

Taxim: An Example-based Simulation Model for GelSight Tactile Sensors and its Sim-to-Real Applications

Zilin Si

CMU-RI-TR-22-55

August, 2022



The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:

Wenzhen Yuan, *chair*
Srinivasa G. Narasimhan
Changliu Liu
Arpit Agarwal

*Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics.*

Copyright © 2022 Zilin Si. All rights reserved.

Abstract

Simulation is widely used in robotics for system verification and large-scale data collection. However, simulating a robot system efficiently and with high fidelity, from sensing, perception to manipulation, has been a long-standing challenge. Tactile sensing, as one of the sensing modalities has shown its essential functionality in robotic applications such as manipulation, while its simulation is still under exploration. This thesis contributes an example-based simulation model for GelSight [121], a dense optical tactile sensor and its applications in robotic perception and manipulation with sim-to-real transfer.

First, we propose Taxim [93], a realistic and efficient simulation model for a vision-based tactile sensor, GelSight [121] which measures the contact geometries from images and forces from marker motions: we simulate the optical response to the deformation of contact by mapping the contact geometries to pixel intensity sampled by the embedded camera. We also simulate the surface markers’ motion caused by the surface stretch of the elastomer under contact. To the best of our knowledge, our simulation framework is the first to incorporate *marker motion field simulation* that derives from elastomer deformation together with the *optical simulation*, creating a comprehensive and computationally efficient tactile simulation framework.

Second, we present OBJECTFOLDER 2.0 [37], a large-scale multisensory dataset of common household objects in the form of implicit neural representations from vision, audio and touch where touch representation is learned from virtual objects based on Taxim simulation model. We show that models learned from simulation in our dataset successfully transfer to their real-world counterparts in three perception tasks: object scale estimation, contact localization and shape-reconstruction.

In the end, we integrate simulation of robot dynamics and Taxim simulation for vision-based tactile sensors by modeling the physics of contact. This contact model uses simulated contact forces at the robot’s end-effector to inform the generation of realistic tactile outputs. We demonstrate the effectiveness of our system on a sim-to-real grasp stability prediction task on various objects. Experiments reveal the potential of applying our simulation framework on more complicated manipulation tasks.

Acknowledgments

I would like to first thank my advisor, Prof. Wenzhen Yuan for her encouragement, guidance and support on all of my ways during my Master's. She has been guiding me to grow to a robotics researcher, and without her, I could not have gone forward so far to achieve what I have done today. I also want to thank all my collaborators, especially Suddhu, Arpit, Raj, Ruohan, Dr. Stuart Anderson, Prof. Michael Kaess, Prof. Jiajun Wu, Prof. Jeanette Bohg, Prof. Li Fei-Fei during my research journey. I incredibly appreciate and enjoy the time working with them.

My labmates have been amazing and I am grateful to have them being around to motivate and help each others. I would like to thank the rest of my thesis committees, Prof. Srinivasa G. Narasimhan, Prof. Changliu Liu for their time, feedback, and discussion on this work. I thank Prof. Michael Kaess, Prof. John Dolan, and Rachel Burcin for opening the opportunity for me to join RISS program and do research in RI at the first place.

Lastly, I would like to thank my families and friends as always for their love and support to me. I have such a wonderful life journey with them. I cannot say enough appreciation to Haomin's and Bobo's accompany. They are the best.

I acknowledge the funding support from Facebook AI research for this work.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Contributions	2
2	Background	5
2.1	Tactile Sensor Simulation	5
2.2	Object Dataset with Tactile Sensing	6
2.3	Grasping with Tactile Sensing	8
3	Simulation for GelSight Tactile Sensors	11
3.1	Overview	11
3.2	Optical Simulation	11
3.2.1	Shadow Simulation	15
3.3	Markers' Motion Simulation	16
3.4	Experiments	18
3.4.1	Experiment Setup and Data Collection	18
3.4.2	Optical Simulation	19
3.4.3	Marker Motion Field Simulation	23
4	Sim-to-Real Applications	27
4.1	Multi-sensory Dataset	27
4.1.1	Improved Multisensory Simulation and Implicit Representations	28
4.1.2	Touch	30
4.1.3	Sim2Real Object Transfer	32
4.2	Grasp Stability Estimation	37
4.2.1	Simulation Framework	37
4.2.2	Sim2Real Grasping Prediction	40
4.2.3	Experiments	43
5	Conclusions	49
	Bibliography	51

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

3.1	The pipeline of our proposed example-based simulation model.	12
3.2	(a) Demo of the photometric stereo method: for a surface point p under the light l , the reflected light intensity captured by the camera is determined by the surface reflectance and the surface normal n_p (b) The GelSight sensor [28] we aim to simulate and (c) its schematic diagrams of the optical structure.	12
3.3	Data collection setup (a, b, c) and data examples (d, e) to build the optical simulation model. The GelSight is placed on an XYR optical stage and an indenter with a certain shape object is mounted on a vertical linear stage for precisely indenting on the GelSight. We calibrate the polynomial table with less than 100 data points using a spherical indenter and collect shadow masks with 10 data points using a pin indenter.	14
3.4	Shadow synthesis. (a) A unit shadow case observed under the lighting. (b) We approximate the object as the composition of unit pin case and then attach the shadow caused by each pin. (c), (d) and (e): We collect a set of shadow masks and synthesize the shadow around the contact area.	15
3.5	Elastic deformation calibration and simulation for the gelpad. We calibrate the deformation of gelpad under a unit pin with 0.5 mm diameter indenting in ANSYS to get the dense nodal displacement results (left). In simulation, we compose each node's displacement from elastic deformation in three steps: (a) initial displacement boundary conditions on active nodes, (b) active nodes' virtual displacements, and (c) resultant nodal displacements for both active nodes in contact area and passive nodes in non-contact area.	16
3.6	Dataset of objects designed in Solidworks (a) and 3D printed (b) for contact experiments. The objects are of different shapes. Their base sizes are either 10mm \times 10mm or 15mm \times 15mm.	19
3.7	Optical simulation comparison among our method, TACTO [106], Phong [40] and physics [5] with the real data.	20

3.8	Optical simulation results with different indentation depths and locations. The locations differ over the gelpad surface while the depths differ as 0.5mm, 1.0mm, 1.5mm. The MSE error is shown below each pair.	21
3.9	The GelSight outputs when it contacts objects with rich textures. With the ground-truth geometry [39], our simulation model generates images that are very similar to the real ones.	22
3.10	Optical simulation results (right) for a DIGIT sensor (left).	23
3.11	Optical simulation results for objects from the Google Scan dataset [29]. We touch the objects with a GelSight mounted on a robot arm given certain contact locations. Most artifacts in the simulated images come from the coarse mesh files of the objects.	23
3.12	The simulated marker motion field in a dense mesh in comparison with the FEM data. The heat maps show the marker motion field in X, Y and Z directions where positive Z is the direction of normal loads and X, Y are the direction of shear loads.	24
3.13	Marker motion field simulation with optical simulation results. We visualize the marker motions (scaled up by 20 for better visualization) on the dataset under different normal displacements and shear displacements.	25
4.1	OBJECTFOLDER 2.0 contains 1,000 implicitly represented objects each containing the complete multisensory profile of a real object. We virtualize each object by encoding its intrinsics (texture, material type, and 3D shape) with an <i>Object File</i> implicit neural representation. Then we can render its visual appearance, impact sound, and tactile readings based on any extrinsic parameters. We successfully transfer the models learned from our virtualized objects to three challenging tasks on their real-world counterparts. This opens a new path for multisensory learning in computer vision and robotics, where OBJECTFOLDER 2.0 serves as a rich and realistic object repository for training real-world models.	28
4.2	By querying each <i>Object File</i> implicit neural representation with the corresponding extrinsic parameters we can obtain the multisensory data for the object.	29

4.3	Each <i>Object File</i> implicit neural representation network contains three sub-networks: VisionNet, AudioNet, and TouchNet. Compared with OBJECTFOLDER 1.0, we greatly accelerate VisionNet inference by representing each object with thousands of individual MLPs; for AudioNet, we only predict the parts of the signal that are location-dependent instead of directly predicting the audio spectrograms, which significantly improves the rendering quality and also accelerates inference; our new TouchNet can render tactile readings of varied rotation angles and gel deformations, whereas only a single tactile image can be rendered per vertex in 1.0.	30
4.4	Comparing the visual, acoustic, and tactile data rendered from OBJECTFOLDER 1.0, OBJECTFOLDER 2.0 (Ours), and the corresponding ground-truth simulations for the YCB mug. See Supp. for more examples.	31
4.5	Illustration of real-world objects used in experiments and our hardware set-up for collecting real-world impact sounds and tactile data.	32
4.6	Qualitative results for contact localization with touch readings and impact sounds. Top: in simulation, bottom: real-world experiments. The candidate contact locations are shown as green particles in the particle filter. After several iterations shown from left to right in each row, the green particles converge to the ground-truth contact location shown as the red particle.	35
4.7	Qualitative results for visual-tactile shape reconstruction in simulation (Sim) and real-world (Real) for the square tray and the coffee mug.	36
4.8	Our proposed simulation framework includes physics simulation, contact simulation and tactile simulation. Physics simulator handles the robot dynamics which provide the contact forces and poses. Contact models map them to indentation depths and shear displacements of the contact, and generate the contact map to feed into the tactile simulator. Tactile simulation renders the RGB tactile images.	37
4.9	Linear mapping from the contact force to the soft body deformation. (a) From the experiments we found the linear mapping from the normal force to the indentation volume on soft body, which can be further calculated as the indentation depth. (b) The linear mapping from the shear force to the shear displacement of the contact area.	39
4.10	Grasp pipeline for both simulation and real experiments. We initialize the robot on top of the object, move the gripper down to a preset height, close the gripper with a preset grasping force to grasp the object, and then lift it. We record the tactile readings from a GelSight sensor after grasping.	40

4.11	Grasping configurations including the grasping location, height and force.	40
4.12	Grasp stability prediction networks. (a) We input single tactile images to a feature extractor (CNN) and a classifier (MLP) to predict the grasp results. (b) We input a sequence of tactile images to feature extractors (CNN), a LSTM module and then a classifier (MLP) to predict the grasp results.	41
4.13	We classify grasp as successful grasp (a) and failed grasp (b), (c) including translational and rotational slip.	42
4.14	Examples of tactile readings under different grasping scenarios. Different grasping locations as marked on the object and grasping forces F lead to different grasping outcomes. We show the sequence of the tactile readings during grasping, where geometries of contact can be used to predict the grasping outcomes.	43
4.15	We optimize the friction coefficient of object surface by matching the grasping labels between simulated and real data under the same configuration of grasping heights and forces as shown in (a) and (b). We evaluate possible friction coefficients for several objects and choose the friction values that minimize mislabeling as shown in (c).	44

List of Tables

3.1	Image similarity metrics between simulation and real data for optical simulation. We compare our method with methods from Physics-based model [5], Tacto [106] and Phong’s model [40] on L1, MSE, SSIM and PSNR metrics. Our method performs the best on all the metrics.	20
3.2	Speed test for optical simulation on CPU. We compare our method with the Physics-based model [5], Tacto [106] and Phong’s model [40]. Our method runs with the fastest speed.	22
4.1	Time comparison for rendering one observation sample for each modality, in seconds.	29
4.2	Comparing with OBJECTFOLDER 1.0 on the multisensory data rendering quality. ↓ lower better, ↑ higher better.	30
4.3	Results on object scale prediction. We report the average difference between the predicted and the ground-truth scales of the objects in centimeters.	33
4.4	Results on audio-tactile contact localization. We report the mean distance w.r.t. the ground-truth contact locations in centimeters.	33
4.5	Results on visuo-tactile shape reconstruction. We report the Chamfer-L1 distance w.r.t. the ground-truth meshes in centimeters.	35
4.6	The result of grasp stability prediction. We test the prediction accuracy for both sim-to-sim and sim-to-real transfers with a single tactile image and sequences of tactile images. We compare the performance with TACTO [107].	45
4.7	Ablation study of dataset size, friction coefficient on potted meat can, and center of mass on scissor. 0.45 and -0.03 are the best parameters for friction and center of mass we used in our experiments.	47

Chapter 1

Introduction

1.1 Motivation

Simulation has been widely applied in robotics. It enables roboticists to quickly generate large amounts of realistic data, without costly equipment, manual labour, and the risk associated with real-world experiments. With growing interest in robot simulation, well-developed simulation frameworks such as Gazebo [3], PyBullet [23], MuJoCo [104], Drake [101], SOFA [6], NVIDIA Isaac Gym [68] have been widely used in the robotics community. They can simulate dynamic rigid-body, soft-body, vision and laser sensors with varying levels of accuracy and speed. However, none of them have integrated simulation of tactile sensing which form an irreplaceable part of robotic systems. Tactile sensing provides rich contact information including contact shapes, textures, forces that benefits various perception and manipulation tasks such as shape reconstruction [105], pose estimation [8], grasping [46], slip detection [118] etc. Therefore a simulation system for robots with tactile sensing is on demand to accelerate the application of tactile sensing.

Tactile sensing has been a long-standing research topics not only for human but also for robots, and researchers have explored different functionalities for tactile sensors applied on robots such as piezo-resistive [98], capacitive [67], biomimetic [110] etc. Recent advancements in vision-based tactile sensors, such as GelSight [49], [121], have made high-resolution tactile sensing available. These sensors use a piece of soft elastomer, or *gelpad*, as the contact medium for interacting with the environment.

There is typically a printed marker array on the gelpad surface that moves as the surface stretches and is a good indicator of the contact forces and torques. The sensor utilizes optical components, including LEDs and an embedded camera to capture the illumination change caused by the change of light reflection on the gelpad surface when the sensor contacts an external object, as shown in Fig 3.2. Simulating those vision-based tactile sensors, which contains modeling both the mechanical response of the soft gelpad and the optical response to the deformation, is challenging. There have been previous studies on simulating different components of vision-based tactile sensors separately. For instance, Ding et al. [27] built a physics soft body simulation for the TacTip [109] sensor to indicate pins' motion on the soft membrane; Agarwal et al. [5] and Gomes et al. [40] applied physics-based models for vision-based tactile optical simulation; whereas Wang et al. [106] integrated the optical simulation of tactile sensors with the physics simulation engine PyBullet. However, the work mentioned above lack the ability to simulate the intrinsic noise of the real sensors. They also demand heavy computation while are difficult to generalize to new sensors.

The eventual goal of using simulation is to assist solving problems in real world, therefore sim-to-real transfer tasks make a good indicator to evaluate simulation frameworks. Robotic perception tasks with tactile sensing such as shape mapping [100], pose estimation [107] and edge/surface following [20] only used simulation of tactile readings based on geometry. However, manipulation tasks such as grasping requires not only geometric tactile simulation but also accurate contact dynamics simulation, where the previous simulation frameworks would fail. To this end, we see the demand of an integrated robot simulation framework with modeling dense vision-based tactile sensors and its sim-to-real applications on both perception and manipulation tasks. The availability of such a simulation system can vastly benefit the robotics community to easily get virtual access to the tactile sensors, iterate designs and implement robotic experiments with tactile sensors in simulation to minimize the efforts in the real life.

1.2 Contributions

In this thesis, we present a vision-based tactile sensor simulation model and its sim-to-real applications from two aspects: first in Chapter 3, we present Taxim, an example-based simulation model for GelSight tactile sensors that combines *optical*

simulation for optical response to the gelpad deformation and *marker motion field simulation* for mechanical response to the soft gelpad. Second we present the sim-to-real transfer with Taxim on various perception and manipulation tasks in Chapter 4.

Our simulation model overcomes the constraints of the previous simulation work in that it is computationally lightweight and generates very similar signals to the real sensors in spite of the intrinsic noise of the sensors. Taxim is calibrated with less than 100 contact examples, so that it can easily migrate to other vision-based tactile sensors with similar designs as GelSight. Our main contributions in Chapter 3 are:

1. A polynomial table mapping function to simulate the *optical response* of a GelSight sensor by mapping geometries to pixel intensity in tactile images and an accumulation approach to simulate the shadow caused by the illumination.
2. A model to simulate the *marker motion field* using the linear displacement relationship and the superposition principle for the gelpad’s elastic deformation.

In Chapter 4, we further apply Taxim on various sim-to-real robotic perception and manipulation tasks including object scale prediction, contact localization and shape reconstruction along with a multisensory object dataset 4.1 and grasping 4.2 and show its potentiality to benefit the research with tactile sensing.

