\boldsymbol{p}_i^E = k_{i1}\boldsymbol{x}_{s_{i1}}^E + k_{i2}\boldsymbol{x}_{s_{i2}}^E + k_{i3}\boldsymbol{x}_{s_{i3}}^E \tag{2}$$

where k_{i1}, k_{i2}, and k_{i3} are the triangular barycentric weights, which satisfy $k_1, k_2, k_3 \geq 0, k_1 + k_2 + k_3 = 1$. Note that $\{(k_{i1}, k_{i2}, k_{i3})\}$ and $\{(s_{i1}, s_{i2}, s_{i3})\}$ are computed before the physics simulation.

The transformation from the elastomer frame to the world frame is set at the initialization of the simulation according to the sensor positions. For each time step in the simulation of manipulation, the vertices of the deformed elastomer in the world frame $\{\boldsymbol{x}_i'^W\}$ are computed by the solver, and then the displaced marker coordinates $\{\boldsymbol{p}_i'^W\}$ in the world frame can be computed using $\{\boldsymbol{x}_i'^W\}$, $\{(k_{i1}, k_{i2}, k_{i3})\}$, and $\{(s_{i1}, s_{i2}, s_{i3})\}$.

In order to generate the marker flow in the image, we first compute the marker coordinate in the camera frame $\boldsymbol{p}_i'^C = [x_i', y_i', z_i']^{\mathrm{T}}$ as

$$\boldsymbol{p}_i'^C = \mathbf{R}_W^C \boldsymbol{p}_i'^W + \boldsymbol{t}_W^C \tag{3}$$

where $\mathbf{R}_W^C$ and $\boldsymbol{t}_W^C$ are the rotation matrix and the translation vector of the world frame relative to the camera frame. Then, the corresponding pixel coordinate (u_i', v_i') is computed using the camera pinhole model as

$$\begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix} = \frac{1}{z_i'} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} \tag{4}$$

where f_x, f_y, c_x, and c_y are the camera intrinsic parameters. The marker's initial pixel coordinate before contact (u_i, v_i) can be computed similarly.

To ensure that the 3-D vertex positions $\{\boldsymbol{p}_i'^C\}$ and projected pixel coordinates $\{(u_i', v_i')\}$ are within certain bounds, we impose constraints in both the simulation and the reality:

1) In simulation, the episode terminates immediately and is considered failure if either the pose error exceeds the limit or the simulation step fails to converge due to excessive contact force.
2) In reality, the episode terminates immediately and is considered failure if either the pose error exceeds the limit or the average marker displacement in the contact area is larger than the threshold.

D. Tactile Feature Extraction and Pretraining

For real tactile sensors, the correspondence between the initial and displaced markers is computed by detecting and tracking the markers in the camera stream, which may fail when the sensor's elastomer undergoes large or rapid deformation. It poses difficulties in designing the tactile feature extractor, which the network should be able to deal with a marker displacement set that is orderless and has a variable cardinality. Moreover, the marker displacements are high dimensional, which limits the RL sampling efficiency. Therefore, we propose an autoencoder structure as the tactile feature extractor based on PointNet [43], and pretrain the network on a large amount of simulated data to extract a low-dimensional and task-agnostic feature from the marker displacements.

1) Tactile Feature Extraction: The structure of the proposed tactile feature extractor is shown in Fig. 4. The input of the network is the concatenation of the original marker positions and the displaced ones. If the number of detected and tracked markers is less than n, padding by replicating will be used. Then, a shared multilayer perceptron (MLP) and a max-pooling layer will function as an approximation of a symmetrical function over the input, to deal with the orderless input. The extracted 512-dim global feature is further encoded by another MLP to be a k-dim encoded tactile feature.

It should be noted that we do not use the difference between the original and displaced marker positions as the network input, because the raw marker positions contain spatial information of the tactile signals, which is crucial for inferring the contact status.

2) Autoencoder Pretraining: To pretrain the encoder for use in various tactile manipulation tasks, we design a corresponding decoder to reconstruct the marker positions. Since the ideal extracted feature should be an overall feature extracted from all marker points and independent of the specific marker distribution, we design the proposed decoder to reconstruct all the marker positions from the original marker positions and the latent feature. As shown in Fig. 4, the encoded k-dim feature is repeated and concatenated with the original marker positions, and then fed into a shared MLP to reconstruct the displaced marker positions. We use $L2$ loss between the reconstructed marker positions and the ground truth for training.

E. Reinforcement Learning Policy

Training a one-step manipulation policy with supervised learning is feasible in simulation, where the ground truth state is accessible. However, we find that it suffers from inferior generalizability and Sim2Real transferrability, which is consistent as the finding in [4]. A possible explanation is that the tactile observations are insufficient to capture the full manipulation state. In contrast, RL can optimize the action policy based on the estimated long-term reward and learn better decisions with partial tactile observations. Therefore in this work, we utilize RL to train the contact-rich manipulation policy. We adopt an off-policy RL method, Twin-Delayed-DDPG (TD3) [44], as the RL algorithm due to its sample efficiency.

1) RL Network Structure: The actor network in TD3 maps the observation to the action, and the critic network evaluates the Q value of the action. Fig. 5 illustrates the structure of the actor network. We use two tactile sensors in all the manipulation experiments in this article, so the observation consists of left and

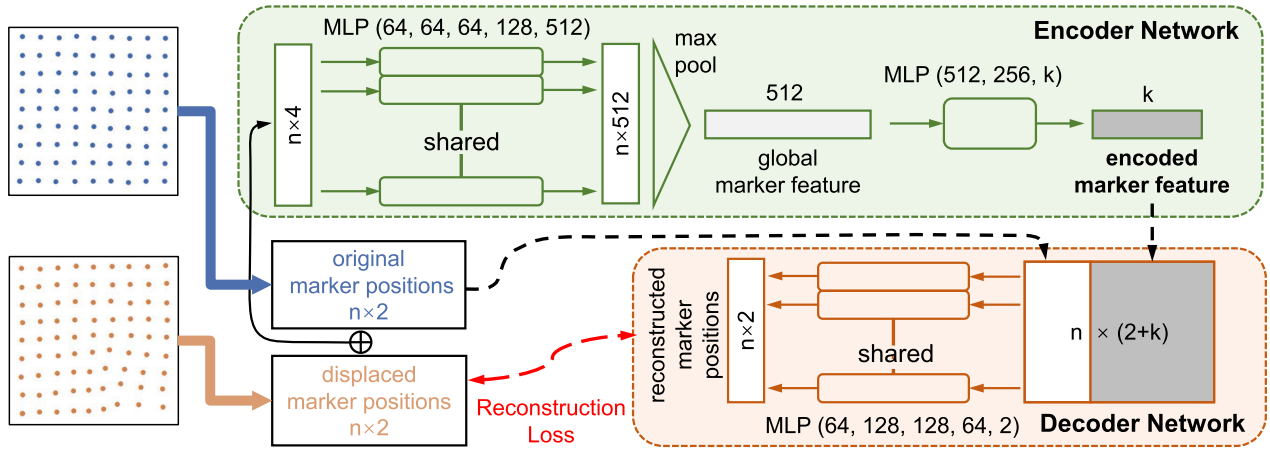


Fig. 4. Overview of the autoencoder structure. To deal with marker permutations and loss of tracking, the PointNet [43] structure is utilized. The encoder takes in concatenated marker positions as input and extracts the k-dim marker flow feature. To reconstruct the deformed marker position, the decoder concatenates the original marker positions with the k-dim feature, and outputs the reconstructed marker positions.

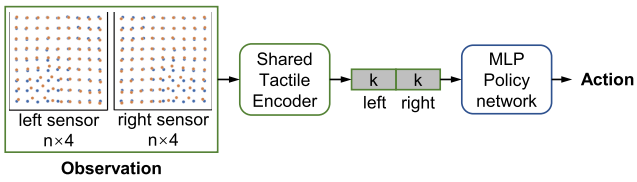


Fig. 5. Structure of the actor network in the reinforcement learning policy. Considering two-finger manipulation, the actor network takes in both sensor's tactile signals. After the shared encoder, the two k-dim features are concatenated and put into the MLP policy network.

right marker displacements. The shared feature extractor transforms the two marker displacements into two latent k-dim features. Then, the MLP policy network concatenates and processes the two features to predict actions. The critic network, on the other hand, uses privileged ground-truth states from simulation instead of raw tactile observations to accelerate training [45].

The definition of actions and ground-truth states may vary for different tasks, while the structures of the actor and critic network remain the same. The task-specified parameters will be introduced in the experiment section.

2) Feature Extractor Fine-Tuning: In Section III-D, we propose to pretrain the tactile feature extractor before training the RL policy to improve the sample efficiency and training stability. However, because the tactile interactions in the pretraining dataset may differ from those in the specific task, we introduce a short self-supervised fine-tune stage of the tactile feature extractor when training the RL policy.

We modify the TD3 algorithm, such that after sampling from the replay buffer, we first compute the reconstruction loss in Section III-D, and then update the feature extractor and the decoder parameters. Next, we freeze the tactile feature extractor and update the critic network and the MLP part in the actor network following the TD3 algorithm. The feature extractor fine-tuning stage continues until a maximum number of updates are reached, or until the reconstruction loss is smaller than a preset threshold.

F. Domain Randomization

Marker-based visuotactile sensors have inherent challenges for Sim2Real transfer due to the discrepancy between simulation and reality. The discrepancy arises from three main sources: 1) the large deformation and aging effects of the elastomer that serves as the sensing surface, which make it difficult to model its physics and identify its material parameters accurately; 2) the camera-imaging process that converts tactile interactions to tactile signals, which depends on various factors such as marker distributions and camera parameters; and 3) the motion errors, uncertainties, or latencies that may occur for real robots, such as the mismatch between commanded and actual robot velocities. To address these challenges, we employ domain randomization techniques in the Sim2Real learning process. We categorize the domain randomization into three subdomains: physical domain, optical domain, and task domain.

