RTTF: Rapid Tactile Transfer Framework for Contact-Rich Manipulation Tasks

Qiwei Wu, Xuanbin Peng, Jiayu Zhou, Zhuoran Sun, Xiaogang Xiong and Yunjiang Lou

***Abstract*— An increasing number of robotic manipulation tasks now use optical tactile sensors to provide tactile feedback, making tactile servo control a crucial aspect of robotic operations. This paper presents a rapid tactile transfer framework (RTTF) that achieves optical-tactile image sim2real transfer and robust tactile servo control using limited paired data. The sim2real aspect of RTTF employs a semi-supervised approach, beginning with pretraining the latent space representations of tactile images and subsequently mapping different tactile image domains to a shared latent space within a simulated tactile image domain. This latent space, combined with the proprioceptive information of the robotic arm, is then integrated into a privileged learning framework for policy training, which results in a deployable tactile control policy. Our results demonstrate the robustness of the proposed framework in achieving task objectives across different tactile sensors with varying physical parameters. Furthermore, manipulators equipped with tactile sensors, allow for rapid training and deployment for diverse contact-rich tasks, including object pushing and surface-following.**

I. INTRODUCTION

In robot-object interaction, tactile feedback provides the robot with the ability to perceive the physical properties of objects, enabling more effective control and manipulation. Many robots now incorporate tactile feedback from optical tactile sensors and use learning-based approaches to achieve tactile servo control. Training tactile servo control policies in simulation environments can avoid excessive wear and tear when tactile sensors come into contact with objects, so many studies have developed tactile simulation environments [1]–[4]. However, achieving effective sim-to-real policy transfer for tactile servo control still poses several key challenges. Some research uses image translation methods to bridge the gap between simulated and real tactile images [5]–[7], but these techniques typically require a large amount of paired data for training and can be difficult to train effectively due to the high resolution of tactile images from optical sensors. End-to-end tactile servo control based on reinforcement learning is often considered a more effective method [8]–[10]. Therefore, how to obtain a simple and adaptive policy model becomes crucial. Furthermore, the wide variety of optical tactile sensors presents a challenge in developing tactile control policy models that can adapt to different sensor types without requiring additional training.

In this study, as shown in Fig. 1, we proposed a rapid tactile transfer framework (RTTF) that enables rapid transfer and deployment when replacing tactile sensors in the tactile control system, even when the sensors are entirely different except for similar geometric structures. Additionally, RTTF supports zero-shot expansion to other tasks.

Our contributions can be summarized as follows: 1) We have addressed the transfer problem of changing various tactile sensors and performing different tasks within a tactile control system. 2) We introduce RTTF, which exploits implicit spatiotemporal tactile information to enable control policy transfer and adaptation across different sensors, tasks, and physical environments. 3) Our framework significantly reduces the data dependency of tactile control policies during sim2real and improves robustness.

Our framework is notable for its ease of deployment and low implementation cost. For a homemade tactile sensor, transferring to the same task requires only a few minutes of data collection. Moreover, both transfer and policy models can be trained in about 1 hour on a consumer single GPU.

The manuscript is structured as follows. In Sec. II, the related work is first reviewed and then the proposed framework is described in Sec. III. This includes the Tactile Latent Space Mapping (TLSM) approach for sim-to-real in Sec. III-A and the privileged learning for policy training in Sec. III-B. Sec. IV presents the experiments, including the hardware employment, the simulation results, and the results of employing the learned policy on the real-world system. Finally, Sec. V concludes our work.

II. RELATED WORK

A. Tactile Sim2Real

The sim2real problem for optical tactile sensors [11]–[15] is often regarded as an image translation problem from the simulation domain to the real domain. This task commonly employs generative models like Generative Adversarial Network (GAN) [16] techniques and diffusion models [17] for image translation, which are also frequently applied to address domain adaptation challenges in tactile sensing [5]–[7]. Methods based on GANs or Diffusion Models typically involve training a generative model using a substantial amount of paired tactile images. Due to the instability of the generative model, the required volume of paired data typically reaches a scale of 1000× to achieve precise reconstruction of tactile images. However, this approach does not

*This work was supported partially by Shenzhen Fundamental Research Program under the following research Grants KQTD20190929172545139, JSGG20210420091602008 and partially supported by GuangDong Research Foundation 2022A1515011521, (Corresponding authors: Xiaogang Xiong and Yunjiang Lou)

Q. Wu, X. Peng, X. Xiong, J. Zhou, Z. Sun and Yunjiang Lou are with the School of Mechanical Engineering and Automation of Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China. (E-mail: xiongxg@hit.edu.cn; 21s053091@stu.hit.edu.cn; lourj@hit.edu.cn)

Manuscript 463 submitted to 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 28, 2024.

