Appendix for Efficient Tactile Simulation with Differentiability for Robotic Manipulation

A Penalty-based Tactile Simulation and Derivatives

In this section, we give the details of the forward dynamics and backward gradient derivation of our differentiable penalty-based tactile simulation. In §A.1, we give the formulation of the equations of motion for forward dynamics with the BDF1 time stepping scheme. In §A.2, we introduce our penalty-based tactile model and derive its analytical derivatives, which are necessary for our implicit forward time integration and backward gradients computation. In §A.3, we show in detail how we compute the analytical gradients of the whole simulation through reverse-mode backward differentiation.

A.1 Equations of Motion

We give the formulation of equations of motion $g(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1}, \boldsymbol{u}_t, \boldsymbol{q}_t)$ here. We follow the reduced-coordinate rigid-body dynamics formulation of DiffRedMax [1] and use the BDF1 implicit time integration scheme [2] with step size h to step forward the simulation. Mathematically, at each time step t, we take a state in reduced coordinate representation $(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1})$ and the joint-space action $\boldsymbol{u}_t$, and get the state $(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$ by solving the following equation with Newton's Method:

$$\left.\begin{aligned}\boldsymbol{q}_t &= \boldsymbol{q}_{t-1} + h\dot{\boldsymbol{q}}_t \\ \dot{\boldsymbol{q}}_t &= \dot{\boldsymbol{q}}_{t-1} + h\ddot{\boldsymbol{q}}_t(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{u}_t)\end{aligned}\right\} \Rightarrow \underbrace{\boldsymbol{q}_t - \boldsymbol{q}_{t-1} - h\dot{\boldsymbol{q}}_{t-1} - h^2\ddot{\boldsymbol{q}}_t(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{u}_t)}_{g(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1}, \boldsymbol{u}_t, \boldsymbol{q}_t)} = 0 \tag{6}$$

with

$$\ddot{\boldsymbol{q}}_t(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{u}_t) = \mathsf{M}_r^{-1}(\boldsymbol{q}_t)\left[\mathsf{f}_r(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t) + \mathsf{J}^\top(\boldsymbol{q}_t)\mathsf{f}_m(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t) + \mathsf{f}_{QVV}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t) + \boldsymbol{u}_t\right], \tag{7}$$

where M_r is the generalized mass matrix in reduced coordinates, J is the Jacobian, f_r is the generalized force vector generated by joint-space effects (*e.g.,* joint damping), f_m is the maximal wrench (*e.g.,* gravity, Coriolis forces, contact forces, external forces, etc.), f_{QVV} is the quadratic velocity vector, and $\boldsymbol{u}_t$ is the joint-space action. Whenever we need the velocity, we compute it from the positions: $\dot{\boldsymbol{q}}_t = (\boldsymbol{q}_t - \boldsymbol{q}_{t-1})/h$. We will not go into details of Eq. 7 since it can be found in many rigid body dynamics tutorials.

A.2 Penalty-based Tactile Model and the Derivatives

As introduced in §3.2, our penalty-based tactile model contains two parts: first, we compute the contact forces (*i.e.,* normal force and friction force) at the tactile point's location with a penalty-based approach; then, we project the contact force into the local coordinate frame of the tactile point to acquire the desired *shear* and *normal* tactile force magnitudes:

$$\boldsymbol{f}_n = (-k_n + k_d\dot{d})d\boldsymbol{n}, \quad \boldsymbol{f}_t = -\frac{\boldsymbol{v}_t}{\|\boldsymbol{v}_t\|}\min(k_t\|\boldsymbol{v}_t\|, \mu\|\boldsymbol{f}_n\|) \tag{8a}$$

$$\boldsymbol{f} = \boldsymbol{f}_n + \boldsymbol{f}_t \tag{8b}$$

$$T_{sx} = \boldsymbol{f}^\top\boldsymbol{x}, \qquad T_{sy} = \boldsymbol{f}^\top\boldsymbol{y}, \qquad T_n = \boldsymbol{f}^\top\boldsymbol{z}. \tag{8c}$$

In order to complete the backward gradient computation, we need to compute the derivatives of the tactile forces with respect to the state of the simulation (*i.e.,* $\frac{\partial T_{\{sx,sy,n\}}}{\partial \mathbf{q}_t}$ and $\frac{\partial T_{\{sx,sy,n\}}}{\partial \dot{\mathbf{q}}_t}$). We drop the subscript t for brevity. Let $\mathbf{q}_m$ be the state of the simulation in maximal coordinates (which can be converted to reduced coordinates through the Jacobian [3]). We have

$$\frac{\partial T_{\{sx,sy,n\}}}{\partial \boldsymbol{q}} = \{\boldsymbol{x}^\top, \boldsymbol{y}^\top, \boldsymbol{z}^\top\}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}} \tag{9a}$$

$$= \{\boldsymbol{x}^\top, \boldsymbol{y}^\top, \boldsymbol{z}^\top\}\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}_m}\mathsf{J} + \frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{q}}_m}\frac{\partial \dot{\boldsymbol{q}}_m}{\partial \boldsymbol{q}}\right) \qquad \left(\frac{\partial \boldsymbol{q}_m}{\partial \boldsymbol{q}} = \mathsf{J}\right) \tag{9b}$$