In Section 4.1, we introduce a large multisensory dataset of 3D objects in the form of implicit neural representations, which is 10 times larger in scale compared to existing work. We significantly improve the multisensory rendering quality for vision, audio, and touch, while being orders of magnitude faster in rendering time. We also show that learning with our virtualized objects can successfully transfer to a series of real-world tasks, offering a new path and testbed for multisensory learning for robotics.

In Section 4.2, we integrate simulation of robot dynamics and vision-based tactile sensors by modeling the physics of contact. We convert the contact forces to the contact deformation and then integrate the Taxim [93] to improve the tactile simulation system’s fidelity. We leverage our simulation framework on a sim-to-real grasp stability prediction task. We learn a model given tactile images to predict the grasp outcomes completely in simulation and then perform zero-shot sim-to-real transfer to test on tactile images from real grasp experiments. This reveals the potentiality of our simulation system on more complicated manipulation tasks with sim-to-real transfer.

CHAPTER 1. INTRODUCTION

Chapter 2

Background

2.1 Tactile Sensor Simulation

Tactile Sensor Simulation The majority of tactile sensors use a soft medium for contact where the measurement of its deformation can indicate the contact information. Therefore the sensor simulation has been mostly focused on simulating the elastomer deformation. Typically, elastic soft body simulation is modelled with finite element methods (FEM) [64], mass-spring model [48], particles [108] or learning methods [15]. To simulate tactile sensors that use soft medium, a traditional way is to build an approximation model of the soft bodies. Pezzementi et al. [85] and Moio et al. [72] simulated the low dimensional tactile sensor signals with a point spread function model and a soft contact model with a full friction description respectively. However, they are not applicable to the high-resolution vision-based tactile sensors such as GelSight. Narang et al. [75] used a finite element model to simulate the BioTac sensor [115] and contact data in ANSYS. As an extension work, they simulated the deformation of BioTac in Isaac gym [68] and projected this deformation to electronic signal with a generative learning framework in [74]. Sferrazza et al. [90, 91] built a synthetic dataset with a finite element model of the vision-based tactile sensors, trained a network to predict the 3D contact force distribution in simulate and realized sim-to-real transfer. For Taxim, instead of using an accurate finite element model with high-computing cost, we approximate the deformation of the soft medium on the GelSight sensor with pyramid Gaussian kernels which is efficient and also gives

acceptable accuracy.

Optical Simulation for Tactile Sensors For vision-based tactile sensors like GelSight, optical simulation is essential as it is used to measure geometries of the contacted objects. To simulate the optical system of GelSight, Gomes et al. [40] and Hogan et al. [47] used Phong’s model to simulate the reflection and illumination. Agarwal et al. [5] applied ray tracing to simulate the light paths within the sensor that form tactile images. Wang et al. [106] presented TACTO, an open-source simulator using pyrender to simulate DIGIT [55] sensors and bridged it to a physics simulator PyBullet. Compared to those physics-based methods, Taxim is data-driven, so that it is computationally efficient and can better simulate the intrinsic noise of the real sensors.

Marker Motion Field Simulation for Tactile Sensors The movement of marker array on GelSight or other vision-based tactile sensors is caused by the planar stretch of the elastomer surface. They also make the key component of many vision-based tactile sensors such as TacTip [109]. In manipulation tasks such as slip detection [119] and grasping stability prediction [11], the marker motion serves as an essential feature. For TacTip, Ding et al. [27] simulated the dynamics of its soft membrane in Unity so as to extracted markers’ motion. They evaluated the simulation on sim-to-real robot tasks. Church et al. [19] simulated the depth maps to represent the contact geometries instead of optical tactile images. They also used a Generative Adversarial Networks (GANs) to realize real-to-sim translation for TacTip sensors. Unlike the above work, we explicitly simulate the marker motion field by using FEM offline and applying the superposition principle [22] online. Our method does not require extensive training data but only a gelpad FEM model, and it approximates the marker motion field well with high accuracy and low computation cost.

2.2 Object Dataset with Tactile Sensing

Object datasets Objects are modeled in different ways across different datasets. Image datasets such as ImageNet [26] and MS COCO [59] model objects in 2D. Datasets of synthetic 3D CAD models such as ModelNet [112] and ShapeNet [16]

focus on the geometry of objects without modeling their realistic visual textures. Pix3D [99], IKEA Objects [58], and Object3D [114] align 3D CAD models to objects in real images, but they are either limited in size or make unignorable approximations in the 2D-3D alignment. BigBIRD [94] and YCB [13] directly model real-world objects but only for a small number of object instances. ABO [21] was recently introduced, containing 3D models for over 8K objects of real household objects, but it focuses only on the visual modality, similar to the other datasets above.

Alternatively, OBJECTFOLDER 2.0 [37] contains 1,000 3D objects in the form of implicit neural representations, each of which encodes realistic visual, acoustic, and tactile sensory data for the corresponding object. Compared to OBJECTFOLDER 1.0 [36], our dataset is not only 10 times larger in the amount of objects, but also we significantly improve the quality of the multisensory data while being 100 times faster in rendering time. Furthermore, while OBJECTFOLDER 1.0 only performs tasks in simulation, we show that learning with our virtualized objects generalizes to the objects’ real-world counterparts.

Implicit neural representations Coordinate-based multi-layer perceptrons (MLPs) have attracted much attention lately and have been used as a new way to parameterize different types of natural signals. They are used to learn priors over shapes [17, 70, 83]; represent the appearance of static scenes [71, 95], dynamic scenes [78, 84], or individual objects [41, 77]; and even encode other non-visual modalities such as wavefields, sounds, and tactile signals [36, 96].

We also use MLPs to encode object-centric visual, acoustic, and tactile data similar to [36], but our new object-centric implicit neural representations encode the intrinsics of objects more realistically and flexibly. Furthermore, inspired by recent techniques [38, 44, 60, 62, 76, 87, 116] on speeding up neural volume rendering [51], we largely reduce the rendering time of visual appearance, making inference of all sensory modalities real-time.

Multisensory learning A growing body of work leverages other sensory modalities as learning signals in addition to vision, with audio and touch being the most popular. For audio-visual learning, inspiring recent work integrates sound and vision for a series of interesting tasks, including self-supervised representation learning [53, 79, 80], audio-

visual source separation [31, 32, 34, 122], sound localization in video frames [89, 103], visually-guided audio generation [33, 73], and action recognition [35, 113]. For visuo-tactile learning, the two sensory modalities are used for cross-modal prediction [57] and representation learning [56, 86]. Touch is also used to augment vision for 3D shape reconstruction [97, 100], robotic grasping [11, 12], and object contact localization [63]. Earlier work on modeling multisensory physical behavior of 3D objects [82] proposes a system to directly measure contact textures and sounds, but mainly for the purpose of better modeling virtual object interaction and creating animations.

OBJECTFOLDER 2.0 [37] is a potential testbed for various multisensory learning tasks involving all three modalities. Different from the works above, instead of learning with certain sensory modalities for a particular task, our goal is to introduce a dataset of implicitly represented objects with realistic visual, acoustic, and tactile sensory data, making multisensory learning easily accessible to the computer vision and robotics community.

2.3 Grasping with Tactile Sensing

Grasping with tactile sensing Tactile sensing has been widely applied to grasping tasks given their rich contact information. Schill et al. [88] learnt a classifier from 6 tactile sensors on a robot hand to continuously estimate the grasp stability during the grasp until reach the stable grasp. Bekiroglu et al. [10] learnt a latent variable probabilistic model from vision, tactile and action parameters. They used the conditional of this model to estimate the grasp stability. Calandra et al. [11] showed that visuo-tactile deep neural network model can improve the ability to predict the grasp. And furthermore, the author proposed an action-conditional model to learn the regrasping policies from visual and tactile sensing in [12]. Note in all the above cases, a data collection stage in the real world was performed to build a learning model by grasping various objects. This process is time consuming and requires a human in the loop to reset the moving objects during grasping.

Alternatively, to overcome the limitation in the data size, an exploration-based grasping method with visual and tactile data was developed and tested in the simulation [25]. Bekiroglu et al. [9] learnt a probabilistic model of grasp stability, given tactile signal, joint configuration of the hand, object shape class and approach

vector of the hand. They trained and tested the model on both simulated and real data but without sim-to-real transfer. Hogan et al. [46] proposed a grasp quality metric based on tactile images and used the simulated regrasp candidates along with their tactile images to search for grasp adjustment. To predict the grasp stability given tactile images, our solution is to train completely in simulation, given a high fidelity simulation system and test directly on a real robot.

Simulation of Robot with Tactile Sensors There are a lot of physics engines available to robotics practitioners [30] which allow full robot simulation with multiple sensing modalities. However, tactile simulation is limited in those simulators as deformable soft surfaces in tactile sensors are hard to simulate accurately and efficiently. Due to above, tactile simulation is still a challenging research area. Existing work [74, 75, 85, 90] build the mechanics models to simulate the soft body deformation for tactile sensors. For vision-based tactile sensors like GelSight [120], optical simulation [5, 40, 47, 93] is also essential as it is used to measure shapes of objects in contact.

To integrate the tactile simulation with physics simulation for manipulation tasks, Moision et al. [72] simulated a low-resolution tactile sensor considering soft contacts and full friction description and applied it on grasping tasks. Kappassov et al. [52] presented a tactile simulation framework for tactile arrays in Gazebo simulation environment considering the effect of contact forces and showed tactile servoing applications. Wang et al. [107] presented TACTO, a simulator using pyrender to simulate optical tactile sensors and combined it to a physics simulator PyBullet. Compared to those simulation frameworks, we combine the simulation of contact physics including forces, frictions and slips with the vision-based tactile simulation of contact shapes, which allows the potential to simulate the more complicated grasping scenarios with slip and efficient sim-to-real transfer.

Sim2Real Learning The control policies learned in simulation can be applied to real robots by framing the problem as transfer learning between the data distribution of simulation and real worlds. Dang et al. [24] presented a learning approach to estimate the grasp stability and make hand adjustments based on low-resolution tactile sensing data from robot hand. They realized sim-to-real transfer but their testing objects are rather limited in number and have simpler shapes. In [20], the authors

CHAPTER 2. BACKGROUND

proposed an image-conditioned generator network that translates between real and simulated images. They used this network to transfer policies trained in simulation for various tactile manipulation tasks. In [111], the authors set up two multi-fingered hands with tactile sensors in PyBullet, used it to learn a reinforcement learning policy for grasping and transferred the learnt policy to real world environments. Mahler et al. [65] considered the problem of bin picking by simulating quasi-static physics in PyBullet with a parallel-jaw gripper. They posed the sim2real task as a transferring GQ-CNN features between simulation and real world data from Dex-Net 2.0 dataset [66]. We train a grasp stability prediction model based on dense vision-based tactile images in our simulation framework and show that the model can be successfully transferred to the real data without any explicitly transfer step.

Chapter 3

Simulation for GelSight Tactile Sensors

3.1 Overview

We construct and employ our simulation models for both optical response and marker motion of GelSight sensors. To simulate the sensor’s optical response, we build a polynomial table to map the contact geometries to the image intensities and collect shadow masks to attach the shadow. Then we apply the superposition principle based on the loading displacement of each finite unit to simulate the markers’ motion. Their combination replicates the contacting of objects on the tactile sensor. For both parts, we calibrate our simulator with examples from a real sensor. We show the pipeline for building and applying the simulation model in Fig. 3.1.

3.2 Optical Simulation

We simulate the optical response of the GelSight sensor as a result of the contact geometry with a model using the examples-based photometric stereo [45] method. Photometric stereo uses the linear reflection function to derive the illumination of the object with the light sources and shape of the illuminated surface. Example-based photometric stereo does not require prior knowledge of lights sources but instead

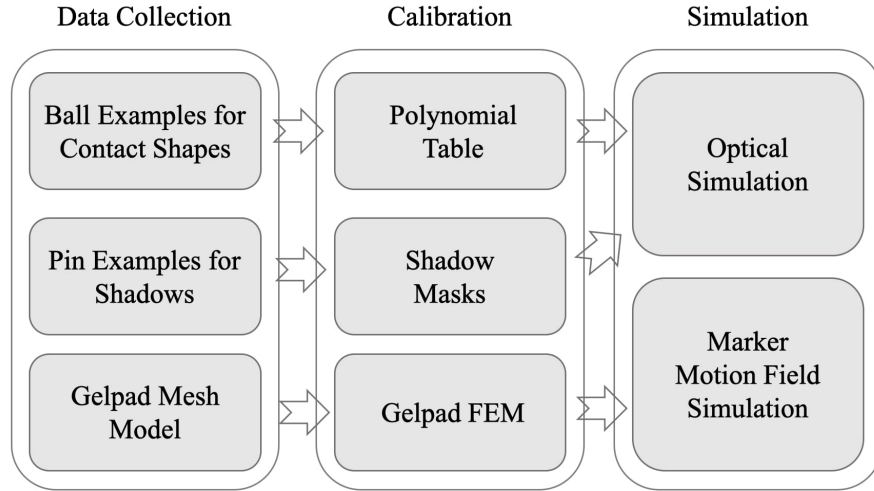


Figure 3.1: The pipeline of our proposed example-based simulation model.

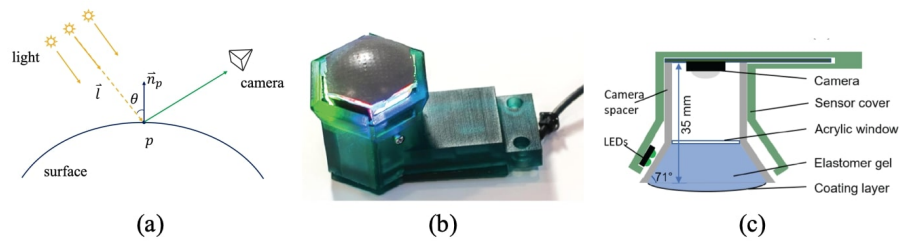


Figure 3.2: (a) Demo of the photometric stereo method: for a surface point p under the light l , the reflected light intensity captured by the camera is determined by the surface reflectance and the surface normal n_p (b) The GelSight sensor [28] we aim to simulate and (c) its schematic diagrams of the optical structure.

uses the imaging of the reference objects. We use a lookup table as the baseline and a polynomial table as our proposed method to map the contact shapes to image intensities.

Lookup Table Mapping The gelpad has a homogeneous diffuse internal surface which makes the reflection function spatial-invariant. The linear reflection function used by photometric stereo is formalized as $I_p = \rho \mathbf{n}_p \cdot \mathbf{l}$, where at a point p , the observed light intensity I_p caused by reflection is a product of the albedo ρ , the surface normal \mathbf{n}_p and the light direction and intensity \mathbf{l} , as shown in Fig. 3.2 (a).

The previous equation implies that the reflected light intensity I and the surface normal \mathbf{n} are linearly correlated. Alternatively, instead of solving the equation from the given lighting conditions, an intensity-shape lookup table can be built as follows

$$I = \sum_l a^l \mathbf{n} \quad (3.1)$$

where a^l is the coefficient of the light l , and can be derived from a calibration process similar to [49, 50]. Here, we define the lights of the same color—any of red, green or blue—as one light source, even though they are contributed by multiple LEDs.

Polynomial Table Mapping The linear lookup table works well with the point lights which are far from the object, whose directions are parallel and intensities are uniform for all the points on the illuminated surface. However, the LEDs in the GelSight are close to the sensor surface so that the emitted light is not strictly parallel and uniform. To compensate for the complicated lighting conditions, we introduce a non-linear model for the reflection as proposed in [7]. The reflection function can then be rewritten as

$$I = \sum_l f_{\mathbf{n}}^l(x, y) \quad (3.2)$$

where \mathbf{n} is the normal vector representative of the surface shape and (x, y) is the 2D location on the image plane.

From experiments, we found that in practice a second order polynomial function is sufficient to approximate the non-linearity. Thus, the non-linear function is represented as:

$$f_{\mathbf{n}}^l(x, y) = \mathbf{w}_{\mathbf{n}}^l \mathbf{b} \quad (3.3)$$

where $\mathbf{w}_{\mathbf{n}}^l$ is a 1×6 vector that represents the parameters to model the polynomial table, and $\mathbf{b} = [x^2, y^2, xy, x, y, 1]^T$.