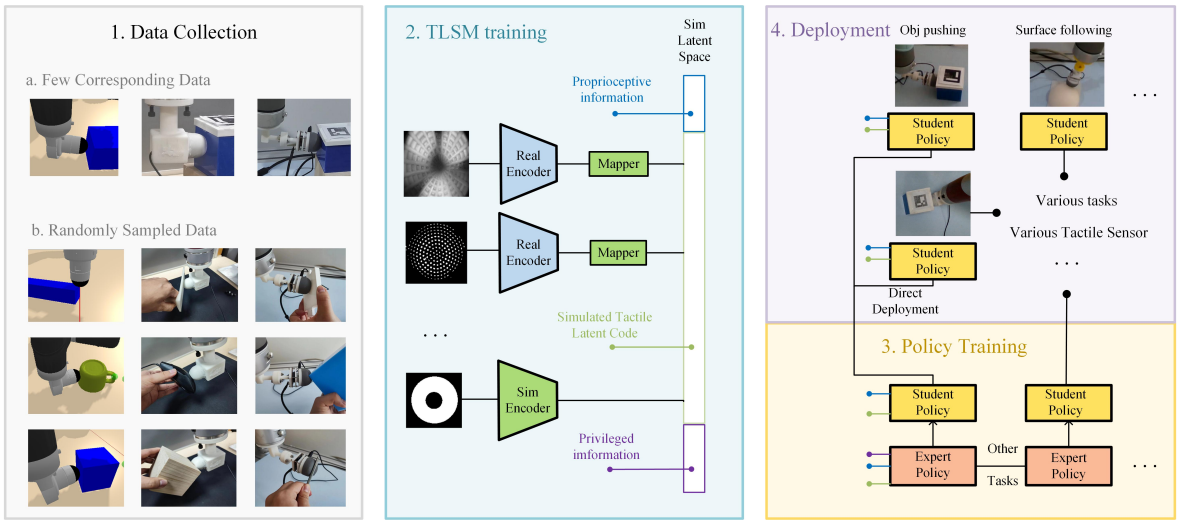


Fig. 1. RTTF is divided into four parts. First, collect paired and unpaired tactile data. Second, the tactile features of different domains are mapped to a unified simulation hidden space through Tactile Latent Space Mapping (TLSM) training. Then, the tactile feedback control policy is obtained using the unified hidden space features through reinforcement learning training. Finally, the policy model is directly deployed through mapping on the actual system.

take into account the data cost and information redundancy inherent in tactile images themselves.

Self-supervised representation learning can reduce data redundancy while capturing spatial features of tactile images. Some studies [18], [19] suggest that obtaining the implicit representation of tactile information first and then deriving policies from this implicit representation is a more data-efficient approach. However, policies acquired through tactile representation learning cannot be directly deployed when there is a change in sensor types, necessitating retraining. In addition, the low-dimensional tactile latent space obtained by the self-supervised method also leads to information loss.

The semi-supervised approach LSM (Latent Space Mapping) can map different latent space representations to the same latent space. Some studies [20], [21] have successfully applied LSM to image translation. We have designed TLSM (Tactile Latent Space Mapping) in RTTF that includes tactile data collection, model structure and model training. TLSM enables us to train a mapping model with minimal tactile data on top of the self-supervised pre-trained model, mapping all similarly structured self-made tactile sensor domains to a common simulated domain. As a result, we can directly apply policies from the simulated domain without additional training.

B. Tactile Control Policy

Tactile servo control relies on valuable feedback from optical tactile sensors. In certain approaches, a supervised learning process is used to train a feature extraction network [22]–[24]. Subsequently, the extracted information, such as contact depth and contact pose, is incorporated into either model-based or model-free servo control methods. For model-free servo control methods with greater generalization capabilities, reinforcement learning has gained attention due to its emphasis on interaction with the environment. The works in [8]–[10] have made progress in tasks such as typing, grasping, and pushing using reinforcement learning methods in tactile servo control. Indeed, the continuous nature of interactions with objects poses challenges for reinforcement learning, as it introduces Partially Observable Markov Decision Process (POMDP) problems [25]. This can make it challenging to train tactile feedback control policies using reinforcement learning methods.

To address these challenges, we drew inspiration from existing work [26]–[28] and integrated temporal and spatial latent features of tactile feedback through a teacher-student framework in RTTF. This integration allows our decision-maker to make decisions based on spatiotemporal features while retaining the ability for rapid policy transfer. Compared to other methods, RTTF enhances the policies' generalization and robustness, allowing the same pre-trained policy to be applied across tactile sensors with diverse physical attributes and various interacted objects.

III. The Proposed Approach

We use the semi-supervised learning approach to map images from real optical tactile sensors to the latent space of simulated tactile images. To achieve this, we initially employed VAE (Variational Autoencoder) to learn the latent space representations of various tactile sensor domains. Subsequently, we use the method of LSM to learn the mapping between these latent spaces. Additionally, by having the latent space representation of the simulated image domain, we combine the latent space representation with the privileged learning framework to train a tactile servo policy within the simulated environment. This ultimately leads to the creation of the tactile servo controller, as illustrated in Fig. 2. In this controller, we map the real tactile domain to the latent space of the simulated domain and apply the tactile servoing policy. RTTF only requires a cost-effective data collection process. Through the methods of latent space mapping and teacher-student learning, we obtain a struc-

Manuscript 463 submitted to 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 28, 2024.

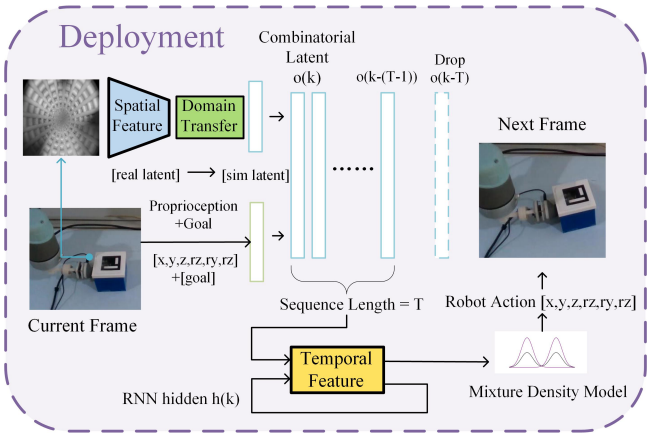


Fig. 2. The workflow of tactile feedback controller after training in RTTF. At each time step k, the controller receives the pose of the robotic arm's end-effector, and in some tasks, target coordinate information is also included. The controller also receives latent space representations of both simulation and real tactile features. These features are combined into a single feature vector, denoted as $o(k)$. The feature vector sequence of the previous T frames is also considered as input for the decision network to predict the mixture density model at time k. Finally, the mixture density model is sampled, and the output is mapped to specific tasks to determine the movement that should be taken. The total inference time only takes about 10ms in a single CPU.

turally concise and effective controller. Simultaneously, our framework achieves temporal-spatial signal extraction for tactile perception, decision-making, and adaptability. Next, we will introduce how all our networks are developed.

