

# Sim2Real Bilevel Adaptation for Object Surface Classification using Vision-Based Tactile Sensors

Gabriele M. Caddeo<sup>1,3,†</sup>, Andrea Maracani<sup>1,3,4,†</sup>, Paolo Didier Alfano<sup>2,4,†</sup>,  
Nicola A. Piga<sup>1</sup>, Lorenzo Rosasco<sup>2,4</sup> and Lorenzo Natale<sup>1</sup>

**Abstract**—In this paper, we address the Sim2Real gap in the field of vision-based tactile sensors for classifying object surfaces. We train a Diffusion Model to bridge this gap using a relatively small dataset of real-world images randomly collected from unlabeled everyday objects via the DIGIT sensor. Subsequently, we employ a simulator to generate images by uniformly sampling the surface of objects from the YCB Model Set. These simulated images are then translated into the real domain using the Diffusion Model and automatically labeled to train a classifier. During this training, we further align features of the two domains using an adversarial procedure. Our evaluation is conducted on a dataset of tactile images obtained from a set of ten 3D-printed YCB objects. The results reveal a total accuracy of 81.9%, a significant improvement compared to the 34.7% achieved by the classifier trained solely on simulated images. This demonstrates the effectiveness of our approach. We further validate our approach using the classifier on a 6D object pose estimation task from tactile data.

## I. INTRODUCTION

Perception of object properties is a fundamental requirement to accomplish everyday tasks. Humans usually have a sense of the object surfaces through vision but sometimes they have to integrate or substitute this information with tactile information. Knowing the kind of surface they are dealing with while manipulating an object can be of primary importance to understand the pose of the object or to decide the next actions.

Among the available tactile sensors, vision-based tactile sensors [1], [2] represent the state of the art when it comes to help robots perceive object properties as they produce high-resolution RGB tactile images of the surface in contact. Such images can be exploited with Deep Learning techniques, despite that they require a huge amount of data for training, which can be difficult to collect in the real world. Although simulators [3], [4] have been proposed to overcome this issue, they hardly reproduce the effect of mechanical properties, e.g., gel deformation, and light distribution with high fidelity.

In this work, we aim to fill the gap between simulated and real images by translating the simulated images to the real domain using a Diffusion Model (DM) trained on few real *unlabeled* images coming from a DIGIT sensor [1]. We propose a surface classifier which can distinguish among four

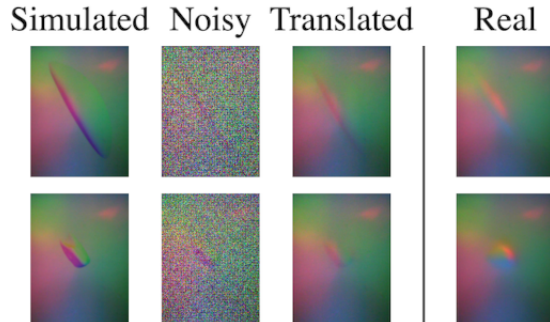


Fig. 1. Our pipeline uses a Diffusion Model to translate simulated images towards the real domain so as to reduce the Sim2Real gap.

classes: flat, curve, edge and corner. The classifier is trained with simulated images that are sampled on the surface of objects from the YCB Model Set [5] and then translated with the DM. To label the images, we sample a point cloud on the object mesh and we employ an *automatic* procedure to evaluate, for each point, the local curvature from which we extract the labels. Notably, the overall procedure to generate training data does not require manual annotations, as it uses a small set of unlabelled data from real contacts and a larger dataset that is acquired in simulation and automatically labelled.

We test the classifier on real tactile images acquired on ten YCB objects. We compare our model by training the classifier with simulated images or with images converted with an alternative DM-based model from the literature [6] having more complex training requirements than ours. The experiments show that our method can achieve better accuracy in the classification task. Moreover, we employ our classifier within a pipeline for 6D object pose estimation [7] from multiple tactile sensors.

We summarize our contributions as follows:

- An image translation procedure to fill the Sim2Real gap using a DM which easily generalizes to different vision-based tactile sensors;
- An object-agnostic surface type classifier trained using simulated images and few real images;
- A method to automatically label the object surfaces given the object mesh, that we use to train the above classifier.

## II. RELATED WORK

This work draws from the literature on both Sim2Real translation for vision-based tactile sensors and perception of object properties using the same sensors.

