DIFFTACTILE: A PHYSICS-BASED DIFFERENTIABLE TACTILE SIMULATOR FOR CONTACT-RICH ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce DIFFTACTILE, a physics-based and fully differentiable tactile simulation system designed to enhance robotic manipulation with dense and physically-accurate tactile feedback. In contrast to prior tactile simulators which primarily focus on manipulating rigid bodies and often rely on simplified approximations to model stress and deformations of materials in contact, DIFFTACTILE emphasizes physics-based contact modeling with high fidelity, supporting simulations of diverse contact modes and interactions with objects possessing a wide range of material properties. Our system incorporates several key components, including a Finite Element Method (FEM) -based soft body model for simulating the sensing elastomer, a multi-material simulator for modeling diverse object types (such as elastic, plastic, cables) under manipulation, a penalty-based contact model for handling contact dynamics. The differentiable nature of our system facilitates gradient-based optimization for both 1) refining physical properties in simulation using real-world data, hence narrowing the sim-to-real gap, and 2) efficient learning of tactile-assisted grasping and contact-rich manipulation skills. Additionally, we introduce a method to infer the optical response of our tactile sensor to contact using an efficient pixel-based neural module. We anticipate that DIFFTACTILE will serve as a useful platform for studying contact-rich manipulations, leveraging the benefits of dense tactile feedback and differentiable physics. The source codes of DIFFTACTILE will be publicly available. For extensive qualitative results, please refer to our project website[1].

1 INTRODUCTION

In the goal of enabling robots to perform human level manipulation on a diverse set of tasks, touch is one of the most prominent components. Tactile sensing, as a modality, is unique in the sense that it provides accurate, fine-detailed information about environmental interactions in the form of contact geometries and forces. Where, its efficacy has been highlighted by prior research, providing crucial feedback in grasping fragile objects (Ishikawa et al., 2022), enabling robots to perform in occluded environment (Yu & Rodriguez, 2018), and detecting incipient slip (Chen et al., 2018) for highly reactive grasping, there are still advances in tactile sensing to be made especially in the form of simulation.

Physics-based simulation has become a significant practical tool in the domain of robotics, by mitigating the challenges of real-world design and verification of learning algorithms. However, existing robotic simulators either lack simulation for tactile sensing or limit interactions to rigid bodies. To accurately simulate tactile sensors, however, which are inherently soft, it is essential to accurately model soft body interaction's contact geometries, forces, and dynamics. Prior works that attempted to simulate contact geometries and forces (Si & Yuan, 2022) for tactile sensors typically under (quasi-)static scenarios, where they successfully applied it to robotic perception tasks such as object shape estimation (Suresh et al., 2022), and grasp stability prediction (Si et al., 2022), but manipulation tasks that are highly dynamic have not been thoroughly explored. Other

[1] https://difftactile.github.io/

prior works approach contact dynamics by either approximating sensor surface deformation using rigid-body dynamics (Xu et al., 2023) or using physics-based soft-body simulation methods such as Finite Element Method (FEM) (Narang et al., 2021). However, these methods are still limited to manipulating rigid objects.

In this work, we aim to build a differentiable tactile simulator, DIFFTACTILE , that supports contact-rich robotic manipulation of rigid, deformable, and articulated objects. Differentiability is a key component of our work, as it provides fine-grained guidance for efficient and flexible skill learning (Huang et al., 2021; Xian et al., 2022), as well as providing a means to perform system identification to close the sim-to-real gap (Li et al., 2023). We implement DIFFTACTILE in Taichi (Hu et al., 2019) which enables massively parallel GPU computing and auto-differentiation. To demonstrate the capability and versatility of our simulator, we evaluate it on a diverse set of manipulation tasks including handling fragile, deformable, highly dynamic objects that cannot be addressed with prior tactile simulators. We summarize our contributions below:

Figure 1: Grasping a deformable object in the real world and in DIFFTACTILE.

- We introduce DIFFTACTILE , a platform supporting various tactile-assisted manipulation tasks. We model tactile sensors by FEM, objects with various materials (rigid, elastic, and plastic) by Moving Least Square Material Point Method (MLS-MPM), and cable by Position-Based Dynamics (PBD). We simulate the contact between sensors and objects with a penalty-based contact model. In addition, we accurately simulate the optical response of tactile sensors with high spatial variation via a learning-based method.
- Our system is differentiable and can reduce the sim-to-real gap with system identification. From a sequence sample of real data, we can optimize our simulation's sensor material and contact model parameters with differential physics and validate it with more general real-world scenarios.
- We demonstrate the improvement of skill learning efficiency with tactile feedback. We evaluate stable and adaptive grasps of objects with diverse geometry and material properties, and four contact-rich manipulation tasks.

2 RELATED WORK

Tactile simulation The most recent work on tactile simulation is built upon existing rigid-body simulators. For example, Tacto (Wang et al., 2022), Tactile-Gym (Church et al., 2022; Lin et al., 2022) were built upon PyBullet (Coumans & Bai, 2016). An efficient tactile simulation (Xu et al., 2023) was built upon DiffRedMax (Xu et al., 2021), where a penalty-based contact model was used to simulate the force distribution for tactile sensors. Even though it is computationally efficient to use rigid body simulation, these tactile simulators approximate contact dynamics for soft bodies at the cost of fidelity.

Alternatively, methods exist that use Finite Element Method (FEM) to accurately simulate soft body dynamics. A physics-based tactile simulator (Narang et al., 2021) was developed for SynTouch BioTac sensors (SynTouch) by using FEM in Isaac Gym (Makoviychuk et al., 2021). A grasp simulator also used the FEM in Isaac Gym (Kim et al., 2022) with incremental potential contact (IPC) model to handle contact dynamics. Taxim (Si & Yuan, 2022) used a superposition method to approximate the FEM. We also model tactile sensors with the FEM to maintain the simulator's physical accuracy and extend the contact model to handle multi-material objects beyond rigid objects.