A. Tactile Latent Space Mapping

First, we outline the preprocessing of the tactile data and the TLSM method. We implement and optimize the networks using PyTorch.

1) ***Preprocessing****:* We collected unpaired data with a data size of 1000×, which is often employed in methods that require paired or labeled data. The unpaired data consisted of tactile images for all possible contact situations in all the tasks we chose, including different contact depths, contact angles, contact locations, and contact surface shapes. We then selected the paired data based on the specific task acquisition; for the push task, we used 10% of the unpaired data to pair with the simulation data, and for the surface-following task, we used 40% of the unpaired data to pair with the simulation data.

Our data pairing process is to synchronize the tactile sensors in the real and simulated environments to maintain roughly the same angle to the contact surface, make a shallow-to-depth contact, save the tactile images from this process, and then change to another angle. The whole process takes only 1 to 2 minutes.

The end of the tactile sensor is made of soft material with low light transmission, which is largely unaffected by light and occlusion, and the inner texture is relatively simple. Therefore, we believe that a greyscale image using a single channel of 128×128 pixels is sufficient to represent the tactile features. While binary images have more distinctive features, they also suffer from more information loss, and we show in the experimental section the impact of tactile binary features and tactile greyscale features on our study.

For unpaired data in real-world scenarios, we denote it as X_{real} and paired data as $\hat{X}_{real}$. In simulated scenarios, unpaired data is denoted as X_{sim}, and paired data as $\hat{X}_{sim}$.

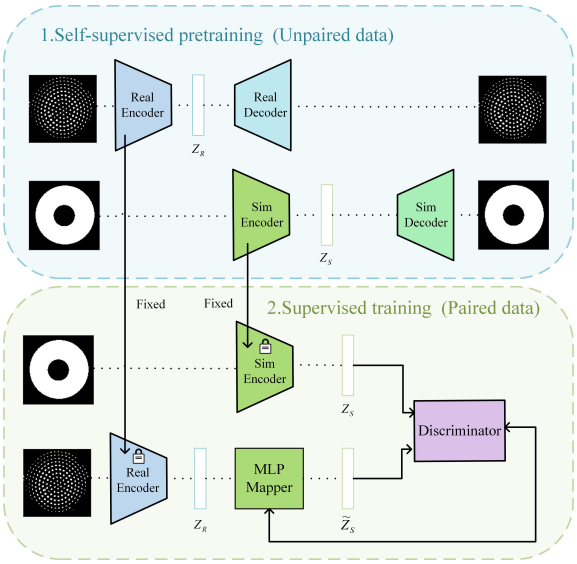


Fig. 3. The framework of TLSM. We begin by training VAE_{real} and VAE_{sim} through unsupervised learning using unlabeled tactile images. The Real Encoder's and Sim Decoder's parameters are fixed during this training phase. Next, a mapper model is obtained through supervised adversarial training using a small set of paired tactile images. Finally, in the real-world setting, we use a Real Encoder and Mapper to process the tactile images.

2) ***Tactile Latent Space Learning****:* Variational self-encoder contains two neural networks: An encoder and a decoder. They all consist of a four-layer CNN structure akin to the posenet [29] (the decoder employs deconvolutional layers). We use a pair of variational self-encoder networks VAE_{sim} and VAE_{real}, which contains encoders E_{sim}, E_{real} and decoders G_{sim}, G_{real}. Given a data sample $x_{sim} \in X_{sim}$ and its encoded representation z_{sim}, the loss function for this stage can be represented as follows:

$$\begin{aligned}\mathscr{L}_{\text{sim}}^{\text{VAE}} = &-E_{q(z_{\text{sim}}|x_{\text{sim}})}\left[\log p\left(x_{\text{sim}} \mid z_{\text{sim}}\right)\right] \\ &+\mathrm{KL}\left(\mathrm{q}\left(\mathrm{z}_{\text{sim}} \mid x_{\text{sim}}\right) \| p\left(z_{\text{sim}}\right)\right),\end{aligned} \tag{1}$$

where p represents the probability distribution of the data, q is an approximation of p, and KL represents the KL divergence. Similarly, as (1), we can get the loss $\mathscr{L}_{\text{real}}^{\text{VAE}}$ in the same way. After training the VAE_{sim} and VAE_{real}, we represent the data as latent codes z_{sim} and z_{real}.

3) ***Tactile Latent Space Mapping****:* We then construct a four-layer MLP (Multilayer Perceptron) network $F_{real2sim}$ to achieve TLSM from z_{real} to z_{sim}. To train this mapping network $F_{real2sim}$, we adopt the VAE-GAN framework and use paired data $\hat{X}_{sim}$ and $\hat{X}_{real}$. A GAN also contains two neural networks: a generator G and a discriminator D. The generator maps a latent code to an image, and the discriminator outputs a probability that the generated image is real.

Manuscript 463 submitted to 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 28, 2024.

We define our generator G as a network combination of the Encoder E_{real}, the mapper $F_{real2sim}$ and the decoder G_{sim}. The parameters of E_{real} and G_{sim} are fixed after latent space learning procedure. Given a data sample $\hat{x}_{real} \in \hat{X}_{real}$, we obtain the encoded result $\hat{z}_{real}$ by employing the Encoder E_{real}, i.e., $\hat{z}_{real} = E_{real}(\hat{x}_{real})$. Then the encoded result $\hat{z}_{real}$ is mapped to $\hat{z}_{fakesim}$ through the mapper $F_{real2sim}$ and decoding the corresponding result with decoder G_{sim}. The final output is a fake simulated tactile image, $\hat{x}_{fakesim} = G_{sim}(F_{real2sim}(E_{real}(\hat{x}_{real})))$ generated from $\hat{x}_{real}$. For the discriminator, we independently construct a separate network D, which is trained to distinguish between authentic and generated fake images from $\hat{x}_{sim}$ and $\hat{x}_{fakesim}$.