1) Physical Domain: The behavior of the tactile sensor, including sensitivity, contact forces, and deformation states, depends on the elastic modulus, Poisson's ratio and friction coefficient. However, due to the age-dependent properties of the elastomer, the actual elastic modulus of the real sensor can hardly be obtained. Moreover, friction in the real world is a complex interaction between the object and the sensor that is difficult to model accurately. Therefore, to improve the Sim2Real transferrability, we randomize the elastic modulus, Poisson's ratio, and friction coefficient within an heuristically determined range.

2) Optical Domain: Due to the low elastic modulus of the tactile sensor elastomer, fabrication-induced deformations may result in variations in the marker distribution. However, since the markers merely serve as a medium to transform the tactile interaction into image features, the exact marker distribution should not influence the underlying tactile perception. To this end, we introduce random perturbations to marker positions.

We assume that the markers form a uniform grid and the randomization includes grid spacing, global translation, global rotation, local perturbation, random noise, and loss of tracking.

Grid spacing changes the distance between adjacent markers that forms a uniform grid. *Global translation* and *global rotation* are applied to the whole marker grid. *Local perturbation* allows a marker to deviate from its ideal position as a grid point. These aforementioned four parameters can emulate the fabrication variances of the sensor elastomer. On the contrary, *random noise* and *loss of tracking* are related to the postprocessing process using computer-vision algorithms, which *random noise* affects the extracted marker positions in each frame and *loss of tracking* allows a marker to disappear at a given probability. The combination of these parameters provides enough degrees of freedom to cover all possible cases for real sensors, promoting the Sim2Real transfer. For camera parameters, we do not apply randomization because camera parameters will have small variation range if mass production is realized; the randomization on markers is sufficient to deal with small camera parameter variations.

3) Task Domain: This domain mainly includes motion errors, uncertainties, or latencies that may occur for real robots. For example, when tactile feedback serves as a stopping signal, the actual position where the robot stops is affected by the robot's speed, acceleration, and latencies, etc.; when a gripper grasps an object, the object may deviate from the ideal grasping position. We randomize the parameters in this domain according to the specific task settings, which will be introduced in the experiment section.

IV. Experiment: Data Collection, Training, and Performance of Autoencoder

In Section III-D, we have proposed a tactile feature autoencoder to extract low-dimensional and task-agnostic features for tasks. In this section, the data collection, training of the autoencoder are presented in detail. The performance of the autoencoder is validated on several datasets from simulated manipulation tasks.

A. Autoencoder Data Collection

To make the tactile encoder more generalizable, a diverse dataset for pretraining should be collected. We design an independent simulation scene to collect the possible interactions between the sensor and objects. As shown in Fig. 6(a), a simple scene is created where a cuboid object is interacting with the sensor elastomer. Despite the scene's simplicity, there are a number of parameters to be randomized. First, the size, position, and orientation of the cuboid object are randomized. Second, in the first stage of interaction, the indentation depth is randomized. Third, after the indentation is finished, the sensor will make a relative movement to the object and the translational and rotational component of the movement are randomized. For parameters in physics domain and optical domain, the randomization techniques in Section III-F are used.

From each run of the aforementioned simulation, we extract three keyframes: the initial tactile reading, the one after indentation and the final one. From these three keyframes, six pairs of tactile readings can be combined as the training data for the autoencoder. Some samples of the collected training data are shown in Fig. 6(b). In total, 400 K pairs of original-displaced markers are collected, among which 70% are used for training, 15% for validation, and 15% for testing.

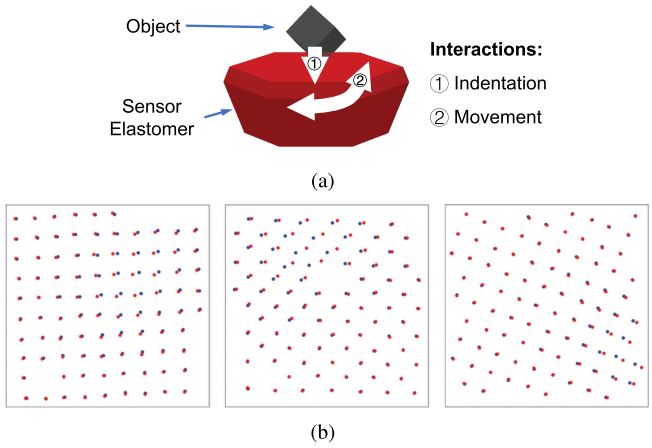


Fig. 6. Data collection of the proposed tactile autoencoder. (a) Simulation scene for collecting autoencoder pretraining data. (b) Some samples of the collected marker flow training data. Markers in red are the displaced ones.

B. Autoencoder Training

1) Training Details: We use $L2$ loss between the ground-truth displaced marker positions and the reconstructed ones to train the tactile encoder and decoder simultaneously

$$L = \frac{1}{2n}\sum_{i=0}^{n-1}[(u_i' - \hat{u}_i')^2 + (v_i' - \hat{v}_i')^2] \tag{5}$$

where $\hat{u}_i', \hat{v}_i'$ are the reconstructed marker positions. For our sensor, in the region of interest there are approximately 100 markers so we select $n = 128$ as the input size of points. We use Adam optimizer and set the learning rate as 2e$-$4, with a batch size of 128. The training lasts for 2000 epochs to ensure convergence.

2) Determination of Latent Feature Dimension: The dimension k of the extracted latent feature is a hyperparameter that needs to be determined in advance. We compare different latent dimensions k from 4 to 256, to find an appropriate value that has both sufficient representative capability and compactness. In Fig. 7, we present the smoothed validation loss over training epochs. It shows that when the latent dimension k is low, i.e., $k = 4, 8, 16$, the reconstruction loss is obviously higher than other cases, which means the encoder does not have enough representative capability. When $k = 32$, the reconstruction loss is also higher than those when $k = 64, 128, 256$, but the difference is minor. Therefore, we choose $k = 32$ to balance representative capability and compactness.

C. Autoencoder Performance on Unseen Data

We test the pretrained autoencoder on the datasets collected in the specific manipulation tasks, namely *peg-in-hole insertion*, *plug adjustment*, and *lock opening*, which will be introduced in the following sections. The datasets are from the replay buffer

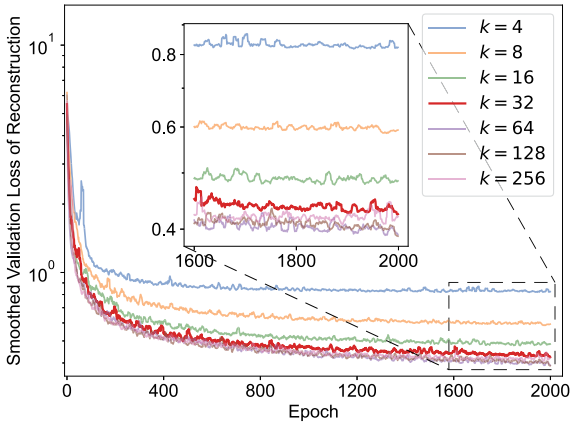


Fig. 7. Reconstruction loss in validation during autoencoder training, where the latent feature has different dimensions. The loss is in log scale and is smoothed using 10-element moving average. $k = 32$ is chosen as the latent feature dimension because it has both compactness and sufficient representative capability.

TABLE I
RECONSTRUCTION LOSS OF THE AUTOENCODER ON SEEN AND UNSEEN DATASETS

Dataset	Reconstruction loss
pre-training dataset (seen)	0.410
dataset from peg-in-hole task	0.597
dataset from plug adjustment task	0.837
dataset from lock opening task	0.975

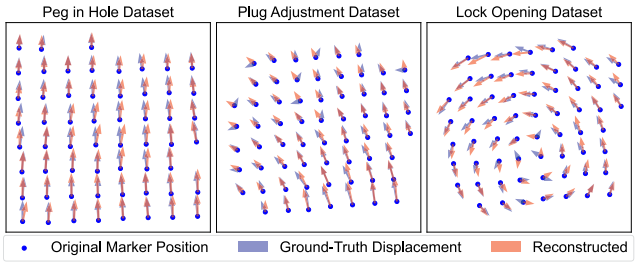


Fig. 8. Visual comparison between the ground-truth and reconstructed marker displacements in different datasets. The datasets are collected from the replay buffers in RL training. The arrows representing the displacements are 2× scaled.

collected during the RL training process and each contains 400 K test samples.

Table I summarizes the average reconstruction loss of the autoencoder on seen and unseen datasets. In the pretraining dataset, the reconstruction loss is 0.410, which means the reconstructed marker positions have a root-mean-squared distance of $\sqrt{2 \times 0.410} \approx 0.9$ pixel from the ground-truth ones [see (5)]. Considering that the markers have random noises in simulation, this proves that the trained autoencoder has sufficient capability of extracting compact and representative tactile features. Table I also shows that in unseen datasets, the reconstruction losses are less than 1, proving that the trained autoencoder is generalizable. Comparison between the ground-truth and reconstructed marker displacements is visualized in Fig. 8.

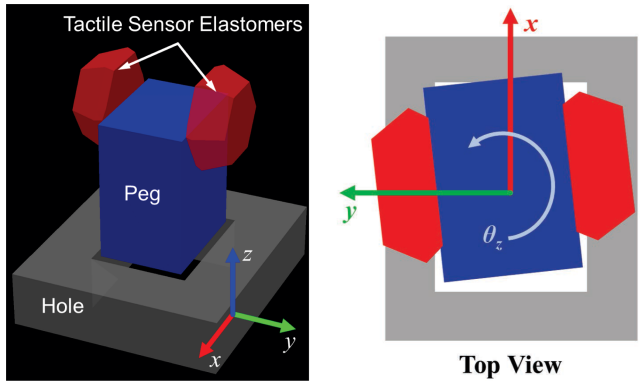


Fig. 9. (a) Our simulation environment of the peg-in-hole insertion task. Visualization is realized by SAPIEN [46]. (b) Top view of the task. The initial pose error has three degrees of freedom.