Calibration Calibration entails fitting the parameters in the polynomial table from real data. Since these parameters vary for different sensors, this process has to be done per sensor. During calibration, we press a small ball with a known radius over the surface and manually locate contact areas in the tactile images as shown in Fig 3.3 (b) and (d). The surface normal at each point in the circular area can be easily

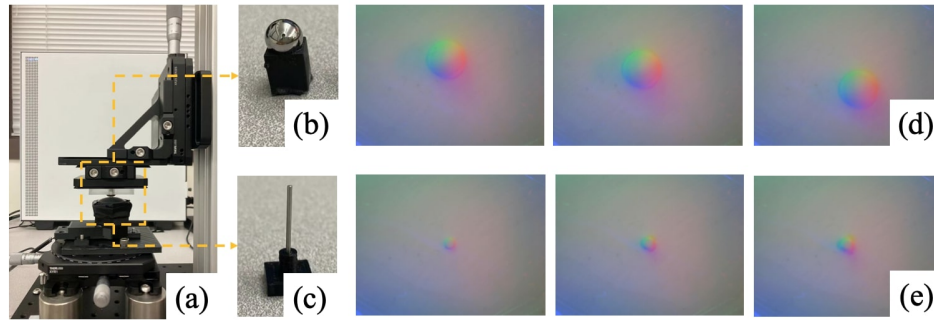


Figure 3.3: Data collection setup (a, b, c) and data examples (d, e) to build the optical simulation model. The GelSight is placed on an XYR optical stage and an indenter with a certain shape object is mounted on a vertical linear stage for precisely indenting on the GelSight. We calibrate the polynomial table with less than 100 data points using a spherical indenter and collect shadow masks with 10 data points using a pin indenter.

calculated based on the ball’s geometry. We discretize the 3D surface normal vectors to a 125×125 table with the magnitude and direction of the surface normal as the two dimensions. The parameters in polynomial table can then be solved via least squares with the set of intensity-shape-location pairs $(I_p, \mathbf{n}_p, x_p, y_p)$ from collected data. We fill invalid values in the table by interpolation.

Simulation We simulate the visual outputs in three steps: collision detection, deformation approximation, and optical simulation. A collision is detected when an object comes in contact with the gelpad. From this contact, the local shape, represented as a height map, is constructed from the object’s shape in the contact area, and gelpad’s shape in non-contact area. Additionally, we need to simulate the soft body deformation from the height map. An approximation of soft body simulation is applied with pyramid Gaussian kernels. The shape in contact area is kept unchanged to maintain the fine textures and the boundaries between the contact and non-contact areas are smoothed using pyramid Gaussian kernels from large to small. From the height map, the normal vector for each point can be extracted and mapped to an intensity value with the calibrated polynomial table to synthesize the tactile images.

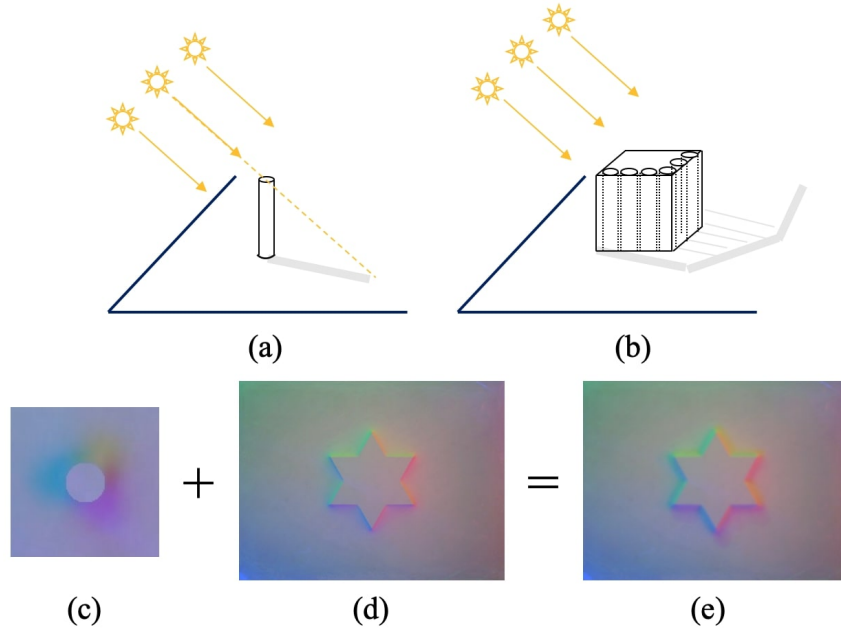


Figure 3.4: Shadow synthesis. (a) A unit shadow case observed under the lighting. (b) We approximate the object as the composition of unit pin case and then attach the shadow caused by each pin. (c), (d) and (e): We collect a set of shadow masks and synthesize the shadow around the contact area.

3.2.1 Shadow Simulation

Other than the illumination change that is modeled with photometric stereo method, the shadow is another factor causing the change of the pixel intensity in the tactile images. According to the design of the sensor, the shadows are caused by three groups of LED lights: red, green and blue lights. We simulate the shadow from those light sources respectively. We simplify shadow casting by collecting the “unit” shadow case, and then simulate the shadow by accumulating the shadows caused by each geometrical “unit”. Since each light beam is traveling independently in the space, without considering inter-reflection, the shadow cast by them can be linearly accumulated.

A “unit” shadow is the shadow cast by a standing pin, as shown in Fig. 3.4 (a). For objects with different geometries, we consider them as the accumulation of “unit” shadows placed side by side with different heights, as illustrated in Fig. 3.4 (b). Therefore, given the tactile images with shadow cast by indenting a pin normally

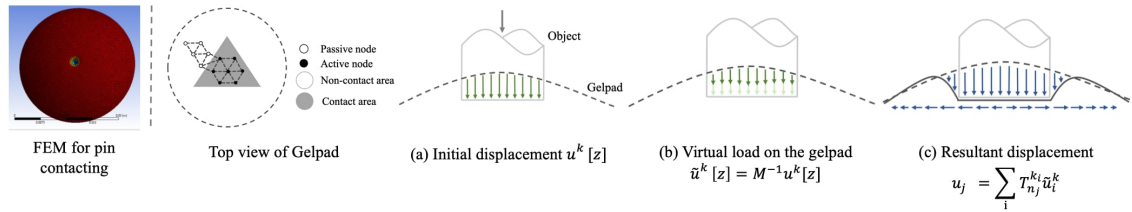


Figure 3.5: Elastic deformation calibration and simulation for the gelpad. We calibrate the deformation of gelpad under a unit pin with 0.5 mm diameter indenting in ANSYS to get the dense nodal displacement results (left). In simulation, we compose each node’s displacement from elastic deformation in three steps: (a) initial displacement boundary conditions on active nodes, (b) active nodes’ virtual displacements, and (c) resultant nodal displacements for both active nodes in contact area and passive nodes in non-contact area.

onto the gelpad with different depths as shown in Fig. 3.3 (c) and (e), we extract a set of shadow masks on three dominant directions caused by three light sources. For a general case, the shadow mask is attached for all three color channels and all points within the contact area if the neighbors are lower than that point.

3.3 Markers’ Motion Simulation

We simulate the markers’ motion on the gelpad surface caused by the deformation of the soft gelpad from contacting. In this work, we consider the deformation under normal and shear loads. We employ the linear displacement relationship and superposition principle [22] to compose the deformation of the surface with loads on each finite unit of the contact surface. Although the markers are sparsely spread on the surface, we mesh the surface with dense nodes in simulation, track each node’s motion and then locate the markers’ motions. With the dense solution, the method can be applied to markers at any locations. Nodes in the gelpad surface mesh are classified into two categories: *active nodes* who come in contact with the object and are applied external forces and constrained by internal elastic forces; *passive nodes* who are in non-contacted area and only constrained by the internal elastic forces.

The linear displacement relationship assumes that any two nodes can influence each other in a linear way. By considering two nodes n_i and n_j with displacement in 3D as \mathbf{u}_i and \mathbf{u}_j , the n_j can be passively influenced by n_i as $\mathbf{u}_j = T_{n_j}^{n_i} \mathbf{u}_i$, where $T_{n_j}^{n_i}$ is

a 3×3 tensor representing the mutual influence. The superposition principle states that a node n_i 's displacement \mathbf{u}_i is an aggregation of all active nodes' influence to it. Assume we have active nodes $K = \{k_1, k_2, k_3, \dots, k_m\}$, where \mathbf{u}_i^k represent active nodes' initial displacement under external loads. Under the linear displacement relationship and superposition principle, any node n_j 's displacement \mathbf{u}_j can be composed as

$$\mathbf{u}_j = \sum_{i=1}^m T_{n_j}^{k_i} \mathbf{u}_i^k \quad (3.4)$$

However, before applying the superposition principle by using the active nodes' initial displacement \mathbf{u}_i^k , we need to amend them to virtual displacements under the virtual loads because they not only are constrained by external loads but also influence each other. For instance, if all the active nodes' displacements are initialized such that they only move along the z direction *i.e.* $\mathbf{u}^k = [0, 0, dz]$, the following equation holds:

$$\mathbf{u}_j^k[z] = \sum_{i=1}^m T_{k_j}^{k_i}[3, 3] \tilde{\mathbf{u}}_i^k[z] \quad (3.5)$$

where \mathbf{u}_j^k is the initialized displacement, and $\mathbf{u}_j^k[z]$ is its component along z -direction; $\tilde{\mathbf{u}}_i^k$ is the virtual displacement under the virtual load; $T_{k_j}^{k_i}[3, 3]$ is the last element in the tensor $T_{k_j}^{k_i}$. Therefore, it is able to solve virtual displacements for active nodes by stacking all the equations as:

$$\begin{aligned} \mathbf{u}^k[z] &= M_z \tilde{\mathbf{u}}^k[z] \\ \begin{bmatrix} \mathbf{u}_1^k[z] \\ \mathbf{u}_2^k[z] \\ \dots \\ \mathbf{u}_m^k[z] \end{bmatrix} &= \begin{bmatrix} T_1^1[3, 3] & T_2^1[3, 3] & \dots & T_m^1[3, 3] \\ T_1^2[3, 3] & T_2^2[3, 3] & \dots & T_m^2[3, 3] \\ \dots & \dots & \dots & \dots \\ T_1^m[3, 3] & T_2^m[3, 3] & \dots & T_m^m[3, 3] \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_1^k[z] \\ \tilde{\mathbf{u}}_2^k[z] \\ \dots \\ \tilde{\mathbf{u}}_m^k[z] \end{bmatrix} \end{aligned} \quad (3.6)$$

Then $\tilde{\mathbf{u}}^k[z]$ is solved by matrix inversion as $\tilde{\mathbf{u}}^k[z] = M_z^{-1} \mathbf{u}^k[z]$.

The x , y components of the active nodes' displacement can be amended using the same approach, but with $T[1, 1]$ or $T[2, 2]$ for the x or y directions respectively. Later, we apply the superposition principle to get the final resultant displacements

for all nodes with

$$\mathbf{u}_j = \sum_i T_{n_j}^{k_i} \tilde{\mathbf{u}}_i^k \quad (3.7)$$

Calibration The tensor $T_{n_j}^{k_i}$ depends on the gelpad’s physical properties, and therefore can be measured in advance. The markers on the real gelpad are sparsely distributed which can not be used to generate dense meshes. Instead, we calibrate the arbitrary $T_{n_j}^{k_i}$ in a Finite Element Method (FEM) software ANSYS. In ANSYS, we generate the dense mesh of the gelpad and measure the deformation when there is a load on a unit node, as shown in Fig. 3.5 (left). We then use the measurement of the deformation to calibrate T . Since the markers are not printed on the top surface of the gelpad, we extract the second layer’s mesh which is 0.5mm below the top surface from the simulated model as reference. To fully calibrate the 3×3 tensors, we simulate an active node’s motion in z-direction only, a combination of z-direction and x-direction and a combination of z-direction and y-direction. Then we solve all the tensors $T_{n_j}^{k_i}$ using least squares from these three sets of unit case.

Simulation We employ the marker motion simulation in three steps, by: 1) applying the initial displacements on the active nodes under the external loads, 2) getting active nodes’ virtual displacements with the superposition principle, and then 3) calculating the resultant displacements at each node using the superposition principle with virtual displacements of active nodes. This process is demonstrated in Fig. 3.5 (a), (b), (c).

3.4 Experiments

We perform a set of experiments to evaluate the similarity between the simulated tactile data and that from real sensors.

3.4.1 Experiment Setup and Data Collection

To collect well-controlled contact data with a real GelSight sensor, We set up an optical platform, as shown in Fig. 3.3 (a). The GelSight is placed on a XYR stage, and an indenter is mounted on a vertical linear stage positioned above the GelSight.

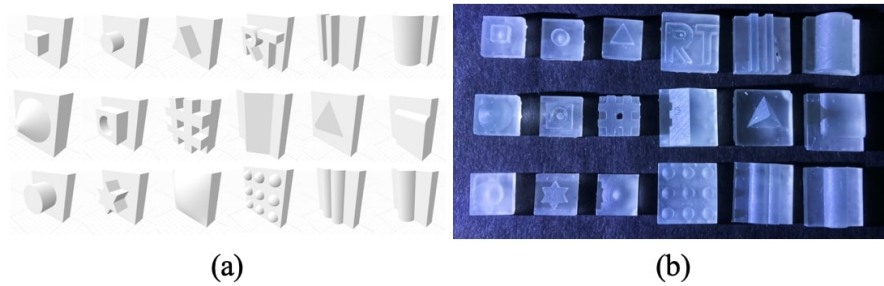


Figure 3.6: Dataset of objects designed in Solidworks (a) and 3D printed (b) for contact experiments. The objects are of different shapes. Their base sizes are either $10\text{mm} \times 10\text{mm}$ or $15\text{mm} \times 15\text{mm}$.

We manually control the contact location and depth by adjusting the stages. The XYR stage enables horizontal movement and the vertical stage adjusts the indenting depth. Both are with 0.01mm precision. We use a dome-shaped gelpad for both the real sensor and the simulated sensor.

We evaluate our simulation using objects with different shapes and textures. The objects are designed in Solidworks [4], output as mesh files for simulation (Fig. 3.6 (a)) and 3D printed for collecting data from the real sensor for comparison (Fig. 3.6 (b)).

3.4.2 Optical Simulation

To calibrate the optical simulation model, we collect 50 data points on different locations of gelpad surface with a 4mm -diameter spherical indenter, as shown in Fig. 3.3 (b), (d); to calibrate the shadow simulation model, we collect 10 data points of different pressing depths with a 1mm -diameter pin indenter, as shown in Fig. 3.3 (c), (e). The calibration process is simple and easy to conduct manually without precise control of contact locations which can be accomplished within 1 hour. And it can be used until any components of the sensor are replaced or the sensor is broken.

We simulate the tactile images on the aforementioned dataset and compare our method with three other methods: the physics-based model [5], TACTO [106] and Phong’s model [40] as shown in Fig. 3.7. We evaluate our method by comparing the simulated images with the real images in pixel-wise level, against the three methods mentioned above on four metrics: mean absolute error (L1), mean squared error



Figure 3.7: Optical simulation comparison among our method, TACTO [106], Phong [40] and physics [5] with the real data.

(MSE), structural index similarity (SSIM) and peak signal-to-noise ratio (PSNR). The simulated images are cropped to size 400×400 around the indenting area to eliminate the background’s effect. Also, due to the precision of the operation with the real sensor, the ground truth tactile images are not well aligned with the simulated images. So we manually align the images using GIMP [102]. The quantitative results are summarized in the Table. 3.1. From the table, our method outperforms all the other methods.

Different indentation depths and locations Our optical simulation model

	L1 ↓	MSE ↓	SSIM ↑	PSNR ↑
Tacto [106]	10.861	215.861	0.808	25.495
Phong’s [40]	8.163	123.249	0.832	27.763
Physics [5]	7.409	90.623	0.759	28.687
Ours	5.565	58.358	0.882	30.974

Table 3.1: Image similarity metrics between simulation and real data for optical simulation. We compare our method with methods from Physics-based model [5], Tacto [106] and Phong’s model [40] on L1, MSE, SSIM and PSNR metrics. Our method performs the best on all the metrics.