<sup>1</sup>Humanoid Sensing and Perception, Istituto Italiano di Tecnologia, Genoa, Italy. name.surname@iit.it

<sup>2</sup>IIT@MIT, Istituto Italiano di Tecnologia, Genoa, Italy. name.surname@iit.it

<sup>3</sup>DIBRIS, Università di Genova, Via All'Opera Pia, 13, Genoa, Italy.

<sup>4</sup>MaLGA Center, DIBRIS, Università di Genova, Genoa, Italy.

†Equal contribution.

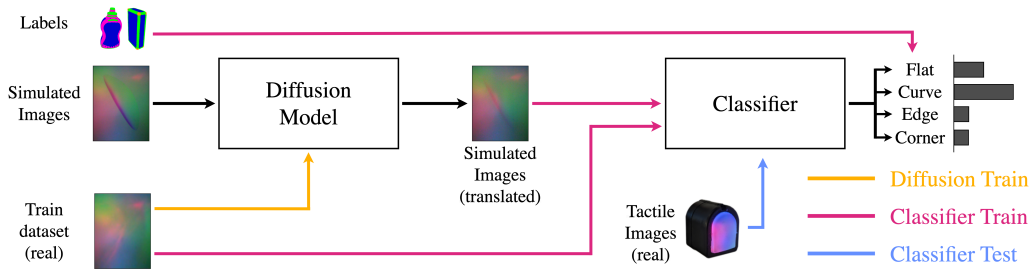


Fig. 2. Overview of our pipeline for object surface classification. The Diffusion Model, trained on real images, translates the simulated images. Such images are used to train the classifier which categorizes surfaces from real images at test time. The colored arrows indicate the time level at which we require specific input. During inference, only the test image is needed

**Sim2Real** Some work attempts to reduce the Sim2Real gap by carefully emulating data from real sensors. In [8] and [9] the authors mimic the behaviour of the GelSight sensor using Phong’s model. In [3], the authors present a simulator for the DIGIT sensor based on OpenGL using pyrender. A limitation of these methods is that they do not take into account the mechanical properties of the sensors. In [4] the authors integrate the dependency on the mechanical properties of the gel by adding a calibration procedure based on the sensor output. Other approaches attempt to mitigate the domain shift between simulated and real images. In [10], the authors try to bridge the Sim2Real gap by employing a CycleGAN [11] to simulate the complex light transmission observed in real sensors, a characteristic not captured by the simulator. In contrast, our work utilizes simulated images, from TACTO [3], which undergo a transformation via a DM trained on real images, to mimic the real deformation of the gel and light transmission of the sensor. A similar approach is considered in [6], where, however, the DM is trained with additional conditioning depth images that are obtained from a network [12] that had been previously trained. In our case, we dispense with the need for this network and rely solely on RGB images. In the experimental section, we compare with a variant of our pipeline where the DM for image translation is substituted with that of [6].

### Object perception with vision-based tactile sensors

To the best of our knowledge, no work directly classifies the surfaces of an object using vision-based tactile sensors. Nonetheless, we refer to other works which infer similar properties of the object such as shape estimation or identification of possible points of contact on the surface. In [13] the authors retrieve the shape of unknown objects with an active tactile exploration method. However, these methods need to fuse the proprioception with multiple touches to reconstruct the local surface. In [14] the authors integrate vision with touch to infer the shape of an object. In our work, no visual information is needed. In [15] the authors train a network to retrieve the location and shape of the contacts to infer the shape of the object while the sensor slides along the object’s surface. In [16], the authors reconstruct the normals of the local surface of small objects thanks to a network trained with real and simulated images. On the contrary, we concentrate on objects whose size is not comparable to that of the sensor surface, thus producing more ambiguous tactile images and

on general classes of surfaces. In [12], [7] the authors identify the possible contact point on the object surface by comparing the input image with a database of images that are sampled on the object surface in advance, while our work does not need any database at inference time.

### III. METHOD

Our approach leverages unlabeled real-world data and labeled simulated data to obtain good classification performance in real-world scenarios. To achieve this goal, we devise an automated method for acquiring and labeling synthetic data. We incorporate two levels of adaptation to mitigate domain shift and enhance performance. Specifically, we employ a probabilistic DM to translate the simulated images. We further adapt the model features through an adversarial process using the Domain-Adversarial Training of Neural Networks (DANN) method [17].

We remark that we always remove the background signal of the DIGIT, i.e. its RGB output when not in contact with an object, from both real and simulated images. As DIGIT sensors can exhibit slight background variations owing to manufacturing differences, this ensures that the methodology remains agnostic to the specific background.

This section offers a comprehensive overview of all the components illustrated in Fig. 2 within our pipeline.