$(\frac{\partial \dot{\boldsymbol{q}}_m}{\partial \boldsymbol{q}} = \frac{\partial \mathsf{J}}{\partial \boldsymbol{q}} \otimes \dot{\boldsymbol{q}}$, which is provided by DiffRedMax.)

and

$$\frac{\partial T_{\{sx,sy,n\}}}{\partial \dot{\boldsymbol{q}}} = \{\boldsymbol{x}^\top, \boldsymbol{y}^\top, \boldsymbol{z}^\top\} \frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{q}}} \tag{10a}$$

$$= \{\boldsymbol{x}^\top, \boldsymbol{y}^\top, \boldsymbol{z}^\top\} \left(\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{q}}_m} \mathsf{J} \right). \qquad \left(\frac{\partial \dot{\boldsymbol{q}}_m}{\partial \dot{\boldsymbol{q}}} = \mathsf{J} \right) \tag{10b}$$

Let $\mathbf{B}_1$ denote the body the tactile point attaches to, and $\mathbf{B}_2$ be the body the tactile point has contact with. We first derive the gradients of the contact normal force $\boldsymbol{f}_n$.

$$\frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{q}_m} = d\mathbf{n} \frac{\partial\big(-k_n + k_d \dot{d}\big)}{\partial \boldsymbol{q}_m} + (-k_n + k_d \dot{d}) \boldsymbol{n} \frac{\partial d}{\partial \boldsymbol{q}_m} + (-k_n + k_d \dot{d}) d \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{q}_m} \tag{11a}$$

$$= k_d d \mathbf{n} \frac{\partial \dot{d}}{\partial \boldsymbol{q}_m} + (-k_n + k_d \dot{d}) \boldsymbol{n} \frac{\partial d}{\partial \boldsymbol{q}_m} + (-k_n + k_d \dot{d}) d \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{q}_m} \tag{11b}$$

and

$$\frac{\partial \boldsymbol{f}_n}{\partial \dot{\boldsymbol{q}}_m} = k_d d \mathbf{n} \frac{\partial \dot{d}}{\partial \boldsymbol{q}_m}. \tag{12}$$

The values d, $\dot{d}$, $\boldsymbol{n}$ are only related to the bodies $\mathbf{B}_1, \mathbf{B}_2$. We assume that there is a distance function of body $\mathbf{B}_2$ (can be either an analytical function if $\mathbf{B}_2$ is a primitive shape or a signed distance field if $\mathbf{B}_2$ is an arbitrary shape), so their corresponding derivatives can be computed easily from the distance function.

Next, we derive the gradients for the contact friction force $\boldsymbol{f}_t$. Since the contact friction force is composed of the static friction force ($\boldsymbol{f}_s = -k_t \boldsymbol{v}_t$) and dynamic friction force ($\boldsymbol{f}_d = -\frac{\boldsymbol{v}_t}{\|\boldsymbol{v}_t\|} \mu \|\boldsymbol{f}_n\|$), we derive the derivatives for each of them separately.

For the static friction force, the derivatives are:

$$\frac{\partial \boldsymbol{f}_s}{\partial \boldsymbol{q}_m} = -k_t \frac{\partial \boldsymbol{v}_t}{\partial \boldsymbol{q}_m} \tag{13}$$

and

$$\frac{\partial \boldsymbol{f}_s}{\partial \dot{\boldsymbol{q}}_m} = -k_t \frac{\partial \boldsymbol{v}_t}{\partial \dot{\boldsymbol{q}}_m}, \tag{14}$$

where the derivatives of $\boldsymbol{v}_t$ can also be acquired from the distance function.

For the dynamic friction force, the derivatives are:

$$\frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{q}_m} = -\mu \frac{\boldsymbol{v}_t}{\|\boldsymbol{v}_t\|} \frac{\partial \|\boldsymbol{f}_n\|}{\partial \boldsymbol{q}_m} - \mu \|\boldsymbol{f}_n\| \frac{\partial\big(\boldsymbol{v}_t / \|\boldsymbol{v}_t\|\big)}{\partial \boldsymbol{q}_m} \tag{15a}$$

$$= -\mu \frac{\boldsymbol{v}_t}{\|\boldsymbol{v}_t\|} \frac{\boldsymbol{f}_n^\top}{\|\boldsymbol{f}_n\|} \frac{\partial \boldsymbol{f}_n}{\partial \boldsymbol{q}_m} - \mu \|\boldsymbol{f}_n\| \left(\frac{1}{\|\boldsymbol{v}_t\|} \frac{\partial \boldsymbol{v}_t}{\partial \boldsymbol{q}_m} - \frac{\boldsymbol{v}_t \boldsymbol{v}_t^\top}{\|\boldsymbol{v}_t\|^3} \frac{\partial \boldsymbol{v}_t}{\partial \boldsymbol{q}_m} \right) \tag{15b}$$

and

$$\frac{\partial \boldsymbol{f}_d}{\partial \dot{\boldsymbol{q}}_m} = -\mu \frac{\boldsymbol{v}_t}{\|\boldsymbol{v}_t\|} \frac{\boldsymbol{f}_n^\top}{\|\boldsymbol{f}_n\|} \frac{\partial \boldsymbol{f}_n}{\partial \dot{\boldsymbol{q}}_m} - \mu \|\boldsymbol{f}_n\| \left(\frac{1}{\|\boldsymbol{v}_t\|} \frac{\partial \boldsymbol{v}_t}{\partial \dot{\boldsymbol{q}}_m} - \frac{\boldsymbol{v}_t \boldsymbol{v}_t^\top}{\|\boldsymbol{v}_t\|^3} \frac{\partial \boldsymbol{v}_t}{\partial \dot{\boldsymbol{q}}_m} \right). \tag{16}$$

