# Generation of GelSight Tactile Images for Sim2Real Learning

Daniel Fernandes Gomes, Paolo Paoletti, and Shan Luo

***Abstract*—Most current works in *Sim2Real* learning for robotic manipulation tasks leverage camera vision that may be significantly occluded by robot hands during the manipulation. Tactile sensing offers complementary information to vision and can compensate for the information loss caused by the occlusions. However, the use of tactile sensing is restricted in the *Sim2Real* research due to no simulated tactile sensors being available. To mitigate the gap, we introduce a novel approach for simulating a GelSight tactile sensor in the commonly used Gazebo simulator. Similar to the real GelSight sensor, the simulated sensor can produce high-resolution images from depth-maps captured by a simulated optical sensor, and reconstruct the interaction between the touched object and an opaque soft membrane. It can indirectly sense forces, geometry, texture and other properties of the object and enables *Sim2Real* learning with tactile sensing. Preliminary experimental results have shown that the simulated sensor could generate realistic outputs similar to the ones captured by a real GelSight sensor. All the materials used in this letter are available at https://danfergo.github.io/gelsight-simulation.**

***Index Terms*—Deep learning methods, data sets for robot learning, force and tactile sensing, transfer learning.**

## I. Introduction

THE manipulation of objects is prevalent in various applications, e.g., grasping tools, untangling cables and folding pieces of garment. In these tasks, it is essential to track the states of the object being manipulated (shape, pose and the centre of mass etc.). Most methods for robotic manipulation rely on vision-based sensing that allows for a rapid assessment of the scene [1]. However, it can be affected greatly by factors like occlusions and lighting conditions, making the obtained measurements less reliable. In contrast, tactile sensing is not affected by such factors. More importantly, tactile sensing can provide rich contact information between the object and the hand. Nonetheless, tactile sensors are not as widely available as cameras.

Manuscript received October 15, 2020; accepted February 11, 2021. Date of publication March 4, 2021; date of current version April 7, 2021. This letter was recommended for publication by Associate Editor M. Gauthier and Editor D. Popa upon evaluation of the reviewers' comments. This work was supported in part by the EPSRC projects "ViTac: Visual-Tactile Synergy for Handling Flexible Materials" (EP/T033517/1), and in part by the "Robotics and Artificial Intelligence for Nuclear" (EP/R026084/1). *(Corresponding author: Daniel Fernandes Gomes.)*

Daniel Fernandes Gomes and Shan Luo are with the SmARTLab, Department of Computer Science, University of Liverpool, Liverpool L69 3BX, United Kingdom (e-mail: danfergo@liverpool.ac.uk; shan.luo@liverpool.ac.uk).

Paolo Paoletti is with the School of Engineering, University of Liverpool, Liverpool L69 3GH, United Kingdom (e-mail: paoletti@liverpool.ac.uk).

Digital Object Identifier 10.1109/LRA.2021.3063925

To save time and resources, robotic agents can be initially trained within a simulator, so that only a limited number of real experiments are required to fine tune the model before its deployment in the real scenario (*Sim2Real*). A seminal example is [2], where an object detector is pre-trained in randomized simulation environments, and then tested in a real scene. Most of the *Sim2Real* works train models on simulated images that are then transferred to real images. As initial behaviours of the learning agents may be highly unpredictable and can damage delicate tactile sensors, it is also desirable to construct simulated robotic setups equipped with touch sensors, to sense the contact dynamics during manipulation in simulation.

There have been some works to simulate certa in properties of touch sensors, such as friction forces for resistive sensors [3], surface deformations for optical marker-based tactile sensors [4], and tactile contacts (both detection of contacts and contact locations) for capacitive tactile cells in grasping [5]. However, due to the use of dielectrics [4], [5] or tracking sparse points [4], [6], these tactile sensors suffer from low resolution, for instance, a commercial Weiss tactile sensor of 14x6 tactile cells simulated in [3].

In contrast, GelSight optical tactile sensors [7], [8] are able to attain high resolution tactile images, thanks to exploiting the full raw images of the elastomer deformation captured by the embedded camera. It has been widely applied to various perception and manipulation tasks, e.g., geometry and slip measurement [9], localisation [7], texture recognition [10]–[12], and tactile servoing [13]. However, little research has been done on simulating GelSight sensors, which has prevented the exploitation of *Sim2Real* learning for high-resolution tactile sensing.

The main challenge of simulating a GelSight-like sensor is to generate a similar internal view to one captured by the real GelSight camera, that depends on the internal illumination and the membrane deformations of the sensor. To overcome the challenge, in this letter we propose a novel approach for simulating a GelSight sensor in the commonly used robotics simulator *Gazebo*.[1] We leverage a simulated depth camera to capture the surface depth map of the in-contact object. Then, we approximate the heightmap of the deformed membrane by applying Bivariate (2-D) Gaussian filtering. Furthermore, we use the Phong shading model [14] for rendering the sensor internal illumination.

To evaluate our proposed method, we collect a dataset of real tactile images using a GelSight sensor [7] and corresponding virtual tactile images, using a set of small 3D printed objects. As shown in Fig. 1, we use a 3D printer as a Cartesian actuator

[1] http://gazebosim.org/

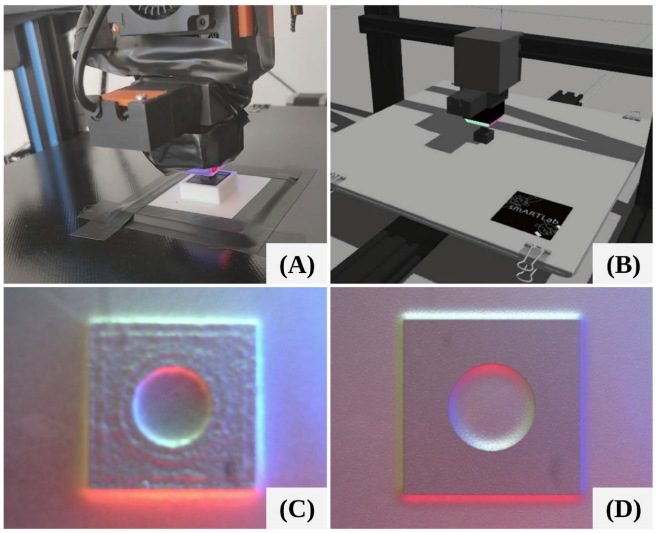


Fig. 1. **(A)** and **(B)**: The real and the simulated experimental setups respectively, in each a GelSight tactile sensor is mounted onto a 3D Printer and contacts a 3D printed object (here is a cube with a hollow cylinder in the centre). **(C)** and **(D)**: The corresponding tactile images captured from the real GelSight sensor in (A) and generated from our proposed simulated GelSight sensor in (B) respectively.