#### A. Acquisition and labeling of simulated data

We employ Poisson disk sampling [18] to extract uniformly distributed point clouds from object meshes augmented with surface normals. For each point we simulate the image produced by the DIGIT, using the simulator [3], while considering several rotations of the sensor around the normal direction and several penetration depths, so as to ensure variability in the collected data.

To automate the labeling process, we devised a simple yet effective algorithm to proficiently categorize each point within the point clouds as either *flat*, *curve*, *edge*, or *corner*. Let suppose to have a point cloud of  $M$  points:  $P = \{p_i\}_{i=1}^M$ , where  $p_i \in \mathbb{R}^3$ . For every point  $p_i \in P$  the algorithm, which operates in four distinct steps, computes automatically the class of the point:

- 1) The neighborhood  $N_i$  of  $p_i$  is computed:  $N_i = \{p \in P : \|p_i - p\|^2 \leq R\}$ , where the radius  $R$  is an hyperparameter of the algorithm.

- 2) We extract the ordered singular values of the  $3 \times 3$  covariance matrix of the points in  $N_i$ , i.e.,  $\sigma_1^{(i)} \geq \sigma_2^{(i)} \geq \sigma_3^{(i)}$ .
- 3) We define the curvature level of the point  $p_j$  as the ratio:

$$\text{Curv}(p_i) := \frac{\sigma_3^{(i)}}{\sigma_1^{(i)} + \sigma_2^{(i)} + \sigma_3^{(i)}} \quad (1)$$

This method provides an estimation of the local surface behavior. Intuitively, when the local curvature level is 0, the local area is entirely flat (as the smallest singular value  $\sigma_3 = 0$ ) and the points would be distributed just on the plane defined by the first two eigenvectors corresponding to  $\sigma_1$  and  $\sigma_2$ . Conversely, when the local curvature level is  $1/3$  (the maximum value attainable by the curvature function), the surface displays pronounced curvature, as all three singular values are equal. By setting two thresholds ( $0 < t_1 < t_2 < 1/3$ ) on the local curvature level, we are able to partition and classify all the points into three categories: *flat*, *curve*, and *hard-curve*.

- 4) Finally, points previously classified as *hard-curve* are further separated into *edges* and *corners*, a task that cannot be achieved using curvature levels alone. In this step, we utilize K-means clustering on the *normal* vectors of the neighborhood of *hard-curve* points to classify them as either *edges* or *corners*. Intuitively, edges are defined by two flat surfaces represented by two normal vectors. Consequently, the normal vectors of the neighborhood of edge points should be easily clustered with  $K = 2$ , and the advantage of using  $K = 3$  should be negligible. On the other hand, corner points are defined by three planes, and the normal vectors of the neighborhood cannot be easily clustered with  $K = 2$ . Following this intuition, we compute the following quantity:

$$\Delta\text{loss}_{23} := \text{loss}(K = 2) - \text{loss}(K = 3), \quad (2)$$

where  $\text{loss}(K = q)$  is the final loss of K-Means when  $K$  is set to  $q$ . A threshold on the value of  $\Delta\text{loss}_{23}$  is set to partition *edges* and *corners*: low values of this metric identify edges, while high values identify corners.

We show the effectiveness of the overall approach for two YCB objects in Fig. 3.

### B. Image-level adaptation

As simulated images and actual images acquired with a DIGIT sensor exhibit notable differences (see Fig. 1), we propose an unsupervised image-to-image translation method to address the domain shift between these two domains. Our approach leverages the reverse process of a probabilistic DM. DMs are latent variable models that involve two processes: the forward (diffusion) process and the reverse process. Let  $x_0$  represent an image (in our work, a real image acquired with a DIGIT sensor). The forward process is a fixed Markov chain that gradually introduces Gaussian noise to  $x_0$  over a

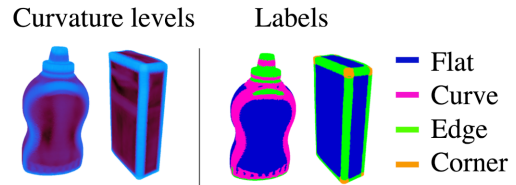


Fig. 3. On the left: curvature levels (Eq. 1) for the YCB objects “mustard bottle” and “sugar box”. On the right: the resulting labels of surface types.

predefined number of steps  $T$ , as determined by a variance schedule defined by the parameters  $\{\beta_j\}_{j=1}^T$ . Specifically, at any given time step  $t \in 1, \dots, T$ , the forward process adds Gaussian noise to the image according to the following transition:

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1 - \beta_t} \cdot x_{t-1}, \beta_t I) \quad (3)$$

where  $\mathcal{N}$  is a Gaussian distribution and  $I$  is the identity matrix. The reverse process is also a Markov chain with Gaussian transitions that can be learned using a model, such as a neural network. This allows us to recover the image from the previous step ( $x_{t-1}$ ) given the noisy image at step  $t$  ( $x_t$ ). By iterating this process, we can ultimately obtain the fully denoised image,  $x_0$ . To be more specific, the reverse transitions are as follows:

$$p(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)) \quad (4)$$

Here,  $\mu(\cdot, \cdot)$  and  $\Sigma(\cdot, \cdot)$  represent functions that can be learned during the training process of the DM. In our work, we utilized a U-Net architecture [19], which is commonly employed in the literature.

Following the training, it becomes possible to sample an image with random Gaussian noise (bearing a strong resemblance to the images generated in the forward process at step  $T$ ) and iteratively denoise it. This iterative process enables the generation of an image from the distribution of the training dataset. For more intuitions and a more detailed description of probabilistic DMs the reader can refer to [20] and [21].

To narrow the domain gap between simulated and real images, our approach encompasses the following steps:

- 1) We train a U-Net model solely with unlabeled real images acquired with the DIGIT sensor to acquire knowledge of the reverse process.
- 2) Once trained, we introduce a moderate level of noise to the simulated images, advancing in the forward process until step  $T' < T$ , with the aim of altering the *style* of the images while preserving their semantic information.
- 3) We employ the reverse process learned in step 1 to denoise the images, moving from step  $T'$  to step 0. As the DM was trained using real images, the output comprises images that retain the semantic information of the simulated ones but exhibit a style more akin to real images as shown in Fig. 1.

Unlike previous works, such as [6], our diffusion method does not necessitate additional information during training or inference. It operates exclusively on raw images without any

conditioning. The *semantic conditioning* for image generation is accomplished, as mentioned earlier, by introducing a controlled level of noise into the simulated images, as it is similarly done in [22]. This preserves crucial information while preventing complete degradation.

### C. Feature-level adaptation

Despite the significant reduction in domain shift achieved by the DM, some residual differences between the domains still exist. To address this, we adopted a classical yet effective adversarial approach, known as Domain-Adversarial Training of Neural Networks (DANN), to facilitate the learning of domain-invariant representations. Specifically, we utilized a Vision Transformer (ViT) [23], pretrained with DINO v2 method [24], as a feature extractor, keeping it fixed throughout the training process. We introduce and train a bottleneck layer (a linear layer, followed by Layer Normalization [25] and a GELU [26] activation) to map ViT features into a domain-invariant space, along with a classifier that maps these bottleneck features to our target classes, as depicted in Fig. 4.

During training, we employ a discriminator to distinguish between real and simulated images, while the bottleneck layer is optimized to deceive the discriminator by making the features from both domains indistinguishable. To be more precise, we define  $\phi(\cdot)$  to be our feature extractor (ViT) that maps images to features,  $\beta(\cdot)$  as our bottleneck that reduces the dimension of features to 256,  $\gamma(\cdot)$  as a classifier that maps bottleneck features to our four classes and  $\psi(\cdot)$  as a discriminator that maps bottleneck features to a domain label: 0 for simulated images and 1 for real images. During training we sample a batch of real images (without labels)  $x_{real}$  and a batch of simulated images (with labels)  $(x_{sim}, y_{sim})$ . We compute the bottleneck features ( $\hat{z}$ ), the predictions of the network  $\hat{y}_{sim}$  and the domain predictions of the discriminator  $\hat{d}_{all}$  as follow:

$$\hat{z}_{real} = \phi(\beta(x_{real})) \quad (5)$$

$$\hat{z}_{sim} = \phi(\beta(x_{sim})) \quad (6)$$

$$\hat{y}_{sim} = \gamma(\hat{z}_{sim}) \quad (7)$$

$$\hat{d}_{all} = \psi(\hat{z}_{all}) \quad (8)$$

where  $\hat{z}_{all}$  is the concatenation of  $\hat{z}_{sim}$  and  $\hat{z}_{real}$  along the batch dimension of the two inputs.