Differentiable physics-based simulation. Differentiable physics-based simulation has become popular in recent years as it allows for efficient gradient-based policy learning compared to traditional sampling-based algorithms. PlasticineLab (Huang et al., 2021), FluidLab (Xian et al., 2022), SoftZoo (Wang et al., 2023) were presented with differentiability for soft body manipulation, fluid manipulation, and soft robot co-design, respectively, by leveraging Moving Least Square Material Point Method (MLS-MPM) (Hu et al., 2018). Tacchi (Chen et al., 2023) also used MLS-

Tactile simulator	Object model		Backend	Method	Optical Simulation	Differentiability
	Rigid	Soft				
Tacto (Wang et al., 2022)	✓		PyBullet	Rigid body	✓	
(Xu et al., 2023)	✓		DiffRedMax	Rigid body	✓	✓
Tacchi (Chen et al., 2023)	✓		Taichi	MPM	✓	
Taxim (Si & Yuan, 2022)	✓		PyBullet	FEM	✓	
(Narang et al., 2021)	✓		Isaac Gym	FEM		
IPC-GraspSim (Kim et al., 2022)	✓	✓	Isaac Gym	FEM		
Ours	✓	✓	Taichi	FEM	✓	✓

Table 1: Comparison with other state-of-the-art tactile simulators. We show DIFFTACTILE is the only tactile simulator supporting multi-material while being based on differential physics.

MPM to simulate the soft body deformation for GelSight (Yuan et al., 2017), a type of vision-based tactile sensor but did not present differentiability and contact dynamics modeling. It is shown that differential physics can be applied for system identification (Ma et al., 2023) to fine-tune the simulator's physical parameters and reduce the sim-to-real gaps. However, it remains unclear whether the gradient-based approach can benefit to improve the flexibility and efficiency of tactile-assisted manipulation skill learning.

Optical Simulation Taxim (Si & Yuan, 2022) showed that data-driven approaches to reconstructing the optical response of a vision-based tactile sensor to contact significantly outperforms model-based methods like (Wang et al., 2022; Chen et al., 2023; Agarwal et al., 2021; Gomes et al., 2021). However, there is a divergence in data-driven approaches. Methods like (Higuera et al., 2023; Chen et al., 2022; Zhong et al., 2023) use an image generation technique like generative models to perform style transfer from a simulated image to the style of a real deformation. However, these methods are rather data intensive since it needs a large variation of real world example deformations to generalize well. Methods like Taxim instead takes a pixel-based approach which uses a polynomial lookup table to map surface normals to RGB directly. This is a data-efficient approach, however this method makes assumptions about the sensors bidirectional reflectance distribution function (BRDF), which limits its applicability to sensors with low spatial variance.

We compare our work with state-of-the-art tactile simulations in Table 1. We show that our work, to the best of our knowledge, is the only work that is system-wise differentiable and can accurately model the soft body dynamics and contact dynamics, supports a broad categories of objects including rigid, elastic, plastic, and cables, and provide a data-efficient approach to simulate optical responses for vision-based tactile sensors.

3 TACTILE SIMULATION

3.1 SYSTEM OVERVIEW

DIFFTACTILE models the soft contact between tactile sensors and objects to provide dense tactile feedback including contact force distribution, contact surface deformation, and optical response. We present four key modules of our system: 1) a Finite Element Method (FEM)-based tactile sensor model in Section 3.2, 2) a learning-based method to simulate the optical response of tactile sensors with high spatial variation in Section 3.3, 3) rigid, elastic, and elasto-plastic object models using Moving Least Square Material Point Method (MLS-MPM), and cable model using Position-Based Dynamics (PBD) in Section 3.4, 4) a penalty-based contact model in Section 3.5.

3.2 TACTILE SENSOR SIMULATION

We model the deformation of the tactile sensor's soft elastomer under contact forces with FEM. We discretize the sensor soft elastomer to tetrahedron elements and then apply boundary conditions at the base of the sensor with position or velocity control. Since most tactile sensors' elastomers including ours are made from hyper-elastic materials, we apply the Neo-Hookean constitutive model in our simulation to capture the non-linearity of the material property. The energy density function Ψ and the first Piola-Kirchhoff stress tensor $\mathbf{P}$ used for governing equations are defined as:

$$\begin{aligned} \Psi(I_1, J) &= \frac{\mu}{2}(I_1 - 3) - \mu log(J) + \frac{\lambda}{2} log^2(J) \\ \mathbf{P}(\mathbf{F}) &= \mu(\mathbf{F} - \mathbf{F^{-T}}) + \lambda log(J)\mathbf{F}^{-T} \end{aligned} \tag{1}$$

where $\mathbf{F} \in \mathbf{R^{3\times3}}$ is the deformation gradient, $I_1 = tr(\mathbf{F}^T\mathbf{F})$ is the first isotropic invariants, and additional invariant $J = det(\mathbf{F})$. Note that our tactile simulation can be easily customized with different shapes, sizes, and materials by replacing the input mesh model or constitutive model.

To get tactile outputs including visual images and marker motions for vision-based tactile sensors, we first extract the deformed surface mesh from each simulation step's FEM solution. Then we interpolate the marker's locations by weighting surface node locations given a set of initial markers captured from a real sensor. We project markers in 3D to a 2D image plane given camera models.

3.3 OPTICAL SIMULATION

We reconstruct the optical response of a vision-based tactile sensor to contact using a data-driven approach. We model the surface of the sensor as a height function $z = f(x, y)$, and represent the continuous spatially-varying reflectance function of the surface as a 4D vector-valued function whose input is the 2D viewing direction (d = θ, φ) and 2D surface normals (x = $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial x}$), and whose output is the reflected color $c = (r, g, b)$. We approximate our reflectance function with a multilayer perceptron (MLP) f_θ whose input is augmented with a positional encoding $\gamma(d)$ and $\gamma(x)$ rather than directly d and x to enable the network to better fit data with high frequency variation (Mildenhall et al., 2021). Formally the encoding function used is:

$$\gamma(p) = \sin(2^0\pi p),\ \ \cos(2^0\pi p),\ \ ...,\ \ \sin(2^{L-1}\pi p),\ \ \cos(2^{L-1}\pi p) \tag{2}$$

Our rendering scheme finally consists of approximating the deformation caused by the contact indentation using pyramid Gaussian kernels as proposed in (Si & Yuan, 2022).

3.4 OBJECTS SIMULATION

We aim to support broader categories of objects beyond rigid objects for more diverse manipulation applications. We leverage Moving Least Square Material Point Method (MLS-MPM) (Hu et al., 2018) to simulate rigid, elastic, elasto-plastic objects. MLS-MPM has been shown to be efficient in simulating soft bodies. For elastic objects, we implement both corotated linear elasticity and Neo-Hookean elasticity models. For elasto-plastic objects, we use von Mises yield criterion to model plasticity upon elasticity. For rigid objects, we first treat objects as elastic using MLS-MPM, and then we add rigidity constraints by calculating object transformation and enforcing the shape of the object.