to perform accurate tapping motions and an equivalent setup is reproduced in the *Gazebo* simulator. Both qualitative and quantitative analyses are performed to compare the difference between real and virtual tactile images, which is as low as 8.39% on average in the Mean Absolute Error (MAE) and a similarity of 0.859 in the Structural Similarity Index Measure (SSIM). To illustrate advantages of using our simulation model, *Sim2Real* learning was conducted to classify a set of 21 objects and the results show that the model trained with only generated tactile images, augmented with random texture perturbations, achieves a high accuracy of 76.19% when applied to real tactile images.

Our proposed simulation model of the GelSight sensor was firstly presented in the workshop letter [15] and this letter extends our previous work by including a detailed introduction to the simulation model, a rectified Gaussian filtering, and more thorough experiments. Though the *Gazebo* simulator is used in this work, thanks to the simple model and rendering methods we propose, the simulated model can be easily modified and transferred to other simulators such as Unity[2] and PyBullet.[3]

## II. Related Work

### A. Simulation of Optical Tactile Sensors

Tactile sensors of a wide range of working principles have been developed in the last decades [16], [17]: resistive, capacitive, ultrasonic, magnetic, piezo-electric and optical tactile sensors. Compared to other types, optical tactile sensors, that use cameras to capture the deformations of membranes, have the potential to produce higher resolution tactile images. This advantage is fully exploited by GelSight-like sensors that use the captured raw image, for instance, to reconstruct the contacted surface geometry. In opposition, marker-based optical tactile sensors only track the displacement of sparse markers/pins printed in the soft deformable membrane [18]–[21]. Consequently, the generation of synthetic tactile images depends on the corresponding tactile sensor working principle, especially in simulating the soft membrane physical properties.

It is challenging to simulate the deformation of the elastomer in the contact with another surface. A few approaches take advantage of machine learning algorithms to directly approximate the desired quantities to be measured such as contact forces and incipient slip [22]. In [4], Finite Element Analysis was used instead to model the deformations of the elastomer. In addition to the distribution of the deformations, the holding torque around the contact surface and the stick-slip phenomenon are modeled using the LuGre dynamic friction model in [3]. For simulating marker-based sensors, the markers displacement can be obtained and tracked [23]. The pseudo tactile images can also be generated from real-world data of another modality, for example, visual camera images [10]. In our work, we use Bivariate Gaussian filtering to generate the protruding surface description, so as to mimic the surface deformation of the elastomer. It is easy to compute and can be used in real-time simulations, which enables fast collection of tactile data in simulation and is ideal for *Sim2Real* applications.

### B. Sim2Real Transfer Learning

Data-driven (or learning-based) approaches, introduce two main advantages over handcrafted ones: require less domain-specific knowledge to develop, and have the potential of continuous online improvement. However, one of their major drawbacks is the requirement of extensive amount of data that is often costly to obtain. To address this issue, Transfer Learning has been proposed: a model pre-trained on a more general domain is fine-tuned to the target domain. It has been used in [24] to learn a sensor-specific calibration layer, instead of learning the entire model for every sensor. In the context of robotics, it has evolved into *Sim2Real* transfer learning, i.e., training an agent in a simulated environment, followed by its fine-tuning in the real environment, as simulated data is cheaper to obtain than real data. For example, in [5], a Barrett Hand mounted on a robot arm is trained via Sim2Real transfer learning to grasp using proprioceptive and tactile feedback (only contact locations are used).

## III. The GelSight Working Principle

The GelSight sensors [25] are built using a soft transparent membrane, coated with an opaque elastic paint and placed over a rigid transparent glass, as illustrated in Fig. 2. When an object is pressed against the tactile membrane, the soft elastomer distorts and the geometry of the object is indented onto the elastomer. A view of the pressed surface can then be obtained by a camera enclosed within the opaque rigid shell. Light sources (LEDs) are placed inside the shell to illuminate the elastomer internal surface, making the sensor readings immune to external light variations.

To facilitate the stereographic image processing, light from different colored LEDs is shone from different directions. In the GelSight (2014) [7], four sets of LEDs (red, green, blue and

[2]https://unity.com/
[3]https://pybullet.org/wordpress/

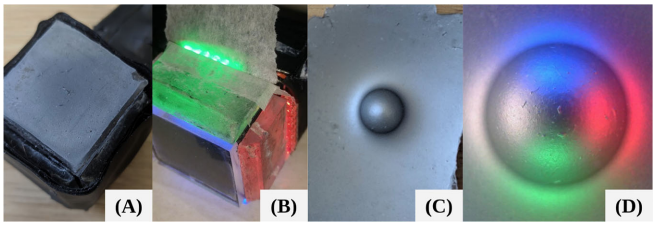


Fig. 2. Multiple views of the GelSight sensor [7]: **(A)** the sensor exterior; **(B)** the LEDs and the light guiding plates embedded in the sensor; **(C)** the tactile membrane after being pressed by a ball; **(D)** the corresponding tactile image captured by the sensor.