The bottleneck and the classifier are trained to minimize the following loss:

$$L_{cls} = \text{CE}(y_{sim}, \hat{y}_{sim}) - \alpha \cdot \text{BCE}(d_{all}, \hat{d}_{all}) \quad (9)$$

where  $d_{all}$  are all the domain labels (0s and 1s based on the domain of the inputs), CE is the cross-entropy loss, BCE is the binary cross-entropy loss and  $\alpha$  is a hyperparameter that we set to 1.2 in all of our experiments.

The discriminator, on the other hand, is trained to minimize:

$$L_{dis} = \alpha \cdot \text{BCE}(d_{all}, \hat{d}_{all}) \quad (10)$$

For this reason the training proceeds as an adversarial game between the bottleneck and the discriminator. In practice,

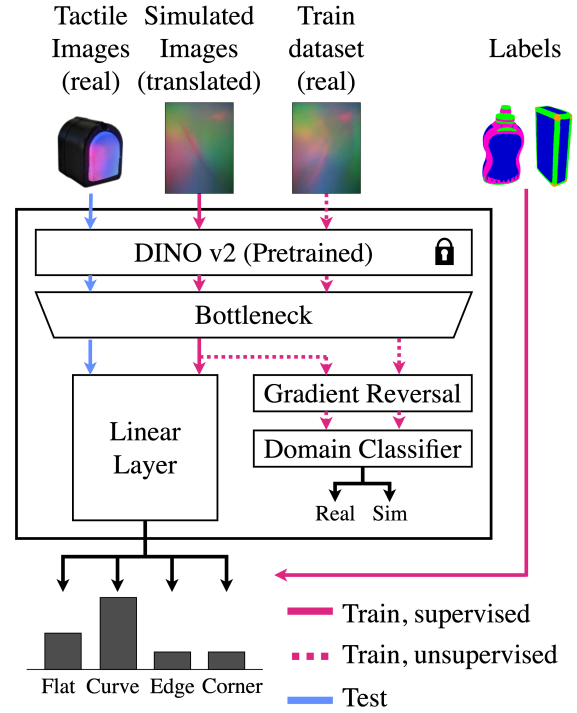


Fig. 4. Diagram of the classification architecture.

we trained the whole network (with the exception of the feature extractor that is fixed) in an end-to-end fashion using a Gradient Reversal Layer (GRL) as proposed in DANN [17] (refer to this work for more details).

### D. Training and testing datasets

For our experiments, we collected three different datasets: The first dataset comprises 5,000 real images acquired using a DIGIT sensor on the surfaces of randomly selected everyday objects, excluding all objects from the YCB Model Set [5]. We refer to this *unlabeled* dataset as **Train<sub>real</sub>**.

The second dataset consists of 50,000 simulated images, 12,500 per class, acquired from randomly selected points sampled on the meshes of ten YCB objects (*cracker box, tomato soup can, tuna fish can, pudding box, gelatin box, banana, bleach cleanser, bowl, power drill, wood block*) and translated to the real domain as in Sec. III-B. We employ the algorithm described in Section III-A to associate a ground truth label for each image, and we refer to this dataset as **Train<sub>sim</sub>**.

The third dataset includes 792 real images acquired using our setup from ten 3D printed YCB objects (*master chef can, sugar box, mustard bottle, tuna fish can, potted meat can, banana, pitcher base, bleach cleanser, bowl, power drill*) (see Fig. 5). For every tactile image we manually label the type of surface depending on the actual type of contact. We use the latter as the ground-truth label for evaluation purposes. We refer to this dataset as **Test<sub>real</sub>**.

To demonstrate the generalization capabilities of our approach, we intentionally include 5 overlapping objects (*tuna fish can, banana, bleach cleanser, bowl, power drill*) in both the **Train<sub>sim</sub>** and **Test<sub>real</sub>** datasets. As a result, our

TABLE I  
RESULTS OF THE CLASSIFICATION EXPERIMENTS:  
ACCURACY FOR EACH OBJECT AND AVERAGED.

Image transl.	None		Tactile Diff. [6]		Ours	
	✗	✓	✗	✓	✗	✓
<b>Train w/ DANN</b>						
master chef can	52.3%	41.2%	58.7%	80.9%	66.6%	<b>92.0%</b>
sugar box	35.0%	35.0%	84.0%	71.0%	57.9%	<b>93.0%</b>
mustard bottle	41.0%	48.0%	34.0%	60.0%	63.0%	<b>71.0%</b>
tuna fish can	57.5%	68.1%	59.0%	<b>80.3%</b>	74.2%	75.7%
potted meat can	35.0%	43.0%	54.0%	76.0%	<b>87.0%</b>	84.0%
banana	2.4%	29.2%	29.2%	7.3%	56.0%	<b>78.0%</b>
pitcher base	40.6%	39.0%	85.9%	90.6%	64.0%	<b>98.4%</b>
bleach cleanser	38.0%	43.0%	39.0%	64.0%	69.0%	<b>84.0%</b>
bowl	40.3%	43.5%	62.9%	88.7%	53.2%	<b>90.3%</b>
power drill	3.1%	33.3%	50.0%	38.54%	23.9%	<b>60.4%</b>
<b>Mean</b>	<b>34.7%</b>	<b>42.4%</b>	<b>55.6%</b>	<b>66.6%</b>	<b>61.6%</b>	<b>81.9%</b>

