2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

DOT-Sim: Differentiable Optical Tactile Simulation with Precise Real-to-Sim Physical Calibration

Yang You[1], Won Kyung Do[1], Aiden Swann[1], Rika Antonova[1,2], Monroe Kennedy[1], Leonidas Guibas[1]

***Abstract*— Simulating optical tactile sensors presents significant challenges due to their high deformability and intricate optical properties. To address these issues and enable a physically accurate simulation, we propose DOT-Sim: *Differentiable Optical Tactile Simulation*. Unlike prior simulators that rely on simplified models of deformable sensors, DOT-Sim accurately captures the physical behavior of soft sensors by modeling them as elastic materials using the Material Point Method (MPM). DOT-Sim enables rapid calibration of optical tactile sensor simulation using a small number of demonstrations within minutes, which is substantially faster than existing methods. Compared to current baselines, our approach supports much larger and non-linear deformations. To handle the optical aspect, we propose a novel approach to simulating optical responses by learning a residual image relative to the real-world idle state. We validate the physical and visual realism of our method through a series of zero-shot sim-to-real tasks. Our experiments show that DOT-Sim (1) accurately replicates the physical dynamics of a DenseTact optical tactile sensor in reality, (2) generates realistic optical outputs in contact-rich scenarios, and (3) enables direct deployment of simulation-trained classifiers in the real world, achieving 85% classification accuracy on challenging objects and 90% accuracy in embedded tumor-type detection, and (4) allows precise trajectory following with policy trained from demonstrations in simulation with an average error of less than 0.9 mm.**

I. Introduction

Tactile sensing plays a critical role in enabling robots to perceive and interact with their environment in a physically grounded manner. Among various tactile modalities, optical tactile sensors such as GelSight [1] and DenseTact [2], [3] have gained increasing popularity due to their high spatial resolution and ease of fabrication. However, the soft and deformable nature of these sensors, combined with their complex internal light transport mechanisms, poses significant challenges for simulation [4]. Accurate simulation of tactile sensors is essential for data generation, algorithm training, and sim-to-real transfer, yet remains underexplored.

Existing tactile simulation frameworks [2], [5], [6], [7] often rely on simplified approximations of contact geometry or assume fixed mappings from force to image output. This is problematic, since many optical tactile sensors with flexible surface materials exhibit highly nonlinear responses to contact, due to both material elasticity and internal optical effects. Capturing these aspects with high fidelity is critical for enabling robust downstream applications, including perception, classification, and control. Hence, existing methods fall short in this regard.

To overcome these challenges, we introduce **DOT-Sim**: ***D**ifferentiable **O**ptical **T**actile **Sim**ulation* – a physically grounded, differentiable simulation framework tailored for optical tactile sensors. Our method proceeds in two stages. First, we model the physical behavior of a tactile sensor using the Material Point Method (MPM), a particle-based continuum simulation approach well-suited for soft, elastic materials. We calibrate the physical parameters of the sensor, such as Young's modulus and Poisson's ratio, by aligning simulated deformations with a small number of real-world tactile observations. This calibration is made efficient through the use of differentiable simulation and gradient-based optimization. Second, we model the sensor's optical response by simulating its depth and surface normals via virtual camera rendering, and then learn a residual mapping from the simulated rendering to the real tactile image. Rather than predicting the raw image, we predict only the difference from an idle frame, thereby significantly improving the efficiency of learning to generate realistic tactile images.

Through extensive experiments, we show that DOT-Sim: 1) replicates the physical behavior of a recently developed DenseTact sensor with high fidelity; 2) generates realistic optical renderings in a variety of contact scenarios; 3) enables successful zero-shot sim-to-real transfer for downstream tasks, such as classification and trajectory following. DOT-Sim improves over the strongest baseline by 17.34% in average PSNR for optical image quality, achieves absolute accuracy gains of 28.24% & 44.83% on unseen indenter & tumor-type classification tasks respectively, and enables precise trajectory following in reality by controllers trained from demonstrations in simulation, with an average error of $< 0.9\,\mathrm{mm}$. Our framework opens new possibilities for employing tactile sensing, especially in scenarios that require extra precision for perception and interaction.

II. Background and Related Work

a) Optical Tactile Sensors: Optical tactile sensors such as GelSight [1], [8], DIGIT [9], and DenseTact [2], [3] have become popular due to their high spatial resolution and rich contact feedback. They have been used in a variety of downstream tasks including object recognition [10],

[1]Stanford University, United States. Email: {yangyou, wkdo, swann, monroek, guibas}@stanford.edu

[2]University of Cambridge, United Kingdom. Part of this work was completed while the author was with Stanford University. Email: rika.antonova@cst.cam.ac.uk

Leonidas Guibas and Yang You acknowledge support from the Toyota Research Institute University 2.0 Program, ARL grant W911NF-21-2-0104, a Vannevar Bush Faculty Fellowship, and a gift from the Flexiv corporation. Yang You is also supported in part by the Outstanding Doctoral Graduates Development Scholarship of Shanghai Jiao Tong University. Aiden Swann is supported by NSF GRFP Fellowship No. DGE-2146755. This work was also supported in part by NSF Grant No. 2142773 and 2220867.