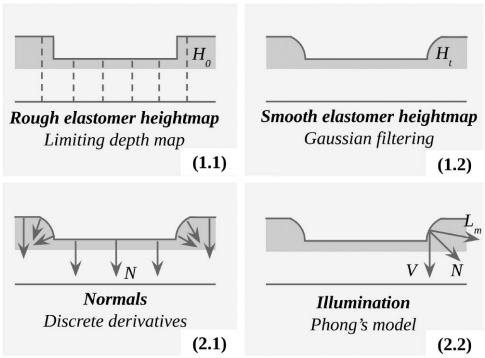


Fig. 3. The two steps in our proposed approach: 1) the elastomer heightmap is first approximated from a depth map captured by a depth camera, by **(1.1)** limiting the depth map and **(1.2)** smoothing it using Gaussian filtering; 2) then the elastomer internal illumination is rendered by **(2.1)** computing its surface normals as discrete derivatives and **(2.2)** applying Phong's illumination model.

white) are used, while in [9] only RGB LEDs are considered. Other variants such as *GelSlim* [26] are equipped only with white LEDs. As a result, different GelSight sensors produce different tactile images. In this work, we leverage the Phong's model to render the internal illumination, which can be parameterized to model any set of directional light sources. Thanks to this, our approach can be trivially configured to simulate any GelSight-like sensor.

## IV. The Proposed Simulation Model

As described in Section III, the core component of any GelSight-like sensor is its deformable tactile membrane that is internally illuminated by multi-color LEDs. We propose a novel approach to generate such tactile images directly from depth maps that can be easily captured in most of the off-the-shelf simulators. As illustrated in Fig. 3, the proposed simulation model consists of two main steps: (1) the heightmap of the elastomer is first computed from the depth map of the object that is in contact with the elastomer; (2) the internal illumination of the elastomer is then computed using Phong's model [14].

### A. The Elastomer Heightmap From the Camera Depth Map

To obtain the elastomer heightmap, a structured-light based depth camera is placed at the same position as the RGB camera in the real sensor, as shown in Fig. 4. The simulated camera captures a depth image $D$ of the object in contact with the elastomer. The obtained depth map is then thresholded by the maximum distance $d_{\max}$ to which the elastomer would be able

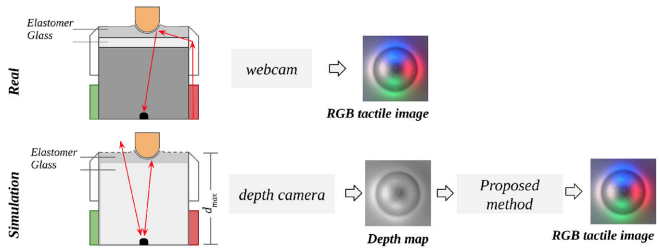


Fig. 4. **Top:** In the real GelSight tactile sensor, the webcam installed in its core directly captures the RGB tactile images. **Bottom:** In the proposed simulated GelSight sensor, a depth map is first captured by a depth camera, from which the virtual tactile image is generated using the proposed simulation method. Note that in the simulated model, both the elastomer and glass are strategically placed such that they are invisible to the depth camera, as detailed in Section V-B.

to contact, resulting in the elastomer height map $H_0$:

$$H_0(x,y) = \begin{cases} D(x,y) & \text{if } D(x,y) \leq D_{\max} \\ d_{\max} & \text{otherwise} \end{cases} \tag{1}$$

where $(x, y)$ is the location of a pixel in $D$.

$H_0$ captures the indentation caused by the object in contact with the elastomer, however, it contains sharp edges resulted from the thresholding. In contrast, the real elastomer presented smoother edges inherent from its elastic properties that generate the color gradients around the in-contact indentation, as shown in Fig. 2-D. To approximate such smooth contact edges, without resorting to more computationally expensive algorithms, 2D (Bivariate) Gaussian filters $G(x, y)$ are applied incrementally over $H_0$:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

$$H'_t = H_{t-1} * G \tag{3}$$

$$H_t(x,y) = \begin{cases} H'_t(x,y) & \text{if } H'_t(x,y) \geq H_0(x,y) \\ H_0(x,y) & \text{otherwise} \end{cases} \tag{4}$$

where $\sigma$ is the standard deviation of $G$, $H_t$ is the elastomer surface approximation after $t \in [1, T]$ steps, $*$ stands for the convolution operation. In each step: 1) $G$ is first applied to the elastomer surface approximation from the previous step $H_{t-1}$ to obtain the smoothed surface $H'_t$ (Equation 3); 2) $H'_t$ is merged with $H_0$ to preserve the sharp features within the in-contact region (4).

To mimic the bump contouring that is raised around the in-contact region by the deformation caused by the contact force, a final heightmap $H_{DoG}$ is computed based on the Difference of Gaussians (DoG):

$$H_{DoG} = 2H_{narrow} - H_{wide} \tag{5}$$

where $H_{narrow}$ is an heightmap approximation computed with a smaller $\sigma$ than the one used in $H_{wide}$. See Section VI-B on the discussion about the elastomer deformation approximation variants.

### B. The Internal Illumination of the Elastomer

The generation of RGB tactile images $I$ from the heightmap $H$ of the elastomer can be interpreted as the inverse problem of the surface reconstruction problem [25], as the former consists of finding the mapping function $I \rightarrow H$ while the latter $H \rightarrow I$. In both cases, the relationship between the two can be described by:

$$I(x, y) = R\left(\frac{\partial H}{\partial x}, \frac{\partial H}{\partial y}\right) \tag{6}$$

where $R$ is the reflectance function that models both the lighting conditions (i.e., illumination of the LEDs) and the reflectance properties of the surface material (i.e., the elastomer coating paint). Here, it should be noted that the color observed at a given pixel is directly correlated with the orientation of the corresponding point on the elastomer.

In [25], the mapping of the two points in the image space and the elastomer is built through a calibration process. In our case, we get $R$ using the Phong's illumination model [14]. Phong's model is an empirical model of local illumination that has been developed in the context of 3D Computer Graphics to describe how a given surface reflects light as a combination of the diffuse and specular reflections.