algorithm performs well on objects seen during training and also on new objects. We used the unlabeled  $\text{Train}_{real}$  dataset for training the DM, while the classifier is trained (with DANN) using both  $\text{Train}_{sim}$  and the unlabeled  $\text{Train}_{real}$ . The testing phase was exclusively conducted on  $\text{Test}_{real}$ .

#### IV. EXPERIMENTAL RESULTS

We evaluate the performance of the classifier by considering the accuracy for each object and the F1-Score when analysing the results class-wise. In the results we also consider a variation of the pipeline where the proposed DM is substituted with the state-of-the-art alternative [6]. Moreover, we perform several ablation studies in order to investigate the role of the DM and of the DANN procedure. Beyond the classification task, we apply our method within a pipeline for 6D object pose estimation from multiple tactile sensors [7], to show its effectiveness on a practical task. In order to collect the  $\text{Test}_{real}$  dataset, we use a DIGIT sensor mounted on a 7-DoF Franka Emika Panda robot to touch ten 3D-printed YCB objects in several configurations. We show the setup and examples of the printed objects in Fig. 5.

In order to evaluate the outcome of the 6D object pose estimation experiments, we also collect the pose of the object in the robot root frame. To do so, we employ a fiducial system based on a RealSense camera and ArUco markers. To make sure that the object does not move while touching it we fixed it to a table as shown in Fig. 5. We remark that the necessity to fix the objects is the only reason for printing them in 3D. The results of further experiments with the original objects are provided in the supplementary video.

Code and data will be made publicly available online<sup>1</sup>.

##### A. Experiments on surface classification

Table I presents the results in terms of *accuracy* for each considered object for the test sensor. In Table II, we instead detail the results for each class using the F1-Score defined as follows:

$$\text{F1-Score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

where *precision* and *recall* are defined as usual.

In order to investigate how the image translation and the DANN procedure contribute to the performance we repeated

<sup>1</sup><https://github.com/hsp-iiit/sim2real-surface-classification>

TABLE II  
RESULTS OF THE CLASSIFICATION EXPERIMENTS:  
F1 SCORE FOR EACH SURFACE TYPE.

Image transl.	None		Tactile Diff. [6]		Ours	
	✗	✓	✗	✓	✗	✓
<b>Train w/ DANN</b>						
flat	46.5%	28.6%	61.5%	81.3%	86.4%	<b>91.1%</b>
curve	3.6%	31.1%	18.6%	30.3%	45.2%	<b>73.5%</b>
edge	28.%	56.2%	78.1%	74.4%	58.1%	<b>83.2%</b>
corner	0.0%	0.0%	46.6%	26.3%	21.1%	<b>68.3%</b>



Fig. 5. On the left: the testing setup with the DIGIT sensor touching the “bleach cleanser” object. On the right: example of YCB objects and their 3D printed counterparts.

the experiments using different training configurations of the pipeline. As regards the image translation we trained the classifier using:

- simulated images, indicated as “None”;
- translated images from the DM proposed in [6], indicated as “Tactile Diffusion”;
- translated images from the DM proposed in this work, indicated as “Ours”.

For each configuration above, we repeated the experiments with the DANN procedure enabled or disabled. The results in Table I show that the best performance, except few cases, is achieved when using the proposed DM as the translation mechanism combined with the DANN procedure. Using DANN, the proposed DM increases the accuracy by  $\approx 40$  points with respect to the “None” case and by  $\approx 15$  points with respect to the method [6]. As regards the DANN procedure, on average it helps increase the accuracy in all the configurations. The best improvement, of  $\approx 20$  points, is obtained in conjunction with the proposed DM. Table II shows that the proposed pipeline achieves the best performance on all surface types. Remarkably, both the DM and DANN are essential to improve performance on all classes, and specifically to gain acceptable performance on the class corners.