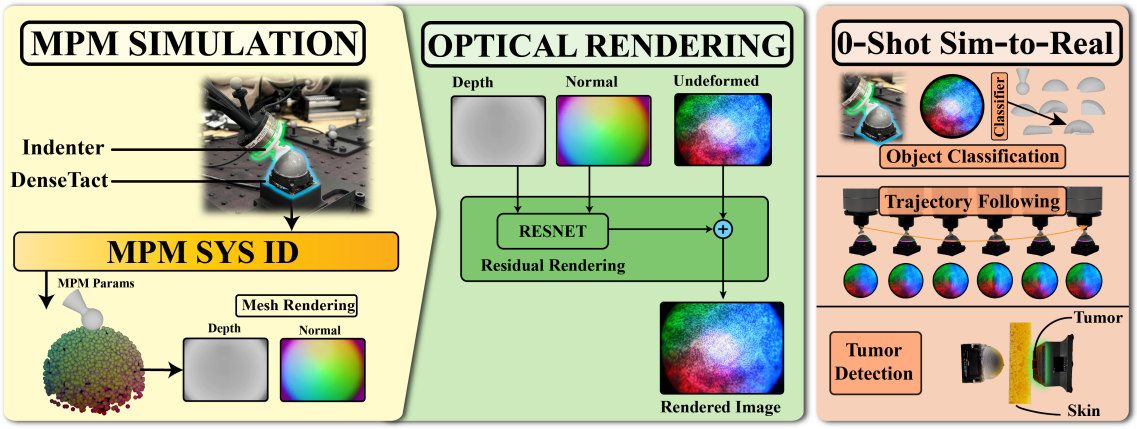


Fig. 1: Optical tactile sensors with flexible surface materials are challenging to simulate due to complex physical deformation and internal optical effects. We propose DOT-Sim – a framework that enables physically grounded and realistic tactile image generation by modeling both deformation and optical appearance. This enables zero-shot sim-to-real transfer for tasks such as indenter classification, trajectory following, and tumor detection.

manipulation [11], and shape reconstruction [1], [12]. Such sensors encode surface deformation as images, enabling compatibility with standard vision-based learning techniques. Despite variations in physical form factors, gel elasticity, and calibration methods, optical tactile sensors share a common structure: a soft deformable gel sensing element, an internal light emitting module, and camera with appropriate lenses (to observe the deformation of the gel sensing element). The gel element is built from clear gel materials (e.g. PDMS, silicone, or thermal plastics) coated with a reflective material to simplify sensing and capture the sensor deformations robustly. Markers or randomized patterns may be applied between the clear gel materials and the reflective surface. The camera captures sensor surface deformation, and the resulting images are used to extract shape information and contact force distributions. The complex interplay between deformation, lighting, and gel properties enables these images to be highly expressive, but also creates modeling and simulation challenges.

Simulator	Phy. Sim	Backend	Optical Sim	Sim→Real
Tacto [5]	PyBullet	PyBullet	OpenGL	None
Taxim [6]	FEM	N/A	Calibrated LUT	None
DiffTactile [4]	FEM	Taichi	Learned reflectance	Marker only
DOT-Sim (ours)	MPM	Warp	ResNet-based model	Full optical

TABLE I: Overview of the key existing optical tactile simulators and comparison with our work. We use a differentiable physical model and a fully learned optical model to achieve efficient and accurate calibration (real-to-sim alignment). Note **Sim→Real** refers to policy learning, not only image generation. This table includes the most notable methods.

b) Physical Modeling: Tacto [5] is a widely used rendering-based tactile simulator that approximates contact using analytical geometry and heuristics for visual appearance. While computationally efficient, it lacks physical realism and fails to generalize to complex contacts or deformations. In [13], a penalty-based collision model is used to improve simulation speed at the cost of lower accuracy. DiffTactile [4] uses a finite element method (FEM) to preform physics simulation of the deformable sensor. DiffTaichi [14] and ChainQueen [15] introduced differentiable simulation frameworks using the Material Point Method (MPM), but were primarily targeted at general-purpose deformable body simulation and not optimized for tactile sensors. Our approach builds upon MPM but focuses on dense, real-world calibration of tactile dynamics and optical appearance.

c) Optical Simulation: Rendering the optical properties of a sensor is very difficult due to the complex reflective properties and shadows which occur within the sensor. Many works choose to forgo optical simulation entirely and instead simulate simplified proxies, such as markers on the sensor surface, when obtaining policies for sim-to-real tasks [13], [4], [16]. Several recent works utilize simplified optical models to render the sensor. In Tacto [5], sensor images are rendered using OpenGL. These images can then be used to augment the real image. Taxim [6] utilizes a polynomial lookup table to color its sensor images. While these methods are fast, they lack visual accuracy. Several simulators use learning for optical simulation [4], [16] by either learning to generate images from scratch, or learning one component of an otherwise analytical rendering method. While these methods do employ learning for image generation, the resulting images are not accurate enough for policy conditioning. Instead, they utilize marker motion, which is easier to simulate and can be tracked on the real sensor. This compromises the expressive signal produced by the sensor and ultimately reduces the set of tasks which can be trained in sim. Furthermore, these methods learn small MLP neural networks, which operate on a per-pixel basis. In contrast, our method uses a single ResNet, which outputs the entire rendered image. High accuracy can be achieved by applying GANs to the output of a lower fidelity simulator [17]. However, the indenter set used in [17] is designed to be very easy to discriminate. Furthermore, [17] solves a fundamentally easier problem since the visual output of the GelSight used there is uniform, compared to the

DenseTact's pattern surface. Moreover, our method provides a full pipeline including physical and optical simulations.

d) System Identification and Sim-to-Real Transfer for Tactile Perception: Several recent works have explored learning tactile dynamics in a differentiable manner to facilitate calibration (system identification), e.g. PhysDreamer [18] and SpringGaus [19]. DiffTactile [4] applies this method to tactile sensing, utilizing a differentiable FEM simulation. DiffTactile optimizes their FEM simulation based only on tactile marker pose tracking and force input across a few trajectories. In contrast, we rely on a high fidelity Abaqus model, which generates full deformed meshes across several thousand indentations, enabling more direct supervision.

Table I gives a summary of recent tactile simulators and contrasts them with our proposed method.