V. EXPERIMENT: PEG-IN-HOLE TASK

A. *Definition of* Monotone Insertion

The original version of the tactile-RL insertion task is proposed in [4]. In this task, a gripper with tactile sensors is controlled to insert an object into a rectangle-shaped hole with a random initial pose misalignment (see Fig. 9). The random pose misalignment is 3-DoF, consisting of the 2-DoF $x-y$ plane translation and the rotation around z-axis [see Fig. 9(b)]. Each insertion attempt is a downward motion along the $-z$ direction. If getting blocked, the gripper will *retreat to the initial height* and adjust the pose according to the trained policy, followed by another insertion attempt. This task is later completed in a Sim2Real way in [14].

As discussed in [4], [14], this task is difficult in that the hole is unchamfered, and thus the robot must recognize nuances in the tactile readings to decide the insertion pose adjustment, in the case where the object hits the hole at four corners.

We introduce a novel variant of the tactile insertion task, which we call *monotone insertion*. Unlike the original task, our variant does not allow the gripper to retract during the insertion process. Instead, the gripper descends by a fixed distance at each step. Thus, the gripper motion is *monotone* along the z-axis. This modification makes the task more realistic, intuitive, and efficient. However, it also poses higher demands on the simulator, which has to capture the continuous deformation of the elastomer under insertion. We evaluate our method on the *monotone insertion* task variant in Sections V-D, V-G, V-H, V-I, and V-K. For a fair comparison, we also report our method's performance on the original version of the task in Section V-J.

B. *Experimental Setup*

In [4], there are cuboid, hexagonal, cylindrical, elliptical objects, while the holes are all rectangular. In this work, the task is stricter, that the holes have the same shape as the objects (pegs). Three representative shapes we choose to test in reality are shown in Fig. 10(b). The object to insert is grasped by a parallel gripper (Robotiq Hand-E), and the object width is unified to be 30 mm. The clearance between the object and the hole is

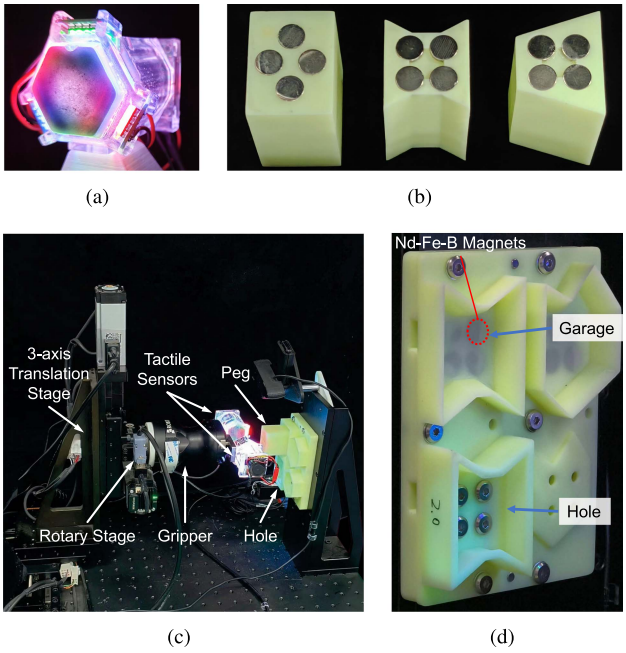


Fig. 10. Devices for the peg-in-hole experiment. (a) Our self-made GelSight-type tactile sensor. (b) These three shapes, namely *Cuboid*, *Concave*, and *Trapezoid*, are used to test the trained policy in reality. The magnets are used for pose resetting. (c) We use a combined motorized stage and a parallel gripper to execute the tasks. (d) Mechanism for autoresetting the pose. The garage has large chamfers to tolerate possible peg pose errors. There are also magnets at the bottom of the garage for pose resetting.

2 mm, which is about 6.7% of the object width, similar to [4], [14]. Correspondingly, the initial error range is set to [±5 mm, ±5 mm, ±10°]. In our monotone insertion task, the gripper will move down 0.125 mm at each step; the maximum number of allowed pose corrections is 8. Note that the maximum attempt number in [4] and [14] is 15, showing that our monotone task is more difficult and efficient. The four-axis motion (three-axis translation, one-axis rotation) is provided by a translation stage with orthogonal axes and a rotary stage [see Fig. 10(c)].

To better facilitate evaluation of the trained RL policies in reality, we design an autoresetting mechanism as shown in Fig. 10(d). When insertion fails, the pose of the peg may change largely because of the large contact force caused by incorrect actions, which will impair the performance of the following episodes. Therefore, to autonomously reset the pose, we use a "garage" with large chamfers for the peg. We glue Nd-Fe-B magnets at the bottom of the garage and the peg to provide enough attractive forces for resetting the peg's pose. It should be noted that the magnets will not interfere with the peg-in-hole insertion process because the screws to fasten the hole are made of 304 stainless steel and thus will not be attracted by the magnets. Whether to do pose resetting is determined by the average marker displacements in both sensors' contact area.

Each test in reality contains 57 episodes starting with a series of predefined initial errors. The predefined initial errors are uniformly sampled within the range [±5 mm, ±5 mm, ±10°], ensuring fair comparison across different tests. The metrics summarized after each test are the success rate and the average number of pose correction attempts in all successful episodes. Note that when success rate is high, the average number of attempts may be also large because more challenging cases are taken into calculation. So the success rate should be the first to compare; when the success rates of two policies are similar, the average number of attempts can be compared to reveal their optimality.

C. Details of RL Policy Training

At the beginning of each episode, the original marker positions of each tactile sensor are stored before any insertion attempt. At each step, the sensor's displaced marker positions corresponding to the original ones are recorded through a marker-tracking algorithm. Then the original marker positions and displaced ones are concatenated to form the observation. Observations from the two sensors are fed into the actor network. The output action is a 3-dim vector $[a_x, a_y, a_{\theta_z}]$. Note that the action is defined relative to the peg (or equivalently the gripper), not to the hole. The state input for the critic network is the peg's ground-truth offset $[e_x, e_y, e_{\theta_z}]$.

One episode terminates under three conditions: 1) the insertion succeeds, 2) the pose error is larger than a threshold in any axis, or 3) the number of insertion attempts is over a limit. We set the max error in x and y axis to be 12 mm, and 15° in θ_z. The maximum number of attempts is 8 for our monotone insertion task and 15 for the original version of the task.

The reward function at each step is shown in (6). It consists of four parts: the decrease in error $e_{t-1} - e_t$, the constant penalty at each step $P = 1$, the final success reward R_{final}, and the penalty for too large error R_{fail}. e_t is calculated from the error in each axis, where e_x and e_y are in mm and e_{θ_z} is in degrees. To prevent the policy from quick failure, the penalty for large error is used where $t_{\max}$ is the maximum allowed number of attempts

$$
\begin{aligned}
R_t &= e_{t-1} - e_t - P + R_{\text{final}} + R_{\text{fail}} \\
e_t &= \sqrt{e_x^2 + e_y^2 + e_{\theta_z}^2} \\
R_{\text{final}} &= \begin{cases} 10, & \text{if success} \\ 0, & \text{otherwise} \end{cases} \\
R_{\text{fail}} &= \begin{cases} -2(t_{\max} - t)P, & \text{if } |\mathrm{e_{x,y}}| \geq 12 \\ & \text{or } |\mathrm{e}_{\theta_z}| \geq 15 \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{6}
$$

The domain randomization method is mainly introduced in Section III-F. For randomization in the task domain, the indentation depth into each sensor's surface varies from 0.5 to 1.25 mm, and the peg's relative position to the sensors varies from −1 to 1 mm in x direction and from −5 to 5 mm in z direction.

As mentioned in Section III-E, we adopt TD3 as the RL algorithm. For unmodified TD3, we use Stable-Baselines3 [47] as a default implementation. For the proposed fine-tuning method of tactile feature extractor, we make some modifications on it as introduced in Section III-E. To efficiently train the RL policy, we use 20 parallel environments with 500 K environment steps in total. We note here for comparison that Dong et al. [4] used startup bootstrap and a carefully designed curriculum learning

scheme to efficiently train the policy in reality (therefore no randomization) with 3K steps; Xu et al. [14] used PPO algorithm in simulation and the policy is trained for 5M steps. Considering that zero-shot Sim2Real policies usually have large variance, in the following experiments we train and test the policy several times to provide more reliable results.

D. Advantage of FEM Simulation Over Rigid-Body Simulation

We first compare our FEM-based simulator with the state-of-the-art work [14] on tactile Sim2Real in the monotone peg-in-hole insertion task. Based on their open-source code,[1] we implement the monotone insertion task using our experimental settings. We modify the object size, the hole size, the tactile readings according to our real devices. The domain randomization parameters are unchanged. The level of noise added to tactile readings is adjusted to be of a similar ratio to the noiseless value as in the original code. To keep most of Xu et al.'s [14] implementation, we do not change the RL algorithm PPO, the convolutional RNN policy, and the reward function. For Sim2Real transfer, we also apply the normalization technique proposed in [14].

It should be noted that the open-source simulator DiffHand [48] used in [14] currently only supports the collisions where one object must be a primitive shape[2] (e.g., cube, sphere, etc.). Therefore, only the cuboid object is considered in this comparison.