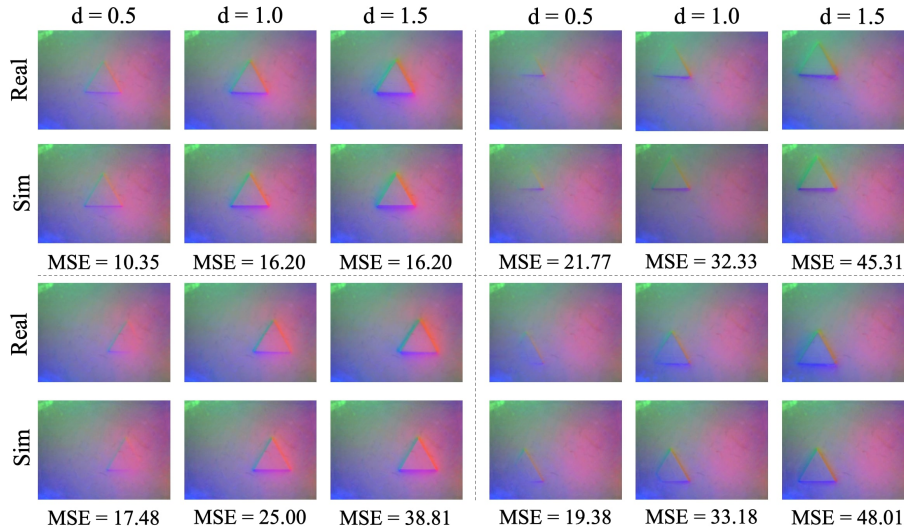


Figure 3.8: Optical simulation results with different indentation depths and locations. The locations differ over the gelpad surface while the depths differ as 0.5mm, 1.0mm, 1.5mm. The MSE error is shown below each pair.

works well for different indentation depths and locations. One example is shown in Fig. 3.8. From MSE errors, we can see errors increase when the indentation become farther from the center and deeper. This comes from the less calibration data in areas far from the center and the approximation model of soft body deformation becomes less effective for large displacements.

Fine texture simulation Our model can simulate the contact cases with fine-textured objects, as shown in Fig. 3.9.

Simulation on various sensors and objects Note that tactile images look different in Fig. 3.7, Fig. 3.8, Fig. 3.11 and Fig. 3.13. This is because we use 4 different GelSight sensors and manufactures lead to the difference. However, our model works well on all of them. We also apply our model on a DIGIT sensor [55] and the results are shown in Fig. 3.10. In addition, We test our model on various objects from the Google Scan dataset [29] and some results are shown in Fig. 3.11.

Speed test We test all the simulation techniques, mentioned above, on a AMD Ryzen Threadripper 2950X 16-Core Processor CPU. We input height maps with the size 480×640 , and output the simulated tactile images of the dataset. We then record the average running time of all the methods, as shown in the Table 3.2. Our method is the most computationally lightweight on CPU and achieves the real-time

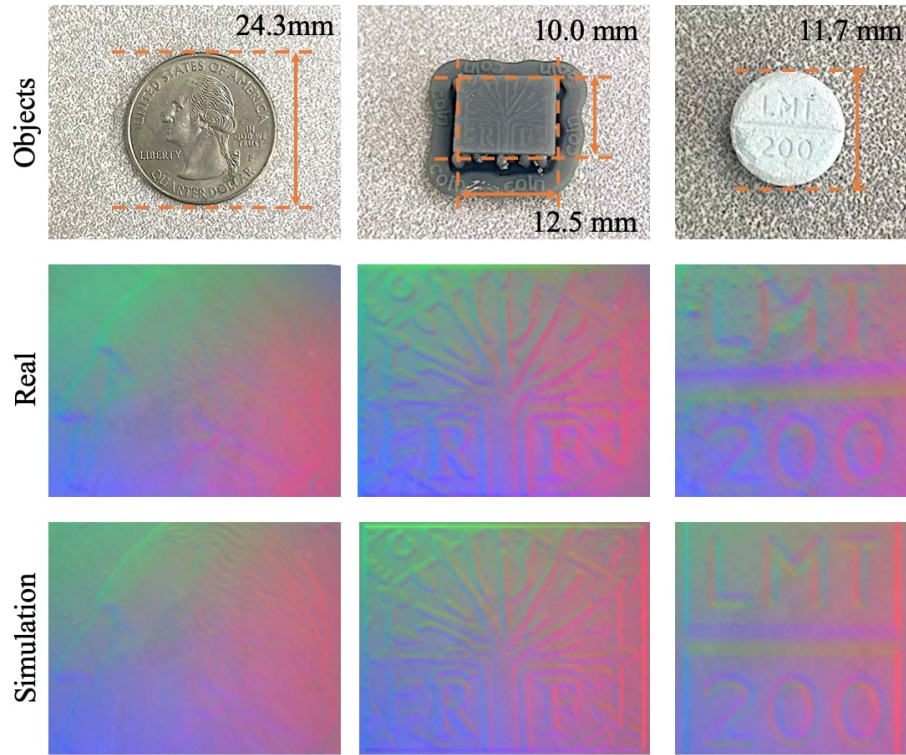


Figure 3.9: The GelSight outputs when it contacts objects with rich textures. With the ground-truth geometry [39], our simulation model generates images that are very similar to the real ones.

data transferring speed from real sensors. However, Tacto [106] and Physics model [5] can be largely accelerated on GPUs but not considered here for evaluation. Our method can be potentially optimized for GPU computation as well and we will work on that for the next step.

	Ours w/o shadows	Ours w/ shadows	Physics [5]	Tacto [106]	Phong's [40]
Speed (fps)	18.1	9.6	0.1	1.9	3.8

Table 3.2: Speed test for optical simulation on CPU. We compare our method with the Physics-based model [5], Tacto [106] and Phong's model [40]. Our method runs with the fastest speed.

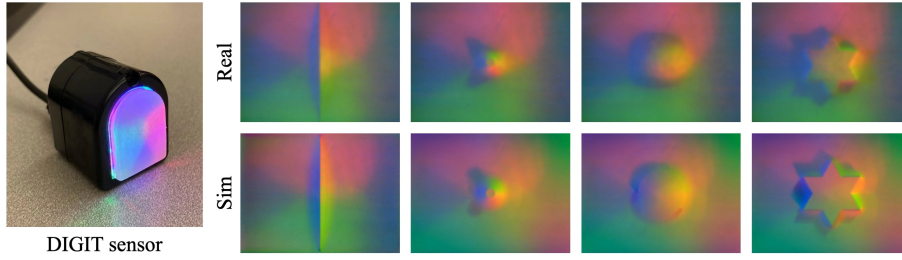


Figure 3.10: Optical simulation results (right) for a DIGIT sensor (left).

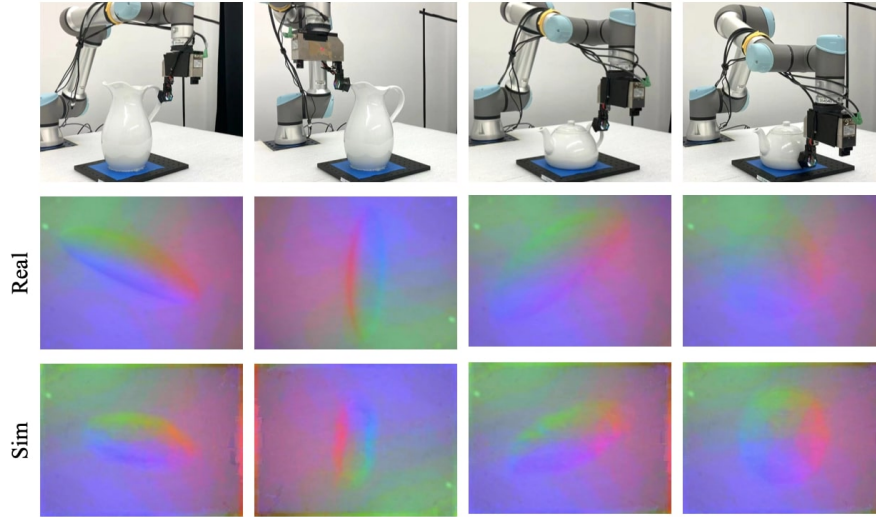


Figure 3.11: Optical simulation results for objects from the Google Scan dataset [29]. We touch the objects with a GelSight mounted on a robot arm given certain contact locations. Most artifacts in the simulated images come from the coarse mesh files of the objects.

3.4.3 Marker Motion Field Simulation

We evaluate the simulation results with two references: 1) the dense displacement map generated by the FEM simulation, and 2) the sparse displacement map collected from a real sensor.

The contact cases are with objects in Fig. 3.6 under combinations of different normal loads and shear loads. The load displacement varies from 0.3 mm to 0.8 mm.

Comparison with FEM simulation As illustrated in the four sets of comparison from Fig. 3.12, the dense mesh vertices displacements on X, Y, Z are simulated from both the FEM (Fig. 3.12 (b) left) and our methods (Fig. 3.12 (b) right). The

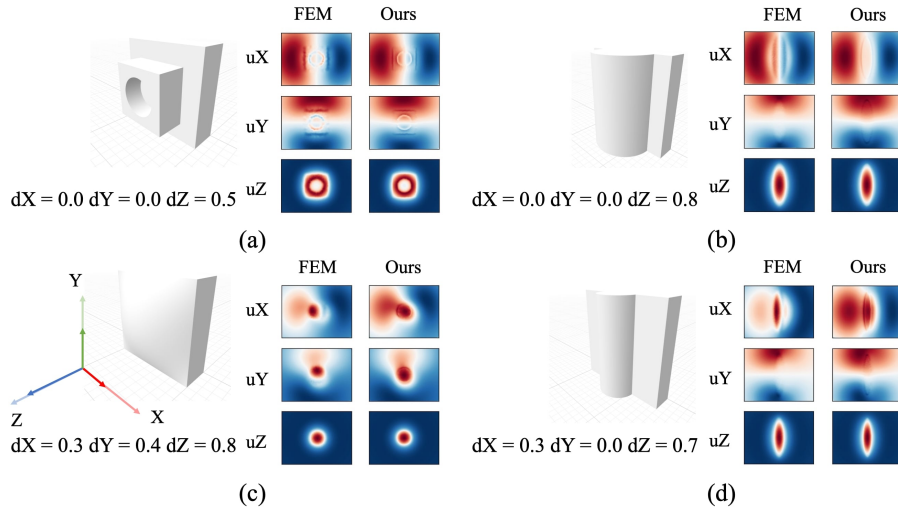


Figure 3.12: The simulated marker motion field in a dense mesh in comparison with the FEM data. The heat maps show the marker motion field in X, Y and Z directions where positive Z is the direction of normal loads and X, Y are the direction of shear loads.

color red means negative displacement value and blue means positive values. The average interpolated pixel-wised L1 errors over the gelpad surface on dataset are 3.58×10^{-3} mm for X-axis, 3.32×10^{-3} mm for Y-axis, 5.43×10^{-3} mm for Z-axis, and 5.40×10^{-3} mm for XY (gelpad surface).

Comparison with both real data and FEM simulation Examples of results are shown in Fig. 3.13. The mean of marker motion’s magnitude L1 errors on dataset is 1.00×10^{-2} mm between real & FEM, 1.02×10^{-2} mm between real & ours, and 3.96×10^{-3} mm between FEM & ours. We weight the marker motion’s angular errors based on the its magnitude because smaller marker motion is easier being affected by the system noise. The weighted mean of marker motion’s angular L1 errors is 12.94° between real & FEM, 14.57° between real & ours, and 4.89° between FEM & ours. From the experimental results, the FEM model and our model match well, but there is still a gap from the simulation to the real gelpad soft body model. Three reasons observed from the experiments causing the errors are: 1) The gelpad is hand-manufactured and it is not perfectly matched with the FEM model in ANSYS. 2) The marker motions tracked from the real sensor’s data have the noise in marker extraction and tracking. 3) When shear loads are present, our model cannot model

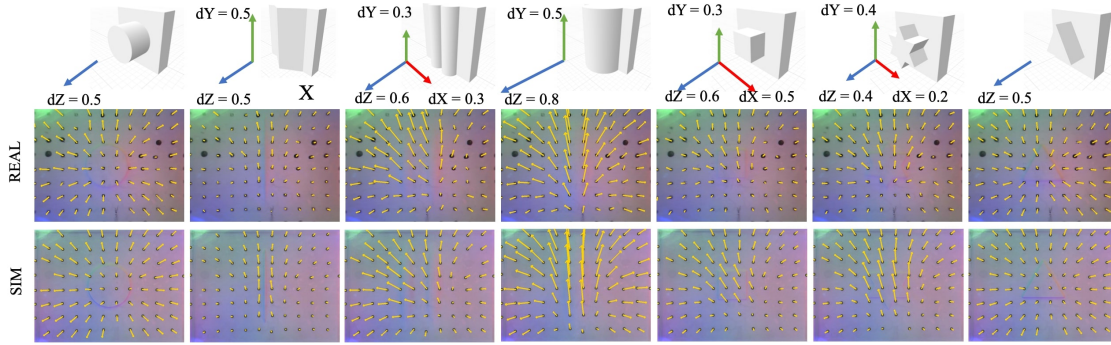


Figure 3.13: Marker motion field simulation with optical simulation results. We visualize the marker motions (scaled up by 20 for better visualization) on the dataset under different normal displacements and shear displacements.

the partial slip but it is very common for the real contact cases.

Speed Testing Our dense marker motion field simulation runs 9.22 seconds on average tested with CPU only. The FEM simulation in ANSYS with CPU costs 2 to 4 hrs for difference cases. In addition, according to Narang et al. [74]’s 5.57 seconds per sim for BioTac sensor in Isaac Gym with GPU acceleration, our simulation has a reasonable low computing demand.

We show some final results that combine the optical simulation and marker motion simulation in Fig. 3.13.

Chapter 4

Sim-to-Real Applications

4.1 Multi-sensory Dataset

OBJECTFOLDER 2.0 [37] contains 1,000 3D objects in the form of implicit neural representations. Among the 1,000 objects, we use all 100 objects from OBJECTFOLDER 1.0 [36], which consists of high quality 3D objects from 3D Model Haven [1], YCB [13], and Google Scanned Objects [29]. The recently introduced ABO dataset [21] is another rich repository of real-world 3D objects, containing about 8K object models with high-quality 3D meshes, which come from Amazon.com product listings. For each object, we obtain metadata such as category, material, color, and dimensions on the real product’s publicly available webpage. We filter the dataset by material type and only keep objects of the following materials: ceramic, glass, wood, plastic, iron, polycarbonate, and steel. We visually inspect each object’s product images to make sure the metadata is correct and keep the object if its material property is approximately homogeneous. These steps ensure that the selected objects are acoustically simulatable as will be described in Sec. ???. In the end, we obtain 855 objects from the ABO dataset. Additionally, we obtain 45 objects of polycarbonate material type from Google Scanned Objects.

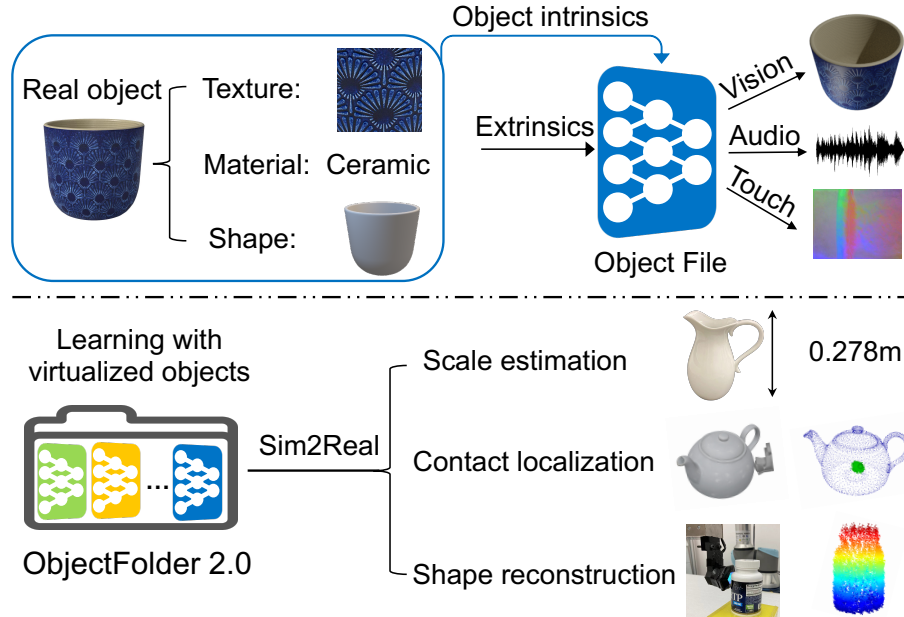


Figure 4.1: OBJECTFOLDER 2.0 contains 1,000 implicitly represented objects each containing the complete multisensory profile of a real object. We virtualize each object by encoding its intrinsic (texture, material type, and 3D shape) with an *Object File* implicit neural representation. Then we can render its visual appearance, impact sound, and tactile readings based on any extrinsic parameters. We successfully transfer the models learned from our virtualized objects to three challenging tasks on their real-world counterparts. This opens a new path for multisensory learning in computer vision and robotics, where OBJECTFOLDER 2.0 serves as a rich and realistic object repository for training real-world models.