A.3 Backward Gradients Computation through Adjoint Method

Since we use an implicit time integration scheme for forward dynamics, the core step of gradient computation is to differentiate through the nonlinear equations of motion. We re-write the finite-horizon tactile-based policy optimization problem here for convenience.

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \ \mathcal{L} = \sum_{t=1}^{H} \mathcal{L}_t\big(\boldsymbol{u}_t, \boldsymbol{q}_t, \mathbf{v}_t(\boldsymbol{q}_t)\big) \tag{17a}$$

$$\text{s.t.} \ g(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1}, \boldsymbol{u}_t, \boldsymbol{q}_t) = 0 \qquad \text{(Equations of Motion)} \tag{17b}$$

$$\boldsymbol{u}_t = \pi_\theta\big(\tilde{\boldsymbol{q}}_{t-1}, \tilde{\mathbf{v}}_{t-1}(\boldsymbol{q}_{t-1}), T_{t-1}(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1})\big). \qquad \text{(Policy Execution)} \tag{17c}$$

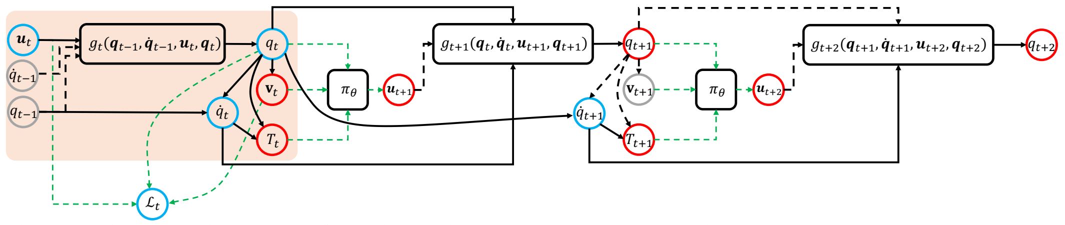

Figure 6: Computation graph of the simulation with BDF1 time stepping around time step t. We illustrate the computation graph for gradient derivations of $\partial\mathcal{L}/\partial\boldsymbol{q}_t$ and $\partial\mathcal{L}/\partial\boldsymbol{u}_t$. The boxes (*e.g.,* g, π_θ) represent functions, and the circles represent data/values. The grey circles are the data unrelated to the gradient derivation at step t. The red circles are the data $(\cdot)$ that we already have the gradient $\partial\mathcal{L}/\partial(\cdot)$ for when we arrive at step t during backward propagation. The blue circles are the data related to the gradient computation at step t. The green arrows are the data flows computed by PyTorch, and the black arrows are the data flows computed by our simulator. The dashed arrows are the data flows whose gradients computations are not handled by the simulator or are not related to the derivation at the current step. The orange-shaded part is our simulation layer for step t in the PyTorch computation graph.

Here, H is the task horizon, $\mathcal{L}_t$ is a step-wise task-dependent reward function, $\boldsymbol{u}$ is the action (*e.g.,* joint torques), $\boldsymbol{q}$ is the simulation state (*i.e.,* joint angles), and $\mathbf{v}$ is the derived auxiliary simulation variables (*e.g.,* fingertip positions) which themselves are a function of $\boldsymbol{q}$. Eq. 17b describes the nonlinear equations of motion (§A.1). Eq. 17c represents the inference of the control policy π_θ to obtain the desired action given the partial observation of the simulation state $\tilde{\boldsymbol{q}}$, partial observation of the simulation computed variables $\tilde{\mathbf{v}}$, and the tactile force values T from Eq. 8.

We embed our simulator as a differentiable layer into the PyTorch computation graph and use reverse mode differentiation to backward differentiate through dynamics time integration. To illustrate the gradient derivation, we draw the computation graph in Fig. 6. The computation steps such as loss/reward computation and policy inference (*i.e.,* green arrows in the figure) are computed by PyTorch, and the dynamics-related computation (*i.e.,* black arrows) are processed by our simulator in C++. Each step of our simulation can be regarded as a function (shown in the orange shaded box in Fig. 6) in the computation graph:

$$(\boldsymbol{q}_t, \mathbf{v}_t, T_t) = \text{Sim}(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1}, \boldsymbol{u}_t). \tag{18}$$

We compute the gradients $d\mathcal{L}/d\theta = \sum_t (\partial\mathcal{L}/\partial\boldsymbol{u}_t)(\partial\boldsymbol{u}_t/\partial\theta)$ for policy optimization. The first gradient, $\partial\mathcal{L}/\partial\boldsymbol{u}_t$, which includes the simulation dynamics and tactile derivatives, is derived analytically. The second gradient, $\partial\boldsymbol{u}_t/\partial\theta$, is computed by PyTorch's auto-differentiation.