III. The Proposed Approach: DOT-Sim

Simulation of optical tactile sensors is challenging due to the manufacturing variability of physical components (e.g. the gel elements) and the complexity of accurately modeling optical components. To achieve accurate simulation, we propose a two-stage framework that integrates physically grounded modeling with data-driven optical rendering, as illustrated in Figure 1. Instead of directly learning a mapping from forces to tactile images, we embed strong physical priors by first modeling the sensor's mechanical deformation using the Material Point Method (MPM). We then calibrate the sensor's physical parameters via differentiable physics, leveraging a small number of real-world interactions between the sensor and objects. This process yields paired data of sensor images and corresponding deformed sensor meshes, which can be obtained from the standard software provided with most optical tactile sensors.

After obtaining a physically accurate model, we simulate the sensor's mechanical deformation under forces from interaction with objects. To replicate optical rendering, we position a virtual camera inside the simulated sensor to render depth and surface normals. Then, we input these into a neural network that predicts a residual optical image relative to a reference (idle) frame. This substantially reduces approximation error compared with direct image regression.

A. Calibration of the Physical Properties of Tactile Sensors

To simulate the mechanical behavior of a tactile sensor, we model it as a collection of particles governed by the Material Point Method (MPM); [20] provides a recent overview of MPM. Each particle represents a small volume of the sensor and carries physical properties including volume V_p, mass m_p, position $\boldsymbol{x}_p^t$, velocity $\boldsymbol{v}_p^t$, deformation gradient $\boldsymbol{F}_p^t$, and local velocity field gradient $\boldsymbol{C}_p^t$ at time step t. MPM is a hybrid Eulerian-Lagrangian method, which also maintains another set of grid node mass m_i and velocity $\boldsymbol{v}_i$, where i is the grid index. We consider the case where the tactile sensor is in contact with rigid indenters. Let $\tilde{\boldsymbol{x}}^t$ and $\tilde{\boldsymbol{v}}^t$ denote the position and velocity of the indenter at time t. The dynamics of the full system are:

$$\{m_i\}, \{\boldsymbol{v}_i\} := P2G\big(\{m_p\}, \{\boldsymbol{x}_p^t\}, \{\boldsymbol{v}_p^t\}, \{\boldsymbol{F}_p^t\}, \{\boldsymbol{C}_p^t\}, \theta, \Delta t\big) \tag{1}$$

$$\{m_i\}, \{\boldsymbol{v}_i\} := CM\big(\{m_i\}, \{\boldsymbol{v}_i\}, \tilde{\boldsymbol{x}}^t, \tilde{\boldsymbol{v}}^t, \theta, \beta, \Delta t\big) \tag{2}$$

$$\{\boldsymbol{x}_p^{t+1}\}, \{\boldsymbol{v}_p^{t+1}\}, \{\boldsymbol{F}_p^{t+1}\}, \{\boldsymbol{C}_p^{t+1}\} := G2P\big(\{m_i\}, \{\boldsymbol{v}_i\}, \theta, \Delta t\big) \tag{3}$$

$$\tilde{\boldsymbol{x}}^{t+1}, \tilde{\boldsymbol{v}}^{t+1} := FK(\tilde{\boldsymbol{x}}^t, \tilde{\boldsymbol{v}}^t, \beta, \Delta t). \tag{4}$$

Here, $P2G$ and $G2P$ denote the MPM *particle to grid* and *grid to particle* transfer operations. CM is a contact model of the sensor-indenter interaction dynamics, FK is the forward kinematics function for the indenter. Parameter vector $\theta = (E, \nu)$ encapsulates the physical properties of the sensor. In this work, we focus on calibrating Young's modulus E and Poisson's ratio ν, following prior work such as PhysDreamer [18]. Indenter properties β are fixed, thus treating the indenter as rigid and unaffected by contact forces to reflect real-world conditions (where indenters are typically actuated by robot arms or human hands, and are not influenced by sensor deformation).

To estimate $\theta = (E, \nu)$, we collect a small set of real-world demonstration sequences (19 videos), recording the indenter position in a calibrated OptiTrack marker tracking system, and then generate the pseudo–ground-truth mesh deformation with the Abaqus 2024 Finite Element Analysis (FEA) simulator [21], which provides accurate but computationally expensive deformation results. We rely on Abaqus to generate pseudo ground-truth because directly capturing the deformed point cloud during indentation is highly challenging due to severe occlusions from the indenter, limited depth-sensor resolution, and measurement noise. Following prior works [22], [23], we align our MPM simulator with Abaqus outputs, which have been demonstrated to be accurate. For Abaqus simulations, we specify physical parameters such as the silicone–plastic friction coefficient and Yeoh hyperelastic material constants (for the gel model). These are obtained through standardized uniaxial and biaxial tests based on Densetact specifications [24]. Notably, these parameters are distinct from the learnable θ in our MPM model. Since the Abaqus solver is slow, complex, and lacks GPU acceleration, it is impractical to use it directly as the physical simulator, especially in reinforcement learning where simulation data must be generated online. Instead, we calibrate our MPM simulator to match Abaqus' accurate but costly deformations, enabling efficient and scalable physics simulation.