In Fig. 6 we present some qualitative results. Specifically, for each tactile image, we show the name of the predicted class and all the points sampled on the object mesh, as per Sec. III-A, having the same label as the predicted one. These results indicate that a combination of the proposed classifier and the automatic labeling procedure of Sec. III-A is useful to provide hypotheses on the contact location of the sensor on the object surface.

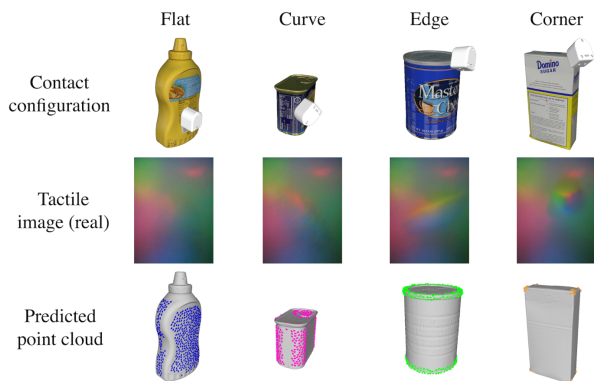


Fig. 6. Visualization of the predicted surface type in terms of the object-sampled points having the same label as the predicted one.

### B. Experiments on 6D object pose estimation

For these experiments we make use of the algorithm presented in [7] that estimates the 6D pose of an object in contact with  $N$  tactile sensors given input tactile images and the pose of the sensors from the robot proprioception. Specifically, [7] extracts several hypotheses on the 3D location of each sensor on the object surface given the input tactile image. Then, it uses gradient-based optimization to fuse the hypotheses from all sensors resulting in a set of candidate 6D poses of the object. For a more detailed description the reader can refer to [7]. In this paper, we substitute the hypothesis extraction part, based on a convolutional autoencoder and a set of per-object databases of latent features, with the proposed classifier. We choose to run the experiments using  $N = 3$  sensors. As in [7], we considered four different configurations of sensors for every object under study combining the poses collected for the classification experiments. We compare the performance against a purely geometric baseline, as done in [7]. The baseline skips the hypothesis extraction part and executes the optimization assuming that the  $N$  sensors might touch the object everywhere. We don't compare against the convolutional autoencoder of [7] due to its poor performance on real images. Indeed, unlike [7], our evaluation is based on real tactile images and their related poses collected with the robot. To evaluate the performance we compare the output pose with the ground truth pose in terms of the positional error and a variant of the ADI-AUC metric [27], introduced in [7], which reports on the rotational error. Table III shows that using the tactile feedback, in the form of the proposed classifier, helps halve the positional error and increasing the rotational metric by more than ten points, on average. In Fig. 7 we show a sample outcome of the experiment and compare it against the baseline. The ground-truth pose is shown as the greyed-out mesh while the estimated pose as the red point cloud. As can be seen, although the pose estimated by the baseline is compatible with the *position* of the contacts, two out of three sensors are in contact with the wrong part of the object. Instead, the pose estimated by our method is compatible with all contact positions and types. This further demonstrates the advantages of using tactile feedback, in the form of the proposed classifier, for this task.

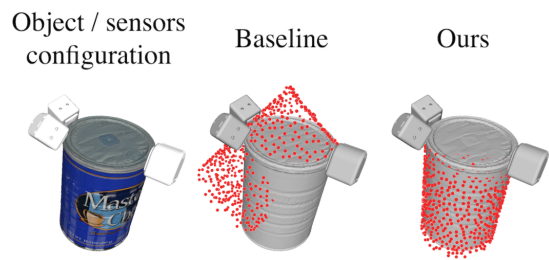


Fig. 7. Visualization of the outcome of the 6D object pose estimation experiment using real tactile images.

TABLE III

RESULTS OF THE OBJECT POSE ESTIMATION EXPERIMENTS: AVERAGED POSITIONAL ERROR AND ADI-AUC WITH THRESHOLD SET TO 2 cm.