From Phong's model, $I(x, y)$ observed at a given point of the sensor elastomer is given by three components: ambient, diffuse and specular light, as

$$I(x, y) = k_a i_a + \sum_{m \in L} (k_d(\hat{L}_m \cdot \hat{N}) i_{m,d} + k_s(\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,s}) \tag{7}$$

$$\hat{R}_m = 2(\hat{L}_m \cdot \hat{N})\hat{N} - \hat{L}_m \tag{8}$$

where $L$ is the set of light sources (i.e., LEDs), $\hat{L}_m$ is the emission direction of a given light source $m$; $i_a$ is the intensity of the ambient light that is not caused by any of the LEDs; $i_{m,d}$ and $i_{m,s}$ are the intensities of the diffuse and specular reflections of light source $m$ respectively; $k_a$, $k_d$, $k_s$ and $\alpha$ are all reflectance properties of the surface; $\hat{R}_m$ is the direction that a perfectly reflected ray of the light would take; $\hat{V}$ is the direction pointing towards the camera. Given that our camera is pointing perpendicularly to the elastomer, $\hat{V}$ is set to $< 0, 0, 1 >$. The surface normals $\hat{N}$ are computed using the discrete partial derivatives of the heightmap, as in the surface reconstruction [25]:

$$\begin{aligned} \hat{N} &= < \frac{\partial H}{\partial x}, \frac{\partial H}{\partial x}, -1 > \\ &= < \frac{H}{2r} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \frac{H}{2r} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T, -1 > \end{aligned} \tag{9}$$

where $r$ is a pixel-to-meter ratio obtained through a basic calibration process, detailed in Section V-D.

## V. Experimental Setup

### A. Real World Setup

To produce the necessary real reference dataset, a GelSight sensor [7] is mounted onto a Fused Deposition Modeling (FDM) 3D printer *A30* from Geeetech. One object from a 3D printed object set is placed onto the 3D printer bed and the printer is set to tap the object. As shown in Fig. 1-A, the sensor is fixed to the Tool Center Point of the printer, using a customised 3D printed fixture. Another fixture is attached to the printer base to ensure that all objects stay in the same position during the dataset collection.

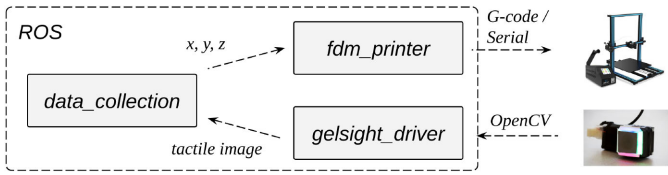


Fig. 5. To collect the (real) reference dataset, the ***data_collection*** node publishes position commands to the ***fdm_printer*** node while receiving tactile images from the ***gelsight_driver*** node. In turn, the ***fdm_printer*** node issues G-code commands to the 3D printer via serial communication, while the ***gelsight_driver*** node uses the *OpenCV* library to capture the images from the sensor's webcam.

To automate data collection using the 3D printer, we create three nodes in the Robot Operating System (ROS) [27], i.e., *data_collection*, *fdm_printer* and *gelsight_driver*. The *data_collection* node orchestrates the data collection process by publishing to the *fdm_printer* node and subscribing to the *gelsight_driver* node, as illustrated in Fig. 5. The *fdm_printer* driver node is implemented within the *ros_control* framework and has a custom *hardware_interface* that sends *G-code* commands to control the 3D printer via serial communication. The *gelsight_driver* is implemented as a vanilla ROS publisher node, using the OpenCV library to interface with the webcam in the tactile sensor.

### B. Virtual World Setup

Similar to the *Real World* setup, the *Virtual World* setup is also comprised of a simulated FDM printer, a simulated GelSight sensor and a set of virtual objects, as shown in Fig. 1-B. The entire setup is run in the *Gazebo* simulator with its default Bullet physics engine. We choose *Gazebo* as it is widely in robotics applications and can be easily integrated with ROS. However, since our approach only requires a simulated depth camera, it can be easily adapted to any other simulators that offer support to depth cameras or the lower level *raycasting* operations.

The virtual 3D printer is modeled after the real one, with the Unified Robot Description Format (URDF). A mesh of the 3D printer is adapted to be its rigid links: the bed (x-axis), the y-axis link and the frame (z-axis). For the actuators, the *ros_control_gazebo_plugin* is used. From the printer's URDF description, the plugin sets the appropriate *hardware_interface* that establishes the communication between *Gazebo* and *ROS* via the *ros_control* framework.

The GelSight sensor is also modeled using the URDF specification. In this case, the mesh used to 3D print the real sensor shell is also used in the specification, together with six extra elements to model the sensor, i.e., the glass, the elastomer and four light guiding plates. The *Gazebo*'s *Openni Kinect* plugin is used to install the virtual depth camera at the core of the sensor. As detailed in Section IV, our proposed approach is used to generate the RGB tactile images from the obtained depth maps from the depth camera.

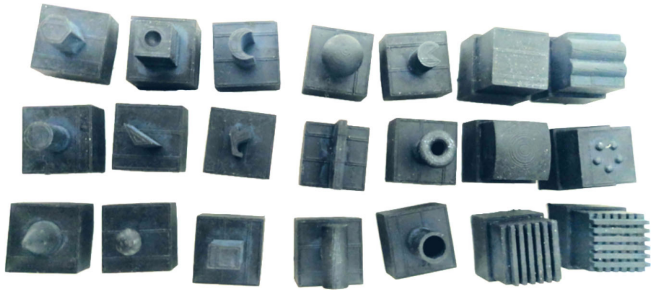

Fig. 6. The objects set. 1st row: Hexagon, Dot-in, Moon, Large Sphere, Pacman, Flat Slab, Wave; 2nd row: Cylinder, Triangle, Random Prism, Line, Torus, Curved Surface, Dots; 3 rd row: Cone, Small Sphere, Rectangular Prism, Side Cylinder, Open Shell, Parallel lines and Crossed Lines.

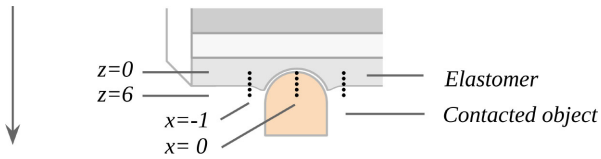


Fig. 7. 2D illustration of the data collection motion. The GelSight sensor installed on the 3D printer is moved from top to bottom, contacting the object in a grid pattern and at multiple heights.