We then optimize θ by differentiable minimization of the Chamfer distance between the simulated and real point clouds. The trajectories are collected by poking the sensor at various angles and depths, and differentiable MPM simulations are run in parallel for all sequences. To obtain a robust estimate, we take the median of the estimated (E, ν) values over all sequences. The calibration process is highly efficient, completing within a few minutes on a single A5000 GPU, significantly faster than previous approaches [3]. Figure 2 illustrates the calibration process. For the differentiable

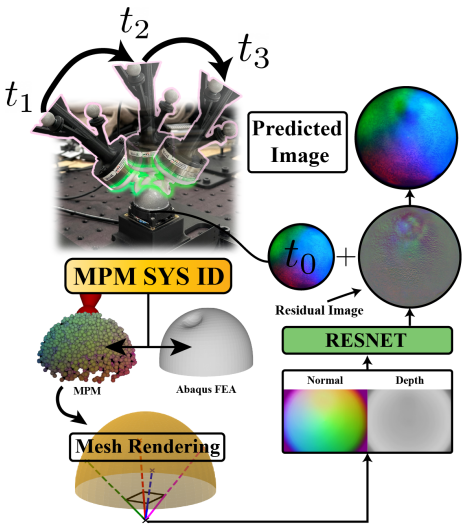


Fig. 2: Sensor calibration.

minimization, we re-parameterize E by optimizing $\log(E)$ instead. Both $\log(E), \nu$ have a learning rate of 0.1, and are optimized for 30 iterations.

B. Optical Simulation: A Residual Approach

While MPM simulation provides accurate modeling of physical deformation, the final output of a real-world tactile sensor is an optical image. Directly simulating light transport through the elastomer is challenging due to complex and sensor-specific optical properties, such as non-uniform gel coloration and internal reflections. To address this, we propose a hybrid approach that combines physics simulation for geometry with a neural rendering model for optics. Specifically, we first render depth and normal maps from the simulated sensor by placing a virtual camera at the center of the sensor base and casting rays through the upper hemisphere, mimicking the imaging process of real-world tactile sensors (see bottom left part of Figure 1). Given these simulated depth and normal maps, we input them into a neural network (with the U-Net architecture) that predicts the corresponding optical RGB image (middle part of Figure 1).

Although directly predicting the RGB image is feasible, it does not exploit the key observation that most regions of the tactile image remain static during contact, with deformation-induced signals manifesting primarily as localized residuals. Therefore, we predict a *residual image* relative to an idle (contact-free) reference frame. This residual image is defined as the difference between a contact frame and the idle frame, i.e., the frame at $t = 0$ when the indenter is not in contact with the sensor. We add the residual image to the real-world idle image to obtain the final optical output.

We employ the PyTorch implementation of *DeepLabV3 ResNet50* [25] as our backbone. The training objective is a simple ℓ_2 loss computed on pixel-wise RGB values. For all experiments, we use a batch size of 8 and a learning rate of 3e-4, with a weight decay of 1e-4, using the Adam optimizer. The model is trained for 100 epochs.

IV. Experimental Results

We evaluate DOT-Sim through a series of experiments designed to assess the realism and effectiveness of both its physical simulation and optical rendering. Section IV-A evaluates the accuracy of the physical simulation. Section IV-B assesses the fidelity of the optical simulation by comparing rendered outputs with real-world tactile images, and also presents an ablation comparing to direct regression without residual prediction. Section IV-C shows performance in downstream tasks, such as indenter classification, tumor detection, and trajectory tracking with sim-to-real transfer.

Since DOT-Sim can handle large sensor deformation, for our experiments we selected the DenseTact 2.0 sensor, which is a soft tactile sensor that can undergo significant deformation. Smaller deformations would be easier to handle, so the method should also work well with sensors that experience deformations smaller than those of the DenseTact 2.0. Furthermore, DenseTact 2.0 has a randomized pattern on its surface, which makes it particularly challenging to simulate with existing optical or physical simulators. Like most optical tactile sensors, DenseTact 2.0 consists of a camera, camera-gel mount, and the gel to sense deformation. The hemispherical gel is made of clear silicone, coated with a randomized pattern and reflective surface (this enables effective observation of contact-induced deformations). Given an optical image of the deformation, the corresponding deformed sensor mesh is generated by finite element analysis of the gel. This provides near ground-truth deformation of the sensor given a known indenter pose and shape.

A. Physical Accuracy of 3D Deformation

We evaluate the physical realism of the simulated sensor deformation by comparing the particle-based simulated surface against the observed DenseTact point cloud across various indenters. This evaluation is conducted using the evaluation dataset from all indenters in Medium (M) setting.

a) Evaluation Metrics: Following the evaluation protocols in [26], [27], we report performance using four standard metrics: L2 Chamfer Distance (L2 CD), Significant L2 Chamfer Distance (Sig. L2 CD), Earth Mover's Distance (EMD), and F-Score at 1mm. For each evaluation, we uniformly sample 2,048 points from both the predicted and ground-truth point clouds. L2 Chamfer Distance measures the average normed distance between each point in one point cloud and its nearest neighbor in the other, serving as an overall geometric similarity. Significant L2 Chamfer Distance is a variant of L2 considering only the top 1% furthest nearest neighbor for each point. Earth Mover's Distance computes the minimum cost of transforming one point cloud into the other. F-Score at 1mm evaluates the harmonic mean of precision and recall under a 1mm threshold, indicating the accuracy of point-level correspondences. For baselines, we compare with DiffTactile, Tacto and Taxim.

b) Results: Table II compares discrepancy between the simulated surface and the observed DenseTact point cloud. DOT-Sim achieves consistently better performance

Method	L2 CD mm ↓	Sig. L2 CD mm ↓	EMD mm ↓	F-Score @1mm ↑
*DiffTactile	-	-	-	-
Tacto	1.77	4.21	1.33	63.59
Taxim	1.74	3.97	1.31	64.69
DOT-Sim	**1.71**	**3.82**	**1.29**	**69.89**

TABLE II: Discrepancy between the simulated surface and the observed point cloud. Lower is better for CD and EMD; higher is better for F-Score. *DiffTactile: see explanation in the Results paragraph.

than both Taxim and Tacto across all metrics, demonstrating improved fidelity in modeling 3D surface deformation. Notably, we achieve a lower EMD (1.29 vs. 1.31 mm) and higher F-Score at 1mm (69.89 vs. 64.69), indicating a closer alignment. Note that L2 CD and EMD appear close to the baselines because they are averaged over the space, which masks the substantial improvement in the small regions directly under the indenter. Significant L2 error and F-Score compute error only over the largely deformed voxels, so are the more relevant metrics, and on these DOT-Sim shows a substantial reduction in error. We attempted to use the open-source code released by the authors of DiffTactile, but observed that the FEM sensor simulation was overly stiff, showing no noticeable deformation in the provided examples. Most apparent 'deformation' stemmed from the penalty-based contact model rather than true FEM response. Although the paper proposes a differential system identification method to tune FEM parameters, no implementation was included in the codebase, and the publication lacked sufficient detail for reproduction.