Metric	Positional error (cm) ↓		ADI-AUC <sub>2cm</sub> (%) ↑	
Method	Baseline	Ours	Baseline	Ours
master chef can	2.97	<b>0.69</b>	93.54	<b>97.23</b>
sugar box	3.61	<b>3.12</b>	<b>93.75</b>	72.61
mustard bottle	2.97	<b>1.83</b>	69.83	<b>96.61</b>
tuna fish can	2.18	<b>1.07</b>	<b>96.08</b>	95.97
potted meat can	1.81	<b>1.26</b>	94.71	<b>96.63</b>
banana	6.72	<b>2.39</b>	48.49	<b>96.93</b>
pitcher base	6.71	<b>2.73</b>	25.0	<b>47.19</b>
bleach cleanser	4.97	<b>2.96</b>	70.34	<b>94.28</b>
bowl	<b>1.74</b>	2.08	<b>97.29</b>	96.27
power drill	8.36	<b>4.09</b>	67.77	<b>73.76</b>
<b>Mean</b>	4.21	<b>2.22</b>	75.68	<b>86.75</b>

### V. LIMITATIONS

The rigidity of the elastomer of the DIGIT sensor requires a more than moderate force, when interacting with objects, to highlight surface differences. Although this fact depends on the sensor itself, it might impact the effectiveness of our method if the contact forces are too weak.

We acknowledge that our method makes use of a DM whose training and image-translation times are non-negligible. Nonetheless, one of the advantages of the proposed method is that it is trained on images without background, thus it can be used on different devices without a re-train.

### VI. CONCLUSION

In this work, we tackled the Sim2Real gap in the context of vision-based tactile sensors to classify local surfaces of objects. Our method combines a bilevel adaptation, at the image and feature level, with an automatic labeling procedure, allowing us to train the classifier using a supervised approach while avoiding manual annotation. The extensive experiments on real data and the comparison against a state-of-the-art method validate the robustness and the effectiveness of our approach that we also applied successfully in the context of 6D object pose estimation from tactile data.

As future work, we plan to exploit our classification approach for several other robotic tasks and to study new adaptation mechanisms to further increase the classification accuracy. Moreover, we plan to handle classification of simultaneous surfaces under touch.

### ACKNOWLEDGMENT

We acknowledge financial support from the PNRR MUR project PE0000013-FAIR.

## REFERENCES

- [1] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, D. Jayaraman, and R. Calandra, "DIGIT: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, jul 2020.
- [2] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017.
- [3] S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra, "Tacto: A fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3930–3937, 2022.
- [4] Z. Si and W. Yuan, "Taxim: An example-based simulation model for gelsight tactile sensors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2361–2368, 2022.
- [5] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.
- [6] C. Higuera, B. Boots, and M. Mukadam, "Learning to read braille: Bridging the tactile reality gap with diffusion models," 2023.
- [7] G. M. Caddeo, N. A. Piga, F. Bottarel, and L. Natale, "Collision-aware in-hand 6d object pose estimation using multiple vision-based tactile sensors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 719–725.
- [8] D. F. Gomes, P. Paoletti, and S. Luo, "Generation of gelsight tactile images for sim2real learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4177–4184, 2021.
- [9] F. R. Hogan, M. Jenkin, S. Rezaei-Shoshtari, Y. Girdhar, D. Meger, and G. 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*, 2021, pp. 1218–1227.
- [10] W. Chen, Y. Xu, Z. Chen, P. Zeng, R. Dang, R. Chen, and J. Xu, "Bidirectional sim-to-real transfer for gelsight tactile sensors with cyclegan," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6187–6194, 2022.
- [11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.
- [12] S. Suresh, Z. Si, S. Anderson, M. Kaess, and M. Mukadam, "Midas-touch: Monte-carlo inference over distributions across sliding touch," in *Conference on Robot Learning*. PMLR, 2023, pp. 319–331.
- [13] A.-H. Shahidzadeh, S. J. Yoo, P. Mantripragada, C. D. Singh, C. Fermüller, and Y. Aloimonos, "Actexplore: Active tactile exploration on unknown objects," 2023.
- [14] W. Xu, Z. Yu, H. Xue, R. Ye, S. Yao, and C. Lu, "Visual-tactile sensing for in-hand object reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8803–8812.
- [15] Y. Chen, A. E. Tekden, M. P. Deisenroth, and Y. Bekiroglu, "Sliding touch-based exploration for modeling unknown object shape with multi-fingered hands," *arXiv preprint arXiv:2308.00576*, 2023.
- [16] P. Sodhi, M. Kaess, M. Mukadam, and S. Anderson, "Patchgraph: In-hand tactile tracking with learned surface normals," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2164–2170.
- [17] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.
- [18] K. B. White, D. Cline, and P. K. Egbert, "Poisson Disk Point Sets by Hierarchical Dart Throwing," in *2007 IEEE Symposium on Interactive Ray Tracing*, Sep. 2007, pp. 129–132.
- [19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.
- [20] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.
- [21] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [22] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," in *International Conference on Learning Representations*, 2021.
- [23] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [24] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.
- [25] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
- [26] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.
- [27] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.