For comparison, we train our proposed method using the proposed RL scheme (without pretraining, for fairness) in Section III-E. With the tactile encoder and the randomization in optical domain, the trained policy can be directly transferred to reality *without any normalization*. This advantage can be attributed to the following two reasons. First, in our simulation, the synthesized marker displacement is the same as the real tactile readings, so there is no need to do conversion or normalization as Xu et al. [14] did. Second, with the domain randomization technique, the proposed tactile encoder can extract features from raw tactile readings, regardless of the specific marker distribution.

We train Xu et al.'s [14] policy and our method ten times with different random seeds and evaluate the trained policy in our real environment. The success rate and number of pose correction attempts in each test are presented in Fig. 11, while each group's average number and standard deviation are summarized in Table II. To compare the Sim2Real gap of the two simulators, we also show the evaluation results in simulation in Fig. 11. Although the two methods have similar success rate in simulation, our method performs significantly better in the real environment. The difference between our simulation and reality is small, in terms of both success rate (94.5% in Sim, 95.1% in Real) and attempt number (3.63 in Sim, 3.55 in Real). This proves our method has a smaller Sim2Real gap in the monotone peg-in-hole insertion task. The reason is that, in the monotone insertion task, the peg and the hole keep contacting

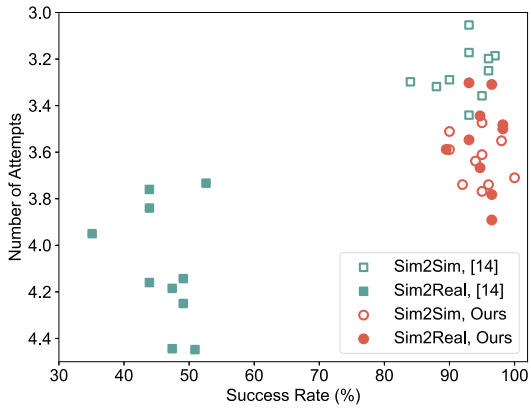


Fig. 11. Distribution of all the Sim2Sim and Sim2Real results of our method and Xu et al.'s [14] method on the monotone peg-in-hole insertion task. The metrics are the success rate (higher is better) and the number of attempts (lower is better). Our method has a higher success rate and a smaller Sim2Real gap.

TABLE II
SIM2SIM AND SIM2REAL COMPARISON BETWEEN OUR METHOD AND [14] ON THE MONOTONE PEG-IN-HOLE INSERTION TASK

	Success Rate (%) ↑	Number of Attempts ↓
Sim2Sim, [14]	92.50 ± 4.09	3.26 ± 0.11
Sim2Sim, ours	94.50 ± 3.20	3.63 ± 0.10
Sim2Real, [14]	46.33 ± 4.96	4.09 ± 0.26
Sim2Real, ours	**95.08 ± 2.70**	**3.55 ± 0.19**

until a successful insertion, so the tangential deformation and the relative displacement of the peg to the sensors always exist in the elastomer during insertion. Our FEM-based simulation method is able to capture the complex and continuous deformation, thus achieving a smaller Sim2Real gap. In rigid-body-based simulation proposed by Xu et al. [14], however, the penalty-based tactile model cannot account for the tangential deformation because their tangential forces are approximated by friction. Therefore, during the sequence of movements in the monotone insertion task, the tactile forces, especially the tangential restoration forces by the elastomer, cannot maintain, which differs largely from the reality. In conclusion, Xu et al.'s [14] method sacrifices physical accuracy for efficiency, which is the reason why our method performs better in this task.

E. Simulation Speed

We conduct a comparison of the simulation speed of our method and that of Xu et al. [14] on the monotone peg-in-hole insertion task. For our method, the tetrahedral mesh of the sensor elastomer comprises 129 vertices and 412 elements. In the simulation scene, there are 878 elements in total. We set the simulation timestep Δt as 0.1 s and each environment step typically contains approximately five timesteps. The simulation is run on a PC with Intel Core i7-13700 K CPU[3] and 64 GB memory using eight parallel environments, achieving a speed of 43 environment steps per second. For comparison, Xu et al.'s [14]

[1] [Online]. Available: https://github.com/eanswer/TactileSimulation
[2] [Online]. Available: https://github.com/eanswer/TactileSimulation/issues/3
[3] The actual frequency when running simulation is about 5 GHz.

TABLE III
SIMULATION PARAMETERS WITH AND WITHOUT DOMAIN RANDOMIZATION (DR)

Sub-domain	Parameter Name	w/ DR	w/o DR
Physical	elastic modulus (MPa)	[0.3, 1]	0.65
	Poisson's ratio	[0.3, 0.47]	0.4
	friction coefficient	[0.5, 1]	0.75
Optical	grid spacing (mm)	[1.95, 2.05]	2
	global translation* (mm)	[0, 1]	0
	global rotation (rad)	[-0.1, 0.1]	0
	local perturbation* (mm)	[0, 0.1]	0
	random noise**(pixel)	0.5	0
	lose-tracking probability	0.01	0
Task	indentation depth (mm)	[0.5 1.25]	0.8
	peg offset x (mm)	[-1 1]	0
	peg offset z (mm)	[-5 5]	0

* global translation and local perturbation are applied in both directions
** random noise is Gaussian, in both directions

TABLE IV
ABLATION STUDY RESULTS FOR DOMAIN RANDOMIZATION

	Success Rate (%) ↑	Number of Attempts ↓
w/ all domains	**95.08 ± 2.70**	**3.55 ± 0.19**
w/o physical domain	80.70 ± 10.44	3.90 ± 0.67
w/o optical domain*	10.01 ± 4.71	3.34 ± 0.87
w/o optical domain**	56.84 ± 12.87	3.61 ± 0.38
w/o task domain	61.57 ± 13.64	4.35 ± 0.45

* the simulated marker positions are NOT aligned with real sensors
** the simulated marker positions ARE aligned with real sensors

rigid-body-based method reaches 151 steps per second in the same conditions. Our single real experimental device can reach approximately 0.23 steps per second. The speed of the real-world pipeline is mainly limited by the acceleration and deceleration of the stage because we implemented a conservative acceleration strategy to ensure operational stability. These results demonstrate that while our FEM-based simulation is less efficient than rigid-body-based simulation, the difference between the two methods does not reach an order of magnitude. More importantly, our simulation method is two orders of magnitude faster than reality, which can greatly accelerate the policy training.

F. Ablation Study for Domain Randomization

In this section, an ablation study is conducted to validate the domain randomization techniques. In Section III-F, the parameters are classified into three subdomains: physical domain, optical domain, and task domain. We therefore design three experiments, each of which has nonrandomized parameters in one subdomain, respectively. The randomization range of the parameters, and the nonrandomized values for ablation study, are summarized in Table III. Other conditions of training are kept the same as in Section V-D. In the experiments, training and test are repeated ten times for each condition, and the results are summarized in Table IV.

Table IV shows that the randomization in the optical domain has the largest influence. This is because without optical domain randomization, the tactile feature extractor is unable to correctly interpret the tactile readings from a different marker distribution. We note here that, the default marker distributions used in the simulation are different from those in reality, which leads to a significant performance degradation. Once the simulated markers are aligned with the real two sensors used in test, the Sim2Real success rate rises to 56.84%, which is better than [14] (46.33%). Furthermore, the Sim2Real performance deteriorates in the absence of the randomization in the other two domains, as indicated by the decline in success rate and the increase in the number of attempts. This proves that the randomization in the physical and task domain plays a crucial role in enhancing the robustness and optimality of the trained policy.

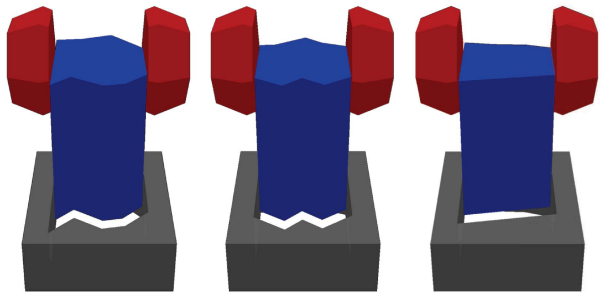


Fig. 12. Three examples of randomly generated shapes. We do not necessarily assume symmetry or convexity of the shape. These arbitrary shapes used in simulation result in a generalizable peg-in-hole insertion policy.

G. Generalizable Peg-in-Hole Policy for Arbitrary Shapes

For real peg-in-hole applications, the shapes of pegs and holes are widely distributed and the precise geometry information is usually unavailable. Therefore, we aim to train a peg-in-hole policy that is generalizable to arbitrary shapes. In order to achieve this goal, we randomize the shapes of pegs and holes during RL training, which is costly in the real world but requires minimal effort in simulation.

We randomly generate 10 000 polygons as the shape of the peg and the hole, as shown in Fig. 12. Note that we do not assume the shape's symmetry or convexity. In training of the RL policy, a shape will be randomly selected when the environment resets. To validate the generalizability of the trained policy, we test the policy in reality using three predefined shapes, namely *Cuboid*, *Concave*, and *Trapezoid* [see Fig. 10(b)]. These three shapes are chosen because the cuboid shape is more difficult to insert compared with hexagonal or circular shapes [4]; the concave shape and the trapezoid shape represent the objects with concavity and nonsymmetry, respectively.