Now we show how to compute $\partial\mathcal{L}/\partial\boldsymbol{u}_t$. We compute $\partial\mathcal{L}/\partial\boldsymbol{u}_t$ in reverse order, starting from the last time step. At time step t, we assume that we have the following gradients computed: $\partial\mathcal{L}/\partial\mathbf{v}_{t,t+1,\ldots}$, $\partial\mathcal{L}/\partial T_{t,t+1,\ldots}$, and $\partial\mathcal{L}/\partial\boldsymbol{q}_{t+1,t+2,\ldots}$ (red circles in Fig. 6). To compute the gradient backpropagation at step t, we need to compute $\partial\mathcal{L}/\partial\boldsymbol{q}_t$ and $\partial\mathcal{L}/\partial\boldsymbol{u}_t$:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{q}_t} &= \frac{\partial\mathcal{L}_t}{\partial\boldsymbol{q}_t} + \frac{\partial\mathcal{L}}{\partial\boldsymbol{q}_{t+1}}\left(\frac{\partial\boldsymbol{q}_{t+1}}{\partial\boldsymbol{q}_t} + \frac{\partial\boldsymbol{q}_{t+1}}{\partial\dot{\boldsymbol{q}}_t}\frac{\partial\dot{\boldsymbol{q}}_t}{\partial\boldsymbol{q}_t}\right) + \left(\frac{\partial L}{\partial T_{t+1}}\frac{\partial T_{t+1}}{\partial\dot{\boldsymbol{q}}_{t+1}} + \frac{\partial L}{\partial\boldsymbol{q}_{t+2}}\frac{\partial\boldsymbol{q}_{t+2}}{\partial\dot{\boldsymbol{q}}_{t+1}}\right)\frac{\partial\dot{\boldsymbol{q}}_{t+1}}{\partial\boldsymbol{q}_t}\\ &\quad + \frac{\partial\mathcal{L}}{\partial\mathbf{v}_t}\frac{\partial\mathbf{v}_t}{\partial\boldsymbol{q}_t} + \frac{\partial\mathcal{L}}{\partial T_t}\left(\frac{\partial T_t}{\partial\boldsymbol{q}_t} + \frac{\partial T_t}{\partial\dot{\boldsymbol{q}}_t}\frac{\partial\dot{\boldsymbol{q}}_t}{\partial\boldsymbol{q}_t}\right),\end{aligned} \tag{19a}$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{u}_t} = \frac{\partial\mathcal{L}_t}{\partial\boldsymbol{u}_t} + \frac{\partial\mathcal{L}}{\partial\boldsymbol{q}_t}\frac{\partial\boldsymbol{q}_t}{\partial\boldsymbol{u}_t}. \tag{19b}$$

The derivatives $\partial\mathbf{v}_t/\partial\boldsymbol{q}_t$ can be computed from the functions $\mathbf{v}(\boldsymbol{q})$ easily. The derivatives $\partial T_t/\partial\boldsymbol{q}_t$, $\partial T_t/\partial\dot{\boldsymbol{q}}_t$, and $\partial T_{t+1}/\partial\dot{\boldsymbol{q}}_{t+1}$ have been shown in §A.2. The derivatives of $\partial\dot{\boldsymbol{q}}_t/\partial\boldsymbol{q}_t$ and $\partial\dot{\boldsymbol{q}}_{t+1}/\partial\boldsymbol{q}_t$ can be computed from the BDF1 equations (Eq. 6).

$$\frac{\partial\dot{\boldsymbol{q}}_t}{\partial\boldsymbol{q}_t} = \frac{1}{h}\mathrm{I} \tag{20a}$$

$$\frac{\partial\dot{\boldsymbol{q}}_{t+1}}{\partial\boldsymbol{q}_t} = -\frac{1}{h}\mathrm{I}. \tag{20b}$$

To computing the remaining derivatives $\partial\boldsymbol{q}_{t+1}/\partial\boldsymbol{q}_t$, $\partial\boldsymbol{q}_{t+1}/\partial\dot{\boldsymbol{q}}_t$, $\partial\boldsymbol{q}_{t+2}/\partial\dot{\boldsymbol{q}}_{t+1}$, and $\partial\boldsymbol{q}_t/\partial\boldsymbol{u}_t$ we must differentiate through the implicit function $g(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1}, \boldsymbol{u}_t, \boldsymbol{q}_t) = 0$. We show the derivation for $\partial \boldsymbol{q}_t / \partial \boldsymbol{u}_t$ and how to compute Eq. 19b efficiently through the adjoint method; the same approach can be used for computing others and Eq. 19a.