For another group of deformable objects such as cables and clothes, it is common to simulate them with Position Based Dynamics (PBD) (Müller et al., 2007). We also incorporate cable objects in our simulation by using PBD, where we constrain the stretch, bending, and self-collision.

3.5 PENALTY-BASED CONTACT MODEL

We handle contact dynamics between sensors and objects with a penalty-based contact model similar to (Xu et al., 2023). At each simulation step, we first check contact collision by pairing the surface triangle mesh from FEM with surface nodes from the object's particles (with either MPM or PBD). For each pair, we calculate the sign distance field d and normal directions $\mathbf{n}$ from the node to the triangle mesh. If d is negative, the node is penetrating the surface mesh and we need to apply normal penalty force to both mesh nodes and particle node to constrain the contact. In addition, we apply static or dynamic friction forces to the pair based on their relative velocities and normal forces. We represent our contact model as:

$$\begin{aligned} \mathbf{f_n} &= -(k_n + k_d \mathbf{v_n}) d\mathbf{n} \\ \mathbf{f_t} &= -\frac{\mathbf{v_t}}{||\mathbf{v_t}||} \min(k_t ||\mathbf{v_t}||, \mu ||\mathbf{f_n}||) \end{aligned} \tag{3}$$

where $\mathbf{f_n}$ and $\mathbf{f_t}$ are contact forces in the normal and tangential direction with respect to the local surface triangle. $\mathbf{v_n}$ and $\mathbf{v_t}$ are the relative velocities between the pair of the triangle and node in normal and tangential directions. k_n, k_d, k_t and μ are the parameters of contact stiffness, contact damping, friction stiffness, and friction coefficient. Then the contact force $\mathbf{f} = \mathbf{f_n} + \mathbf{f_t}$ is applied to both the triangle nodes and the particle node of the pair as an external force.

FEM-MPM coupling FEM is a mesh-based method and we can extract surface triangle meshes along with their associated node positions, velocities and face normal direction. MLS-MPM is a meshless hybrid Lagrangian-Eulerian method that uses Lagrangian particles and Eulerian grids to simulate continuous materials. Contact collision checking and contact force modeling are between FEM surface mesh nodes and MPM Eulerian grids for efficiency.

In each simulation step, we first pre-compute the internal elastic forces for all tetrahedral meshes from the constitutive law for the FEM sensor model, and advance particles to grids for the MPM object model. Then we check contact collision and calculate external contact forces for all pairs of triangle meshes and grid, and add them to the surface nodes. As post-contact computing, we transfer the velocities and affine coefficients from the grid to particles and do particle advection for MPM object model; and we advect the positions and velocities of the nodes based on the internal elastic forces, external contact forces, and gravity for FEM elements. We also consider the external boundaries such as tables and walls to constrain the positions of the objects.

FEM-PBD coupling Similarly to FEM-MPM coupling, we simply replace the MPM particles with PBD particles for contact collision detection and modeling. For PBD objects, there's no pre-contact computation, but we need to solve the stretch constrain, bending constrain, and self-collision constrain after the contact, and velocity advection based on the updated positions.

3.6 DIFFTACTILE TASKS

We present three sets of tasks with DIFFTACTILE : system identification, grasping, and manipulation. For system identification, we use real-world tactile observations to optimize the simulator's system parameters to reduce sim-to-real gaps. Then we present the grasping of a diverse set of objects with various shapes, sizes, and materials without slipping and damaging. We then demonstrate four contact-rich manipulation tasks including manipulating rigid, deformable, and articulated objects with high dynamics.

System Identification Sim-to-real transfer for robot learning has been a long-standing challenge where the gap in between heavily relies on simulation fidelity. To reduce the gap, we leverage differentiable physics to optimize the physical parameters of material and contact models given example data from the real world. Our optimization targets include Lamé parameters μ and λ of the FEM sensor model, and k_n, k_d, k_t, μ of the contact model. The optimization objectives

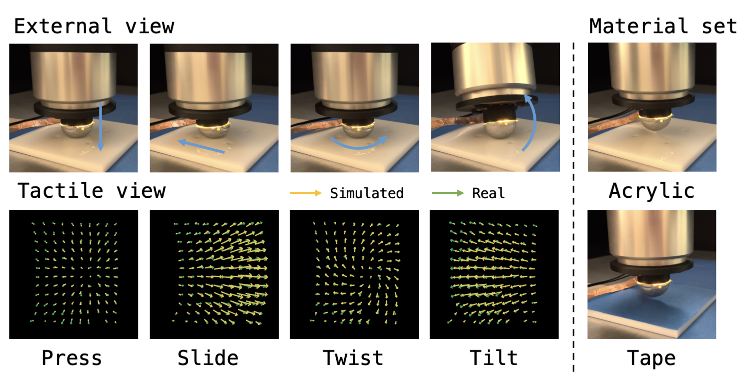

Figure 2: We use tactile readings and force readings from the *press* and *slide* trajectories as the objectives and optimize the FEM sensor model and contact model's physical parameters. We test the optimized parameters on additional *press*, *twist*, and *tilt* trajectories.