During the training process, the parameters of $F_{real2sim}$ and D are updated iteratively by optimizing distinct loss functions alternatively. The generator's loss function consists of two components: the adversarial loss L_G and the latent space mapping loss L_F, where L_F penalizes the deviation between the mapped latent codes $\hat{z}_{fakesim}$ and $\hat{z}_{sim}$ using the L_1 loss, while L_G aims to make the decoded images $\hat{x}_{fakesim}$ from the mapping results $\hat{z}_{fakesim}$ appear more natural. We represent the weight of the loss function L_F as a hyperparameter λ_f, and the loss functions are written as follows:

$$\mathscr{L}_{F_{real2sim}} = L_G + \lambda_f L_F \tag{2a}$$

$$L_G = \mathbb{E}[\log(1 - D(\hat{x}_{fakesim}))] \tag{2b}$$

$$L_F = \|F(\hat{z}_{real}) - \hat{z}_{sim}\|_1. \tag{2c}$$

The discriminator's role is to distinguish between the generated simulated tactile images and the original simulated tactile images, and the loss function is written as:

$$\mathscr{L}_D = \mathbb{E}[\log(D(\hat{x}_{sim}))] + \mathbb{E}[\log(1 - D(\hat{x}_{fakesim}))]. \tag{3}$$

After the TLSM training procedure, the tactile feedback images from the natural environment will be encoded by E_{real}. Then, the latent code will be mapped to the simulation domain by $F_{real2sim}$, which enables us to train the control policy directly in the simulation environment. Moreover, this implies that the feedback images from any structurally similar tactile sensors can be mapped to the same simulated domain through the TLSM training process without being limited to a specific simulation environment.

B. Learning Tactile Control Policies

Reinforcement learning (RL) involves agents interacting with the environment through actions to find optimal policies guided by reward signals. The problem aligns with a partially observable Markov decision process (POMDP). At time step k, the agent relies on the observation $o(k)$ to approximate the previous state $S(k-1)$ and current action $a(k)$, given by $S(k) \approx o(S(k-1), a(k))$. In the simulation, exact object attributes are accessible, allowing precise state determination $S(k) = o'(k)$ and obtaining an expert policy π^e through RL. In real-world environments, the privileged information is unavailable. After imitating the expert π^e, the student policy π^s employs temporal non-privileged information $o(k-(T-1)), o(k-(T-2)), ..., o(k)$(where T represents the size of time window) to estimate implicit dynamic models, thus predicting the actions $a(k)$ to be executed. In the following section, we give the details of the environment setup, expert, and student policies (refer to Tactile-Gym's documentation [1] for environment specifications).

*1) **Environment Setup**:* The environment setup consists of action space, observation space, and reward.

Action Space: The action space is defined as the six-dimensional linear and angular velocities of the robotic arm's end effector.

Observation Space: Let $o_t(k)$ represent the TCP coordinates of the robotic arm at time step k, $z_{sim}(k)$ denotes the latent space of tactile images, $o_g(k)$ stands for the target information, and $o_p(k)$ signifies the privileged information. The privileged information encompasses all known information in the simulation environment, including the positions, velocities, friction forces, and masses of objects in the environment. The fully observed measurement $o'(k)$ and non-privileged observation $o(k)$ are represented as follows:

$$o'(k) = \big(o_t(k), z_{sim}(k), o_g(k), o_p(k)\big) \tag{4a}$$

$$o(k) = \big(o_t(k), z_{sim}(k), o_g(k)\big). \tag{4b}$$

Reward: The Tactile-Gym environment provides different reward functions based on different tasks for environmental rewards. We did not modify any formulas or hyperparameters on top of this foundation.

*2) **Expert Policy**:* As shown in Fig. 4, to train the policy π^e, we employ one of the off-policy reinforcement learning methods, Truncated Quantile Critics (TQC) [30], which can control the overestimation bias in the critic's value estimation by using distributional critics. Given that the total number of time steps taken from the initiation to the completion of a task is T_{end}, the current time step is k, and the current expert action is $a^e(k)$, the optimized policy π^e is achieved through the following approach:

$$\pi^{e*} = \arg\max_{\pi^e} \mathbb{E}\left(\sum_{k=0}^{T_{end}} \gamma^k R_{\pi^e}(o'(k), a^e(k))\right). \tag{5}$$

Our work uses the stable-baselines3 [31] implementation of TQC, which led to the best and most reproducible results. During expert policy training, we randomly selected the physical parameters of the environment and added noise to the proprioceptive information.

*3) **Student Policy**:* : Building upon the pre-trained VAE network, we use a four-layer GRU (Gate Recurrent Unit) and a mixture density layer as the student policy network. The network models a probability density function (PDF) $p(a^s|x)$ as a mixture of m PDFs with the mixing coefficient $\prod = \{\alpha_0, \alpha_1, ..., \alpha_{m-1}\}$ by the following equation:

$$p(a^s|x) = \sum_{i=0}^{m-1} \alpha_i \phi_i(a^s(k)|x) \tag{6}$$

where $\phi_i(a^s|x)$ is a kernel function in the form of a multivariate Gaussian distribution with parameters $\{\mu_i, \sigma_i\}$, a^s represents the output action of student policy that should be taken, and x is the observation sequence $o(k-(T-1)), o(k-$