B. Optical Simulation Accuracy

To quantify the accuracy of DOT-Sim's optical outputs, we compare simulated and real tactile images in various scenarios. We use videos from six indenters captured at 24 FPS. Figure 3 shows the indenters. We report the following metrics: **Mean L2 Norm** (↓), **Significant Pixel L2 Norm** (↓), and **Peak Signal-to-Noise Ratio (PSNR)** (↑), where arrow direction shows whether higher or lower values are better (Appendix gives further details).

We evaluate on three levels of difficulty settings:
Easy (E): Two indenters (#1, #3) are used in both training and evaluation, with a random 80/20 split. **Medium (M):** All six indenters (#1 – #6) are used with a random 80/20 training/evaluation split. **Hard (H):** A leave-two-out setup where four indenters (#2, #4, #5, #6) are used for training, and two unseen indenters (#1, #3) are used for evaluation.

a) Results: We compare our method with DiffTactile [4] and Tacto [5], Table III shows the quantitative results. We observe significant improvements in visual accuracy: as high as 4 points of PSNR. For comparison, DiffTactile [4] reports 1 point PSNR improvement over their own baseline Taxim [6]. Figure 4 shows side-by-side comparisons of simulated and ground-truth tactile images under the Hard (H) setting. DOT-Sim improves over the strongest baseline

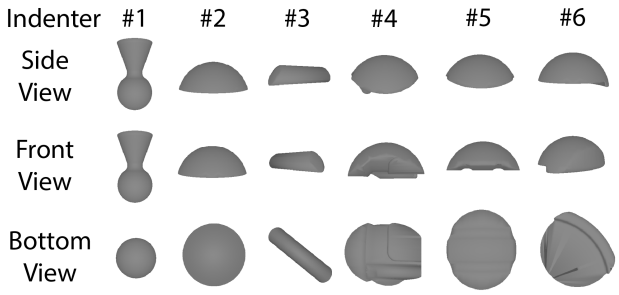


Fig. 3: Indenters.

Setting	Method	Mean L2 ↓ $\times 10^{-2}$	PSNR ↑	Sig. L2 ↓ $\times 10^{-2}$
Easy	DiffTactile	3.80	28.73	4.91
	Tacto (calib)	4.92	26.97	6.18
	DOT-Sim	**2.85**	**32.12**	**3.68**
Medium	DiffTactile	7.28	22.89	8.99
	Tacto (calib)	5.35	26.35	6.71
	DOT-Sim	**3.25**	**31.39**	**4.21**
Hard	DiffTactile	8.63	21.31	10.67
	Tacto (calib)	5.04	26.79	6.35
	DOT-Sim	**3.50**	**30.48**	**4.53**

TABLE III: Comparison of DOT-Sim against baselines.

by **17.34**% on average PSNR. DOT-Sim faithfully captures fine-grained contact features, including anisotropic responses (underrepresented in prior methods) that arise with indenter #3, which exhibits anisotropic geometry.

To evaluate the effectiveness of the proposed residual image prediction, we conduct an ablation study comparing DOT-Sim against a baseline that directly regresses the contact optical frame from the input normals and depths, without residual refinement. The evaluation is done under the Hard (H) setting. Quantitative results are summarized in Table IV, and qualitative comparisons are shown in Figure 5. DOT-Sim achieves a higher PSNR (30.48 vs. 28.89) and lower significant L2 error (4.53×10^{-2} vs. 5.39×10^{-2}), confirming that the residual prediction significantly improves image fidelity, while direct regression produces blurry images.

Method	PSNR ↑	Sig. L2 ↓ $\times 10^{-2}$
w/o Residual	28.89	5.39
DOT-Sim	**30.48**	**4.53**

TABLE IV: Ablating residual image prediction.

C. Performance in Downstream Classification, Detection, and Sim-To-Real Applications

a) Sim-To-Real Zero-Shot Indenter Classification: To assess whether the simulated optical images preserve sufficient discriminative information, we design a classification task where a model is trained to identify the type of indenter solely from simulated tactile images. Importantly, evaluation is performed entirely on real-world tactile images, making this a stringent test of sim-to-real generalization. We consider two settings: a) *in-domain* – the classifier is trained on simulated images of indenters that were also used during

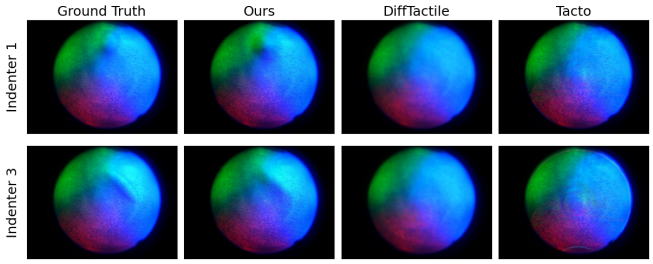


Fig. 4: DOT-Sim captures fine-grained and anisotropic features more faithfully than prior methods in Hard (H) setting.