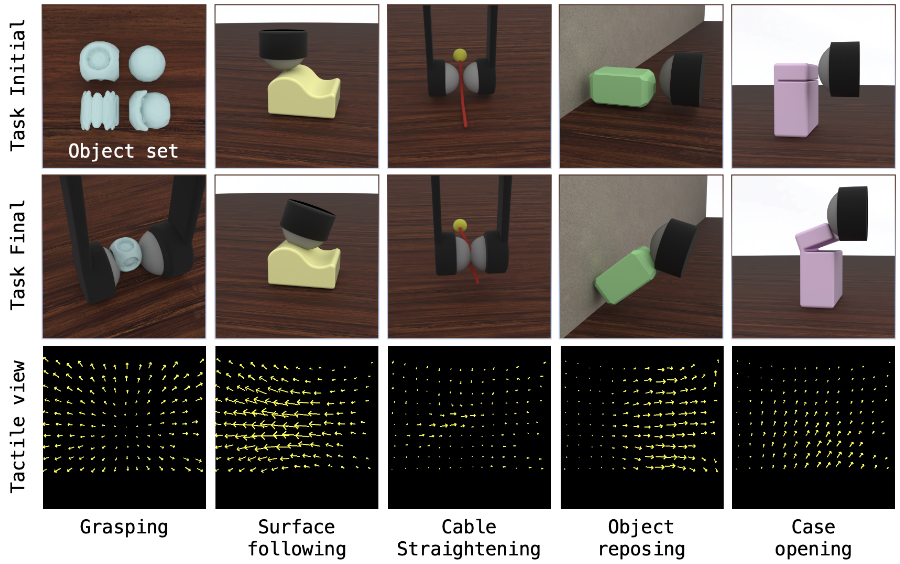

Figure 3: DIFFTACTILE tasks. **Grasping**: We grasp a set of four objects with different geometries and materials. **Surface following**: A sensor travels on the surface while maintaining the contact. **Cable straightening**: A pair of sensors straighten a cable by gripping and sliding from a fixed end. **Object reposing**: A sensor pushes an object to let it stand against a wall. **Case opening**: A sensor opens the cap of a case.

include the 6-axis force readings and tactile marker readings under four different contact scenarios: pressing, sliding, in-plane twisting, and tilt twisting as shown in Fig. 2.

Grasping Grasping is a fundamental skill for robot manipulation. Tactile sensing can enable safer and more adaptive grasping to handle fragile objects such as fruits. As shown in Fig. 3 (grasping), we select a set of objects with various shapes and materials to evaluate the grasping skill within our simulation. We aim to learn to grasp objects without damage or slippage.

Manipulation We present four contact-rich manipulation tasks: follow a surface, straighten a cable, open a case, and repose an object as shown in Fig. 3. Surface following requires the sensor to stay in contact with a 3D surface and travel to an endpoint while maintaining a certain contact force. Cable straightening requires a pair of sensors to first grasp a fixed end of the cable, and then straighten it by sliding towards the other end. Opening a case uses a single sensor to open an articulated object via pushing. Lastly, reposing an object involves using a single sensor to push an object from a lying pose to a standing pose against the wall. These four tasks represent rigid, deformable, and articulated object manipulation.

3.7 ENVIRONMENT SETUP

Initialization We initialize the environment with a single tactile sensor s for system identification, surface following, case opening, and object reposing, and two tactile sensors $\{s_1, s_2\}$ mounted on a parallel jaw gripper for grasping and cable straightening. Both tactile sensors' and objects' shapes are initialized with STL or OBJ mesh models and then voxelized to FEM tetrahedron meshes or MPM/PBD particles. Objects o_i are initialized statically on the tabletop and we add a vertical wall for object reposing. Tactile sensors are initialized statically near objects depending on tasks but without contact. We initialize the poses of tactile sensor at time step $t = 0$ as $T_s(0) = (R_s(0), t_s(0)) \in SE(3)$ where $R_s(0) \in SO(3)$ and $t_s(0) \in \mathbb{R}^3$ and similarly object pose as $T_o(0)$.

State Each tactile sensor s is represented as a FEM entity with N nodes and M tetrahedral elements. For each node n_i, it contacts a 6D state vector $s_i(t) = \{p_i(t), v_i(t)\}$ including a 3D position $p_i(t)$ and a 3D velocity $v_i(t)$. For each element m_i, it contacts a 4D index mapping from the element to its associated four nodes. Both MPM-based and PBD-based objects are represented with particles and each particle o_i also has a 6D state vector $o_i(t) = \{p_i(t), v_i(t)\}$ similarly.

		Press-slide↓	Press-twist-z↓	Press-twist-x↓
	Random	1.69 ± 1.10	1.15 ± 0.51	1.43 ± 0.62
Sim2Sim	**RNN**	1.20 ± 0.42	0.68 ± 0.28	0.90 ± 0.26
	Ours	$\mathbf{0.53 \pm 0.35}$	$\mathbf{0.42 \pm 0.24}$	$\mathbf{0.58 \pm 0.28}$
	Random	3.54 ± 1.73	2.59 ± 0.99	4.47 ± 3.31
Real2Sim	**RNN**	3.29 ± 1.51	2.42 ± 0.90	4.53 ± 3.51
	Ours	$\mathbf{3.08 \pm 1.27}$	$\mathbf{2.38 \pm 0.86}$	$\mathbf{3.99 \pm 2.89}$

Table 2: The pixel-wise tactile marker mean squared errors to evaluate system identification.

Observation We define two types of observations of each simulation step t, the state observation and the tactile observation. State observation includes tactile sensors' and objects' poses $T_s(t), T_o(t)$ and each node's or particle's state $s_i(t), o_i(t)$. For tactile observation, we can output the sensor's surface triangle mesh as a deformation map, the sensor's surface force distribution, or an aggregated three-axis force vector.

Action At each time step t, actions for end-effectors (either tactile sensors or gripper with kinematic chains down to tactile sensors) are queried from the controller as represented as a velocity vector $v_s(t) = \{\Delta R_s(t), \Delta t_s(t)\}$ to update the velocities of the FEM nodes.

Reward/Loss Each task's reward or loss function is formed differently based on the task objectives. We refer the readers to Section 4 for more details.

4 Experiments

4.1 System identification

Experimental Setup and Dataset We collect sequences of contact data from both the real world and simulation with synchronized control poses and velocities of the sensor. As shown in Fig. 2, there are four types of contact patterns collected, press, slide, twist (twist-z), and tilt (twist-x). For this experiment, the sensor interacts with two surfaces with different frictional properties, acrylic and tape.

Experimental results We evaluate two sets of experiments: **Sim2Sim** and **Real2Sim** where we use simulated data or real data respectively as inputs of the system. We optimize the sensor and contact model parameters with *press-slide* sequence, and test on all three sequences. We compare gradient-based trajectory optimization (**Ours**) with two baselines, **Random** and **RNN** as shown in Table 2. Here we use pixel-wise mean squared error (MSE) between predicted and collected tactile markers as evaluation metrics. For **Random**, we randomly select parameters within a practical range; for **RNN**, we input tactile marker readings and force readings and output the predicted system parameters. **Ours** outperforms **RNN** and **Random** on all sequences for both **Sim2Sim** and **Real2Sim**.