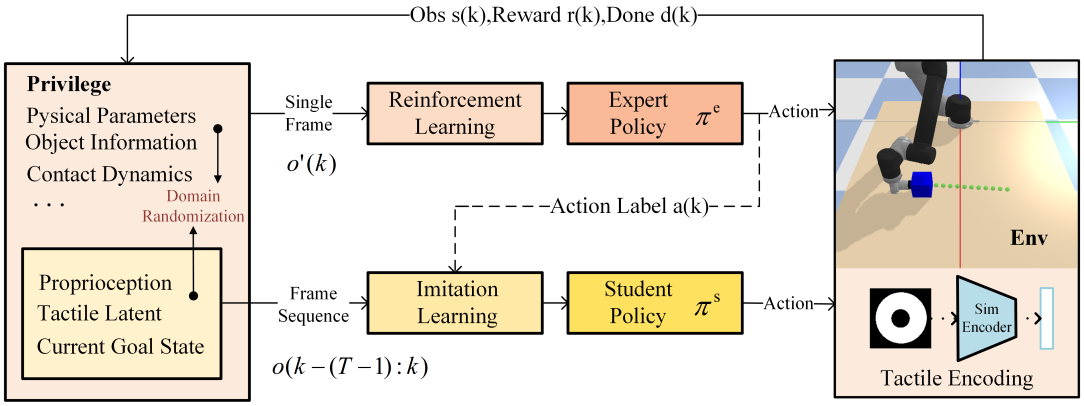


Fig. 4. The privileged learning approach. Firstly, we train an expert decision network through reinforcement learning using complete observational information. Then, we train a student policy based on the Proprioception information, which considers the historical information of observations during training to deal with the POMDP.

$(T-2)),...,o(k)$. The student action a_s will be generated by sampling from $p(a^s|x)$.

The student policy π^s is optimized using the NLL (Negative Log-Likelihood) loss function. The NLL loss function will maximize the likelihood of the sequence of observations x under the action label a_e given by the expert:

$$\mathscr{L}_{NLL} = -\log\left(\sum_{i=0}^{m-1} \alpha_i \phi_i(a^e(k)|x)\right). \tag{7}$$

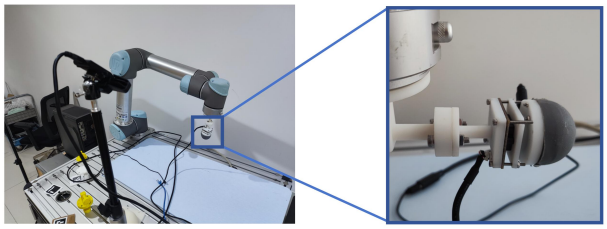
In this work, the DAGGER method [32] is employed, where the dataset is continuously aggregated with the incoming data from the training rollouts of the student policy. Labels are obtained by querying the expert policy for the visited states. At each training iteration, the student policy is updated by performing an optimization step with batches sampled from the aggregated dataset.

IV. EXPERIMENTS

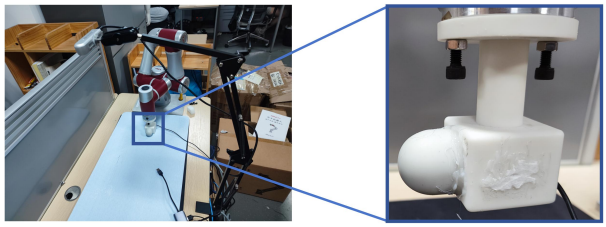
A. System and robot setup

1) Sensors: The set of hardware systems consists of four components: a custom-made tactile sensor, one 6DoF robotic arm, an external camera that is only used during testing phases, and a computer system for processing. To facilitate comparative experiments, we conducted trials using two distinct robotic arms, each equipped with one of the two different tactile sensors. Our two experimental setups are shown in Fig. 5.

We adapted optical tactile sensors, Tactip and Insight, to create affordable variants named W-Tactip (Fig. 5(b)) and R-Insight (Fig. 5(a)). The W-Tactip is a low-cost device which includes a two-layer soft gel tip, a micro-camera with LEDs, and 3D-printed connectors. The W-Tactip's internal micro-camera and LED capture images for binarized data emphasizing internal textures during contact. The R-Insight uses grayscale processing for precise texture analysis. Both sensors capture image data using cameras and LEDs. In the simulations, we use the Tactip simulations from Tactile-gym. The images from these tactile sensors are then mapped to a common simulation latent space.



(a) The robotic arm and R-Insight sensor system.



(b) The robotic arm and W-Tactip sensor system.

Fig. 5. Two sets of robotic arm-tactile sensor systems. These two systems are identical in all aspects except for the tactile sensors, and external cameras are not involved in the decision-making process of the tactile control policy.

2) Robotic Arm: In our experiments, we selected the 6-DOF Jaka and UR5 robotic arms. We adjusted the movement speeds of these two robotic arms to match the motion speeds in the simulation environment.

3) Computer System: Our computer system uses PyBullet and GPU rendering to provide fast simulation. The training process was conducted on a single Nvidia RTX 3060 Ti GPU in the simulation environment.

B. Simulations

Our experiments are divided into training and testing on two distinct tasks: the push task for goal-driven nonprehensile manipulation and the surface-following task for non-target-driven exploration. The push task involves pushing an object along a trajectory generated by OpenSimplex noise. Each trajectory has ten target points that are 3cm apart. The experiment includes flat and curved paths, with randomized initial object orientations ranging from $-15°$ to $15°$. In the

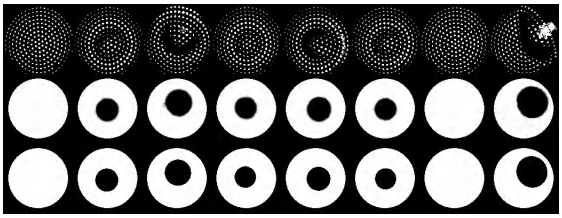
TABLE I
RESULTS OF SIM-TO-REAL.

phase	SSIM(Reconstruction)	SSIM(Real2Sim)	MAE(Reconstruction)	MAE(Real2Sim)
W-tactip(TLSM)	0.844	**0.934**	0.0246	**0.0101**
R-Insight(TLSM)	0.828	**0.922**	0.0048	**0.0168**
W-tactip(Unsupervised)	0.687	0.866	0.0449	0.0610
R-Insight(Unsupervised)	0.770	0.892	0.0064	0.0357

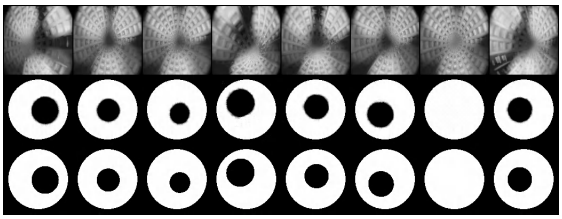
surface-following task, the tactile sensor maintains a vertical attitude while moving forward along a 3D contact surface generated using openSimplex noise.