4.1.1 Improved Multisensory Simulation and Implicit Representations

We propose a new simulation pipeline to obtain the multisensory data based on the objects’ physical properties. Each object is represented by an *Object File*, which is an implicit neural representation network that encodes the complete multisensory profile of the object. See Fig. 4.1. Implicit representations have many advantages compared to conventional signal representations, which are usually discrete. We can parameterize each sensory modality as a continuous function that maps from some extrinsic parameters (e.g., camera view point and lighting conditions for vision,

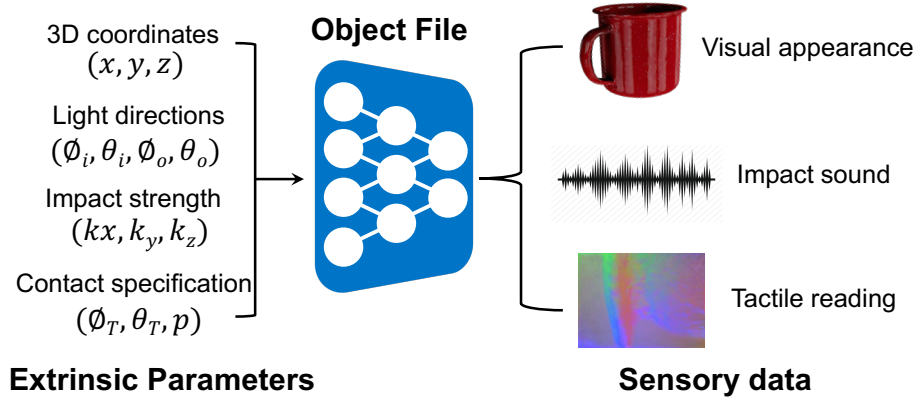


Figure 4.2: By querying each *Object File* implicit neural representation with the corresponding extrinsic parameters we can obtain the multisensory data for the object.

	Vision	Audio	Touch	Total
OBJECTFOLDER 1.0 [36]	3.699	0.420	0.010	4.129
OBJECTFOLDER 2.0 (Ours)	0.062	0.035	0.014	0.111

Table 4.1: Time comparison for rendering one observation sample for each modality, in seconds.

impact strength for audio, gel deformation for touch) to the corresponding sensory signal at a certain location or condition. Implicit neural representations serve as an approximation to this continuous function via a neural network. This makes the memory required to store the original sensory data independent of those extrinsic parameters, allowing the implicit representations to be easily streamed to users. Furthermore, thanks to the continuous property of implicit neural representations, the sensory data can be sampled at arbitrary resolutions.

Each *Object File* has three sub-networks: VisionNet, AudioNet, and TouchNet (see Fig. 4.12). For VisionNet and AudioNet, we would refer the readers to OBJECTFOLDER 2.0 [37] for more details, and we present the details of how we simulate the tactile modality and how we use multi-layer perceptrons (MLPs) to encode the data in this thesis.

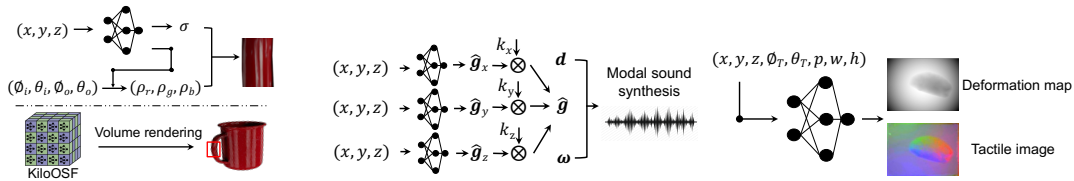


Figure 4.3: Each *Object File* implicit neural representation network contains three sub-networks: VisionNet, AudioNet, and TouchNet. Compared with OBJECTFOLDER 1.0, we greatly accelerate VisionNet inference by representing each object with thousands of individual MLPs; for AudioNet, we only predict the parts of the signal that are location-dependent instead of directly predicting the audio spectrograms, which significantly improves the rendering quality and also accelerates inference; our new TouchNet can render tactile readings of varied rotation angles and gel deformations, whereas only a single tactile image can be rendered per vertex in 1.0.

4.1.2 Touch

Background. We use the geometric measurement from a GelSight tactile sensor [28, 121] as the tactile reading. GelSight is a vision-based tactile sensor that interacts the object with an elastomer and measures the geometry of the contact surface with an embedded camera. It has a very high spatial resolution of up to 25 micrometers and can potentially be used to synthesize readings from other tactile sensors [55, 81]. To simulate tactile sensing with GelSight, we need to simulate both the deformation of the contact and the optical response to the deformation. For our tactile simulation, we aim to achieve the following three goals: 1) Being flexible to render tactile readings for touches of varied location, orientation, and pressing depth; 2) Being fast to efficiently render data for training TouchNet; 3) Being realistic to generalize to real-world touch sensors.

	Vision		Audio		Touch	
	PSNR \uparrow	SSIM \uparrow	STFT Distance ($\times 10^{-5}$) \downarrow	ENV Distance ($\times 10^{-4}$) \downarrow	PSNR \uparrow	SSIM \uparrow
OBJECTFOLDER 1.0 [36]	35.7	0.97	4.94	7.65	27.9	0.64
OBJECTFOLDER 2.0 (Ours)	36.3	0.98	0.19	1.29	31.6	0.78

Table 4.2: Comparing with OBJECTFOLDER 1.0 on the multisensory data rendering quality. \downarrow lower better, \uparrow higher better.

TouchNet. To achieve the three goals above, we adopt a two-stage approach to render realistic tactile signals. First, we simulate the contact deformation map, which

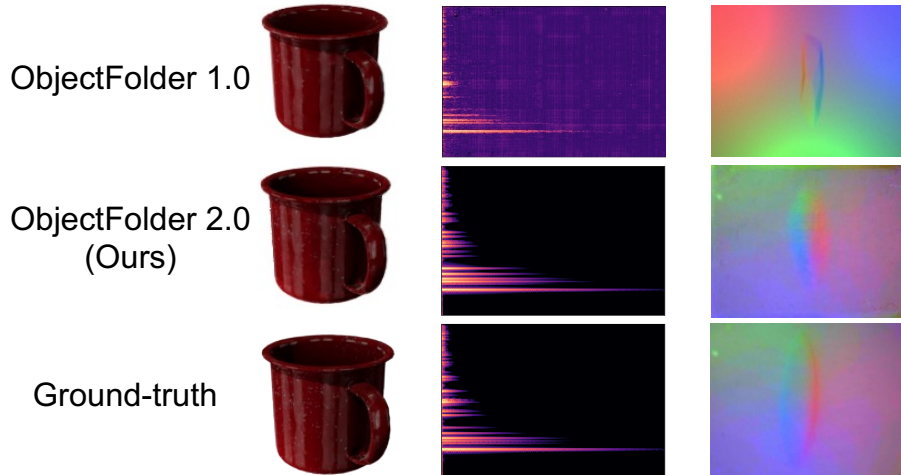


Figure 4.4: Comparing the visual, acoustic, and tactile data rendered from OBJECTFOLDER 1.0, OBJECTFOLDER 2.0 (Ours), and the corresponding ground-truth simulations for the YCB mug. See Supp. for more examples.

is constructed from the object’s shape in the contact area and the gelpad’s shape in the non-contact area to represent the local shape at the point of contact. We simulate the sensor-object interaction with Pyrender [69] to render deformation maps using OpenGL [2] with GPU-acceleration, reaching 700 fps for data generation.

We design TouchNet to encode the deformation maps from contacting each vertex on the object. We represent the tactile readings of each object as an 8D function whose input is a 3D location $\mathbf{x} = (x, y, z)$ in the object coordinate frame, a 3D unit contact orientation parametrized as (θ_T, ϕ_T) , gel penetration depth p , and the spatial location (w, h) in the deformation map. The output is the per-pixel value of the deformation map for the contact. TouchNet models this continuous function as an MLP network $F_T : (x, y, z, \theta_T, \phi_T, p, w, h) \rightarrow d$ that maps each input 8D coordinate to its corresponding value in the deformation map. After rendering the deformation map, we utilize the state-of-the-art GelSight simulation framework— Taxim [92], an example-based tactile simulation model that is calibrated with a real GelSight sensor, to render tactile RGB images from the deformation maps.

Compared to the TouchNet in OBJECTFOLDER 1.0, which can only render a single tactile image along the vertex normal direction per vertex, our new design of TouchNet can generate tactile outputs for rotation angles within $\pm 15^\circ$ and pressing depth in the range of 0.5-2 mm. Furthermore, with the help of Taxim, the mapping



Figure 4.5: Illustration of real-world objects used in experiments and our hardware set-up for collecting real-world impact sounds and tactile data.

from the deformation maps to the tactile optical outputs can be easily calibrated to different real vision-based tactile sensors, producing realistic tactile optical outputs that enable Sim2Real transfer.

4.1.3 Sim2Real Object Transfer

The goal of building OBJECTFOLDER 2.0 is to enable generalization to real-world objects by learning with the virtual objects from our dataset. We demonstrate the utility of the dataset by evaluating on three tasks including object scale estimation, contact localization, and shape reconstruction. In each task, we transfer the models learned on OBJECTFOLDER 2.0 to real-world objects. See Fig. 4.5 for an illustration of the 13 objects used in our experiments, and the hardware set-up for collecting real impact sounds and GelSight tactile readings.

Object Scale Estimation

All sensory modalities of objects are closely related to their scales. We want to demonstrate that learning with our virtualized objects can successfully transfer to scale estimation for a real object based on either its visual appearance, an impact sound, or a sequence of tactile readings. We train on the rendered multisensory data from our dataset, and test on 8 real objects from which we have collected real-world sensory data for all three modalities.

For vision and audio, we train ResNet-18 [43] that takes either an RGB image of the object or the magnitude spectrogram of an impact sound as input to predict

		Virtual Objects	Real Objects
Random		14.5	14.5
1.0 [36]	Vision	0.80	7.41
	Audio	0.57	6.85
	Touch	0.19	4.92
2.0 (Ours)	Vision	0.79	5.08
	Audio	0.20	4.68
	Touch	0.45	3.51

Table 4.3: Results on object scale prediction. We report the average difference between the predicted and the ground-truth scales of the objects in centimeters.

object scale¹. From a single local tactile reading, it is almost impossible to predict the scale of the object. Therefore, we use a recurrent neural network to combine features from 10 consecutive touch readings for tactile-based scale prediction. See Supp. for details.

Table 4.3 shows the results. “Random” denotes the baseline that randomly predicts a scale value within the same range as our models. We compare with models trained on sensory data from OBJECTFOLDER 1.0. Both OBJECTFOLDER 1.0 and our dataset achieve high scale prediction accuracy on virtual objects. However, models trained on our multisensory data generalize much better to real-world objects, demonstrating the realism of our simulation and accurate encoding of our implicit representation networks. Among the three modalities, tactile data has the smallest Sim2Real gap compared to vision and audio.







Modalities												
	Sim	Real	Sim	Real	Sim	Real	Sim	Real	Sim	Real	Sim	Real
Random	6.74	6.74	12.96	12.96	4.28	4.28	9.39	9.39	14.53	14.53	14.21	14.21
Audio	1.88	1.79	0.26	1.16	0.65	4.67	0.23	1.04	0.14	-	0.74	-
Touch	0.04	1.26	0.03	0.78	0.18	1.30	0.04	0.44	0.04	0.91	0.04	3.82
Audio + Touch	0.02	0.59	0.04	0.36	0.09	0.51	0.04	0.63	0.23	-	0.30	-

Table 4.4: Results on audio-tactile contact localization. We report the mean distance w.r.t. the ground-truth contact locations in centimeters.

¹We define the scale of an object as the length of the longest side of the axis aligned bounding box (AABB) enclosing the object.

Tactile-Audio Contact Localization

When interacting with an object of known shape, accurately identifying the location where the interaction happens is of great practical interest. Touch gives local information about the contact location, and impact at varied surface locations produces different modal gains for the excited sound. We investigate the potential of using the impact sounds and/or the tactile readings associated with the interaction for contact localization.

We apply particle filtering [61] to localize the sequence of contact locations from which tactile readings or impact sounds are collected. Particle filters are used to estimate the posterior density of a latent variable given observations. Here, observations are either tactile sensor readings when touching the object or impact sounds excited at the contact locations. The latent variable is the current contact location on the object’s surface. For touch, we extract features from an FCRN network [54] pre-trained for depth prediction from tactile images. For audio, we extract MFCC features from each 3s impact sound. We compare these features with particles sampled from the object surfaces that represent the candidate contact locations. Particles with high similarity scores to the features of the actual tactile sensor reading or impact sound are considered more likely to be the true contact location. In each iteration, we weight and re-sample the particles based on the similarity scores, and then update the particles’ locations based on the relative translations between two consecutive contacts obtained from the robot end-effector. We choose the 10 particles with the highest similarity scores as the candidate contact locations. For each object, we iterate the above process for 5-7 times until the predicted current contact location converges to a single location on the object’s surface. We perform experiments both in simulation and in real world.

Table 4.4 shows the results for six objects of complex shapes. We use the mean Euclidean distance with respect to the ground-truth contact location as the evaluation metric similar to [8]. We compare the localization accuracy for using only touch readings, impact sounds, or their combinations, and a baseline that randomly predicts a surface position as the contact location. We can see that touch-based contact location is much more accurate than using audio. Combining the two modalities leads to the best Sim2Real performance. Fig. 4.6 shows a qualitative example for

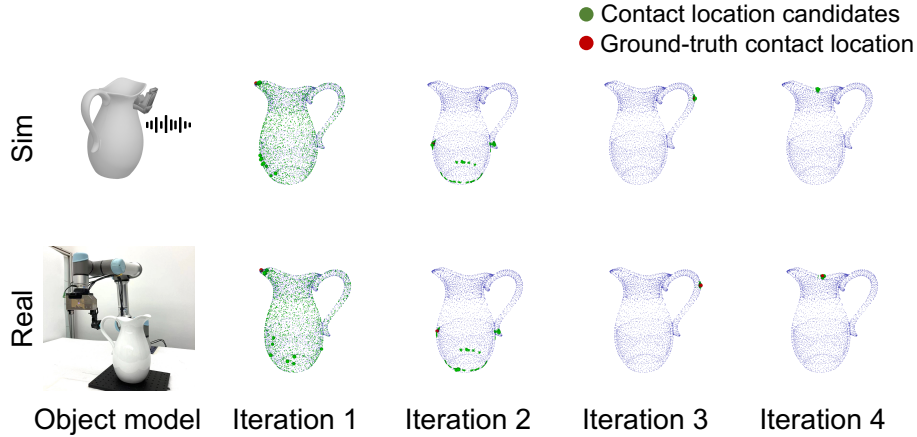


Figure 4.6: Qualitative results for contact localization with touch readings and impact sounds. Top: in simulation, bottom: real-world experiments. The candidate contact locations are shown as green particles in the particle filter. After several iterations shown from left to right in each row, the green particles converge to the ground-truth contact location shown as the red particle.

tactile-audio contact location with the pitcher object.






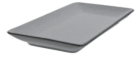
Modalities												
	Sim	Real	Sim	Real	Sim	Real	Sim	Real	Sim	Real	Sim	Real
Average	2.12	2.01	2.97	1.91	4.80	3.26	4.53	4.49	2.44	2.53	2.52	3.29
Vision	0.25	0.32	0.30	0.72	0.51	0.74	0.38	0.66	0.32	0.40	0.49	0.99
Touch	0.24	0.56	0.29	0.80	0.35	0.61	0.38	0.43	0.30	0.41	0.36	1.11
Vision + Touch	0.09	0.25	0.18	0.46	0.26	0.43	0.24	0.32	0.18	0.24	0.23	1.20

Table 4.5: Results on visuo-tactile shape reconstruction. We report the Chamfer-L1 distance w.r.t. the ground-truth meshes in centimeters.

Visuo-Tactile Shape Reconstruction

Single-image shape reconstruction has been widely studied in the vision community [16, 18, 70, 83]. However, in cases where there is occlusion such as during dexterous manipulation, tactile signals become valuable for perceiving the shape of the objects. Vision provides coarse global context, while touch offers precise local geometry. Here, we train models to reconstruct the shape of 3D objects from a single RGB image containing the object and/or a sequence of tactile readings on the object’s surface.