4.2 Optical Simulation

Experimental setup and dataset We manually collect 250 example deformations across the entire sensing surface using a 4mm spherical indenter. The pose of the sphere is manually annotated, and we split the dataset into a training set consisting of 200 examples, with the rest held out for testing.

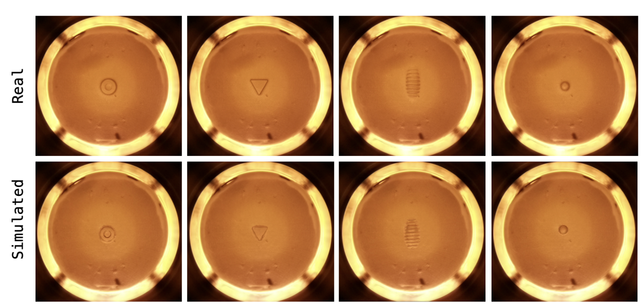

Figure 4: Tactile optical simulation compared with real data capturing various contact geometries.

Experimental results We test our method against a polynomial table mapping from Taxim (Si & Yuan, 2022). We use pixel-wise MSE, L1, SSIM, and PSNR as evaluation metrics. As shown in Table 3, our method outperforms Taxim across

	L1↓	MSE↓	SSIM↑	PSNR↑
Taxim	16.1	85.74	0.998	38.47
Ours	**7.94**	**56.10**	**0.999**	**39.42**

Table 3: Image similarity metrics for our test set. We compare our method to Taxim (Si & Yuan, 2022) on L1, MSE, SSIM and PSNR metrics. Our method performs best across all metrics.

	Loss	$\mathbf{L_{pos}}$↓	$\mathbf{D_{slip}}$↓	$\mathbf{L_{deform}}$↓
Elastic	**w/o tactile**	0.04 ± 0.03	0.18 ± 0.06	N/A
	w/ tactile	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.07 \pm 0.04}$	N/A
Elasto-plastic	**w/o tactile**	0.70 ± 0.53	0.26 ± 0.09	$\mathbf{0.85 \pm 0.25}$
	w/ tactile	$\mathbf{0.24 \pm 0.01}$	$\mathbf{0.2 \pm 0.05}$	0.89 ± 0.48

Table 4: Evaluation of grasping deformable, fragile objects with position, deformation losses and slipping distance by either using or not using tactile observations.

all metrics. Additionally, we verify the generalization and accuracy of our method by rendering a set of test probes with varying geometry, along with example real-world indentations for comparison in Fig. 4. We show our method can capture contact geometries in great detail.

4.3 Grasping

Experimental setup and dataset We evaluate our simulator on grasping fragile objects with various object properties including different shapes, sizes, weights, and material properties. As shown in Fig. 3, we select four objects from EGAD (Morrison et al., 2020) dataset with different shape complexity and assign each object with two different material properties, elastic, and elasto-plastic.

We aim to grasp objects stably and adaptively to avoid slipping and damaging the object with gradient-based trajectory optimization. Here we use two tactile sensors as fingertips and mount them on a parallel jaw gripper. In each trajectory, the gripper first grips the object, then lifts it. Based on our goal, we define the objectives with three types of losses 1) **Position loss** L_{pos}: we set a 3D target position to reach after lifting; 2) **Deformation loss** L_{deform}: we aim to keep the shape of the object during the grasp by using the sign distance field of the object and the L1 distance of the mass distribution between the current object and the target one to penalize the deformation (Huang et al., 2021) 3) **Slipping loss** L_{slip}: we use the shear force detected between the fingertip and the object to penalize the slippage during grasping.

Experimental results We evaluate the grasping with or without tactile feedback on three metrics. We use L_{pos} for both types of objects, and we use L_{deform} for elasto-plastic objects only. In addition, we measure the slipping distance of the object relative to the sensor for both sets of objects, the slipping distance is denoted as D_{slip}. We show in Table 4 that the tactile feedback greatly improves the grasping quality.

4.4 Manipulation

Experimental setup For all four manipulation tasks, we define two different rewards, state reward and tactile reward for manipulation skill learning. We evaluate our system's learning efficiency by comparing gradient-based trajectory optimization with CMA-ES (Hansen et al., 2003) (sampling-based trajectory optimization), SAC (Haarnoja et al., 2018), and PPO Schulman et al. (2017) (model-free RL algorithm).

Surface following We set up a sensor to travel and follow a curved 3D surface. We define the state reward as traveling to a certain position on the 3D surface, and the tactile reward as keeping contact with the surface while maintaining a constant shear motion.

Cable straightening We set up a parallel jaw gripper with two tactile fingers and a cable with one end fixed to the wall while the other end free. The state reward is defined as the distance between the target position (the cable is horizontally straight) and the current position for each node on the cable. The tactile reward is defined as the force applied to the cable to maintain the gripping while being able to slide along the cable.

	Obs	Rew	Manipulation tasks			
	w/ tac	**w/ tac**	**ObjectRepose ↑**	**CableStraighthen ↓**	**CaseOpen ↑**	**SurfaceFollow ↑**
PPO	×	×	4.57 ± 0.06	2.06 ± 0.00	-0.95 ± 0.04	1.33 ± 1.53
	√	×	4.49 ± 0.08	2.07 ± 0.02	-0.93 ± 0.26	1.67 ± 0.58
	×	√	4.64 ± 0.15	2.03 ± 0.04	-0.83 ± 0.06	2.67 ± 1.15
	√	√	4.30 ± 0.15	1.90 ± 0.18	-0.80 ± 0.24	1.33 ± 1.53
SAC	×	×	5.00 ± 0.01	1.50 ± 0.02	-0.68 ± 0.29	11.00 ± 0.00
	√	×	4.90 ± 0.01	2.03 ± 0.02	-0.89 ± 0.09	10.00 ± 5.57
	×	√	4.89 ± 0.11	1.60 ± 0.12	-0.84 ± 0.04	14.00 ± 2.00
	√	√	4.68 ± 0.11	1.36 ± 0.03	-0.95 ± 0.07	1.33 ± 1.53
CMA-ES	N/A	×	4.65 ± 0.14	1.97 ± 0.14	-1.07 ± 0.05	2.33 ± 1.15
	N/A	√	4.50 ± 0.05	1.97 ± 0.15	-0.98 ± 0.07	1.67 ± 1.15
Ours	N/A	×	12.07 ± 12.46	1.27 ± 0.81	$\mathbf{17.11 \pm 0.05}$	4.00 ± 1.00
	N/A	√	$\mathbf{60.82 \pm 0.00}$	$\mathbf{0.89 \pm 0.32}$	9.83 ± 0.38	$\mathbf{51.67 \pm 12.86}$

Table 5: Evaluation of manipulation tasks by comparing gradient-based optimization (**Ours**) with sampling-based optimization (**CMA-ES**), and reinforcement learning approaches (**SAC**, **PPO**).