Our approach involves privileged learning once for each task, training both expert and student policies. In real environment testing, we vary physical parameters like object mass, initial orientation, and shapes to assess policy robustness. For training, the student policy uses 1,000 sets of trajectory sequences with expert action labels. After the imitation learning process, we evaluate expert and student policies using reinforcement learning metrics. The results from extensive testing are summarized as follows. The median rewards of expert policy for push and surface-following are -54.55 and -0.92, while the median rewards of student policy for push and surface-following are -63.25 and -1.31.

C. Sim-to-real Results



(a) The transfer result of W-Tactip.



(b) The transfer result of R-Insight.

Fig. 6. The images provided by R-Insight and W-Tactip are translated into the same type of simulated tactile images. Real tactile images (top), ground truth (middle), and transfer results (bottom). The grayscaling tactile images from the R-Insight (b), and the binary tactile images from the W-Tactip (a).

In the experiment, we initially set a baseline with the data scale of [1], [33], comprising 5,000 unpaired images. Subsequently, we progressively reduced the data scale while ensuring the feasibility of subsequent experiments. Ultimately, we opted for 2,000 unpaired images for self-supervised pretraining. Additionally, 200 images paired with simulated data for the push task and 800 paired images for the surface task were used for TLSM training. For each type of tactile sensor, our entire data collection process takes less than 10 minutes.

We referenced the cycle-consistency based method [34] for unsupervised image translation and conducted comparative experiments while ensuring the parameters of the encoding models are kept the same. In addition, we also used two different tactile sensors with binary features and grey scale features for comparison experiments. The performance of representation learning and TLSM was evaluated using the SSIM (Structural Similarity) and MAE (Mean Absolute Error) between the reconstructed images and ground truth. Given that the learning rate (0.001), iteration count (1,000) for each model, batch size (128), and other hyperparameters are kept consistent, the domain transfer results of two different sensors are shown in Fig. 6 and Table I.

The results demonstrate that the introduction of supervised data significantly improves the quality of image generation, indicating that the model has learned more accurate features. Tactile grey-scale features can be more accurately mapped to the latent space of simulation, whereas tactile binary features have a greater loss of information. Additionally, we observed that unsupervised methods on binary images may lead to correspondences of opposite directions, resulting in significant biases.

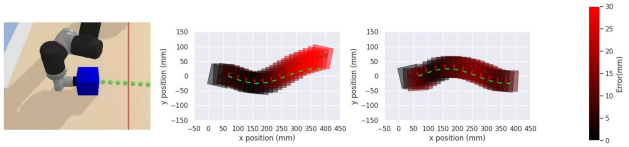
D. Real-world Manipulation Tasks

In our experiments, we investigated the robustness of our policy and compared it with other methods. We selected two methods for comparison. We designed the first method by referencing [18], [33] and call it TRL (Tactile Reinforcement Learning). The first method involves replacing the RNN structure of the student network with an MLP structure, directly trained through RL, as an ablation experiment that does not consider temporal features.

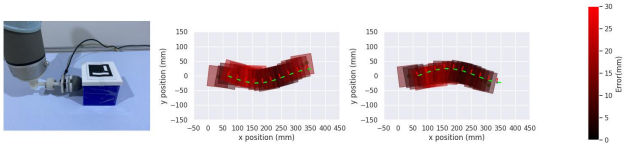
The second method involves training a posenet-based angle predictor using tactile images and contact angle data, with a network structure similar to the latent space encoder. The output includes the contact depth denoted as d, the contact angle in the horizontal plane denoted as θ_x and the contact angle in the vertical plane denoted as θ_y. Then, based on [24], we designed a PID controller to control the tactile system using the predicted contact angle and the relative angles and positions of the object and the target. We refer to this method as PN-PID (PoseNet-PID).

*1) **Push Tasks**:* To evaluate the performance of push tasks, we tracked the objects following the method given in Tactile-Gym. Importantly, the ArUco markers were only used for obtaining quantitative results and not as part of the observation. For the two experimental setups, we ensured the objects' consistency and the working surfaces upon

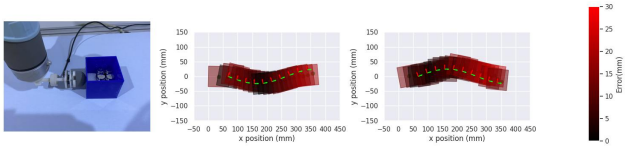
Manuscript 463 submitted to 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 28, 2024.