Method	**In-domain** Accuracy (%)	**Out-of-domain** Accuracy (%)
DiffTactile	65.88	52.94
*Tacto	50.59	50.59
DOT-Sim	**90.48**	**81.18**

TABLE V: Indenter classification accuracy (%) on real-world images using simulated data. *Tacto does not learn optical rendering, so performs the same across both settings (since only the training set differs between in- and out-of-domain).

residual image prediction training, i.e. indenters #1 and #3; b) *out-of-domain* – the classifier is trained on simulated images of indenters that were not used during residual image prediction training. For b), only the indenter meshes are available at test time, real images of these indenters are not seen during training; indenters #2, #4, #5, and #6 are used for residual image prediction training, while the classifier is trained on simulated images from indenters #1 and #3. In both settings, the classifier's training data is generated entirely from simulated optical outputs of our residual prediction module described in Section III-A and III-B. This requires the simulated images to be not only realistic but also sufficiently aligned with the real-world domain for successful downstream task transfer.

Results in Table V show that DOT-Sim significantly outperforms prior methods in both settings. Notably, in the challenging out-of-domain scenario, our approach achieves over 80% accuracy despite the complete absence of real images for the test indenters. It highlights the strong sim-to-real generalization capability of DOT-Sim.

b) Tumor Detection via Tactile Feedback: Given a tactile image resulting from pressing on soft skin, this task is to classify the presence of a tumor. We simulate the presence of a tumor using a convex shape, and its absence using a concave shape, as illustrated on the left side of Figure 7. Skin deformation is simulated using a position-based dynamics (PBD) solver, for balance of speed and realistic elastic behavior. A tactile classifier is trained exclusively on synthetic data generated by DOT-Sim and evaluated on real-world images across three levels of skin stiffness, simulated using foams of varying thickness. For evaluation, we collect 36, 13, and 29 real tactile images corresponding to each stiffness level for Skin 1, Skin 2, Skin 3 respectively. Table VI shows the results (note that for Skin 2, DiffTactile and Tacto have the same 46.15% accuracy, since they both get 6/13 correct).

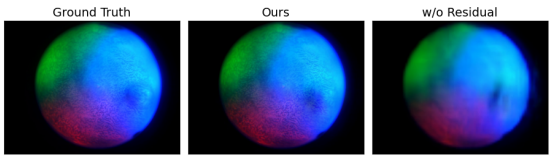


Fig. 5: Qualitative comparison of our residual image prediction vs direct regression without residual learning.

Method	**Skin 1** Acc. %	**Skin 2** Acc. %	**Skin 3** Acc. %
DiffTactile	52.78	46.15	51.72
Tacto	38.89	46.15	44.83
DOT-Sim	**80.56**	**92.31**	**96.55**

TABLE VI: Tumor classification results.

c) Precise Trajectory Following with Sim-to-Real Transfer: To demonstrate how DOT-Sim enables real-world control, we perform trajectory-following via sim-to-real transfer, where an agent mimics simulated demonstrations in a physical environment. Our policy relies solely on synthetic tactile images as input, without requiring force measures.

We train a ResNet-18 policy via behavior cloning in simulation, which outputs a 6-DoF velocity command from each tactile image, consisting of translational and angular velocities, which is sent to the robot's Cartesian controller.

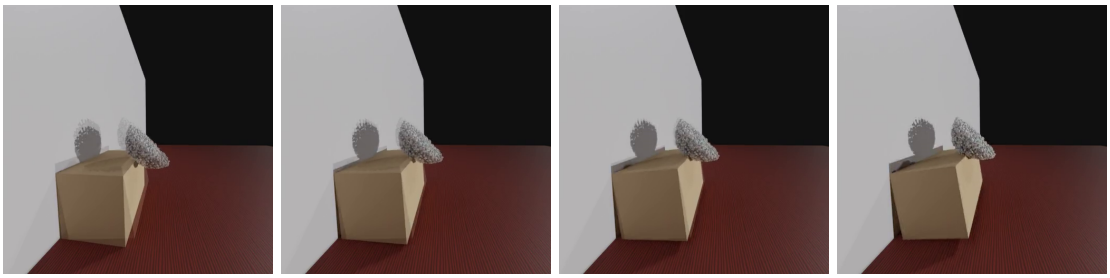
For real-time deployment on the xArm 7, we implement a multi-threaded control system with three threads: (1) image acquisition at 25 Hz, (2) policy inference ($\sim$3.9 ms) followed by synchronization to maintain 25 Hz, and (3) Cartesian control with 100 Hz pose feedback. The low latency ensures synchronization across sensing, inference, and control.

Qualitative results are shown in Figure 7. The transferred policy successfully tracks the demonstration, achieving an average action error of 0.896 ± 0.031 mm over 10 trials.

D. Reinforcement Learning

Although DOT-Sim primarily targets accurate physical and optical simulation, we also demonstrate its utility for control via reinforcement learning. We consider a box-repositioning task: using only DenseTact tactile images, the agent must rotate the box to a target yaw of 10° (i.e., flip by 10°), while maintaining stable contact. Success is declared when the yaw error drops below $< 2^\circ$.

We train a PPO [28] agent with SKRL [29] library in our differentiable DenseTact environment with a discrete 2-DoF action space. At every timestep the agent observes a 3-channel optical image simulated by our algorithm, flattened and encoded by a ResNet-18; the policy (categorical) and value heads share this backbone. The action space consists of 9 planar velocity commands {stay, left, right, up, down, up-left, up-right, down-left, down-right} with a fixed action magnitude. The reward primarily encourages reaching a target yaw of 10° about the sensor axis, with success defined as an angular error $< 2^\circ$; additional shaping penalizes ineffective no-op actions when far from the goal. Episodes are capped at 1s (24 FPS). During training and evaluation we use the calibrated physical parameters (e.g., $E = 27575$, $\nu = 0.303$) as specified in Section III-A. The policy converges quickly (within 15 minutes). Figure 6 visualizes a successful rollout, showing the executed over time.