We apply the implicit function theorem on $g(\boldsymbol{q}_{t-1}, \dot{\boldsymbol{q}}_{t-1}, \boldsymbol{u}_t, \boldsymbol{q}_t) = 0$:

$$\frac{dg}{d\boldsymbol{u}_t} \equiv 0 \Rightarrow \frac{\partial g}{\partial \boldsymbol{q}_t}\frac{\partial \boldsymbol{q}_t}{\partial \boldsymbol{u}_t} + \frac{\partial g}{\partial \boldsymbol{u}_t} \equiv 0 \Rightarrow \frac{\partial \boldsymbol{q}_t}{\partial \boldsymbol{u}_t} = -\left(\frac{\partial g}{\partial \boldsymbol{q}_t}\right)^{-1}\frac{\partial g}{\partial \boldsymbol{u}_t}. \tag{21}$$

Plugging this into Eq. (19b):

$$\frac{d\mathcal{L}}{d\boldsymbol{u}_t} = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{u}_t} - \underbrace{\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}_t}}_{\boldsymbol{b}}\Bigg(\underbrace{\frac{\partial g}{\partial \boldsymbol{q}_t}}_{A}\Bigg)^{-1}\frac{\partial g}{\partial \boldsymbol{u}_t}. \tag{22}$$

To efficiently calculate Eq. (22), we apply the adjoint method to first solve $\boldsymbol{c}$ from the linear equation $A^\top \boldsymbol{c} = \boldsymbol{b}^\top$ and then compute Eq. (19b) as

$$\frac{d\mathcal{L}}{d\boldsymbol{u}_t} = \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{u}_t} - \boldsymbol{c}^\top \frac{\partial g}{\partial \boldsymbol{u}_t}. \tag{23}$$

B Detailed Experiment Setups and More Results

In this section, we give the details of the 5 experiments in §4 of the main text. The subsections correspond between this appendix section and the main experiment section, so that $\S4.1 \leftrightarrow \S B.1$, $\S4.2 \leftrightarrow \S B.2$, etc.

B.1 High-Resolution Tactile Ball Rolling Experiment

In §4.1, we use a ball rolling experiment to show the efficacy of the tactile force field generated by our simulator and to test the simulation speed. The resolution of the tactile marker points is 200×200, and the simulation step size $h = 5$ ms. Here we visualize the lower surface of the pad in Fig. 7(a). The speed of the simulation varies with the resolution of the tactile marker points and the frequency of the tactile force field computation. We report in Table 3 the speeds of the simulation under different tactile marker resolutions and different frequencies of tactile force field acquisition. All the experiments run on a single core of an Intel Core i7-9700K CPU. The simulation speed can be further accelerated by simply parallelizing different environments across multiple CPU cores, and the speed of each individual simulation can be significantly accelerated by computing the tactile force at each tactile marker point in parallel through GPU programming since all the tactile marker points are independent of each other.

Frequency / Resolution	40 Hz (5 steps)	10 Hz (20 steps)
10×10	3477 FPS	3562 FPS
50×50	3167 FPS	3482 FPS
200×200	1050 FPS	2360 FPS

Table 3: Simulation speeds at different tactile points resolutions and the frequencies of the tactile force computation.

B.2 Tactile-Based Stable Grasp Task

In §4.2, we show the usage of shear force information for control and the effectiveness of our tactile simulator in a parallel-jaw bar-grasping task.

Task Specification The task requires a WSG-50 parallel-jaw gripper to stably grasp a bar with *unknown mass distribution* in fewer than 10 attempts. The gripper has two tactile sensors with a tactile marker resolution of 13×10 with 1.5 mm space between adjacent markers (shown in Fig. 7(b)). The bar is composed of 11 blocks. The total mass of the bar is in the range $[51, 120]$ g. Directly randomizing the density of each block results in the configuration where the center of mass

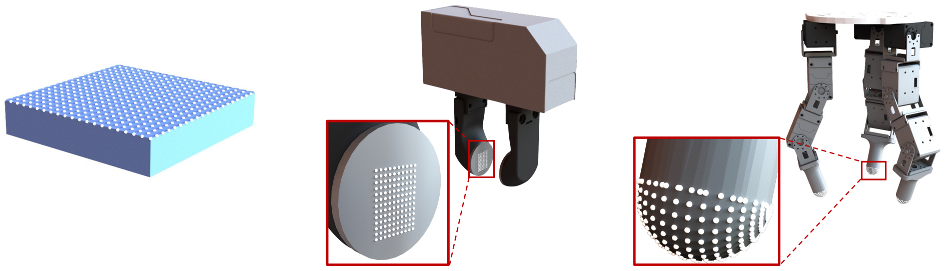
(a) Flat Tactile Pad (§4.1) (b) WSG-50 (§4.2, §4.3, §4.5) (c) D'Claw (§4.4)

Figure 7: Visualization of the tactile sensor layouts. We visualize the tactile sensor layouts of the three tactile manipulators we used in the experiments: (a) The flat tactile pad used in the ball rolling experiment (§4.1) (for better visualization, we only render tactile resolution of 20×20); (b) The WSG-50 gripper with GelSlim sensor used in the stable grasp task (§4.2), box pushing task (§4.3) and tactile RL insertion task (§4.5); (c) The D'Claw tri-finger hand used in the rotating cap task (§4.4).

of the bar is located near the geometric center in most cases. Therefore, to generate the bar with a uniform distribution of the center of mass, we randomize the center of mass location first, and then adjust the density of the blocks to meet the requirement of the center of mass location. We consider a grasp to be a failure if the bar is not lifted up or tilts more than 0.02 rad after the gripper grasps a bar.