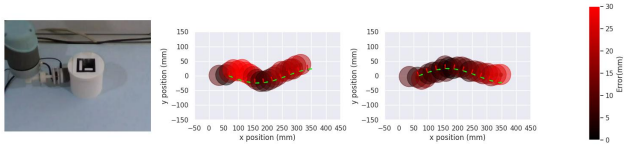
(a) Simulation



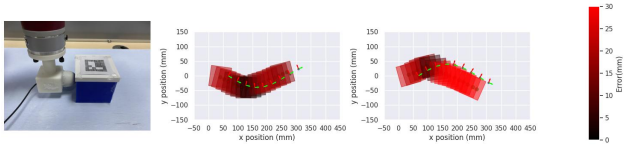
(b) R-Insight Origin



(c) Mass changed (+100g)



(d) Shape change



(e) Sensor change (W-Tactip)

Fig. 7. An example of the results for pushing student policy under random physical parameters.

which the objects were placed. The entire testing area is a rectangular region with the initial TCP coordinates $(0,0)$ as the origin. The x-axis coordinate ranges from 0 to 400mm, and the y-axis coordinate ranges from -100 to 100mm. In our experiments, we tested the performance of policies based on the same pre-trained transfer model and policy model while varying the physical conditions. We segmented the object's movement trajectory and the target trajectory into segments of equal length and calculated the positional error between the object and target positions at these segment points. The results of the robustness tests with different physical parameters are displayed in Fig. 7.

Furthermore, we conducted ablation experiments and comparative studies using both TRL and PN-PID methods. Testing was performed on approximately 50 randomly generated trajectories in both simulated and real environments with various sensors. We computed AE (Average error in mm) and GCR (Goal Completion Rates) of the local goals in the path, shown in Table II. AE greater than 30mm is indicated by "$-$". According to the results, both the PN-PID method and the TRL method face difficulties in adapting to self-made sensor systems when transitioning from simulation to reality. In contrast, our approach shows significant robustness and portability. However, the soft rubber tip of our W-tactip sensor lacks material toughness, which results in poor performance across all methods. Nonetheless, our method still maintains a high success rate.

TABLE II

AVERAGE ERROR IN POLICY TESTING UNDER RANDOM PATHS

Methods	Real(AE)	Real(GCR)	Sim(AE)
RTTF (R-Insight)	**11.67**	**1**	8.56
RTTF (W-Tactip)	23.68	0.80	
PN-PID (R-Insight)	23.38	0.92	8.93
PN-PID (W-Tactip)	–	0.46	
TRL (R-Insight)	–	0.62	8.22
TRL (W-Tactip)	–	0.42	

*2) **Surface-following Tasks**:* To evaluate the performance of surface-following tasks, a spherical object is traversed from its center outwards in a set direction over 360° in 45° intervals. We employed a 3D-printed shape with known ground truth from the CAD model and computed the error for the surface-following task, considering the prescribed standard tactile contact depth (1mm) and the vertical angle (90°) to the surface. Remarkably, the sensor successfully traversed the physical robot, even without texture and frictional forces that were not replicated in the simulation.

The experimental results (Fig. 8) show that in the surface-following task, the PN-PID method exhibits cumulative errors when transferred to a real physical environment. In contrast, our approach dynamically adjusts to the current contact state, achieving a more effective following performance.

V. CONCLUSION

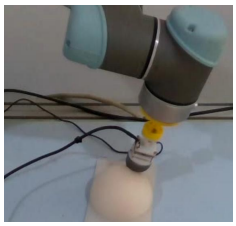
This study proposes a framework called RTTF that addresses both the problem of tactile sim2real and tactile feedback control. In the sim2real problem, the semi-supervised TLSM method reduces data dependency compared to supervised methods while also improving mapping accuracy compared to unsupervised methods. In the tactile feedback control section, RTTF combines spatiotemporal features of tactile signals, demonstrating robust adaptability across different tasks.

Real-world testing of our RTTF framework has shown that it enables rapid policy transfer for comparable tactile sensors with minimal paired data. It can quickly generate task-specific policies in simulated settings using pre-trained models with the same tactile sensor. Although our network architecture may not be optimal, the ability to adjust hyperparameters and replace networks enhances tactile servo control capabilities as required.

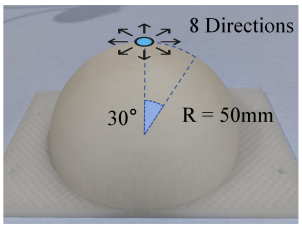
REFERENCES

[1] A. Church, J. Lloyd, R. Hadsell, and N. F. Lepora, "Optical tactile sim-to-real policy transfer via real-to-sim tactile image translation," *CoRR*, vol. abs/2106.08796, 2021. [Online]. Available: https://arxiv.org/abs/2106.08796

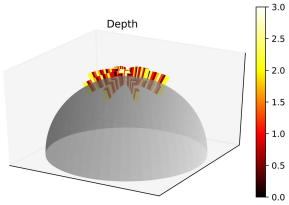
[2] S. Wang, M. Lambeta, P. Chou, and R. Calandra, "TACTO: A fast, flexible and open-source simulator for high-resolution vision-based tactile sensors," *CoRR*, vol. abs/2012.08456, 2020. [Online]. Available: https://arxiv.org/abs/2012.08456



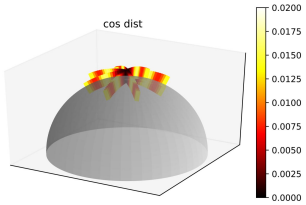
(a) Experiment example



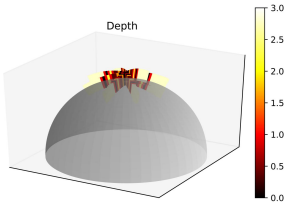
(b) Schematic diagram of the experimental process



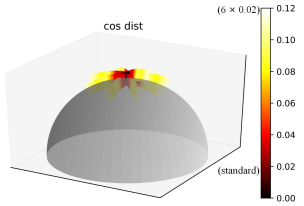
(c) Depth error (mm) of student policy (Ours)



(d) Orientation error (cosine dist) of student policy(Ours)



(e) Depth error (mm) of PN-PID policy



(f) Orientation error (cosine dist) of PN-PID policy

Fig. 8. The results of the student policy and PN-PID policy. The surface-following experiments were conducted on object surfaces without prior information.