We strategically set the glass and the elastomer as two concentric volumes, with the glass being placed inside the elastomer, as in the real sensor shown in Fig. 2. This ensures that the depth camera sensor sits inside these two volumes so that they are not *hit* by the rays cast by the depth camera plugin, making them invisible in the depth maps. This is particularly necessary for the elastomer that needs to be opaque to look similar to the one on the real sensor. The elastomer is modeled without collision/physical geometry, to allow for objects to penetrate; in contrast, the glass is modeled as collision/physics only, to prevent any object from entering the rigid region of the sensor. The set of objects used in the simulation are the meshes of the 3D printed objects in the Real World setup.

### C. Reference (real) Dataset

The reference (real) dataset consists of tactile images collected in normal contacts against 21 3D printed objects, shown in Fig. 6. The objects were printed using a Stereolithography (SLA) 3D printer *Form 2* from the Formlabs and have different shapes on the top. Each object has a maximum volume of $1 \times 1 \times 2$ cm$^3$.

The contacts are located in a 3D grid of $3 \times 3 \times 11$, captured in 1 mm horizontal steps and 0.1 mm vertical steps, as shown in Fig. 7. The grid is horizontally centered with the object, and its highest position ($z = 0$) is where the very first contact is established in a top-down motion. This results in 99 triplets of $< RGB\,Image, Class, Position >$ for each object. Each raw tactile sample has a resolution of $640 \times 480$.

### D. Virtual Datasets and Baseline Parameters Setup

A virtual dataset is also collected using a similar procedure to the reference dataset collection. However, in this case, depth maps $D$ are captured instead, to enable offline fine-tuning of the simulation method parameters. The corresponding synthetic tactile images are then generated by setting the method parameters manually. As in the real sensor, the captured depth-maps have a spatial resolution of $640 \times 480$.

TABLE I
BASELINE CONFIGURATIONS FOR THE VIRTUAL LEDS

| | $i_{m,s}$, $i_{m,d}$ (RGB) | $\hat{L}_m$ (XYZ) | $k_d$ | $k_s$ |
|---|---|---|---|---|
| White | (255, 255, 255) | (0, 1, 0.25) | 0.6 | 0.5 |
| Blue | (115, 130, 255) | (-1, 0, 0.25) | 0.5 | 0.3 |
| Red | (225, 82, 108) | (0, -1, 0.25) | 0.6 | 0.4 |
| Green | (153, 255, 120) | (1, 0, 0.25) | 0.1 | 0.1 |

To mimic the real GelSight sensor, there are four light sources ($L$): white (top), blue (right), red (bottom) and green (left), as shown in Fig. 1-C. The color emitted by each is sampled from corresponding bright regions in a real tactile image, using GIMP. $k_d$ and $k_s$ of each LED are set based on the observed brightness of the corresponding LEDs. The ambient component $i_a$ is sampled from the corresponding position of a background image captured by the real tactile sensor, i.e., an image captured when the sensor is not in contact with any surface; $k_a$ is set to 0.8. These configurations are listed in Table I. To obtain $r$, a cube of side 5 mm is placed near the virtual tactile sensor, and the distance (in pixels) between the first and last in-contact points of the same row is measured. The approximation parameters of the elastomer deformation are set based on visual inspection: the Bivariate Gaussian Kernel has a size of $21 \times 21$ and $\sigma$ is set to 7; a total of $T = 6$ steps are carried out. The maximum observable depth $d_{\max}$ of the sensor is set to 3 cm, based on the real sensor dimensions, which corresponds to the sum of the distance between the webcam and the elastomer (2.6 cm) and the elastomer thickness (0.4cm).

All the materials used in this letter are available at https://danfergo.github.io/gelsight-simulation. This includes the packages for controlling the real and virtual setups, the collected datasets (both in the real world and in simulation), and the *STL* files for the set of objects and fixtures.

## VI. EXPERIMENTAL RESULTS AND EVALUATION

We evaluate the proposed approach with three sets of experiments. Firstly, we compare the generated tactile images against corresponding real samples, with both quantitative and qualitative analyses; and then, we demonstrate the use of our proposed simulated GelSight sensor in Sim2Real learning for a tactile classification task. As shown in Fig. 8, the generated tactile images look very realistic and quite similar to samples collected using a real GelSight sensor, being able to replicate internal light configurations of different sensors.

### A. Alignment of Real and Generated Samples

As we can see in the previous sections, the *Virtual World* and the *Real World*, and the trajectories executed to collect the real and generated tactile samples, are identical. To this end, the relative movement between two consecutive contacts is expected to be the same for both setups. However, due to the inaccurate placement of the objects onto the 3D printer bed center (in the