We evaluate the arbitrary-shape training scheme through an ablation study. RL policies are trained in three training conditions: 1) train on cuboid shape only, 2) train on the exact three test shapes, and 3) train on the proposed arbitrary shapes. The trained policies are tested on the three selected shapes in Fig. 10(b). The training and test in each condition are repeated ten times to get more reliable results. The experimental results are summarized in Fig. 13. From the results, we draw the following conclusions:

1) By training on cuboid only, we obtain a "specialist" policy that performs best and most stably on the cuboid shape

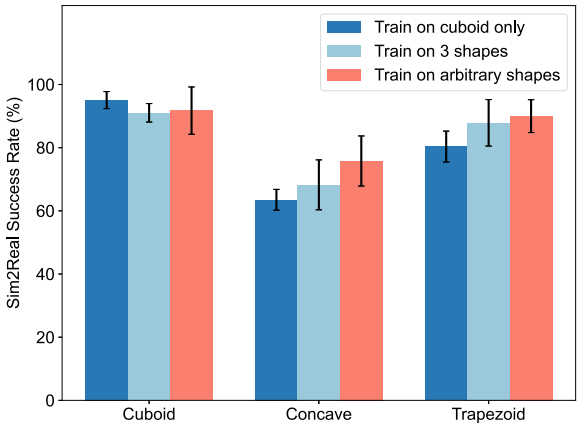


Fig. 13. Ablation study of the proposed arbitrary-shape training scheme. Comparison is done among training with arbitrary shapes, training with the exact three shapes and training with only one shape. The policy trained with arbitrary shapes has the best overall performance. This proves that a "generalist" policy performs better in unknown environments and in Sim2Real.

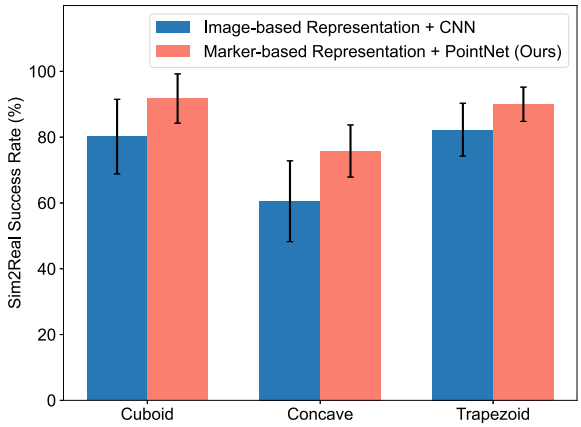


Fig. 14. Ablation study of the proposed tactile representation. Comparison is done between the proposed marker-based tactile representation and the image-based presentation. The success rates are compared over three test shapes. The average success rate and standard deviation are calculated from ten independently trained weights for each representation.

among the three groups. It reaches the highest success rate and the lowest standard deviation in Sim2Real test of the cuboid shape. It can also generalize to the concave and trapezoid shape, but the success rates are lower than the two counterparts.

2) A "generalist" policy is advantageous in unknown environments and is beneficial for Sim2Real. We notice that the policy trained with arbitrary shapes can reach similar success rates as the "specialist" policy on the cuboid shape; on the concave and trapezoid shape, it performs better than the "specialist" policy. We surprisedly find that the "generalist" policy performs better than the policy trained on the exact three shapes. It may be because the real environment exhibits unavoidable deviations from the ideal simulation and therefore the policy trained with the exact three shapes has inferior performance.

H. Advantage of the Proposed Tactile Representation

In this section, we design an ablation study to compare the proposed marker-based tactile representation with image-based tactile representation. In [4], the researchers have found that binary marker images perform better than raw RGB images captured with the sensor. Therefore, in this section, we adopt the binary marker image as the image-based tactile representation. In simulation, the marker image is generated using OpenCV from the calculated marker pixel coordinates. The image resolution is 320×320 and then binarized and downsampled to 256×256 before being fed into CNNs. Like the marker-based representation, the image-based representation also contains two frames, and they are stacked to be a 2-channel binary image. The corresponding actor network contains a shared CNN feature extractor for both left and right sensor images. The dimension of the extracted latent feature and the following MLP network is 32, the same as the marker-based one. Other conditions, such as the randomization parameters, remain the same in this comparison. To implement the randomization of the marker images, the marker positions are randomized first, and then the marker image is generated based on the randomized markers. The pre-training uses the reconstructed displaced markers for self-supervision, and different representations require different decoder networks and different loss functions. In order to eliminate the potential influence of the pretraining, we do not apply it for both representations. Both policies are trained with the proposed arbitrary shapes for 500 K environment steps and the best weights are used for testing in reality.

Fig. 14 shows the comparison result of the tactile representations. It shows that although the randomization parameters are the same, using the proposed marker-based representation is advantageous over the image-based representation. The possible reason is that the convolutional operator in CNN has a translation-invariant property, which makes it difficult for CNN to globally extract spatial features from the tactile observations. Moreover, the randomization in marker distributions impedes the CNN's convergence. On the contrary, with the design of the symmetrical function approximation, the marker-based tactile representation and point cloud learning architecture can inherently deal with the marker position input and extract both global and local tactile features. The randomization enhances its generalizability and further improves Sim2Real performance.

I. Advantage of the Pretrained Tactile Encoder

We follow Section III-E to validate the effectiveness of the proposed pretraining and fine-tuning method. In this experiment, we show that with the pretrained tactile encoder, the policy can achieve considerably high Sim2Real success rate even at very early steps. We test the real success rate on the aforementioned three preset shapes using the trained weight at different RL steps, from 4 to 500 K. The experiments with and without pretraining are repeated three times using different random seeds. The Sim2Real results are presented in Fig. 15. As shown in the figure, at very early steps (i.e., before 10K), the policy with pretraining shows a large advantage over the one without pretraining. At 8K steps, the policies with pretraining can achieve the success rates

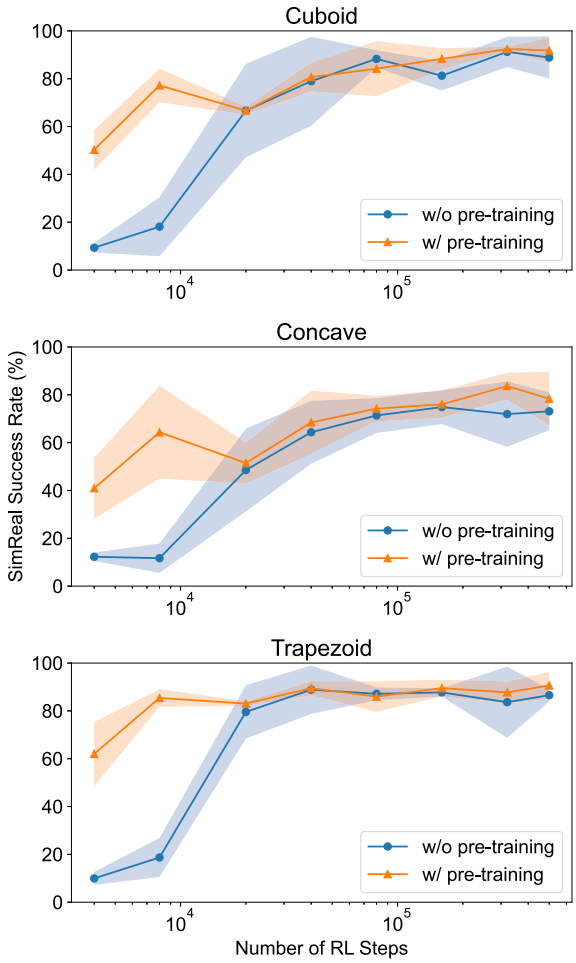


Fig. 15. Comparison between the policy with pretraining and without pretraining. Sim2Real success rates at different RL steps on three preset peg shapes are reported and each line is averaged over three weights trained with different random seeds. The horizontal axes are in log scale.

of 77.2%, 64.3%, and 85.4% for the three shapes, respectively, comparable to those of the policies without pretraining at 40K steps.

J. Comparison in the Original Version of the Insertion Task

In the original version of the peg-in-hole insertion task proposed by [4], [14], if an insertion attempt fails, the gripper will retreat to the initial height, cancelling the contact between the peg and the hole. Although we conduct most of the zero-shot Sim2Real experiments on the proposed *monotone insertion* task, we also provide our method's results on the original version of the task. We repeat the training and testing process ten times to get more reliable results, as shown in Fig. 16. The average success rate and number of attempts are summarized in Table V.

Despite the fact that our sensors, robot, and objects are different from [4], and [14] we have tried our best to reproduce their results on our own hardware. The success rate of [14] that we reproduce is 85.95 $\pm$ 14.97%, slightly higher than what [14] reported (83%). The average attempt number (3.98 $\pm$ 0.49) is also smaller than what [14] reported (4.81). From these statistics,

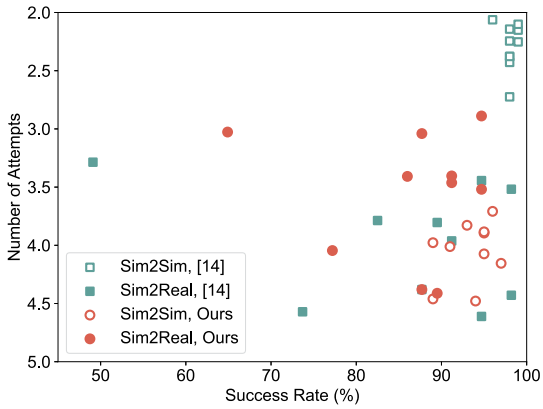


Fig. 16. Comparison of the Sim2Sim and Sim2Real results of our method and Xu et al.'s [14] method on the original peg-in-hole insertion task. The metrics are the success rate (higher is better) and the number of attempts (lower is better).

TABLE V
SIM2SIM AND SIM2REAL COMPARISON BETWEEN OUR METHOD AND [14] ON THE ORIGINAL PEG-IN-HOLE INSERTION TASK

	Success Rate (%) ↑	Number of Attempts ↓
Sim2Sim, [14]	98.10 ± 0.88	2.29 ± 0.20
Sim2Sim, ours	93.40 ± 2.84	4.05 ± 0.26
Sim2Real, [14]	85.95 ± 14.97	3.98 ± 0.49
Sim2Real, ours	86.48 ± **9.09**	**3.56 ± 0.55**

we believe that our reproduction is reasonable and reliable. The difference may be attributed to the hardware differences, e.g., our tactile sensor, the object width, and peg-hole clearance are different; our motion stage may have higher positioning accuracy; our objects and holes are 3D-printed using stereolithography, instead of fused deposition modeling.