[3] M. Bauzá, E. Valls, B. Lim, T. Sechopoulos, and A. Rodriguez, "Tactile object pose estimation from the first touch with geometric contact rendering," *CoRR*, vol. abs/2012.05205, 2020. [Online]. Available: https://arxiv.org/abs/2012.05205

[4] Z. Chen, S. Zhang, S. Luo, F. Sun, and B. Fang, "Tacchi: A pluggable and low computational cost elastomer deformation simulator for optical tactile sensors," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1239–1246, 2023.

[5] C. Higuera, B. Boots, and M. Mukadam, "Learning to read braille: Bridging the tactile reality gap with diffusion models," 2023.

[6] Q. Luu, N. Nguyen, and V. Ho, "Simulation learning and application of vision-based tactile sensing at large scale," *IEEE Transactions on Robotics*, vol. PP, 02 2023.

[7] T. Jianu, D. F. Gomes, and S. Luo, "Reducing tactile sim2real domain gaps via deep texture generation networks," *CoRR*, vol. abs/2112.01807, 2021. [Online]. Available: https://arxiv.org/abs/2112.01807

[8] J. Hansen, F. Hogan, D. Rivkin, D. Meger, M. Jenkin, and G. Dudek, "Visuotactile-rl: Learning multimodal manipulation policies with deep reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8298–8304.

[9] A. Church, J. Lloyd, R. Hadsell, and N. F. Lepora, "Deep reinforcement learning for tactile robotics: Learning to type on a braille keyboard," *CoRR*, vol. abs/2008.02646, 2020. [Online]. Available: https://arxiv.org/abs/2008.02646

[10] M. Yang, Y. Lin, A. Church, J. Lloyd, D. Zhang, D. A. W. Barton, and N. F. Lepora, "Sim-to-real model-based and model-free deep reinforcement learning for tactile pushing," 2023.

[11] B. Ward-Cherrier, N. Pestell, L. Cramphorn, B. Winstone, M. E. Giannaccini, J. Rossiter, and N. F. Lepora, "The tactip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies," *Soft Robotics*, vol. 5, no. 2, pp. 216–227, 2018, pMID: 29297773. [Online]. Available: https://doi.org/10.1089/soro.2017.0052

[12] 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 Robotics and Automation Letters*, vol. 7, no. 4, pp. 9382–9388, 2022.

[13] H. Sun, G. Martius, and K. J. Kuchenbecker, "Sensor arrangement for sensing forces and methods for fabricating a sensor arrangement and parts thereof," Patent PCT/EP2021/050 230, Jan., 2021.

[14] "Gelsight — products," https://www.gelsight.com/products/, 2023.

[15] I. H. Taylor, S. Dong, and A. Rodriguez, "Gelslim 3.0: High-resolution measurement of shape, force and slip in a compact tactile-sensing finger," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 781–10 787.

[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[17] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *CoRR*, vol. abs/2006.11239, 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[18] I. Guzey, B. Evans, S. Chintala, and L. Pinto, "Dexterity from touch: Self-supervised pre-training of tactile representations with robotic play," 2023.

[19] A. Murali, Y. Li, D. Gandhi, and A. Gupta, "Learning to grasp without seeing," *CoRR*, vol. abs/1805.04201, 2018. [Online]. Available: http://arxiv.org/abs/1805.04201

[20] T. Mayet, S. Bernard, C. Chatelain, and R. Herault, "Domain translation via latent space mapping," 2022.

[21] P. Zhang, J. Bao, T. Zhang, D. Chen, and F. Wen, "Semi-supervised image-to-image translation using latent space mapping," 2022.

[22] W. Fan, M. Yang, Y. Xing, N. F. Lepora, and D. Zhang, "Tac-vgnn: A voronoi graph neural network for pose-based tactile servoing," 2023.

[23] F. Ito and K. Takemura, "A model for estimating tactile sensation by machine learning based on vibration information obtained while touching an object," *Sensors*, vol. 21, no. 23, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/23/7772

[24] J. Lloyd and N. F. Lepora, "Goal-driven robotic pushing using tactile and proprioceptive feedback," *CoRR*, vol. abs/2012.01859, 2020. [Online]. Available: https://arxiv.org/abs/2012.01859

[25] K. Åström, "Optimal control of markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 1965. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022247X6590154X

[26] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation," *CoRR*, vol. abs/2101.02680, 2021. [Online]. Available: https://arxiv.org/abs/2101.02680

[27] J. Li, S. Dong, and E. H. Adelson, "Slip detection with combined tactile and visual information," *CoRR*, vol. abs/1802.10153, 2018. [Online]. Available: http://arxiv.org/abs/1802.10153

[28] V. R. Galaiya, M. Asfour, T. E. Alves de Oliveira, X. Jiang , and V. Prado da Fonseca, "Exploring tactile temporal features for object pose estimation during robotic manipulation," *Sensors*, vol. 23, no. 9, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/9/4535

[29] N. F. Lepora and J. Lloyd, "Optimal deep learning for robot touch," *CoRR*, vol. abs/2003.01916, 2020. [Online]. Available: https://arxiv.org/abs/2003.01916

[30] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. P. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," *CoRR*, vol. abs/2005.04269, 2020. [Online]. Available: https://arxiv.org/abs/2005.04269

[31] M. E. A. G. A. K. A. Raffin, A. Hill and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.

[32] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," *CoRR*, vol. abs/1011.0686, 2010. [Online]. Available: http://arxiv.org/abs/1011.0686

[33] 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," 2022.

[34] J. Xing, T. Nagata, K. Chen, X. Zou, E. Neftci, and J. L. Krichmar, "Domain adaptation in reinforcement learning via latent unified state representation," *CoRR*, vol. abs/2102.05714, 2021. [Online]. Available: https://arxiv.org/abs/2102.05714