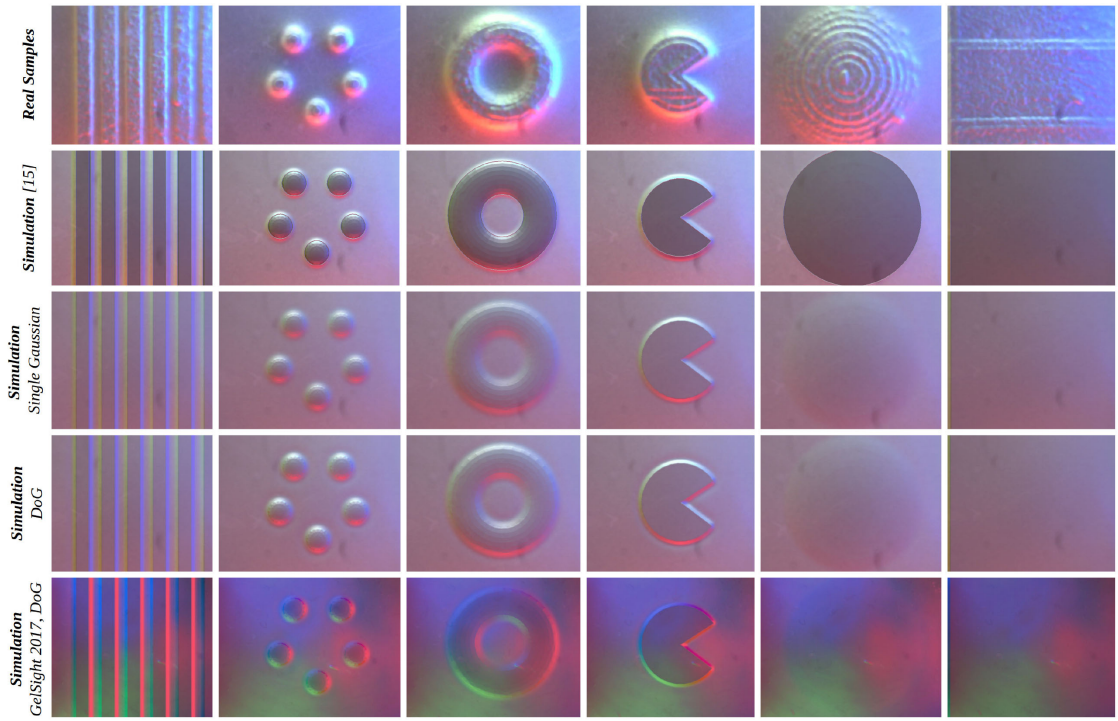


Fig. 8. Samples collected using a GelSight 2014 sensor (top row) and the corresponding simulations: using [15] (2nd row), the "Single Gaussian" (3 rd row) and "Difference of Gaussians" (4th row) for the elastomer heightmap approximation, for a GelSight 2017 [28] sensor (last row). As seen in the listed tactile images, the generated samples look realistic and quite similar to the real ones, being able to replicate internal light configurations of different sensors. For instance, the generated tactile images of the GelSight 2017 blends very well with the background image captured using the real sensor. On the other hand, some differences can be seen in the emitted reflections on areas with squashed, such as on the Torus and the Round Surface shapes.

*Real World* setup), a small positioning error exists between both setups. A scaling error also exists due to inaccuracies in the virtual camera configurations, e.g., the distance from the sensor tactile membrane to the camera and the lens viewing angle.

To mitigate these errors and to ensure a more accurate comparison, we start by using a single alignment pair of the real and virtual tactile images (by positioning the *Dot-in* shape at $< 0, 0, 8 >$) to compute the transformation of the translation and the scaling between the two datasets. Two pairs of corresponding points in the real and virtual frames are selected and constrained to fall in two corresponding vertical lines. A third pair of points is also derived such that the 3 points in each frame form a right isosceles triangle. This constrained formation ensures that the alignment transformation produces only the desired translation and scaling transformation. The OpenCV *getAffineTransform* and *warpAffine* functions are then applied to find the Affine Transformation between the two frames and then the entire dataset. Due to the alignment operation, the transformed (real) frame has a smaller area than its original. As such, both images are cropped to their common area.

Three evaluation metrics are used to evaluate the alignment of the tactile images in the real dataset and the generated dataset: Structural Similarity (SSIM) [29], Peak Signal-to-Noise Ratio (PSNR) and Mean Absolute Error (MAE). As shown in Table II, in the Unaligned case, an average SSIM of 0.731 and an average MAE of 8.40% are obtained, while the Global Alignment method obtains a significantly improved SSIM 0.852 but a slightly worse MAE of 8.80%.

By observing the different frames between aligned and non-aligned pairs, we find that other objects are not aligned as well as the *Dot-in* object. To improve the alignment for each object, a second alignment approach is performed where the Affine transformation is used instead, and it is applied to each object, resulting in aligned tactile samples with a resolution of, on average, $577.15 \times 455.30$. With this Object Alignment, a minimal improvement is obtained for both the SSIM (0.859) and the MAE (8.56%), when compared to the Global Alignment approach, but the MAE is still higher than that of the Unaligned case (8.40%). A probable reason for this is that the affine transformation aligns well for the salient features in the images, achieving a better SSIM, but introduces more distortions to the images at the same time, increasing the absolute errors.

TABLE II
REAL AND GENERATED DATASETS COMPARISON

| | SSIM | PSNR | MAE |
|---|---|---|---|
| Unaligned | $0.731 \pm 0.005$ | $\mathbf{18.85 \pm 3.43}$ | $\mathbf{8.40 \pm 0.04\%}$ |
| Global Align | $0.852 \pm 0.009$ | $18.37 \pm 3.58$ | $8.80 \pm 0.05\%$ |
| Object Align | $\mathbf{0.859 \pm 0.004}$ | $18.58 \pm 3.04$ | $8.56 \pm 0.04\%$ |
| [15] | $0.826 \pm 0.009$ | $17.58 \pm 6.32$ | $10.78 \pm 0.01\%$ |

### B. Analysis of the Elastomer Heightmap Approximation

When approximating the smooth curvature of the sensor elastomer using Gaussian filtering, one issue is that this process also smoothens the existing sharp features within the *in-contact* regions. To mitigate this effect, the initial version of our approach, introduced in the workshop letter [15], segments the tactile image into *in-contact* ($H_0 < d_{\max}$) and *not-in-contact* ($H_0 = d_{\max}$) regions. Then the smoothed and sharp (original)

heightmaps are merged through element-wise multiplication, such that the Gaussian filtering only affects the *not-in-contact* regions. However, in the produced tactile images, shown in Fig. 8 (second row), a sharp contouring around the *in-contact* regions can be noticed that do not reflect the behaviour of the real elastomer.

After extensive analysis, we conclude that this contour artifact is caused by the height discontinuity introduced by the two masks, and the incorrect assumption that elastomer contacts the in-contact object on its entire top-down 2D projection area. This observation leads to the replacement of these masks by the *max* operations. A second issue in the single Gaussian approximation described in [15] is that it does not address the bump contouring around the *in-contact* regions raised by the deformation caused by the contact force. In this letter, we introduce the subtraction of two Gaussians, i.e., Difference of Gaussians, in Equation 5. The heightmaps generated by the "Before Smoothing," smoothed with "Single Gaussian" and smoothed with "Difference of Gaussians" methods are shown in Fig. 10. It can be seen in the figure that the Difference of Gaussians method approximates the elastomer deformation more accurately.

### C. Sim2Real Transfer-Learning for Shape Classification

We then study how models pre-trained using our proposed approach can be applied to real GelSight sensors, i.e., can be used for *Sim2Real* transfer learning. For this goal, we choose tactile image classification as a demonstrative task, i.e., given an image of an object in contact with the sensor, the goal is to predict what class of the 21 objects it belongs to. The real and virtual dataset consist of 2079 ($21 \times 99$) samples each. Splits are randomly defined for training, validation and test with, $21 \times 70$, $21 \times 20$ and $21 \times 9$ samples respectively.