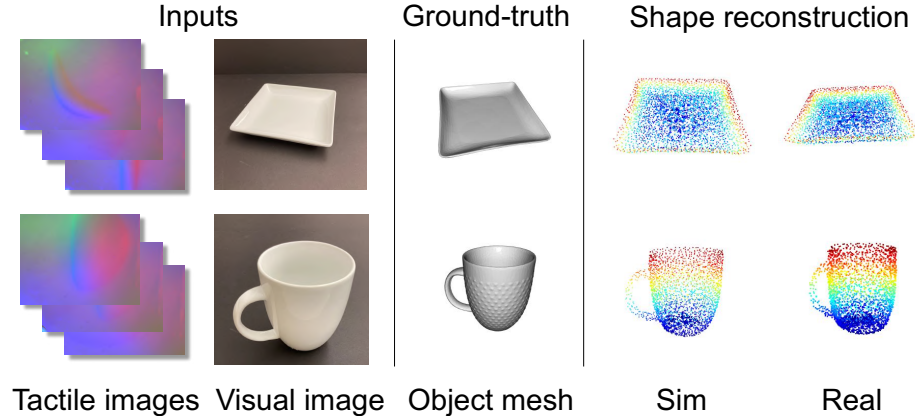


Figure 4.7: Qualitative results for visual-tactile shape reconstruction in simulation (Sim) and real-world (Real) for the square tray and the coffee mug.

We use Point Completion Network (PCN) [117], a learning-based approach for shape completion, as a testbed for this task. For touch, we use 32 tactile readings and map the associated deformation maps to a sparse point cloud given the corresponding touching poses. The sparse point cloud is used as input to the PCN network for generating a dense and complete point cloud. For vision, instead of using a series of local contact maps as partial observations of the object, a global feature extracted from a ResNet-18 network from a single image containing the object is used to supervise the shape completion process. For shape reconstruction with vision and touch, we use a two-stream network that merges the predicted point clouds from both modalities with a fully-connected layer to predict the final dense point cloud.

Table 4.5 shows the results for six objects of different shapes. Compared to the “Average” baseline that uses the average ground-truth mesh of the 6 objects as the prediction, shape reconstructions from a single image and a sequence of touch readings perform much better. Combining the geometric cues from both modalities usually leads to the best Sim2Real transfer performance. Fig. 4.7 shows some qualitative results for shape reconstruction with vision and touch. We can see that the predicted point clouds in both simulation and real-world experiments accurately capture the shapes of the two objects, and matches the ground-truth object meshes well.

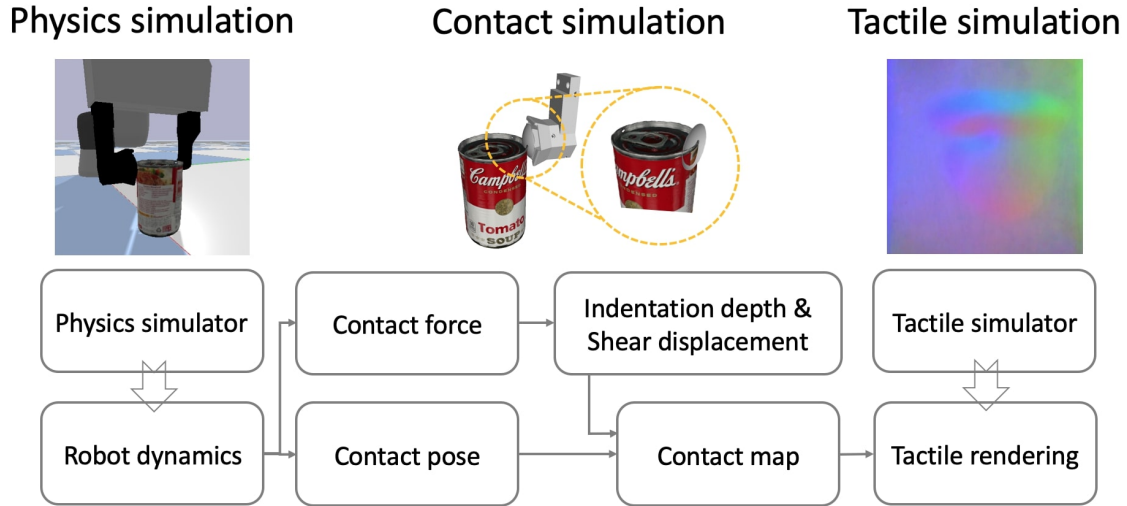


Figure 4.8: Our proposed simulation framework includes physics simulation, contact simulation and tactile simulation. Physics simulator handles the robot dynamics which provide the contact forces and poses. Contact models map them to indentation depths and shear displacements of the contact, and generate the contact map to feed into the tactile simulator. Tactile simulation renders the RGB tactile images.

4.2 Grasp Stability Estimation

4.2.1 Simulation Framework

In this section, we present our integrated simulation framework with tactile sensing. The framework includes three parts as shown in Fig. 4.8: physics simulation in Section 4.2.1, contact simulation in Section 4.2.1 and tactile simulation in Section 4.2.1. We use PyBullet to simulate the physics, and transfer the contact forces and poses to the contact deformation of a GelSight tactile sensor [120], which has a soft surface to interact with the object and an embedded camera to convert the contact geometries to RGB images. Then the tactile simulator renders tactile images according to the contact deformation.

Physics Simulation

Physics simulation reproduces the real world dynamic interaction between the robots and objects, and its accuracy directly impacts sim-to-real transfer. We use PyBullet

as our physics simulation engine. For the grasping task, we load the robot arm, gripper and a GelSight mounted on the gripper with proper geometries and links as shown in Fig. 3.9.

Grasping requires accurate contact simulation, therefore physics parameters setting such as friction, objects’ mass and their center of mass becomes essential. In order to match the simulation with the real grasping scenario, we measure the weights and center of mass of objects in reality as the reference. We then estimate the friction of the contact surface by calibrating with the real world data. Specifically, we search for the best friction coefficient by adjusting friction values in simulation to minimize the grasping label mismatching between real and simulated data under the same grasping configurations.

Contact Simulation

Contact simulation refers to simulating the deformation of GelSight’s soft surface under applied contact forces during the interaction with the object. PyBullet simulates the robot dynamics, but it lacks accurate contact model since the soft body deformation of tactile sensors is not able to be simulated with only rigid body collision. Instead of applying computationally costly soft body simulation, we use an simplified model that maps contact forces to the deformation of the tactile sensor.

When the tactile sensor touches the object, PyBullet calculates the contact force and relative poses between the object and the sensor. To simulate the object’s indentation into the tactile sensor’s surface under the normal loading and sliding on the contact surface under the shear loading, we map the contact force to the indentation depth and the shear motion of the contact. From the experimental measurements on our real GelSight sensor as shown in Fig. 4.9, we characterize the linear mapping’s parameters k_n and k_s between the normal force and the indentation volume, and between the shear force and the shear motion of the contact as:

$$\begin{aligned} V &= k_n F_n \\ D &= k_s F_s \end{aligned} \tag{4.1}$$

where V is the indentation volume, F_n is the normal force; D is the shear displacement, F_s is the shear force.

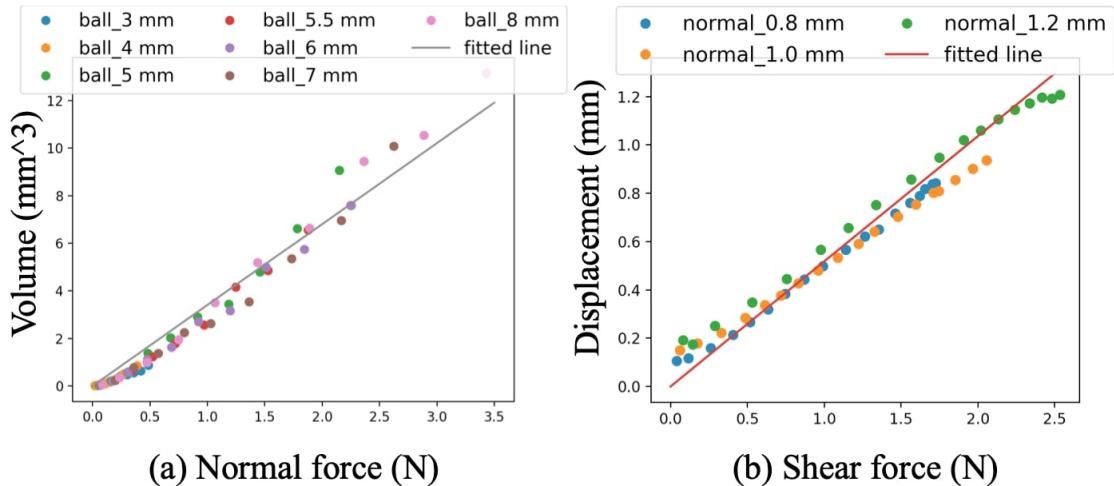


Figure 4.9: Linear mapping from the contact force to the soft body deformation. (a) From the experiments we found the linear mapping from the normal force to the indentation volume on soft body, which can be further calculated as the indentation depth. (b) The linear mapping from the shear force to the shear displacement of the contact area.

Recovering the indentation depth map from the volume V does not have a close-form solution, therefore we use the binary searching to find a best estimated depth map. Given an initial depth map, we integrate the indentation depth $d(A)$ within the contact area A to the volume as:

$$V_{est} = \int d(A)dA \quad (4.2)$$

We iteratively adjust the $d(A)$ to minimize the error between the estimated volume V_{est} and the target volume V to find the best solution.

To simulate the contact shape, we utilize the PyRender as TACTO [107] and place a virtual camera behind the GelSight sensor. We only use the depth camera to capture the 3D contact shape, called *contact map*. Given the indentation depth, shear displacement and the relative poses between the object and the tactile sensor, we move the object along with the normal direction of contact with indentation depth and the tangential direction with the shear displacement in the PyRender and then render the *contact map*.

Note that PyBullet reserves a collision margin between two colliding objects to ensure numerical stability where two objects considered as collided are still separated

by a distance from each other. This margin even changes when the object has non-convex geometric surfaces. Therefore, we adapt this margin in PyRender to get precise contact indentation depth.

Tactile Simulation

After getting the contact map from the contact model, we render the GelSight tactile images with an state-of-the-art example-based simulation model, Taxim [93]. The Taxim model uses a lookup table to map the contact shapes to tactile images. The lookup table is calibrated with a real tactile sensor, which is used for the real grasp experiments.

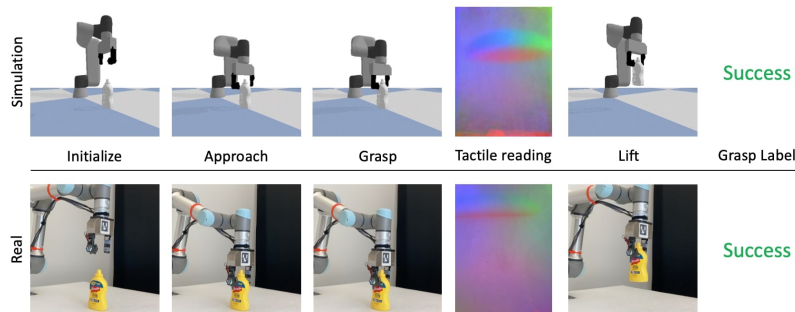


Figure 4.10: Grasp pipeline for both simulation and real experiments. We initialize the robot on top of the object, move the gripper down to a preset height, close the gripper with a preset grasping force to grasp the object, and then lift it. We record the tactile readings from a GelSight sensor after grasping.

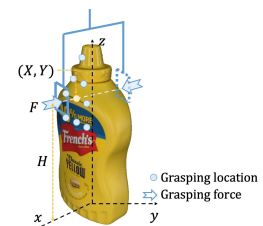


Figure 4.11: Grasping configurations including the grasping location, height and force.

4.2.2 Sim2Real Grasping Prediction

We formalize the grasping stability prediction as a supervise learning problem where we use the tactile images during grasping to predict the grasp outcomes. We use simulated tactile images and labels from our simulation framework to train the learning model and test it on real-world data where we realize zero-shot sim-to-real transfer.

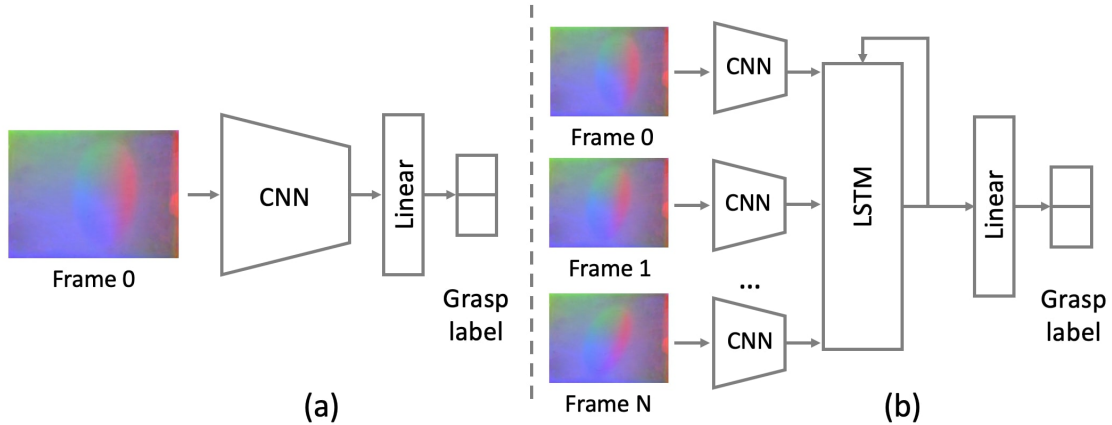


Figure 4.12: Grasp stability prediction networks. (a) We input single tactile images to a feature extractor (CNN) and a classifier (MLP) to predict the grasp results. (b) We input a sequence of tactile images to feature extractors (CNN), a LSTM module and then a classifier (MLP) to predict the grasp results.

Grasp Stability Prediction Model

Grasp stability can be classified into binary labels, success when the object can be stably lifted or failure in other cases. We predict grasp outcomes based on a single or sequential tactile images from a GelSight sensor. We record the sequence of tactile images starting after closing the gripper and lasts for three seconds. And we take the tactile image when the object is grasped but not lifted as the single input.

We build a learning model to extract latent features of tactile images and then use a classifier to predict the grasp stability base on these features. For the model with single tactile inputs, we use a pre-trained ResNet-18 [42] as our feature extractor and a multi-layer perceptual (MLP) as classifier as shown in Fig. 4.12 (a). For the sequential inputs, we feed the sequence of image features to a long short-term memory (LSTM) module and then forward the last hidden state’s output to the classifier as shown in Fig. 4.12 (b).

Grasping Pipeline

Our grasping pipeline includes initialization, approaching, grasping, lifting and labeling as shown in Fig 4.10. We initialize the robot and the gripper on the top of the object. Given the specified grasping configuration, the robot rotates to the target orientation, moves straightly down to the grasping height, and adjust the grasping

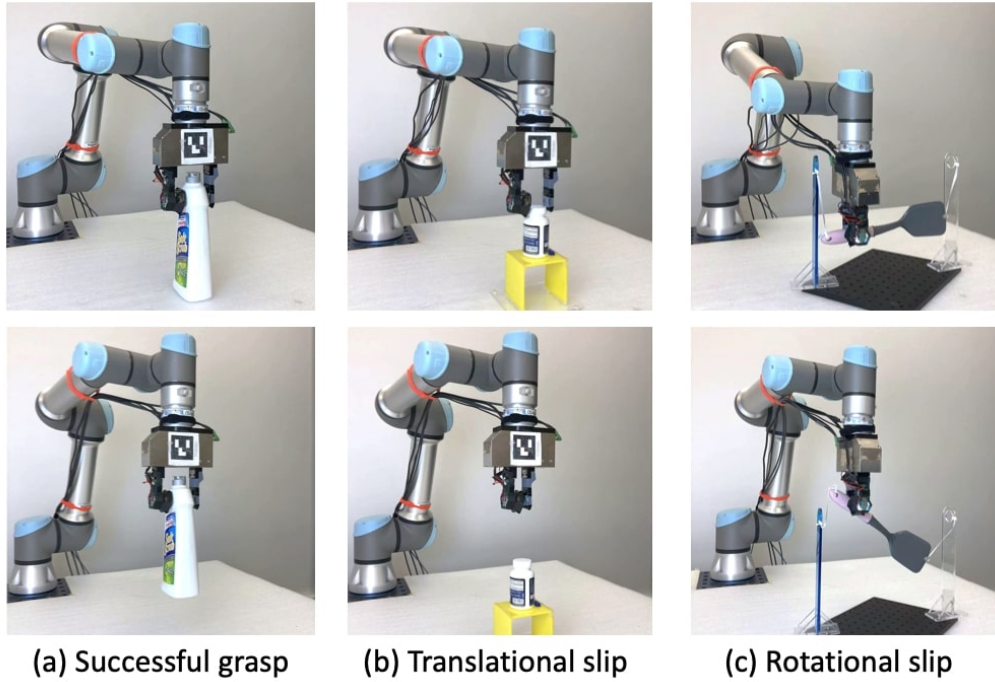


Figure 4.13: We classify grasp as successful grasp (a) and failed grasp (b), (c) including translational and rotational slip.

location on the plane parallel to table. During the grasping, the gripper closes with a certain speed and force. Then till the gripper closes entirely, we record tactile images from the GelSight and use them as our grasping stability prediction model inputs. The robot then lifts the object for 10 cm. During the lifting, if the object remains stable in the gripper, we label the grasp as ‘success’; otherwise, if the object falls, or significantly slips in the gripper, we label the grasp as ‘failure’. There are two different kinds of failure: translational slip or rotational slip coming from lacking grasping forces or wrong grasping locations respectively as shown in Fig. 4.13.