For our method, we achieve a similar Sim2Real success rate as [14] (86.48% compared to 85.95%). While the success rates are similar, our average attempt number is significantly smaller (3.56 compared to 3.98). What is more, as can be noticed, Xu et al.'s [14] method has very few attempts in simulation (2.29) while having more attempts in reality (3.98). On the contrary, our method has fewer attempts in reality (3.56) than in simulation (4.05), which is reasonable because the success rate in simulation is also higher, leading to more challenging cases to be counted when calculating the average attempt number. These results prove that with FEM as the physics simulation method, our simulator has a smaller Sim2Real gap and thus the optimal actions in simulation are closer to those in reality than [14].

K. Generalizability to Different Sensor Instances

The modulus of the sensor's elastomer varies over time due to degradation of its molecular structure [49]. As a result, sensors produced using the same manufacturing process yet at different dates will have different properties. Additionally, it should be noted that there are slight variations in the marker spatial arrangement and camera parameters among different sensor instances. Therefore, the policy's generalizability to different

TABLE VI
SIM2REAL COMPARISON BETWEEN DIFFERENT SENSOR INSTANCES

Sim2Real Condition	Success Rate (%) ↑	Number of Attempts ↓
Sensor Pair 1	91.75 ± 7.48	3.75 ± 0.17
Sensor Pair 2	87.37 ± 11.81	3.66 ± 0.28

TABLE VII
SIM2SIM COMPARISON IN CASE OF ELASTIC MODULUS MISMATCH

Sim2Sim Condition	Success Rate (%) ↑	Number of Attempts ↓
no mismatch in E^*	90.20 ± 3.46	3.84 ± 0.10
$E_{\text{left}} = 0.7E_{\text{right}}$	87.60 ± 4.57	3.95 ± 0.15

* E stands for elastic modulus

sensor instances is critical for real applications. However, it will be highly costly, if possible, to train a generalizable policy using real sensors with different properties. In this section, we demonstrate that the policy learned by our Sim2Real method has good generalizability to different real sensor instances.

Table VI shows the results on two sensor pairs. The policies are trained on arbitrary shapes as described in Section V-G. Ten policies trained with different random seeds are tested on the two pairs. Only the cuboid shape is tested in reality. The Sensor Pair 1 consists of two sensors that are manufactured in a same batch and have similar properties. The Sensor Pair 2 consists of two sensors, where the elastic modulus of the left sensor is different from that of the right sensor. The mismatch violates the common assumption that both sensors in a pair have similar properties and poses additional challenge for control policies, which is again validated by Sim2Sim experiments at the end of this section.

The experimental results show that the learned policies have good performance on the two pairs, demonstrating the effective generalizability of our approach to different sensor instances. This achievement can be attributed to the physics, optical domain randomization, which effectively captures the variations among different sensor instances.

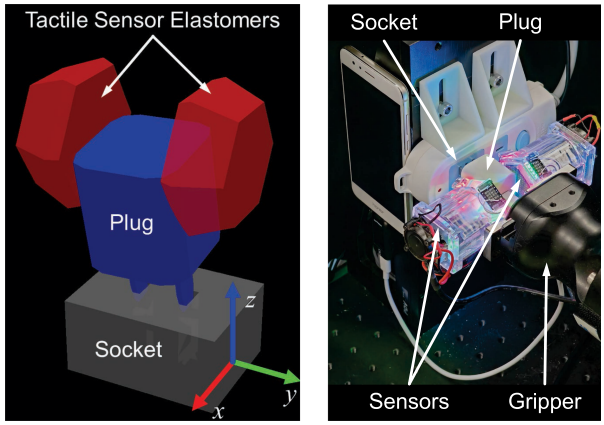
We further replicate the performance decline phenomenon caused by the mismatch in simulation. As shown in Table VII, the success rate decreases from 90.2% to 87.6% when the elastic modulus of the left sensor is 70% of the right sensor. The result demonstrates that our simulation method has a small Sim2Real gap.

VI. OTHER EXPERIMENTS

The previous peg-in-hole insertion task requires accurate tactile simulation and provides quantitative metrics for evaluating Sim2Real performance. In this section, we introduce two additional tasks that more closely resemble real-life scenarios: the *plug adjustment* task and the *lock opening* task.

A. Plug Adjustment Task

1) Task Description: The purpose of this task is to adjust a plug's pose and insert it into the corresponding socket from the initial offsets. Because the hole-searching problem is not the focus in our work, the end of the plug's blades is initialized aligned with the holes of the socket. Fig. 17(a) shows the simulation setup of the task. The two-axis initial offset is defined in the plane parallel to the socket hole surface. The offset range is -8 to 8 mm in x direction and -3 to 3 mm in y direction.



(a) (b)

Fig. 17. Simulation setup and real setup of the plug adjustment task.

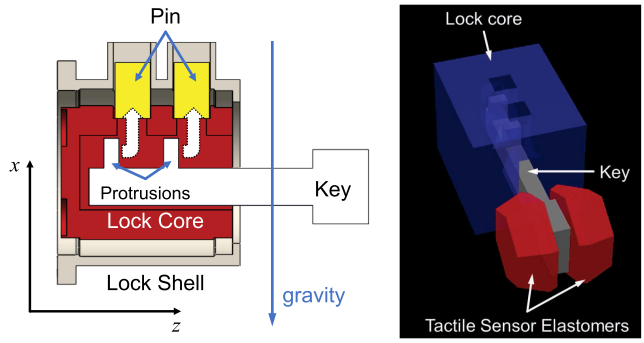
2) Details of RL Policy Training: Similar to the peg-in-hole task, the tactile observation is formed by stacking the original marker positions at the beginning of each episode and the displaced marker positions. Different from the peg-in-hole task, the action is defined as a 2-dim vector $[a_x, a_y]$. The state input for the critic network is a 9-dim vector consisting of the ground-truth plug-socket offset (2-dim), the position (3-dim) and orientation (in the form of quaternions, 4-dim) of the plug.

The terminal state is either that the task succeeds, or that the number of steps is larger than the maximum, or that the offset is larger than a threshold. The task succeeds if the insertion depth is larger than 9 mm and the rotation angle of the peg is smaller than 2° around x-axis and 5° around y-axis.

The reward function shown in (7) consists of four parts: the decrease in error $e_{t-1} - e_t$, the constant penalty at each step $P = 1$, the final success reward R_{final}, and the penalty for too large error R_{fail}. e_t is calculated from the error in each axis, where e_x and e_y are in mm. To prevent the policy from quick failure, the penalty for large error is used where $t_{\max}$ is the maximum allowed number of attempts.

$$
\begin{aligned}
R_t &= e_{t-1} - e_t - P + R_{\text{final}} + R_{\text{fail}} \\
e_t &= \sqrt{e_x^2 + e_y^2} \\
R_{\text{final}} &= \begin{cases} 10, & \text{if success} \\ 0, & \text{otherwise} \end{cases} \\
R_{\text{fail}} &= \begin{cases} -2(t_{\max} - t)P, & \text{if } |\mathrm{e_{x,y}}| \geq 12 \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{7}
$$

The domain randomization method is mainly introduced in Section III-F. For randomization in the task domain, the plug's



(a) (b)

Fig. 18. Lock opening task. (a) Structure of the lock. With the protrusions on the key, it can lift the pins up and thus unlock the lock core. (b) In the simulation scene, the lock core is simplified.

indentation depth into each sensor's surface varies from 0.5 to 1.25 mm, and the plug's relative position to the sensors varies from -1 to 1 mm in x direction and from -5 to 5 mm in z direction.

To efficiently train the RL policy, we use 20 parallel environments with 100K environment steps in total. The proposed pretraining and fine-tuning of the tactile feature extractor are used.

3) Results of Sim2Real Experiments: As shown in Fig. 17(b), in the real plug adjustment task, if the plug is successfully inserted, the smart phone will be charged to indicate the result. We train the RL policy in simulation with three different random seeds and test the policies in reality. Each test has 20 episodes. In all tests, the smart phone is successfully charged. However, in some cases, the plug still has a pose error large than the threshold in Section VI-A2, resulting in a loose contact. So the residual pose error when each episode finishes is manually measured from the captured images to determine whether the episode is successful. As a result, the three tests have the success rates of 95%, 85%, and 90%, respectively, which demonstrates the effectiveness of our Sim2Real method.

B. Lock Opening Task

1) Task Description: Fig. 18(a) shows the structure of the lock assembly. In this task, the rotation of the lock core is constrained to the lock shell by two pins because of gravity. The key has two protrusions at the corresponding position of the pins. The purpose of the task is to use a parallel griper to manipulate the key to lift up the two pins and therefore unlock the core from the shell. Because the initial position of the key to the lock is unknown, a feasible policy must utilize tactile feedback to localize the position of the pins and then fulfill the task.

In simulation [see Fig. 18(b)], the shape of lock core is simplified and the pins are neglected to increase the simulation speed. The real setting of the task is shown in the first column of Fig. 19. The lock core is supported by two ball bearings. When the two pins are successfully lifted by the key, a manually designed rotating motion will be executed by the gripper to turn the lock, and the LEDs will be turned ON by a Hall sensor to indicate the result.

2) Details of RL Policy Training: Similar to the peg-in-hole and plug adjustment tasks, the tactile observation is formed by stacking the original marker positions and the displaced marker positions. The action is defined as a 2-dim vector $[a_x, a_z]$, whose axis is defined as shown in Fig. 18(a). The state input for the critic network is the position of the protrusions on the key $[x_{\text{protrusion\_1}}, z_{\text{protrusion\_1}}, x_{\text{protrusion\_2}}, z_{\text{protrusion\_2}}]$.