Case opening Here we initialize a closed case and we use a tactile sensor to push and open the lid of the case. We define the state reward as the angle of the opened lid and the tactile reward as the push forces to open the lid.

Object reposing A block is placed flat on the table and we aim to use one tactile sensor to flip it 90 degrees to stand against a wall. Similarly to the case opening, we define the state reward as the angle between the object and the floor, and the tactile reward as the push forces to flip the object.

Experimental Results To evaluate the performance of trained policies for different tasks, we design task-specific evaluation metrics. For the surface following, we define the metric as the traveling distance of the sensor in contact with the surface. For cable straightening, the metric is the aggregation distance between the current and target cable nodes' locations. For case opening and object reposing, we define the metric as the opened angle of the lid and the orientation of the object.

We show all experimental results in Table 5 by comparing our proposed gradient-based optimization method with baselines. We show **Ours** outperforms baselines with a large margin to show its learning efficiency. And **w/ tactile** has better performances compared to **w/o tactile** for most trails indicating tactile sensing helps on these contact-rich manipulation tasks.

5 CONCLUSIONS AND FUTURE WORK

We present DIFFTACTILE, a physics-based differentiable tactile simulator to advance skill learning for contact-rich robotic manipulation. By providing models for tactile sensors, multi-material objects, and penalty-based contacts, we greatly extend the capabilities and applicability of robotic simulators. The differentiability of our system aids in reducing the sim-to-real gaps by using system identification and improves the multi-skill learning efficiency by providing gradient-based optimization. We evaluate DIFFTACTILE 's versatility with the grasp of a set of various objects, and manipulation tasks including surface following, cable straightening, case opening, and object reposing. By comparing with the state-of-the-art reinforcement learning and sample-based trajectory optimization approaches, we demonstrate that DIFFTACTILE can enable efficient and flexible skill learning with tactile sensing and can potentially serve as a learning platform for broader tactile-assisted manipulation tasks.

In future work, we would like to deploy the skills learned in our simulator in real-world settings by exploring sim-to-real transfer. In addition, we plan to integrate our tactile simulator into commonly used robotic simulation frameworks to extend its usage on more general manipulation configurations such as adding tactile sensors on dexterous robotic hands for in-hand manipulation. We would also like to investigate robot learning with multi-modalities in simulation such as leveraging vision and touch feedback to improve the robustness of the policies.

References

Arpit Agarwal, Timothy Man, and Wenzhen Yuan. Simulation of vision-based tactile sensors using physics based rendering. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7. IEEE, 2021.

Wei Chen, Heba Khamis, Ingvars Birznieks, Nathan F. Lepora, and Stephen J. Redmond. Tactile sensors for friction estimation and incipient slip detection—toward dexterous robotic manipulation: A review. *IEEE Sensors Journal*, 18(22):9049–9064, 2018. doi: 10.1109/JSEN.2018.2868340.

Weihang Chen, Yuan Xu, Zhenyang Chen, Peiyu Zeng, Renjun Dang, Rui Chen, and Jing Xu. Bidirectional sim-to-real transfer for gelsight tactile sensors with cyclegan. *IEEE Robotics and Automation Letters*, 7(3):6187–6194, 2022.

Zixi Chen, Shixin Zhang, Shan Luo, Fuchun Sun, and Bin Fang. Tacchi: A pluggable and low computational cost elastomer deformation simulator for optical tactile sensors. *IEEE Robotics and Automation Letters*, 8(3):1239–1246, 2023.

Alex Church, John Lloyd, Raia Hadsell, and Nathan F. Lepora. Tactile sim-to-real policy transfer via real-to-sim image translation. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/church22a.html.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

Daniel Fernandes Gomes, Paolo Paoletti, and Shan Luo. Generation of gelsight tactile images for sim2real learning. *IEEE Robotics and Automation Letters*, 6(2):4177–4184, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

Carolina Higuera, Byron Boots, and Mustafa Mukadam. Learning to read braille: Bridging the tactile reality gap with diffusion models. *arXiv preprint arXiv:2304.01182*, 2023.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Fredo Durand. Difftaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations*, 2019.

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

Reina Ishikawa, Masashi Hamaya, Felix Von Drigalski, Kazutoshi Tanaka, and Atsushi Hashimoto. Learning by breaking: food fracture anticipation for robotic food manipulation. *IEEE Access*, 10: 99321–99329, 2022.

Chung Min Kim, Michael Danielczuk, Isabella Huang, and Ken Goldberg. Ipc-graspsim: Reducing the sim2real gap for parallel-jaw grasping with the incremental potential contact model. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 6180–6187. IEEE, 2022.

Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023.

Yijiong Lin, John Lloyd, Alex Church, and Nathan F. Lepora. Tactile gym 2.0: Sim-to-real deep reinforcement learning for comparing low-cost high-resolution robot touch. volume 7 of *Proceedings of Machine Learning Research*, pp. 10754–10761. IEEE, August 2022. doi: 10.1109/LRA.2022.3195195.

Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. *arXiv preprint arXiv:2304.14369*, 2023.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Douglas Morrison, Peter Corke, and Jürgen Leitner. Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation. *IEEE Robotics and Automation Letters*, 5 (3):4368–4375, 2020.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

Yashraj Narang, Balakumar Sundaralingam, Miles Macklin, Arsalan Mousavian, and Dieter Fox. Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6444–6451. IEEE, 2021.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):2361–2368, 2022.

Zilin Si, Zirui Zhu, Arpit Agarwal, Stuart Anderson, and Wenzhen Yuan. Grasp stability prediction with sim-to-real transfer from tactile sensing. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7809–7816, 2022. doi: 10.1109/IROS47612.2022. 9981863.

Sudharshan Suresh, Zilin Si, Joshua G Mangelson, Wenzhen Yuan, and Michael Kaess. Shapemap 3-d: Efficient shape mapping through dense touch and vision. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7073–7080. IEEE, 2022.