The initial grasp location is the geometric center of the bar. The policy executes in an episodic process iterating between open-loop grasp attempts followed by grasp position adjustments. Specifically, the policy observes the tactile sensor readings at the frame the gripper lifts up the bar. Based on this observation input only, the policy outputs a delta change in the grasping location. A scripted grasping controller will then be executed to grasp the bar in the predicted grasping location.

Reward Function The reward function is defined as:

$$\mathcal{R}_t = \begin{cases} 100 & |\alpha| \leq 0.2 \text{ rad and } h > 0.5 \text{ cm (success)} \\ -10|\alpha| & \text{otherwise,} \end{cases} \tag{24}$$

where α is the tilting angle of the bar and h is the lifted height of the bar.

Policy Learning We treat the tactile force field as a multi-channel "image" (resolution is 13×10 and each channel is for the force component in each axis). The policy is modeled as a shallow CNN (*Conv-ReLU-MaxPool-Conv-ReLU-FC-FC*) that takes as input the tactile sensor readings from two tactile pads. We train the policies with PPO [4] using 32 parallel environments with 20K environment steps in total. The PPO parameters are reported in Table 4.

Parameter names	**Value**
learning rate	$3e^{-4}$ (with *linear decay* schedule)
number of rollouts per iteration	32
entropy coefficient	0.01
value loss coefficient	0.5
batch size	256
discount factor γ	0.99
GAE λ	0.95
PPO clip range	0.2

Table 4: The hyperparameter setting of PPO on the tactile-based stable grasp task.

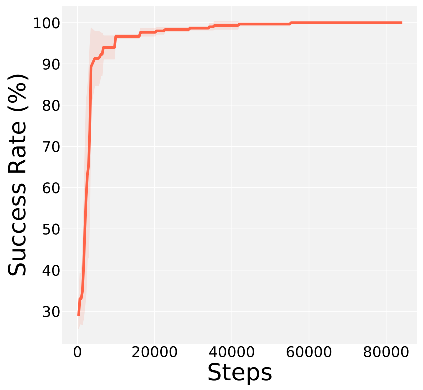

Figure 8: Training Curve of Tactile-based Stable Grasp Task. The curve is averaged from three random seeds. The shaded area is the standard deviation.

Training Results We train the policies with 3 different random seeds. The training curve is plotted in Fig. 8. We test the trained policies for 320 times, and the success rate is $98.5 \pm 1.8\%$ (the success rate is updated from the one reported in §4.2). The average number of attempts taken to stably grasp the bars is 2.1. We further test the learned policies on grasping the bar with a different number of blocks. While the policies are trained with 11 blocks, it achieves 93.3% success rate on 7-block bars and 99.8% on 13-block bars. The generalizability of the learned policies comes from the tactile-based observation, since the rotation pattern of the tactile force field remains consistent no matter how long the bar is.

B.3 Tactile-Based Box Pushing Task

In §4.3, we design a box pushing task similar to [5] to demonstrate how we can leverage the provided analytical gradients to help learn tactile-based control policies better and faster.

Task Specification The task is to use the same WSG-50 parallel jaw gripper as the one in the stable grasp task (with only one finger kept) to push the box to a randomly sampled goal location and orientation. The ranges of the goal location coordinates are $x_g \in [0.15 \text{ m}, 0.25 \text{ m}], y_g \in [-0.2 \text{ m}, 0.2 \text{ m}]$. The range of the goal orientation is $\alpha_g \in [y_g\pi - \pi/16, y_g\pi + \pi/16]$. The initial position of the box is randomly disturbed ($[-0.02 \text{ m}, 0.02 \text{ m}]$ along the direction of the finger surface). A random external force f_{ext} (with $f^x_{\text{ext}}, f^y_{\text{ext}} \in [-1, 1]$ N) is applied continually on the box, which changes every 0.25 s. The control frequency is 40 Hz.

Reward Function The reward function is defined at each control step as

$$\mathcal{R}_t = \mathcal{R}_{\text{pos}} + \mathcal{R}_{\text{rot}} + \mathcal{R}_{\text{touch}} + \mathcal{R}_{\text{u}} \tag{25a}$$

$$\mathcal{R}_{\text{pos}} = -0.01\Big(\frac{\|p_{xy} - [x_g, y_g]\|}{\sigma_{\text{pos}}}\Big)^2, \quad \sigma_{\text{pos}} = 0.01 \text{ m} \tag{25b}$$

$$\mathcal{R}_{\text{rot}} = -0.1\Big(\frac{\alpha - \alpha_g}{\sigma_{\text{rot}}}\Big)^2, \quad \sigma_{\text{rot}} = \frac{\pi}{36} \text{ rad} \tag{25c}$$

$$\mathcal{R}_{\text{touch}} = -\Big(\frac{\|p_{\text{finger}} - p_{\text{box}}\|}{\sigma_{\text{touch}}}\Big)^2, \quad \sigma_{\text{touch}} = 0.02 \text{ m} \tag{25d}$$

$$\mathcal{R}_{\text{u}} = -0.1\|u\|^2, \tag{25e}$$

where p_{xy} is the position of the box in the x-y plane, α is the rotation angle of the box around the vertical axis (z axis), p_{finger} is the position of the center of the gripper finger, p_{box} is the position of the center of the box surface closest to the finger, and u is the policy action (normalized to $[-1, 1]$).