The terminal state is either that the task succeeds or that the number of steps is larger than the maximum. The task succeeds if both protrusions of the key are lifted up to a certain height, i.e., $x_{\text{protrusion\_1}} > h_{\text{threshold}}$ and $x_{\text{protrusion\_2}} > h_{\text{threshold}}$. The maximum number of attempts is set to 10.

The reward function shown in (8) consists of three terms: the constant penalty $P = 1$ at each step, the term related to key position R_{key}, and the final reward R_{final}. R_{key} is calculated from the x and z coordinates of the key's two protrusions, which encourages the RL agent to move the key to align the protrusions with the holes (the positions of the holes are $z_{\text{hole\_1}}$ and $z_{\text{hole\_2}}$) and lift the pins (the upward direction is along the $-x$ axis). In R_{key}, the constant C is to make R_{key} be zero when reaching the success state.

$$\begin{aligned} R_t &= -P + R_{\text{key}} + R_{\text{final}} \\ R_{\text{key}} &= |z_{\text{protrusion\_1}} - z_{\text{hole\_1}}| \\ &\quad + |z_{\text{protrusion\_2}} - z_{\text{hole\_2}}| \\ &\quad - x_{\text{protrusion\_1}} - x_{\text{protrusion\_2}} + C \\ R_{\text{final}} &= \begin{cases} 10, & \text{if success} \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

The domain randomization method is mainly introduced in Section III-F. For randomization in the task domain, the key's indentation depth into each sensor's surface varies from 0.5 to 1.25 mm and the key's relative position to the sensors varies from -5 to 5 mm in x direction and from -8 to 5 mm in z direction.

To efficiently train the RL policy, we use 20 parallel environments with 100 K environment steps in total. The proposed pretraining and fine-tuning of the tactile feature extractor are used.

3) Results of Sim2Real Experiments: To better analyze the trained policy, we present several consecutive frames of the lock opening task in Fig. 19. We divide the lock opening procedure into four stages:

1) *Start stage:* The gripper will grasp the key and make a random movement in x and z direction. This makes the relative position between the start state and the target pin holes unknown to the policy. Thus, the policy need to exploit the tactile feedback to reach the pin hole.
2) *Search stage:* The key maintains contact with the lock core and gradually moves toward the $+z$ direction, until the key's protrusions are inside the target pin holes and the contact disappears.
3) *Lift stage:* The gripper will lift the key to unlock the lock. However, errors in z direction will lead to contacts on the

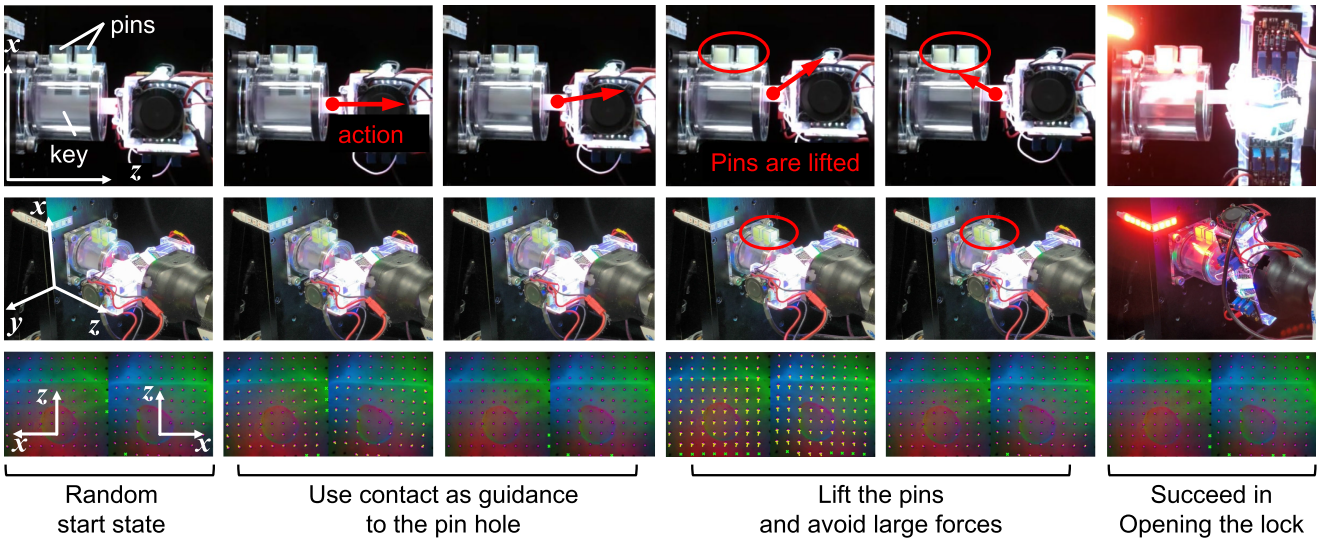


Fig. 19. Execution of lock opening task in reality. Real experimental setup and the definition of axial directions are shown in the first column. Tactile images from the sensors are shown in the bottom row. The actions are plotted in the top row as red arrows. The procedure can be divided into four stages: Start stage, Search stage, Lift stage, and Finish stage. In Start stage, the start position is randomized. In Search stage, the key maintains contact with the lock core until the contact disappears. In Lift stage, the key is lifted and large contact forces should be avoided. Finally, in Finish stage, a manually designed rotating motion is executed to turn the lock.

side of the protrusions, hindering the key from moving up. So the policy will avoid contacts in this stage.

4) *Finish stage:* When the key is lifted up to a certain height and tactile forces are small, we assume that the pins have been lifted up and the lock core can rotate freely. So, a sequence of manually designed rotating motion is executed to turn the lock.

We train the RL policy with three different random seeds and test the policies in reality. Each test has 20 episodes. The three tests have the success rates of 90%, 95%, and 90%, respectively. Therefore, we believe the Sim2Real performance in this task is considerably good and stable.

C. Generation of Depth and RGB Signals

In this work, we mainly focus on the simulation of marker flow for manipulation policy learning. In this section, we show that the simulation can be extended with modest effort to generate depth and RGB signals for GelSight-type visuotactile sensors. Since the deformed vertices can be acquired from the simulator, we can simulate the signals by 1) generating a depth map from the deformed surface, 2) rendering an RGB image from the depth map. Fig. 20 shows the sequence of simulated depth maps and RGB images.

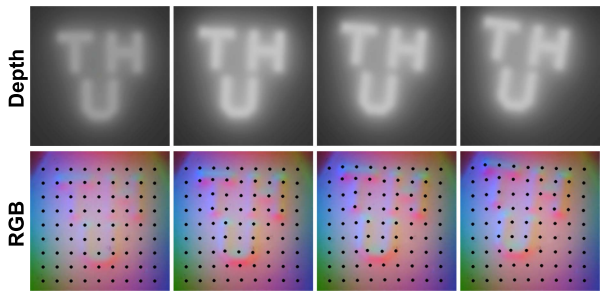


Fig. 20. Sequence of simulated depth maps and RGB images by our simulator. In the simulation, an object contacts with the tactile sensor and then moves along the sensor's surface. The RGB images are blended with the background image captured from a real sensor.

VII. Conclusion

In this article, we present a general-purpose Sim2Real protocol for learning contact-rich manipulation with marker-based visuotactile sensors. Our FEM-based simulation method generates high-fidelity tactile observations and exhibits sufficient robustness for long-term RL training. We propose the use of marker flow as the tactile representation and introduce a point cloud learning architecture for tactile feature extraction. A tailored pretraining and fine-tuning method is employed to achieve high sample efficiency in tactile-based robot learning. Additionally, we summarize a reasonable and diverse domain randomization technique that has been validated for use with our proposed simulation and learning pipeline. The combination of these techniques enables the successful completion of three contact-rich manipulation tasks in a zero-shot Sim2Real manner. The small Sim2Real gap observed in our experiments demonstrates the effectiveness of our method.

Despite the promising results, several limitations remain to be addressed. First, the simulation engine currently employed only supports deformable bodies, resulting in slow performance when complex rigid bodies are present. In addition, the process of adding constraints in our FEM-based simulation is less convenient than in other popular robot simulation environments. Second, while our pipeline can theoretically generalize to other marker-based visuotactile sensors, this has not been validated

due to the limited availability of such sensors. Third, the effectiveness of our proposed learning scheme in more complex tasks remains unknown.

These limitations also reveal some opportunities for further investigation. The simulation efficiency issue regarding rigid bodies could be addressed by incorporating affine-body dynamics [50]. Efforts will be made to integrate our tactile simulation into a general robot simulation environment. We also plan to apply our proposed pipeline to different sensors and more complex manipulation tasks with full robot arms. An intriguing direction for future work is the optimization of tactile sensor design based on the expected application scenarios through simulation. Rapid iterations of sensor design become feasible if a reliable simulation has been built.

REFERENCES

[1] Q. Li, O. Kroemer, Z. Su, F. F. Veiga, M. Kaboli, and H. J. Ritter, "A review of tactile information: Perception and action through touch," *IEEE Trans. Robot.*, vol. 36, no. 6, pp. 1619–1634, Dec. 2020.

[2] F. R. Hogan, J. Ballester, S. Dong, and A. Rodriguez, "Tactile dexterity: Manipulation primitives with tactile feedback," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 8863–8869.

[3] C. Wang, S. Wang, B. Romero, F. Veiga, and E. Adelson, "SwingBot: Learning physical features from in-hand tactile exploration for dynamic swing-up manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5633–5640.

[4] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez, "Tactile-RL for insertion: Generalization to objects of unknown geometry," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6437–6443.

[5] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872.

[6] I. Akkaya et al., "Solving Rubik's cube with a robot hand," 2019, *arXiv:1910.07113*.

[7] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "SIM-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3803–3810.

[8] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.