(a) t_0 (b) t_1 (c) t_2 (d) t_3

Fig. 6: Policy rollout snapshots (left to right). DenseTact learns to drive the box toward the target yaw. The target position is overlaid semi-transparently.

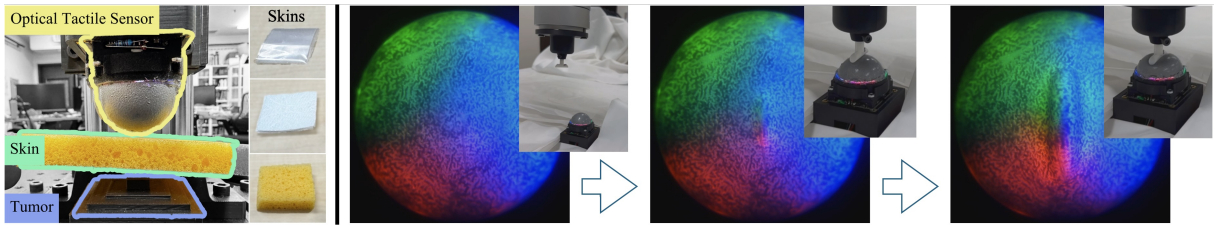


Fig. 7: Experimental setup for tumor detection (left) and sim-to-real trajectory following (right).

V. Conclusion and Limitations

We presented DOT-Sim, a differentiable simulation framework for optical tactile sensors that couples deformation modeling via the Material Point Method with a residual-based optical rendering pipeline. DOT-Sim achieves physically and visually consistent simulations using minimal real-world data, and transfers effectively to downstream tasks such as classification and control. Its modular design further allows integration with any MPM-based engine through a lightweight optical rendering plugin.

Despite these strengths, DOT-Sim has several limitations. First, it struggles to generalize to highly out-of-distribution geometries, especially those with sharp edges or fine surface details, leading to imperfect alignment between simulated and real contacts. Potential remedies include improving MPM efficiency to support denser point sampling, incorporating residual models to refine local deformations, and enlarging the indenter dataset to better capture geometric diversity. Second, the current simulation pipeline is computationally demanding, running at approximately 3 FPS on an NVIDIA A6000 GPU, which restricts its use in real-time control. While our reported configuration does not achieve high FPS, runtime can be significantly improved by tuning MPM parameters such as voxel resolution, contact density, and substeps per frame, yielding substantially higher frame rates with only marginal increases in error (see Table VII).

Voxel res. (mm)	Softness	# substeps	FPS	PSNR (Medium)
1.2	**15**	**100**	**3.6**	**31.39**
1.2	15	20	17.1	30.17
2.4	30	100	3.8	30.98
2.4	30	20	17.2	29.79

TABLE VII: Runtime and accuracy trade-offs with different MPM parameters. Contact density is parameterized by *softness*, where particles within $1/\mathit{softness}$ are considered in contact. The first row corresponds to the parameter settings used in the main experiments.

References

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

[2] W. K. Do and M. Kennedy, "Densetact: Optical tactile sensor for dense shape reconstruction," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6188–6194.

[3] W. K. Do, B. Jurewicz, and M. Kennedy, "Densetact 2.0: Optical tactile sensor for shape and force reconstruction," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 549–12 555.

[4] Z. Si, G. Zhang, Q. Ben, B. Romero, X. Zhou, C. Liu, and C. Gan, "Difftactile: A physics-based differentiable tactile simulator for contact-rich robotic manipulation," *arXiv preprint arXiv:2403.08716*, 2024. [Online]. Available: https://scispace.com/papers/difftactile-a-physics-based-differentiable-tactile-simulator-1q4pbz4xgc

[5] S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra, "Tacto: A fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3930–3937, 2022.

[6] W. Yuan and Z. Si, "Taxim: An example-based simulation model for gelsight tactile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3475–3482, 2022. [Online]. Available: https://scispace.com/papers/taxim-an-example-based-simulation-model-for-gelsight-tactile-f9075kf7

[7] A. Agarwal, T. Man, and W. Yuan, "Simulation of vision-based tactile sensors using physics based rendering," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2021. [Online]. Available: https://scispace.com/papers/simulation-of-vision-based-tactile-sensors-using-physics-4lcow7l4d5

[8] M. K. Johnson and E. H. Adelson, "Retrographic sensing for the measurement of surface texture and shape," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1070–1077.

[9] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, *et al.*, "Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, 2020.

[10] Y. Gao, L. A. Hendricks, K. J. Kuchenbecker, and T. Darrell, "Deep learning for tactile understanding from visual and haptic data." IEEE, 2016, pp. 536–543.