Policy Learning In this task, we explore another tactile observation representation. We flatten the whole tactile force field into a vector and model the policy as an MLP with 2 fully connected hidden layers of 64 units. For the *PPO* policies, we use the PPO parameters reported in Table 5. For the *GD* policies, we use Adam as our optimizer with $\beta_1 = 0.7, \beta_2 = 0.95$. The learning rate of Adam starts from 0.005 and follows a *linear decay* schedule over the episodes.

Parameter names	**Value**
learning rate	$3e^{-4}$ (with *linear decay* schedule)
number of rollouts per iteration	80
entropy coefficient	0
value loss coefficient	0.5
batch size	128
discount factor γ	0.99
GAE λ	0.95
PPO clip range	0.2

Table 5: The hyperparameter setting of PPO on the tactile-based box pushing task.

Experiment Results More visual results comparing different policies are provided in the supplemental video.

B.4 D'Claw Rotate Cap

In §4.4, we train a D'Claw tri-finger hand to open a cap on a bottle, to demonstrate that our method supports tactile sensors on curved surfaces. We put the tactile sensors on the three rounded fingertips in a hemisphere layout, and we use 302 evenly-spaced tactile markers. A close-up view of the tactile sensors is provided in Fig. 7(c).

Task Specification The task is to open a cap using the D'Claw hand. The position (randomized in a $0.04 \times 0.04\,\mathrm{m}^2$ region on the horizontal plane) and the radius (randomized in the range of $[0.2, 0.8]$ m) of the cap are unknown. There is also unknown damping (randomized in the range of $[0.01, 0.7]$) between the cap and the bottle. The task is considered a success if the cap is rotated by $\alpha_g = \pi/4$ rad. The only observation data that the policy gets are the angles of each joint, fingertip positions, and tactile sensor readings. This task is similar to how we open caps by just using proprioception sensory data and tactile feedback on the fingers without knowing the exact size and location of the cap. The control frequency is 40 Hz.

Reward Function The reward function is defined at each control step as

$$\mathcal{R}_t = \mathcal{R}_{\text{touch}} + \mathcal{R}_{\text{rot}} + \mathcal{R}_{\text{u}} + \mathcal{R}_z + \mathcal{R}_{\text{success}} \tag{26a}$$

$$\mathcal{R}_{\text{touch}} = -0.5 N_{\text{no touch}} \tag{26b}$$

$$\mathcal{R}_{\text{rot}} = -\min(\alpha - \alpha_g, 0)^2 \tag{26c}$$

$$\mathcal{R}_{\text{u}} = -0.005\|u\|^2 \tag{26d}$$

$$\mathcal{R}_z = -50 \cdot \mathbb{1}_{p_{i,z} > c_z \text{ for } i \in \{1,2,3\}} \tag{26e}$$

$$\mathcal{R}_{\text{success}} = 50 \cdot \mathbb{1}_{\alpha > \alpha_g}, \tag{26f}$$

where $N_{\text{no touch}}$ is the number of fingers not touching the cap, α is the rotating angle of the cap, and u is the action of the policy, $p_{i,z}$ is the z coordinate of i^{th} fingertip position, and c_z is the z coordinate of the cap's top surface. An episode is terminated when the task is successfully completed, the number of steps exceeds the maximum episode length, or any of the fingertips is above the cap.

Policy Learning The policy takes the hand joint angles, fingertip positions, and the tactile sensor readings as input, and outputs the delta change on the joint angles. We use PPO to train the policy (a shallow CNN) using 32 parallel environments. The PPO parameters are shown in Table 6. To show that the tactile sensors are useful in this task, we also train a baseline policy (a simple MLP policy) where the policy only takes as input the joint angles and fingertip positions.

Parameter names	Value
learning rate	$3e^{-4}$ (with *linear decay* schedule)
number of rollouts per iteration	32
entropy coefficient	0.01
value loss coefficient	0.5
batch size	256
discount factor γ	0.99
GAE λ	0.95
PPO clip range	0.2

Table 6: The hyperparameter setting of PPO on the D'Claw rotating cap task.

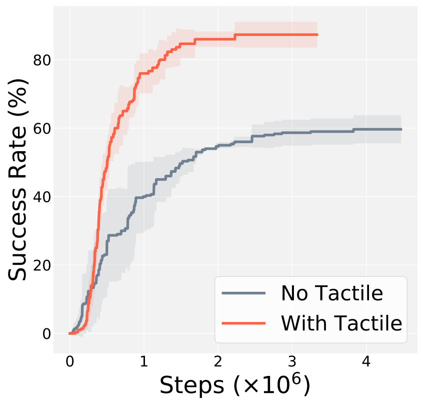

Figure 9: Training Curves of D'Claw Rotating Cap Task. The curves are averaged from three random seeds. The shaded area is the standard deviation.