One aspect to consider in the *Sim2Real* learning is the *Sim2Real* gap that results from characteristics of the real world being not modeled in the simulation. In our case, we find that one major difference between the real and synthetic samples are the textures introduced by the 3D printing process. To mitigate this issue, we create twelve texture maps using GIMP that resemble the textures observed in the real samples, as shown in Fig. 11. By randomly perturbing the captured virtual depth-maps with such textures, we are able to produce an effective data augmentation scheme that significantly improves the *Sim2Real* transition, from a 43.76% classification accuracy to 76.19%, in the real data.

A Convolutional Neural Network (CNN) is implemented, with the first Convolutional layers being extracted from a ResNet-50 CNN [30], pre-trained on ImageNet [31], in order to ease the optimization process. These are followed by two 128-$d$ $FC$-$ReLU$-$BatchNorm$ blocks and a final 21-$d$ FC layer with a Softmax activation, which outputs the predicted object class as one-hot encoded vectors. The network is optimized using the Adadelta optimizer, and a step size of 0.1, by minimizing the Categorical Cross-Entropy loss function.

We report the classification accuracies in Table III and its progress during optimization, in Fig. 9. For the *Real2Real* (94.65%) and *Sim2Sim* (82.73%) baselines, the CNN is trained and evaluated using the real and synthetic data-splits accordingly. For the *Sim2Real* experiments, the CNN is trained using the training split of the synthetic dataset, and is evaluated on the validation and test splits of the real dataset. The validation results are used for assessing the training and choosing the best weights, while the test split is only used for final benchmarking. For all the experiments, the training samples are randomly augmented,[4] and

TABLE III
CLASSIFICATION EXPERIMENT RESULTS SUMMARY

| | Validation | Test |
|---|---|---|
| Real2Real | 95.0% | 94.65% |
| Sim2Sim | 86.42% | 82.73% |
| Sim2Real | 43.80% | 43.45% |
| Sim2Real (texture augmented) | **77.38%** | **76.19%** |

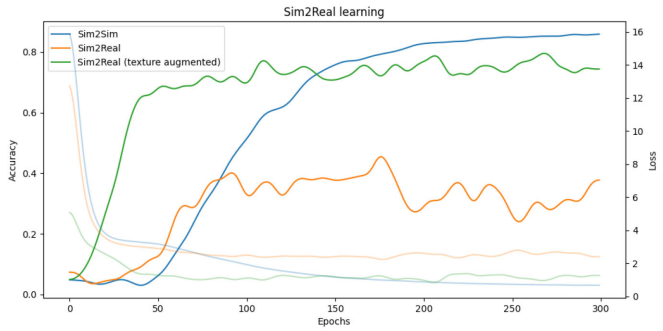


Fig. 9. The evolution of the accuracy and loss values, on the real validation split, while the network is being optimized using synthetic data. As shown, with the texture augmentation scheme described in VI-C, the proposed simulation method can be used for *Sim2Real* transfer learning.

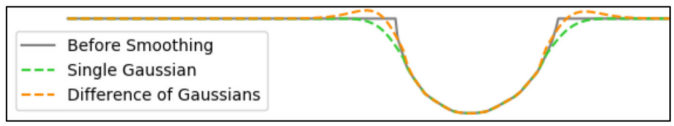


Fig. 10. Elastomer deformation approximation without any smoothing effects ("Before Smoothing"), smoothed with a single Gaussian filter ("Single Gaussian") and smoothed with the DoG ("Difference of Gaussians"). The plotted cross section is extracted from part of an heightmap of the virtual elastomer pressing against the *Dots* object.

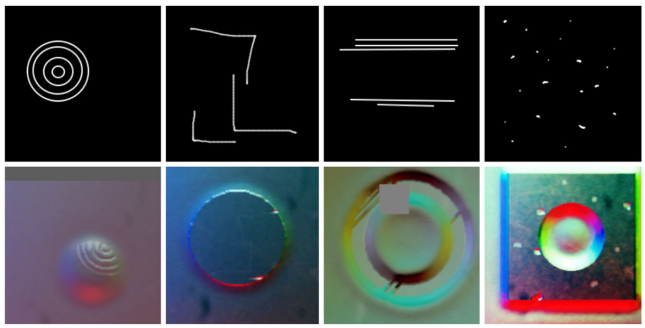

Fig. 11. On the top row, four of the twelve textures created to perturb the captured virtual depth-maps, to address the *Sim2Real* gap. On the bottom row, corresponding augmented samples fed to the Neural Network during training. The augmentation pipeline consists of applying a random geometric transformation (scaling, translation and rotation) to the randomly selected texture, perturbing the raw depth-map with the randomly distorted texture, generating the RGB tactile image using the proposed simulation method, and applying a final random transformation (geometric, color noise and occlusion patches etc.) to this tactile image.

[4]https://github.com/aleju/imgaug

in the *Sim2Real (texture augmented)* case, the synthetic tactile images are also dynamically generated from the depth-map that is perturbed by a randomly selected and randomly distorted texture, as shown in Fig. 11. As the network processes input images of $224 \times 224 \times 3$, during training and evaluation, the tactile images of variable sizes are square cropped and resized to fit the network.

## VII. Conclusion and Discussion

We introduce a novel way of generating synthetic tactile images from a GelSight sensor to enable *Sim2Real* learning with high-resolution tactile sensing. The proposed method was integrated with the widely used Gazebo simulator seamlessly and, as it only depends on the surface function, it can be implemented in any other widely used robotics simulators.