SynTouch. https://syntouchinc.com/.

Shaoxiong Wang, Mike Lambeta, Po-Wei Chou, and Roberto Calandra. Tacto: A fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):3930–3937, 2022.

Tsun-Hsuan Wang, Pingchuan Ma, Andrew Everett Spielberg, Zhou Xian, Hao Zhang, Joshua B Tenenbaum, Daniela Rus, and Chuang Gan. Softzoo: A soft robot co-design benchmark for locomotion in diverse environments. *arXiv preprint arXiv:2303.09555*, 2023.

Zhou Xian, Bo Zhu, Zhenjia Xu, Hsiao-Yu Tung, Antonio Torralba, Katerina Fragkiadaki, and Chuang Gan. Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. In *The Eleventh International Conference on Learning Representations*, 2022.

Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An End-to-End Differentiable Framework for Contact-Aware Robot Design. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.008.

Jie Xu, Sangwoon Kim, Tao Chen, Alberto Rodriguez Garcia, Pulkit Agrawal, Wojciech Matusik, and Shinjiro Sueda. Efficient tactile simulation with differentiability for robotic manipulation. In *Conference on Robot Learning*, pp. 1488–1498. PMLR, 2023.

Kuan–Ting Yu and Alberto Rodriguez. Realtime state estimation with tactile and visual sensing. application to planar manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7778–7785, 2018. doi: 10.1109/ICRA.2018.8463183.

Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

Shaohong Zhong, Alessandro Albini, Oiwi Parker Jones, Perla Maiolino, and Ingmar Posner. Touching a nerf: Leveraging neural radiance fields for tactile sensory data generation. In *Conference on Robot Learning*, pp. 1618–1628. PMLR, 2023.

A Appendix

A.1 Simulation Details

We implement our whole system with Taichi (Chen et al., 2023) along with Python to take advantage of its high computing performance and auto-differentiability. With Taichi, our system can switch between running with CPU or being accelerated by GPU by simply passing an argument to initialize the Taichi environment. Taichi also supports automatic differential features for functions with explicit time integration. Therefore, considering the implementation difficulty and generalizability, our system is implemented with semi-explicit time integration, and without any extra effort, is fully differentiable and can be used for gradient-based trajectory optimization. The simulation pipeline for each simulation step can be seen in Fig. 5.

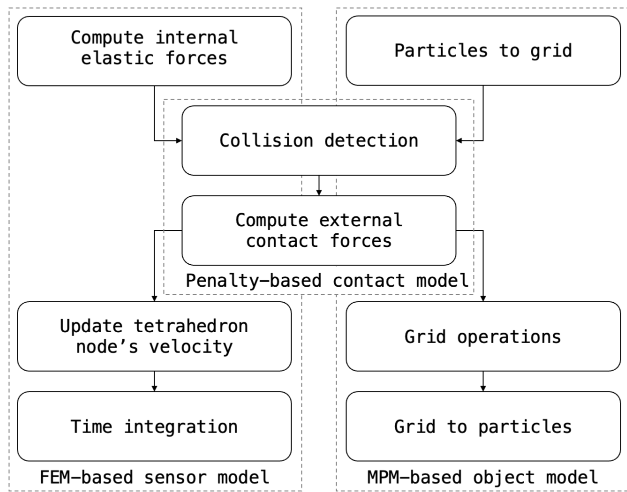

Figure 5: Simulation pipeline for each simulation step. Both the FEM sensor and MPM object have their pre-contact updates, and then we use a two-way coupling to handle collision and calculate contact forces. The contact forces are used for both the FEM sensor and MPM object post-contact.

A.2 Real-world Experiment Setup Details

We use a Gelsight tactile sensor in this work. The sensor is handmade in the laboratory with design flexibility. The soft elastomer is made with SYLGARD 184 silicone elastomer, and in a dome shape with an inner radius of 7.5 mm and an outer radius of 15.0 mm. The sensor uses an Arducam 180-degree fisheye camera to output tactile images.

A.3 System Identification Details

Real-world data collection We collect sequences of contact data from the real world including the 6-axis force readings from a robot arm end-effector, the poses of a Gelsight sensor, and the corresponding tactile images from the tactile sensor. We set up the experiment by mounting a GelSight sensor at the end-effector of an Ur5e robot arm and then controlling the robot arm to get the sensor in contact with a tabletop surface. As discussed in (Yuan et al., 2017), there are four general contact marker patterns under forces/torques that are essential to capture and simulate for tactile sensors including normal force, shear force, in-plane torque, and tilt torque. Therefore we collect three sequences of contact data: *press-slide*, *press-twist-z*, and *press-twist-x*. For each sequence, we start by pressing the sensor normally to a flat surface with a constant velocity 1 mm/s for 10 s to get in contact. Then we slide the sensor along the surface, twist it along the normal direction, or twist it along a horizontal direction to finish *press-slide*, *press-twist-z*, and *press-twist-x* respectively with a constant velocity 1 mm/s or 2 degrees/s for 10 s.

Gradient-based estimation We define the loss including *tactile loss*, the pixel-wised tactile marker distances, and *force loss*, three-axis force errors between the simulated and ground truth data. Since the two losses are on different numerical scales, we use the aggregation with weights

10:1 as the final loss. We use Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$. Learning rates for parameters are $lr_{kn} = 20.0$, $lr_{kd} = 20.0$, $lr_{kt} = 5.0$, $lr_{fc} = 5.0$, $lr_{\mu} = 50.0$, $lr_{\lambda} = 50.0$ depending on their numerical scales. We run 100 optimization steps for each trajectory.

RNN-based estimation We use the Long Short-Term Memory (LSTM) model as the network architecture. The model takes sequential tactile markers and corresponding three-axis forces from each timestep as inputs and predicts the six aforementioned parameters as output. We generate a simulated dataset that includes tactile marker readings, and three-axis contact force readings based on randomized system parameters. Our dataset has 2010 samples, 1800 for training, 200 for validation, and 10 for testing. Each sample's associated system parameters are randomized within pre-determined ranges, which ensures their real-world applicability as shown in Table 6. The model was trained in batch size 32 for 3000 epochs, and using an Adam optimizer with a learning rate of 0.001.

Parameter	Lower	Upper
Kn	10	100
Kd	100	400
Kt	50	150
Fc	5	20
μ	800	1500
λ	7000	10000

Table 6: Range of parameter randomization.