Experiment Results We run each policy type three times with different random seeds, and plot the averaged training curves in Fig. 9. With tactile sensor readings, policies learn significantly faster and achieve an 87.3% success rate, while policies only achieve a 59.7% success rate when tactile sensor information is unavailable.

B.5 Zero-Shot Sim-to-Real: Tactile RL Insertion Task

In §4.5, we conduct a sim-to-real experiment on the tactile-RL insertion task.

Task Specification In this task, a gripper (same as the one in the stable grasp task) is controlled to insert a cuboid object into a rectangle-shaped hole with a random initial pose misalignment (up to 6 mm for translation error and 10° for rotation error). The insertion process is modeled as an episodic policy that iterates between open-loop insertion attempts followed by insertion pose adjustments. The robot has up to 15 pose correction attempts, and the robot only has access to tactile feedback from the sensors installed on both gripper fingers.

Real Robot Setup We use a 6-DoF ABB IRB 120 robot arm with a WSG-50 parallel jaw gripper. On each side of the gripper finger, we mount the GelSlim 3.0 tactile sensors that capture the tactile interaction between the fingers and the grasped object as a high-resolution tactile image. The rectangle-shaped object and hole are 3-D printed. The clearance between the object and hole is 2.25 mm and the initial misalignment between them is randomly sampled within the maximum value of (6 mm, 6 mm, 10°), which is identical to the previous work [6]. We also vary the grasping force between 10∼15 N and the grasping height between 42∼57 mm. To extract the marker tracking information from the raw image tactile measurement, we use a marker detection algorithm from [7]. The speed of the gripper moving down during the insertion attempt is 0.5 mm/s and we capture tactile images every 80 ms.

Reward Function The reward is defined for each insertion attempt as follows:

$$\mathcal{R}_t = \mathcal{R}_{\text{pos}} + \mathcal{R}_{\text{rot}} \tag{27a}$$

$$\mathcal{R}_{\text{pos}} = 10^4 \times \|\text{error}_{\text{position}}\|^2 \tag{27b}$$

$$\mathcal{R}_{\text{rot}} = 20 \times \text{error}^2_{\text{rotation}}. \tag{27c}$$

Policy Learning We train the control policies with PPO [4] for three types of misalignments as mentioned in §4.5. During training, we convert the simulated tactile force field into our normalized tactile flow map representation, and treat the resulting tactile flow map as a 13×10 flow "image" with 4 channels (2 sensors and 2 shear components of tactile forces). For *Rotation* and *Translation* tasks, we sample five tactile frames during the insertion and stack them to obtain the corresponding normalized tactile flow maps to form the policy observation. For *Rotation & Translation* task, we only use the last frame during the insertion attempt as the observation since we observed that the policy tends to overfit to the simulation when more frames are provided. Specifically, when we input more frames of tactile fields, the policy tends to learn to leverage some simulation-only unnoticeable patterns of the tactile field to complete tasks in a tricky way; however those patterns do not sometimes exist in the real robot, and thus we input fewer tactile information to prevent such overfitting. For all three tasks, we model the policy by a convolutional RNN to leverage more information from previous attempts. For better sim-to-real performance, we also apply the domain randomization technique to increase the robustness of the learned policies. Specifically, we randomly change various simulation parameters such as contact parameters, tactile sensor parameters, grasp forces, and grasp height within some ranges, and we also apply random noise to each value in the tactile force observation. The ranges of the parameter randomization are provided in Table 7.

	Parameter names	**Range**
Contact Parameters	k_n	$[2e^3, 1.4e^4]$
	k_t	$[20, 140]$
	μ	$[0.5, 2.5]$
Tactile Parameters	k_n	$[50, 450]$
	k_t	$[0.2, 2.3]$
	μ	$[0.5, 2.5]$
	k_d	$[0, 100]$
Other Parameters	grasp force	$[2.5, 16]$ N
	grasp height	$[-10, 5]$ mm
	tactile force noise	$[-1e^{-5}, 1e^{-5}]$

Table 7: The ranges of the domain randomization parameters.

References

[1] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. 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.

[2] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[3] Y. Wang, N. J. Weidner, M. A. Baxter, Y. Hwang, D. M. Kaufman, and S. Sueda. RedMax: Efficient & flexible approach for articulated dynamics. *ACM Trans. Graph.*, 38(4), July 2019. ISSN 0730-0301. doi:10.1145/3306346.3322952. URL https://doi.org/10.1145/3306346.3322952.

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

[5] A. Church, J. Lloyd, R. Hadsell, and N. F. Lepora. Optical tactile sim-to-real policy transfer via real-to-sim tactile image translation. *arXiv preprint arXiv:2106.08796*, 2021.

[6] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez. Tactile-rl for insertion: Generalization to objects of unknown geometry. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6437–6443. IEEE, 2021.

[7] I. Taylor, S. Dong, and A. Rodriguez. Gelslim3. 0: High-resolution measurement of shape, force and slip in a compact tactile-sensing finger. *arXiv preprint arXiv:2103.12269*, 2021.