The proposed approach generates synthetic tactile images that would otherwise be captured by a flat GelSight sensor, as proposed in [25]. In [25], lights are installed such that a direct mapping between the surface orientation and the captured pixel intensity exists. As such, the assumptions in this work, i.e., a flat elastomer, directional light sources and no shadows, are sufficiently valid for sensors that are constructed following these working principles, such as the ones proposed in [7], [28]. More recently, other sensors have relaxed some of these constraints in favour of finger-shaped surfaces [8], increased measuring areas [26], or domed elastomers for increased contact [9]. As such, for each sensor different generalizations would have to be performed, even though the main pipeline, i.e., elastomer deformation modeling followed by illumination rendering, should still apply. Some GelSight sensors have also been equipped with elastomers containing printed markers. Such markers facilitate the tracking of the elastomer movements and the generation of corresponding force fields. We have not addressed the markers simulation in this work, however, this has already been carried out in [6], for the TacTip sensor. At this point, GelSight sensors are an entire family with multiple variants, and this work represents the first attempt of simulating a GelSight for *Sim2Real* learning applications. Future works should seek to extend the proposed method for other GelSight-like sensors, such as the GelTip [8], [32].

## References


[1] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019, Art. no. eaat 8414.

[2] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

[3] S. Moisio, B. León, P. Korkealaakso, and A. Morales, "Model of tactile sensors using soft contacts and its application in robot grasping simulation," *Robot. Auton. Syst.*, vol. 61, no. 1, pp. 1–12, 2013.

[4] C. Sferrazza, A. Wahlsten, C. Trueeb, and R. D'Andrea, "Ground truth force distribution for learning-based tactile sensing: A finite element approach," *IEEE Access*, vol. 29, no. 7, pp. 173438–173449, Nov. 2019.

[5] B. Wu, I. Akinola, J. Varley, and P. K. Allen, "MAT: Multi-fingeredadaptive tactile grasping via deep reinforcement learning," in *Proc. 3rd Annu. Conf. Robot Learn.*, vol. 100, 2019, pp. 142–161.

[6] Z. Ding, N. F. Lepora, and E. Johns, "Sim-to-Real transfer for optical tactile sensing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1639–1645.

[7] R. Li *et al.*, "Localization and manipulation of small parts using GelSight tactile sensing," in *Proc. Int. Conf. Intell. Robots Syst.*, 2014, pp. 3988–3993.

[8] D. F. Gomes, Z. Lin, and S. Luo, "Blocks world of touch: Exploiting the advantages of all-around finger sensing in robot grasping," *Front. Robot. AI*, vol. 7, pp. 127–141, 2020.

[9] S. Dong, W. Yuan, and E. H. Adelson, "Improved gelsight tactile sensor for measuring geometry and slip," in *Proc. Int. Conf. Intell. Robots Syst.*, 2017, pp. 137–144.

[10] J.-T. Lee, D. Bollegala, and S. Luo, "'Touching to see' and 'seeing to feel': Robotic cross-modal SensoryData generation for visual-tactile perception," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 4276–4282.

[11] S. Luo, W. Yuan, E. Adelson, A. G. Cohn, and R. Fuentes, "ViTac: Feature sharing between vision and tactile sensing for cloth texture recognition," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2722–2727.

[12] G. Cao, Y. Zhou, D. Bollegala, and S. Luo, "Spatio-temporal Attention Model for Tactile Texture Recognition," in *Proc. Int. Conf. Intell. Robots Syst.*, 2020, pp. 9896–9902.

[13] S. Tian *et al.*, "Manipulation by feel: Touch-based control with deep predictive models," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 818–824.

[14] B. T. Phong, "Illumination for computer generated pictures," *ACM Commun.*, vol. 18, no. 6, pp. 311–317, 1975.

[15] D. F. Gomes, A. Wilson, and S. Luo, "Gelsight simulation for Sim-to-Real learning," in *Proc. IEEE Int. Conf. Robot. Autom. ViTac Workshop*, 2019.

[16] S. Luo, J. Bimbo, R. Dahiya, and H. Liu, "Robotic tactile perception of object properties: A review," *Mechatronics*, vol. 48, pp. 54–67, 2017.

[17] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile sensing-from humans to humanoids," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 1–20, Feb. 2010.

[18] B. Ward-Cherrier *et al.*, "The TacTip family: Soft optical tactile sensors with 3D-Printed biomimetic morphologies," *Soft Robot.*, vol. 5, no. 2, pp. 216–227, 2018.

[19] X. Lin and M. Wiertlewski, "Sensing the frictional state of a robotic skin via subtractive color mixing," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2386–2392, Jan. 2019.

[20] K. Sato, K. Kamiyama, N. Kawakami, and S. Tachi, "Finger-shaped GelForce: Sensor for measuring surface traction fields for robotic hand," *IEEE Trans. Haptics*, vol. 3, no. 1, pp. 37–47, Jan.–Mar. 2010.

[21] C. Sferrazza and R. D'Andrea, "Design, motivation and evaluation of a full-resolution optical tactile sensor," *Sensors*, vol. 19, no. 4, pp. 928–950, 2019.

[22] S. Dong, D. Ma, E. Donlon, and A. Rodriguez, "Maintaining grasps within slipping bound by monitoring incipient slip," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 3818–3824.

[23] C. Sferrazza, T. Bi, and R. D'Andrea, "Learning the sense of touch in simulation: a sim-to-real strategy for vision-based tactile sensing," in *Proc. Int. Conf. Intell. Robots Syst.*, 2020, pp. 4389–4396

[24] C. Sferrazza and R. D'Andrea, "Transfer learning for vision-based tactile sensing," in *Proc. Int. Conf. Intell. Robots Syst.*, 2019, pp. 7961–7967.

[25] M. K. Johnson and E. H. Adelson, "Retrographic Sensing for the measurement of surface texture and shape," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1070–1077.

[26] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez, "GelSlim: A high-resolution, compact, robust, and calibrated tactile-sensing finger," in *Proc. Int. Conf. Intell. Robots Syst.*, 2018, pp. 1927–1934.

[27] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *IEEE Int. Conf. Robot. Autom. Open-Source Software? Workshop*, 2009.

[28] S. Dong, W. Yuan, and E. H. Adelson, "Improved gelsight tactile sensor for measuring geometry and slip," in *Proc. Int. Conf. Intell. Robots Syst.*, 2017, pp. 137–144.

[29] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[32] D. F. Gomes, Z. Lin, and S. Luo, "GelTip: A finger-shaped optical tactile sensor for robotic manipulation," *Int. Conf. Intell. Robots Syst.*, pp. 9903–9909, 2020.