Random estimation We also provide a random estimation as our baseline. We use the 10 randomized testing system parameters from our dataset and then compare the simulated tactile markers with the ground-truth markers either from the simulation or real world.

A.4 DiffTactile Task and Evaluation Details

A.4.1 Task Setup Details

Reinforcement Learning (RL) We use object particles' state vector $o_i(t)$ and tactile sensor's pose $T_s(t)$ as state observations. Additionally, we use tactile markers' position in 2D image $m_i = (u_i, v_i)$, three-axis contact force $F(t) = (F_x, F_y, F_z)$, and contact location center $l(t)$ by averaging all in-contact nodes' locations as tactile observations. Given that the total number of markers is 136, we downsample the number of particles to four times the number of markers, ensuring a balanced dimensionality across different input segments. The input vector is formulated by either only state observation or with additional tactile observation. Then it is fed into a Multi-Layer Perceptron (MLP) policy network.

We use stable-baseline3 (Raffin et al., 2021)'s default PPO and SAC as our policy networks. Given an initial trajectory which is the same for all baseline methods, the policy network tasks inputs vector and outputs an action $\Delta v_s(t)$ of the sensor for each time step. Then we update the sensor's velocity as $v_s(t) += \Delta v_s(t)$. We constraint the actions in the range of $[-0.15, 0.15]$ for a reasonable action size.

CMA-ES In each optimization step, we generate 20 new trajectories based on the current trajectory with a standard deviation of 0.15 for a fair comparison with RL. We evaluate each new trajectory's loss and then update the policy based on the evaluation. This then informs the generation of the next optimization step's 20 trajectories. We used the same initial trajectory as RL and ran 100 optimization steps in total for each task.

Gradient-based Optimization (Ours), In each optimization step, we forward the simulation and calculate the defined loss, and then backpropagate the gradients from the loss to the target optimization variables. We then update the target variables with Adam optimizer. To enhance optimization efficiency, we use different learning rates for different optimization variables. The hyper-parameters can be found in Table 7, where lr_p is the learning rate for translation, lr_o is the

Lr	ObjectRepose	CableStraighten	CaseOpen	SurfaceFollow	GraspElastic	GraspPlastic
lr_p	$5e0$	$1e-2$	$1e3$	$5e-7$	$5e-2$	$5e-2$
lr_o	$1e3$	$1e-2$	$1e1$	$5e-5$	$1e-5$	$1e-5$
lr_w	N/A	$1e-2$	N/A	N/A	$5e-2$	$5e-2$

Table 7: Learning rate of ours method in each task

	ObjectRepose	CableStraighten	CaseOpen	SurfaceFollow	GraspElastic	GraspPlastic
α	$1e1$	$1e-2$	$1e1$	$1e2$	$1e2$	$5e-2$
β	$5e-12$	$1e-5$	$5e-12$	$1e0$	$5e0$	$1e1$

Table 8: Coefficient of combined loss α and β of each task

learning rate for orientation, and lr_w is the learning rate for the gripper's width. Note that for tasks where we use a single tactile sensor, the value of lr_w is listed as N/A.

A.4.2 Loss and Reward

For training purpose, we assign task-specific weights to state and tactile losses, denoted as α and β. The final loss is then calculated as $L_{total} = \alpha \times L_{state} + \beta \times L_{tactile}$. The task-specific values for α and β are provided in Table 8. We use the losses discussed in Section 4.3 and Section 4.4 for optimization-based methods including our gradient-based method and CMA-ES; for Model-free RL algorithms, we subtract the cumulative loss of two consecutive steps to obtain the single-step loss, and then calculate the reward to fit the settings of RL algorithms.

A.4.3 Metrics

We design task-specific metrics for evaluations. For **Object Repose**, we measure the angle in degrees of the object from its initial horizontal position. A larger value in this context indicates better performance. In **Cable Straighten**, our metric is the average displacement of each particle on the cable from its target horizontal position. A smaller displacement value indicates a more desirable result. In **Case Open**, we calculate the angle in degrees between the case lid and the horizontal table. The lid, due to gravity, can potentially show a negative value if the training results are suboptimal. Therefore, a larger angle suggests better performance. Lastly, for **Surface Follow**, we evaluate the continuous in-contact distance the sensor travels on the surface within an identical total timestep span. Here, a longer distance means better results.

A.5 Loss Details

We design different losses used for different tasks to obtain state or tactile reward, shown in Table 10.

Grasping The losses we used are defined in Section 4.3. γ and η are set to 0.1.

Surface following We use L_{pos} as the state loss and L_{force} as the tactile loss.

Notion	Explanation
$P(t)$	3D position of the center of the object
$F(t)$	Aggregated three-axis force vector on the surface of the tactile sensor
$F_t(t)$	The shear force with respect to the sensor coordinate frame
$F_n(t)$	The normal force with respect to the sensor coordinate frame
μ	The friction coefficient
$l(t)$	The center location of the contact area on the tactile sensor
$\theta(t)$	The rotated angle of the object from its initial position
$SDF(t)$	The signed distance field of the object
$M(t)$	The mass distribution of the object

Table 9: Explanation of parameters used in loss and metric computation

Loss	Equation
L_{pos}	$\|\|P(t) - P_{target}\|\|^2$
L_{deform}	$\gamma L_{dist} + \eta L_{mass}$
L_{dist}	$SDF(t) \cdot SDF_{target}$
L_{mass}	$\|\|M(t) - M_{target}\|\|$
L_{slip}	$\frac{\|\|F_t(t)\|\|}{\mu \|\|F_n(t)\|\|}$
L_{force}	$\|\|F(t) - F_{target}\|\|^2$
L_{loc}	$\|\|l(t) - l_{target}\|\|^2$
L_{angle}	$\|\|\theta(t) - \theta_{target}\|\|^2$
L_{cable}	$\sum_i \|\|p_i(t) - p_i(target)\|\|^2$

Table 10: Computation equation of losses

Cable straighting L_{cable} is used to obtain the state reward which calculates the sum of the distance between the target position and the current position for each node on the cable. With tactile feedback, our optimization loss comprises $L_{force} + L_{loc}$.

Case opening & **Object reposing** For these two tasks, we use L_{angle} to get the state reward and L_{force} to get the tactile reward.