Grasping Configuration Generation

We collect grasp data with various grasping configurations regarding the grasp location and force. We describe the grasping configuration as a vector (F, H, X, Y) , where F is the grasping force, H is the grasping height, and X, Y is grasping location on the horizontal plane as shown in Fig. 4.11. We define the range of each dimension of grasping, discretize the grasping space and then conduct the grasping.

4.2.3 Experiments

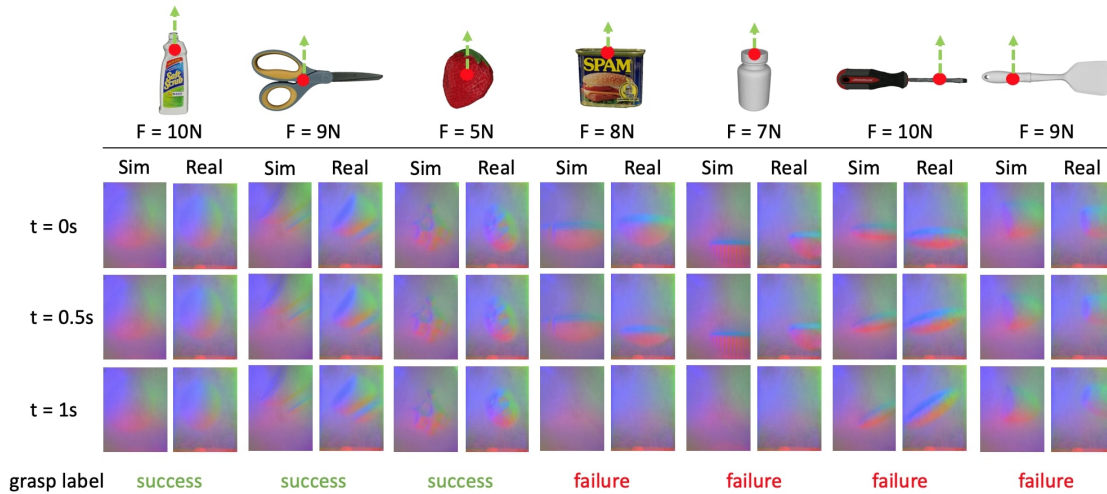


Figure 4.14: Examples of tactile readings under different grasping scenarios. Different grasping locations as marked on the object and grasping forces F lead to different grasping outcomes. We show the sequence of the tactile readings during grasping, where geometries of contact can be used to predict the grasping outcomes.

We conduct both simulated and real grasp experiments and evaluate our simulation framework’s ability to realize sim-to-real transfer based on collected data. Our data collection includes variance in three parameters of grasping: the target object, the grasping location, and the grasping force. We show the grasp stability prediction model’s performance on both single and sequences of tactile images. We provide details of our experiments in the following sections.

Data Collection

We use a UR5e robot with a Weiss WSG-50 gripper, and mount a GelSight [28] sensor on one side of the gripper. We use the objects from YCB [14] and GoogleScan [29] datasets for grasping where the mesh models are available. We conduct the same grasping process in both simulation and real as shown in Fig. 4.10.

We selected twelve objects for both simulation and real-world data collection as shown in Table 4.6. To demonstrate our model’s ability to make predictions over a broad set of objects, we chose objects that provide a range of shapes, surface frictions,

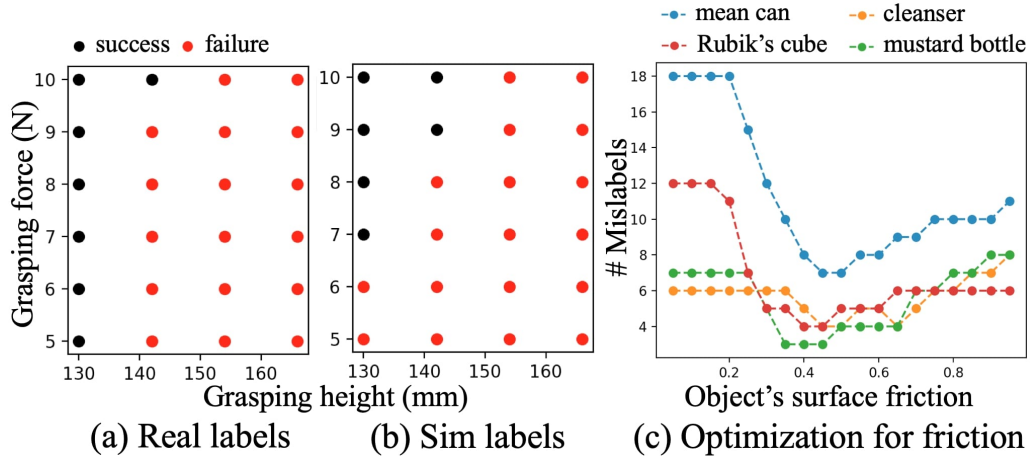


Figure 4.15: We optimize the friction coefficient of object surface by matching the grasping labels between simulated and real data under the same configuration of grasping heights and forces as shown in (a) and (b). We evaluate possible friction coefficients for several objects and choose the friction values that minimize mislabeling as shown in (c).

masses, and center of mass locations.

Testing dataset from real world data For each object, we grasp it at three to five different heights, ranging from the top of the object to the minimum reachable height; we vary the grasping location on the XY plane with three to six values depending on the size of the object; and we linearize the grasping force with six different values ranging from 5N to 10N. We also add mass (water or clay) ranging from 100g to 500g to light objects such as empty bottles to get more grasping failures. These parameters are chosen so that the collected grasp data is approximately balanced with 363 successful and 389 failed grasps in total.

Training dataset from simulated data We use the same set of objects in simulation. For each object, we collected 100 to 150 grasp trials for training and 50 grasp trails for testing following the same configurations mentioned in testing dataset. For each grasp, we record the rendered tactile images and automatically label its corresponding grasp outcomes based on the pose changing of the object.

Some data examples are shown in Fig. 4.14 where we mark the grasp locations, grasp forces, grasp labels and the corresponding tactile readings.













Object													average
Sim2Sim (TACTO)	0.86	0.90	0.96	0.86	0.72	0.94	1.00	0.85	0.92	0.96	0.55	0.65	0.84
Sim2Sim (Single)	1.00	0.95	0.87	1.00	0.93	1.00	0.87	1.00	1.00	1.00	1.00	1.00	0.96
Sim2Sim (Sequence)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Sim2Real (TACTO)	0.54	0.66	0.09	0.68	0.20	0.60	0.33	0.48	0.62	0.37	0.34	0.50	0.42
Sim2Real (Single)	0.95	0.89	0.87	0.81	0.79	0.83	0.75	0.95	0.95	1.00	0.84	0.82	0.86
Sim2Real (Sequence)	1.00	0.91	0.96	0.82	0.83	0.81	1.00	0.93	0.95	1.00	0.87	0.87	0.90

Table 4.6: The result of grasp stability prediction. We test the prediction accuracy for both sim-to-sim and sim-to-real transfers with a single tactile image and sequences of tactile images. We compare the performance with TACTO [107].

Optimization of Friction

To eliminate the gap of sim-to-real transfer, we use a few real-world grasping examples (around 20) for each object to tune the friction coefficient in simulation with an optimization process. As shown in Fig. 4.15, for each object (here we use the mustard bottle as example), we generate the distribution of the grasping outcomes based on the grasping heights and forces for both simulated and real data. Adjusting the friction coefficient in simulation will lead to different distributions, and we search for the best friction coefficient based on how well the generated outcome distribution matches the real-world one. We discretize the friction coefficient search space in the range [0,1] with a step size of 0.05, then we plot the number of mislabeling in Fig. 4.15 (c). For each object, there is a point with the fewest mislabels and we denote the corresponding friction coefficient as the best one to use.

Learning Model and Training Settings

We use Res-Net 18 [42] pre-trained on ImageNet as the tactile feature extractor. For single tactile inputs, we use a two-layer MLP 512-256-2 with ReLu activation and a 0.2 dropout after the first linear layer as classifier. For sequential tactile inputs, we use a three-layer LSTM module with 128 hidden layer size, then a two-layer MLP 128-64-2 with ReLu activation and a 0.2 dropout after the first linear layer as the classifier. For both learning models, we use the Adam optimizer with 1×10^{-4} learning rate, 5×10^{-5} weight decay and a ReduceLROnPlateau scheduler. We train

the model with 8 batch size and 30 epochs. We set the learning rate of the feature extractor as 0.8 of the learning rate of the rest modules to prevent overfitting. We train individual prediction model for each object.

Grasp Stability Prediction with Single Tactile Image

We first test our sim-to-real grasp stability prediction model with single tactile images. The results are reported in Table 4.6 indicated as Sim2Sim (Single) and Sim2Real (Single). We denote the case of training on simulated data and testing on simulated data as “Sim2Sim” and the case of training on simulated data and testing on real data as “Sim2Real”. From the table, we show our Sim2Sim prediction accuracy is 100% excluding objects (Rubik’s cube, cup, spatula, strawberry) which have ambiguous shapes. For those shapes, it is impossible to locate the grasp based on single tactile images since they look similar or even the same.

We also show our Sim2Real prediction accuracy are all above 80% except objects spatula and strawberry. The mesh models of these objects are too coarse compared to real objects, which leads to significant differences between simulated and real tactile images. On average, our Sim2Real gap is less than 10% comparing to our Sim2Sim results.

We conduct the same grasping experiments using TACTO [107] as a baseline and post results in the Table 4.6. The results show that our method outperforms TACTO. A major failure of TACTO is caused by inaccurate contact model: it models the contact that the normal force is linear to the indentation depth, which does not match the physics of the sensor in the real world. In addition, TACTO does not properly calculate the collision margin so that it causes occasional failure in generating tactile signals upon contact.

Grasp Stability Prediction with Sequential Tactile Image

Single tactile images are less informative when objects have ambiguous geometries such as the flat surface of a Rubik’s cube. Therefore we also test our simulation performance on models with sequential tactile images as inputs. Sim2Sim and Sim2Real accuracy results are shown in Table 4.6 indicated as Sim2Sim (Sequential) and Sim2Real (Sequential). Comparing to the single tactile results, we notice that

the Sim2Real performance on objects Rubik’s cube, cup, spatula, and strawberry is improved. And the average performance also improves to 90.7%. This is mostly because the sequential data shows either the stable or the changing contacts from tactile images after lifting which can indicate the object’s motion. And it serves as better features to predict the grasp outcomes.

Effects of Dataset Size, Friction and Center of Mass

We do ablation study on different setting of dataset size, friction and center of mass on a single object in simulation. As shown in Table 4.2.3, the performance of Sim2Sim and Sim2Real both increase along with the increasing dataset size but reach the plateau after 200. This suggests that choosing dataset size between 100 to 200 is sufficient for this task. Comparing to the best choice 0.45 for friction coefficient and -0.03 for center of mass, even though the Sim2Sim accuracy stays high, the Sim2Real accuracy drops quickly with inaccurate parameters. This indicates that the physics settings affect the results significantly and it is necessary to set the proper physics parameters for effective sim-to-real transfer.



Dataset size	50	100	200	500
Sim2Sim Acc	0.875	0.937	1.000	1.000
Sim2Real Acc	0.791	0.833	0.958	0.958
Friction coefficient	0.2	0.45	0.8	
Sim2Sim Acc	1.000	1.000	0.937	
Sim2Real Acc	0.666	0.958	0.541	
Center of Mass	-0.03	0.00	0.03	
Sim2Sim Acc	1.000	1.000	1.000	
Sim2Real Acc	0.958	0.708	0.625	

Table 4.7: Ablation study of dataset size, friction coefficient on potted meat can, and center of mass on scissor. 0.45 and -0.03 are the best parameters for friction and center of mass we used in our experiments.

Chapter 5

Conclusions

In this thesis, we present Taxim, an example-based simulation model for GelSight tactile sensors that combines optical and marker motion field simulation. Our simulation is computationally light weight, easy to set up and use. It can also simply be applied to other GelSight-like vision-based tactile sensors. By calibrating with example data from real sensors, it incorporates the sensor’s illumination features and system noises which significantly decreases the sim-to-real gap.

We further demonstrated various sim-to-real robotic perception and manipulation tasks that leverage Taxim: object scale prediction, contact localization, and shape reconstruction along with a multisensory object dataset, and grasp stability prediction based on tactile sensing. These data-driven tasks require large data collection, and given a simulation model of tactile sensors, these can be conducted in simulation more efficiently.

In this work, we built a simulation model for GelSight tactile sensor, however, there are various tactile sensors with different working principles. In the future, we are going to explore the standard simulation framework for tactile sensors that can be generalized on different kinds of sensors. In addition, we have shown the potentiality of the sim-to-real transfer on robotic tasks with tactile sensing, we would like to further apply our simulation on different applications such as dexterous manipulation which requires even higher fidelity of the simulation.