[9] X. Zhang et al., "Close the optical sensing domain gap by physics-grounded active stereo sensor simulation," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 2429–2447, Jun. 2023.

[10] D. F. Gomes, P. Paoletti, and S. Luo, "Generation of gelsight tactile images for Sim2Real learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 4177–4184, Apr. 2021.

[11] Y. Zhao, X. Jing, K. Qian, D. F. Gomes, and S. Luo, "Skill generalization of tubular object manipulation with tactile sensing and Sim2Real learning," *Robot. Auton. Syst.*, vol. 160, 2023, Art. no. 104321.

[12] 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 Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3930–3937, Apr. 2022.

[13] A. Church et al., "Tactile sim-to-real policy transfer via real-to-sim image translation," in *Proc. Conf. Robot Learn.*, 2022, pp. 1645–1654.

[14] J. Xu et al., "Efficient tactile simulation with differentiability for robotic manipulation," in *Proc. 6th Annu. Conf. Robot Learn.*, 2022, pp. 1488–1498. [Online]. Available: https://openreview.net/forum?id=6BIffCl6gsM

[15] C. Wang, L. Dong, D. Peng, and C. Pan, "Tactile sensors for advanced intelligent systems," *Adv. Intell. Syst.*, vol. 1, no. 8, 2019, Art. no. 1900090.

[16] M. Li, T. Li, and Y. Jiang, "Marker displacement method used in vision-based tactile sensors–from 2-D to 3-D: A review," *IEEE Sensors J.*, vol. 23, no. 8, pp. 8042–8059, Apr. 2023.

[17] Y. Liu et al., "Enhancing generalizable 6 d pose tracking of an in-hand object with tactile sensing," *IEEE Robot. Autom. Lett.*, vol. 9, no. 2, pp. 1106–1113, Feb. 2024.

[18] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, "Cable manipulation with a tactile-reactive gripper," *Int. J. Robot. Res.*, vol. 40, no. 12-14, pp. 1385–1401, 2021.

[19] K. Kamiyama, K. Vlack, T. Mizota, H. Kajimoto, K. Kawakami, and S. Tachi, "Vision-based sensor for real-time measuring of surface traction fields," *IEEE Comput. Graph. Appl.*, vol. 25, no. 1, pp. 68–75, Jan./Feb. 2005.

[20] B. Ward-Cherrier et al., "The TacTip family: Soft optical tactile sensors with 3D-printed biomimetic morphologies," *Soft Robot.*, vol. 5, no. 2, pp. 216–227, 2018.

[21] N. F. Lepora, Y. Lin, B. Money-Coomes, and J. Lloyd, "DigiTac: A DIGIT-TacTip hybrid tactile sensor for comparing low-cost high-resolution robot touch," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9382–9388, Oct. 2022.

[22] S. Cui, R. Wang, J. Hu, J. Wei, S. Wang, and Z. Lou, "In-hand object localization using a novel high-resolution visuotactile sensor," *IEEE Trans. Ind. Electron.*, vol. 69, no. 6, pp. 6015–6025, Jun. 2022.

[23] L. Zhang, Y. Wang, and Y. Jiang, "Tac3d: A novel vision-based tactile sensor for measuring forces distribution and estimating friction coefficient distribution," 2022, *arXiv:2202.06211*.

[24] C. Sferrazza and R. D'Andrea, "Design, motivation and evaluation of a full-resolution optical tactile sensor," *Sensors*, vol. 19, no. 4, 2019, Art. no. 928.

[25] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, 2017, Art. no. 2762.

[26] J. A. Fishel and G. E. Loeb, "Sensing tactile microvibrations with the biotac–comparison with human sensitivity," in *Proc. 4th IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatronics*, 2012, pp. 1122–1127.

[27] Y. Lin, J. Lloyd, A. Church, and N. F. Lepora, "Tactile gym 2.0: Sim-to-real deep reinforcement learning for comparing low-cost high-resolution robot touch," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10754–10761, Oct. 2022.

[28] A. Agarwal, T. Man, and W. Yuan, "Simulation of vision-based tactile sensors using physics based rendering," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 1–7.

[29] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5761–5768, Jul. 2021.

[30] Z. Si and W. Yuan, "Taxim: An example-based simulation model for gelsight tactile sensors," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2361–2368, Apr. 2022.

[31] Y. Narang, B. Sundaralingam, M. Macklin, A. Mousavian, and D. Fox, "Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 6444–6451.

[32] Y. S. Narang, B. Sundaralingam, K. Van Wyk, A. Mousavian, and D. Fox, "Interpreting and predicting tactile signals for the syntouch biotac," *Int. J. Robot. Res.*, vol. 40, no. 12-14, pp. 1467–1487, 2021.

[33] F. Faure et al., "SOFA: A multi-model framework for interactive physical simulation," *Soft Tissue Biomechanical Model. Comput. Assist. Surg.*, vol. 11, pp. 283–321, Jun. 2012. [Online]. Available: https://inria.hal.science/hal-00681539

[34] Q. K. Luu et al., "Simulation, learning, and application of vision-based tactile sensing at large scale," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 2003–2019, Jun. 2023.

[35] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *Int. J. Robot. Res.*, vol. 40, no. 4-5, pp. 698–721, 2021.

[36] W. Chen et al., "Bidirectional sim-to-real transfer for gelsight tactile sensors with cycleGAN," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 6187–6194, Jul. 2022.

[37] X. Jing, K. Qian, T. Jianu, and S. Luo, "Unsupervised adversarial domain adaptation for sim-to-real transfer of tactile images," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, 2023.

[38] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin, "Learning to see before learning to act: Visual pre-training for manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 7286–7293.

[39] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik, "Masked visual pre-training for motor control," 2022, *arXiv:2203.06173*.

[40] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3M: A universal visual representation for robot manipulation," in *Proc. Conf. Robot Learn.*, 2023, pp. 892–909.

[41] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, "Real-world robot learning with masked visual pre-training," in *Proc. Conf. Robot Learn.*, 2023, pp. 416–426.

[42] M. Li et al., "Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 49:1–49:20, Aug. 2020.

[43] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
[44] S. Dankwa and W. Zheng, "Twin-delayed DDPG: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," in *Proc. 3rd Int. Conf. Vis., Image Signal Process.*, 2019, pp. 1–5.
[45] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *Robot., Sci. Syst.*, 2018.
[46] F. Xiang et al., "Sapien: A simulated part-based interactive environment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11097–11107.
[47] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html
[48] J. Xu et al., "An end-to-end differentiable framework for contact-aware robot design," in *Proc. Robot., Sci. Syst., Virtual*, 2021. [Online]. Available: http://diffhand.csail.mit.edu
[49] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground truth force distribution for learning-based tactile sensing: A finite element approach," *IEEE Access*, vol. 7, pp. 173438–173449, 2019.
[50] L. Lan, D. M. Kaufman, M. Li, C. Jiang, and Y. Yang, "Affine body dynamics: Fast, stable, and intersection-free simulation of stiff materials," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 67:1–67:14, Jun. 2022.

Weihang Chen received the B.E. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2018. He is currently working toward the Ph.D. degree with the Department of Mechanical Engineering, Tsinghua University.

His research interests include vision-based tactile sensing and Sim2Real for robotics.

Jing Xu (Member, IEEE) received the Ph.D. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2008.

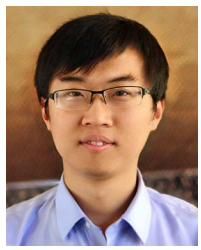
He was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing. He is currently an Associate Professor with the Department of Mechanical Engineering, Tsinghua university, Beijing, China. His research interests include vision-guided manufacturing, image processing, and intelligent robotics.

Fanbo Xiang received the B.S. degree in computer science and the B.S. degree in mathematics from University of Illinois, Urbana-Champaign, Champaign, IL, USA, in 2018. and the M.S. degree in computer science from the University of California San Diego, San Diego, CA, USA, in 2020, where he is currently working toward the Ph.D. degree in computer science.

His research focuses on physical simulation and embodied AI platforms.

Xiaodi Yuan received the B.E. degree in computer science from Tsinghua University, Beijing, China. She is currently working toward the Ph.D. degree with the University of California, San Diego, advised by Prof. Hao Su.

She spent 1.5 years as an Undergraduate Researcher in Prof. Su's Lab. Her recent research focuses on soft-body simulation and its application in robotics.



Hao Su (Member, IEEE) received the B.S. degree in computer science and the Ph.D. degree in mathematics from Beihang University, Beijing, China, in 2006 and 2014, respectively and the Ph.D. degree in computer science from Stanford University, California, USA, in 2018.

He is an Associate Professor of Computer Science and Engineering with the University of California San Diego, San Diego, CA, USA. He is affiliated with the Contextual Robotics Institute and Center for Visual Computing.

Dr. Su served on the program committee of multiple conferences and workshops on computer vision, computer graphics, and machine learning. He is the Area Chair of International Conference on Computer Vision (ICCV), 2019, Conference on Computer Vision and Pattern Recognition (CVPR), 2019, Senior Program Chair of AAAI7apos;19, IPC of Pacific Graphics'18, Program Chair of International Conference on 3D Vision (3DV), 2017, Publication Chair of 3DV'16, and the Chair of various workshops at CVPR, European Conference on Computer Vision, and ICCV. He is also invited as keynote speakers at workshops and tutorials in Neural Information Processing Systems, 3DV, CVPR, Robotics: Science and Systems, IEEE International Conference on Robotics and Automation, S3PM, etc.

Rui Chen (Member, IEEE) received the Ph.D. degree in mechatronical engineering and the B.E. degree in mechanical engineering from Tsinghua University, Beijing, China, in 2020 and 2014, respectively.

He is currently a research Assistant Professor with the Department of Mechanical Engineering, Tsinghua University. His research interests include three-dimensional computer vision and robot learning.