[11] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik,

"General in-hand object rotation with vision and touch," in *Conference on Robot Learning*. PMLR, 2023, pp. 2549–2564.
[12] A. Swann, M. Strong, W. K. Do, G. S. Camps, M. Schwager, and M. K. III, "Touch-gs: Visual-tactile supervised 3d gaussian splatting," *arXiv*, pp. 10 511–10 518, 2024.
[13] J. Xu, S. Kim, T. Chen, A. R. Garcia, P. Agrawal, W. Matusik, and S. Sueda, "Efficient tactile simulation with differentiability for robotic manipulation," in *6th Annual Conference on Robot Learning*, 2022. [Online]. Available: https://openreview.net/forum?id=6BIffCl6gsM
[14] Y. Hu, Z. Li, L. Anderson, T.-M. L. Kaya, D. Ritchie, F. Durand, W. T. Freeman, J. B. Tenenbaum, and W. Matusik, "Difftaichi: Differentiable programming for physical simulation," *International Conference on Learning Representations (ICLR)*, 2020.
[15] Y. Hu, J. Liu, A. Spielberg, J. B. Tenenbaum, W. T. Freeman, J. Wu, D. Rus, and W. Matusik, "Chainqueen: A real-time differentiable physical simulator for soft robotics," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6265–6271.
[16] Y. Zhao, K. Qian, B. Duan, and S. Luo, "Fots: A fast optical tactile simulator for sim2real learning of tactile-motor robot manipulation skills," 2024. [Online]. Available: https://arxiv.org/abs/2404.19217
[17] W. Chen, Y. Xu, Z. Chen, P. Zeng, R. Dang, R. Chen, and J. Xu, "Bidirectional sim-to-real transfer for gelsight tactile sensors with cyclegan," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6187–6194, 2022.
[18] T. Zhang, H.-X. Yu, R. Wu, B. Y. Feng, C. Zheng, N. Snavely, J. Wu, and W. T. Freeman, "Physdreamer: Physics-based interaction with 3d objects via video generation," in *European Conference on Computer Vision*. Springer, 2024, pp. 388–406.
[19] L. Zhong, H.-X. Yu, J. Wu, and Y. Li, "Reconstruction and simulation of elastic objects with spring-mass 3d gaussians," *European Conference on Computer Vision (ECCV)*, 2024.
[20] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle, "The material point method for simulating continuum materials," in *Acm siggraph 2016 courses*, 2016, pp. 1–52.
[21] Dassault Systèmes Simulia Corp., *Abaqus/Standard and Abaqus/Explicit User's Manual*, 2024th ed., Dassault Systèmes SIMULIA, Providence, RI, USA, 2024. [Online]. Available: https://www.3ds.com/products/simulia/abaqus/
[22] C. Zhao, J. Liu, and D. Ma, "ifem2. 0: Dense 3d contact force field reconstruction and assessment for vision-based tactile sensors," *IEEE Transactions on Robotics*, 2024.
[23] W. K. Do, M. Strong, A. Swann, B. Lei, and M. Kennedy III, "Tensortouch: Calibration of tactile sensors for high resolution stress tensor and deformation for dexterous manipulation," *arXiv preprint arXiv:2506.08291*, 2025.
[24] R. W. Ogden, *Non-linear elastic deformations*. Courier Corporation, 1997.
[25] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.
[26] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 international conference on 3D vision (3DV)*. IEEE, 2018, pp. 728–737.
[27] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "Grnet: Gridding residual network for dense point cloud completion," in *European conference on computer vision*. Springer, 2020, pp. 365–381.
[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," pp. 1–12, 2017, arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347
[29] A. Serrano-Muñoz, D. Chrysostomou, S. Bøgh, and N. Arana-Arexolaleiba, "skrl: Modular and flexible library for reinforcement learning," *Journal of Machine Learning Research*, vol. 24, no. 254, pp. 1–9, 2023. [Online]. Available: http://jmlr.org/papers/v24/23-0112.html

APPENDIX

A. Optical Rendering Network Architecture

Our optical rendering network follows a standard encoder-decoder design based on the DeepLabV3-ResNet50 architecture [25], implemented using the PyTorch torchvision library. The input is a 4-channel image composed of rendered surface normals and depth maps (3 for normal, 1 for depth), both normalized to $[0, 1]$. The output is a three-channel RGB image, trained to match ground-truth camera images.

We append a fully connected layer with 3 channels, and train the network using a per-pixel ℓ_2 loss between predicted and target RGB values. No perceptual or adversarial loss is used. The model is optimized with Adam using a learning rate of 3×10^{-4}, weight decay of 1×10^{-4}, and a batch size of 8 for 100 epochs. Input images are resized to 640×480.

We found that the ℓ_2 loss alone yields sharp reconstructions when using accurate geometry and surface rendering, without requiring additional regularization.

B. Evaluation Metrics

a) Physical Accuracy Metrics.: To evaluate the accuracy of predicted deformations in terms of 3D geometry, we adopt four standard point cloud comparison metrics, following [26], [27]. All metrics are computed on point clouds uniformly sampled with 2,048 points from both the predicted and ground-truth surfaces.

- **L2 Chamfer Distance (CD)**: Measures the average distance from each point in one point cloud to its nearest neighbor in the other. It captures overall geometric similarity and is symmetric by averaging both directions.
- **Significant L2 Chamfer Distance**: A variant of Chamfer Distance that focuses on the top 1% largest nearest-neighbor distances for each point, emphasizing outliers and surface discrepancies.
- **Earth Mover's Distance (EMD)**: Computes the optimal bijective assignment between points in the two clouds that minimizes total transport cost. EMD is more sensitive to global shape structure.
- **F-Score @ 1mm**: Computes the harmonic mean of precision and recall under a *1mm* distance threshold. A predicted point is considered a true positive if it lies within *1mm* of any ground-truth point, and vice versa.

These metrics jointly evaluate both average-case performance (CD, EMD) and worst-case or perceptual differences (Sig. CD, F-Score), offering a balanced picture of physical prediction fidelity.

b) Optical Rendering Accuracy Metrics.: To quantify the accuracy of rendered tactile images, we compute:

- **Mean L2 Norm** (↓): The average pixel-wise L2 norm difference between predicted and ground-truth RGB values, normalized to $[0, 1]$.
- **Significant Pixel L2 Norm** (↓): Similar to the physical Sig. CD, we compute the average L2 norm over the top 1% worst-predicted pixels to highlight regions with high rendering error.
- **Peak Signal-to-Noise Ratio (PSNR)** (↑): A standard perceptual metric that expresses the ratio between the maximum possible pixel value and the reconstruction error. Let $\hat{I}, I \in [0, 1]^{H \times W \times 3}$ denote the predicted and ground-truth RGB images. First compute the mean squared error (MSE) as $\|\hat{I} - I\|_2^2$, and then PSNR is given by:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{1}{\text{MSE}} \right).$$