CHAPTER 5. CONCLUSIONS

Bibliography

- [1] 3D Model Haven. <https://3dmodelhaven.com/>. 4.1
- [2] OpenGL. <https://www.opengl.org>. 4.1.2
- [3] Gazebo. <http://gazebo.org/>. 1.1
- [4] Solidworks. <https://www.solidworks.com/>. 3.4.1
- [5] 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)*, pages 1–7. IEEE, 2021. (document), 1.1, 2.1, 2.3, 3.4.2, 3.7, ??, 3.1, 3.4.2, ??, 3.2
- [6] Jérémie Allard, Stéphane Cotin, François Faure, Pierre-Jean Bensusan, François Poyer, Christian Duriez, Hervé Delingette, and Laurent Grisoni. Sofa: an open source framework for medical simulation. In *MMVR 15-Medicine Meets Virtual Reality*, volume 125, pages 13–18. IOP Press, 2007. 1.1
- [7] Maria E Angelopoulou and Maria Petrou. Uncalibrated flatfielding and illumination vector estimation for photometric stereo face reconstruction. *Machine vision and applications*, 25(5):1317–1332, 2014. 3.2
- [8] Maria Bauza, Eric Valls, Bryan Lim, Theo Sechopoulos, and Alberto Rodriguez. Tactile object pose estimation from the first touch with geometric contact rendering. In *Proc. Conf. on Robot Learning, CoRL*, 2020. 1.1, 4.1.3
- [9] Yasemin Bekiroglu, Janne Laaksonen, Jimmy Alison Jorgensen, Ville Kyrki, and Danica Kragic. Assessing grasp stability based on learning and haptic data. *IEEE Transactions on Robotics*, 27(3):616–629, 2011. 2.3
- [10] Yasemin Bekiroglu, Andreas Damianou, Renaud Detry, Johannes A Stork, Danica Kragic, and Carl Henrik Ek. Probabilistic consolidation of grasp experience. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 193–200. IEEE, 2016. 2.3
- [11] Roberto Calandra, Andrew Owens, Manu Upadhyaya, Wenzhen Yuan, Justin Lin, Edward H Adelson, and Sergey Levine. The feeling of success: Does touch sensing help predict grasp outcomes? In *Conference on Robot Learning*, pages

- 314–323. PMLR, 2017. 2.1, 2.2, 2.3
- [12] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, 2018. 2.2, 2.3
- [13] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *ICRA*, 2015. 2.2, 4.1
- [14] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52, Sep 2015. ISSN 1070-9932. doi: 10.1109/mra.2015.2448951. URL <http://dx.doi.org/10.1109/MRA.2015.2448951>. 4.2.3
- [15] Dan Casas and Miguel A Otaduy. Learning nonlinear soft-tissue dynamics for interactive avatars. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–15, 2018. 2.1
- [16] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2.2, 4.1.3
- [17] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2.2
- [18] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 4.1.3
- [19] Alex Church, John Lloyd, Raia Hadsell, and Nathan F Lepora. Optical tactile sim-to-real policy transfer via real-to-sim tactile image translation. *arXiv preprint arXiv:2106.08796*, 2021. 2.1
- [20] Alex Church, John Lloyd, Nathan F Lepora, et al. Tactile sim-to-real policy transfer via real-to-sim image translation. In *Conference on Robot Learning*, pages 1645–1654. PMLR, 2022. 1.1, 2.3
- [21] Jasmine Collins, Shubham Goel, Achleshwar Luthra, Leon Xu, Kenan Deng, Xi Zhang, Tomas F Yago Vicente, Himanshu Arora, Thomas Dideriksen, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. *arXiv preprint arXiv:2110.06199*, 2021. 2.2, 4.1
- [22] Stéphane Cotin, Hervé Delingette, and Nicholas Ayache. Real-time elastic

- deformations of soft tissues for surgery simulation. *IEEE transactions on Visualization and Computer Graphics*, 5(1):62–73, 1999. [2.1](#), [3.3](#)
- [23] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. [1.1](#)
- [24] Hao Dang and Peter K Allen. Stable grasping under pose uncertainty using tactile feedback. *Autonomous Robots*, 36(4):309–330, 2014. [2.3](#)
- [25] Cristiana de Farias, Naresh Marturi, Rustam Stolkin, and Yasemin Bekiroglu. Simultaneous tactile exploration and grasp refinement for unknown objects. *IEEE Robotics and Automation Letters*, 6(2):3349–3356, 2021. [2.3](#)
- [26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. [2.2](#)
- [27] Zihan Ding, Nathan F Lepora, and Edward Johns. Sim-to-real transfer for optical tactile sensing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1639–1645. IEEE, 2020. [1.1](#), [2.1](#)
- [28] Siyuan Dong, Wenzhen Yuan, and Edward H Adelson. Improved gelsight tactile sensor for measuring geometry and slip. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 137–144. IEEE, 2017. ([document](#)), [3.2](#), [4.1.2](#), [4.2.3](#)
- [29] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. *arXiv preprint arXiv:2204.11918*, 2022. ([document](#)), [3.4.2](#), [3.11](#), [4.1](#), [4.2.3](#)
- [30] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 4397–4404. IEEE, 2015. [2.3](#)
- [31] Chuang Gan, Deng Huang, Hang Zhao, Joshua B Tenenbaum, and Antonio Torralba. Music gesture for visual sound separation. In *CVPR*, 2020. [2.2](#)
- [32] Ruohan Gao and Kristen Grauman. Co-separating sounds of visual objects. In *ICCV*, 2019. [2.2](#)
- [33] Ruohan Gao and Kristen Grauman. 2.5d visual sound. In *CVPR*, 2019. [2.2](#)
- [34] Ruohan Gao, Rogerio Feris, and Kristen Grauman. Learning to separate object sounds by watching unlabeled video. In *ECCV*, 2018. [2.2](#)
- [35] Ruohan Gao, Tae-Hyun Oh, Kristen Grauman, and Lorenzo Torresani. Listen to look: Action recognition by previewing audio. In *CVPR*, 2020. [2.2](#)
- [36] Ruohan Gao, Yen-Yu Chang, Shivani Mall, Li Fei-Fei, and Jiajun Wu. Ob-

- jectfolder: A dataset of objects with implicit visual, auditory, and tactile representations. In *CoRL*, 2021. 2.2, 2.2, 4.1, ??, ??, ??
- [37] Ruohan Gao, Zilin Si, Yen-Yu Chang, Samuel Clarke, Jeannette Bohg, Li Fei-Fei, Wenzhen Yuan, and Jiajun Wu. Objectfolder 2.0: A multisensory object dataset for sim2real transfer. In *CVPR*, 2022. (document), 2.2, 2.2, 4.1, 4.1.1
- [38] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021. 2.2
- [39] Ioannis Gkioulekas, Anat Levin, Frédo Durand, and Todd Zickler. Micron-scale light transport decomposition using interferometry. *ACM Transactions on Graphics (ToG)*, 34(4):1–14, 2015. (document), 3.9
- [40] 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. (document), 1.1, 2.1, 2.3, 3.4.2, 3.7, ??, 3.1, ??, 3.2
- [41] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*, 2020. 2.2
- [42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4.2.2, 4.2.3
- [43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4.1.3
- [44] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, 2021. 2.2
- [45] Aaron Hertzmann and Steven M Seitz. Example-based photometric stereo: Shape reconstruction with general, varying brdfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1254–1264, 2005. 3.2
- [46] Francois R Hogan, Maria Bauza, Oleguer Canal, Elliott Donlon, and Alberto Rodriguez. Tactile regrasp: Grasp adjustments via simulated tactile transformations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2963–2970. IEEE, 2018. 1.1, 2.3
- [47] Francois R Hogan, Michael Jenkin, Sahand Rezaei-Shoshtari, Yogesh Girdhar, David Meger, and Gregory Dudek. Seeing through your skin: Recognizing objects with a novel visuotactile sensor. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1218–1227, 2021. 2.1, 2.3
- [48] Chen Hui, Sun Hanqiu, and Jin Xiaogang. Interactive haptic deformation of dynamic soft objects. In *Proceedings of the 2006 ACM international conference*

- on *Virtual reality continuum and its applications*, pages 255–261, 2006. [2.1](#)
- [49] Micah K Johnson and Edward H Adelson. Retrographic sensing for the measurement of surface texture and shape. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1070–1077. IEEE, 2009. [1.1](#), [3.2](#)
- [50] Micah K Johnson, Forrester Cole, Alvin Raj, and Edward H Adelson. Microgeometry capture using an elastomeric sensor. *ACM Transactions on Graphics (TOG)*, 30(4):1–8, 2011. [3.2](#)
- [51] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *SIGGRAPH*, 1984. [2.2](#)
- [52] Zhanat Kappassov, Juan-Antonio Corrales-Ramon, and Véronique Perdereau. Simulation of tactile sensing arrays for physical interaction tasks. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 196–201, 2020. doi: 10.1109/AIM43001.2020.9158822. [2.3](#)
- [53] Bruno Korbar, Du Tran, and Lorenzo Torresani. Co-training of audio and video representations from self-supervised temporal synchronization. In *NeurIPS*, 2018. [2.2](#)
- [54] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *Proc. Intl. Conf. on 3D Vision (3DV)*, pages 239–248. IEEE, 2016. [4.1.3](#)
- [55] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg 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*, 5(3):3838–3845, 2020. [2.1](#), [3.4.2](#), [4.1.2](#)
- [56] Michelle A Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950. IEEE, 2019. [2.2](#)
- [57] Yunzhu Li, Jun-Yan Zhu, Russ Tedrake, and Antonio Torralba. Connecting touch and vision via cross-modal prediction. In *CVPR*, 2019. [2.2](#)
- [58] Joseph J Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing ikea objects: Fine pose estimation. In *ICCV*, 2013. [2.2](#)
- [59] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. [2.2](#)

- [60] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *CVPR*, 2021. 2.2
- [61] Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 1998. 4.1.3
- [62] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2.2
- [63] Shan Luo, Wenxuan Mou, Kaspar Althoefer, and Hongbin Liu. Localizing the object contact through matching tactile features with visual map. In *ICRA*, 2015. 2.2
- [64] Daolin Ma, Elliott Donlon, Siyuan Dong, and Alberto Rodriguez. Dense tactile force estimation using gelslim and inverse fem. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5418–5424. IEEE, 2019. 2.1
- [65] Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Conference on robot learning*, pages 515–524. PMLR, 2017. 2.3
- [66] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017. 2.3
- [67] Perla Maiolino, Marco Maggiali, Giorgio Cannata, Giorgio Metta, and Lorenzo Natale. A flexible and robust large scale capacitive tactile system for robots. *IEEE Sensors Journal*, 13(10):3910–3917, 2013. doi: 10.1109/JSEN.2013.2258149. 1.1
- [68] 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. 1.1, 2.1
- [69] Matthew Matl. Pyrender. <https://github.com/mmatl/pyrender>, 2019. 4.1.2
- [70] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2.2, 4.1.3
- [71] 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. In *ECCV*, 2020. 2.2
- [72] Sami Moision, Beatriz León, Pasi Korkealaakso, and Antonio Morales. Simulation of tactile sensors using soft contacts for robot grasping applications. In *2012 IEEE International Conference on Robotics and Automation*, pages 5037–5043.

- IEEE, 2012. [2.1](#), [2.3](#)
- [73] Pedro Morgado, Nono Vasconcelos, Timothy Langlois, and Oliver Wang. Self-supervised generation of spatial audio for 360° video. In *NeurIPS*, 2018. [2.2](#)
- [74] 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. *arXiv preprint arXiv:2103.16747*, 2021. [2.1](#), [2.3](#), [3.4.3](#)
- [75] Yashraj S Narang, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. Interpreting and predicting tactile signals via a physics-based and data-driven framework. *arXiv preprint arXiv:2006.03777*, 2020. [2.1](#), [2.3](#)
- [76] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H Mueller, Chakravarty R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. 2021. [2.2](#)
- [77] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. [2.2](#)
- [78] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *ICCV*, 2019. [2.2](#)
- [79] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *ECCV*, 2018. [2.2](#)
- [80] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016. [2.2](#)
- [81] Akhil Padmanabha, Frederik Ebert, Stephen Tian, Roberto Calandra, Chelsea Finn, and Sergey Levine. OmniTact: A multi-directional high-resolution touch sensor. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 618–624. IEEE, 2020. [4.1.2](#)
- [82] Dinesh K Pai, Kees van den Doel, Doug L James, Jochen Lang, John E Lloyd, Joshua L Richmond, and Som H Yau. Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001. [2.2](#)
- [83] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. [2.2](#), [4.1.3](#)
- [84] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable

- neural radiance fields. In *ICCV*, 2021. [2.2](#)
- [85] Zachary Pezzementi, Erica Jantho, Lucas Estrade, and Gregory D Hager. Characterization and simulation of tactile sensors. In *2010 IEEE Haptics Symposium*, pages 199–205. IEEE, 2010. [2.1](#), [2.3](#)
- [86] Lerrel Pinto, Dhiraaj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pages 3–18. Springer, 2016. [2.2](#)
- [87] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. 2021. [2.2](#)
- [88] J Schill, J Laaksonen, M Przybylski, V Kyrki, T Asfour, and R Dillmann. Learning continuous grasp stability for a humanoid robot hand based on tactile sensing. In *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1901–1906. IEEE, 2012. [2.3](#)
- [89] Arda Senocak, Tae-Hyun Oh, Junsik Kim, Ming-Hsuan Yang, and In So Kweon. Learning to localize sound source in visual scenes. In *CVPR*, 2018. [2.2](#)
- [90] Carmelo Sferrazza, Adam Wahlsten, Camill Trueeb, and Raffaello D’Andrea. Ground truth force distribution for learning-based tactile sensing: A finite element approach. *IEEE Access*, 7:173438–173449, 2019. [2.1](#), [2.3](#)
- [91] Carmelo Sferrazza, Thomas Bi, and Raffaello D’Andrea. Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4389–4396. IEEE, 2020. [2.1](#)
- [92] Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *arXiv preprint arXiv:2109.04027*, 2021. [4.1.2](#)
- [93] Zilin Si and Wenzhen Yuan. Taxim: An example-based simulation model for gelsight tactile sensors. *IEEE Robotics and Automation Letters*, 2022. ([document](#)), [1.2](#), [2.3](#), [4.2.1](#)
- [94] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird:(big) berkeley instance recognition dataset. In *ICRA*, 2014. [2.2](#)
- [95] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019. [2.2](#)
- [96] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. [2.2](#)

- [97] Edward J Smith, Roberto Calandra, Adriana Romero, Georgia Gkioxari, David Meger, Jitendra Malik, and Michal Drozdal. 3D shape reconstruction from vision and touch. In *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*, 2020. 2.2
- [98] Stefano Stassi, Valentina Cauda, Giancarlo Canavese, and Candido Fabrizio Pirri. Flexible tactile sensing based on piezoresistive composites: A review. *Sensors*, 14(3):5296–5332, 2014. 1.1
- [99] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *CVPR*, 2018. 2.2
- [100] Sudharshan Suresh, Zilin Si, Joshua G Mangelson, Wenzhen Yuan, and Michael Kaess. Efficient shape mapping through dense touch and vision. *arXiv preprint arXiv:2109.09884*, 2021. 1.1, 2.2
- [101] R Tedrake, TDD Team, et al. Drake: Model-based design and verification for robotics, 2019. 1.1
- [102] The GIMP Development Team. Gimp. URL <https://www.gimp.org>. 3.4.2
- [103] Y. Tian, J. Shi, B. Li, Z. Duan, and C. Xu. Audio-visual event localization in unconstrained videos. In *ECCV*, 2018. 2.2
- [104] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 1.1
- [105] Shaoxiong Wang, Jiajun Wu, Xingyuan Sun, Wenzhen Yuan, William T Freeman, Joshua B Tenenbaum, and Edward H Adelson. 3d shape perception from monocular vision, touch, and shape priors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1606–1613. IEEE, 2018. 1.1
- [106] 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. *arXiv preprint arXiv:2012.08456*, 2020. (document), 1.1, 2.1, 3.4.2, 3.7, ??, 3.1, 3.4.2, ??, 3.2
- [107] Shaoxiong Wang, Mike Maroje 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*, 2022. (document), 1.1, 2.3, 4.2.1, 4.6, 4.2.3
- [108] Yikai Wang, Wenbing Huang, Bin Fang, Fuchun Sun, and Chang Li. Elastic tactile simulation towards tactile-visual perception. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2690–2698, 2021. 2.1

- [109] Benjamin Ward-Cherrier, Nicholas Pestell, Luke Cramphorn, Benjamin Winstone, Maria Elena Giannaccini, Jonathan Rossiter, and Nathan F Lepora. The TacTip family: Soft optical tactile sensors with 3D-printed biomimetic morphologies. *Soft robotics*, 5(2):216–227, 2018. [1.1](#), [2.1](#)
- [110] Nicholas Wettels, Veronica J Santos, Roland S Johansson, and Gerald E Loeb. Biomimetic tactile sensor array. *Advanced Robotics*, 22(8):829–849, 2008. [1.1](#)
- [111] Bohan Wu, Iretiayo Akinola, Jacob Varley, and Peter Allen. Mat: Multi-fingered adaptive tactile grasping via deep reinforcement learning. *arXiv preprint arXiv:1909.04787*, 2019. [2.3](#)
- [112] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. [2.2](#)
- [113] Zuxuan Wu, Yu-Gang Jiang, Xi Wang, Hao Ye, and Xiangyang Xue. Multi-stream multi-class fusion of deep networks for video classification. In *ACMMM*, 2016. [2.2](#)
- [114] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *ECCV*, 2016. [2.2](#)
- [115] Tomonori Yamamoto, Nicholas Wettels, Jeremy A. Fishel, Chia-Hsien Lin, and Gerald E. Loeb. Biotac -biomimetic multi-modal tactile sensor. *Journal of the Robotics Society of Japan*, 30(5):496–498, 2012. doi: 10.7210/jrsj.30.496. [2.1](#)
- [116] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. [2.2](#)
- [117] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, 2018. [4.1.3](#)
- [118] Wenzhen Yuan, Rui Li, Mandayam A. Srinivasan, and Edward H. Adelson. Measurement of shear and slip with a gelsight tactile sensor. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 304–311, 2015. doi: 10.1109/ICRA.2015.7139016. [1.1](#)
- [119] Wenzhen Yuan, Rui Li, Mandayam A Srinivasan, and Edward H Adelson. Measurement of shear and slip with a gelsight tactile sensor. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 304–311. IEEE, 2015. [2.1](#)
- [120] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762,

2017. [2.3](#), [4.2.1](#)
- [121] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017. ([document](#)), [1.1](#), [4.1.2](#)
- [122] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. The sound of pixels. In *ECCV*, 2018. [2.2](#)