Supplementary material for TouchSDF: A DeepSDF Approach for 3D Shape Reconstruction Using Vision-Based Tactile Sensing

Author
University

In this appendix, we provide additional details regarding our methodology, experiments and results.

A Tactile Simulator

Physics-based simulation has been crucial in robotics, as advances in computation have improved simulation accuracy and speed. While most simulations focus on rigid-body dynamics and vision sensors, specific environment suites have been introduced to enhance realism and explore new research directions [1, 2, 3, 4, 5, 6]. In our work, we chose to build our methods within the Tactile Gym framework, a tactile simulator built on top of PyBullet. This choice was made because of PyBullet's fast GPU rendering, reliable solvers, successful sim-to-real applications, and open-source nature. Moreover, Tactile Gym exposes various functions for rendering tactile images using three high-resolution vision-based sensors: TacTip [7], DIGIT [8], and DigiTac [9]. In this work, we focus on the use of the TacTip sensor, as its curved profile helps capture depth maps for concave surfaces.

Our experiments are conducted using a simulated 6DoF robotic arm. This allows us to capture local geometries anywhere on the object surface at arbitrary poses. When the sensor is in contact with the object, a tactile image is extracted. Our simulated tactile images represent penetration maps at touch locations and are obtained by rendering the difference between the current depth image and a reference depth image taken when the sensor is not in contact. We collect images rendered a 256×256 resolution and captured using PyBullet's synthetic camera. The chosen resolution allows the capture of highly-rich contact details and to predict local surfaces while running at more than 1000 FPS on a Tesla 2080Ti.

B DeepSDF

DeepSDF [10] is an implicit neural representation used to model geometries as a signed-distance function using a fully-connected neural network. Specifically, a function $f_\theta(\mathbf{x}, \mathbf{z})$ maps a coordinate $\mathbf{x}$ and a latent vector $\mathbf{z}$ to a scalar representing the signed distance to the geometry under consideration. The shape $\mathcal{S}$ is defined as the zero level set of $f_\theta(\mathbf{x}, \mathbf{z})$ such that $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 | f_\theta(\mathbf{x}, \mathbf{z}) = 0\}$. To encode multiple shapes, a latent vector $\mathbf{z}$ is optimised for each shape jointly to the network parameters θ, which are shared for all the geometries. The simultaneous optimisation of network and latent vector defines the auto-decoder framework. Unlike an encoder-based inference, where an observation $\mathcal{O}$ is mapped to a latent vector $\mathbf{z}$ in one step, an auto-decoder lacks an encoder. At training time, an auto-decoder jointly optimises the per-observation latent code $\mathbf{z_i}$ and the network parameters θ by minimising the desider loss function, which is usually a metric of reconstruction error. At test time, observations $\mathcal{O}$ are collected, and $\mathbf{z_i}$ is iteratively refined to minimise the reconstruction error while freezing the network parameters.

C Data preparation and collection

C.1 Collecting data for local surface prediction

The objects loaded into our simulator were randomly sampled from 6 categories of the ShapeNet-CoreV2 dataset proportionally to the number of per-category instances present in the original dataset. Specifically, we sampled 200 bowls (150 training + 50 testing), 300 bottles (240 training + 60 testing), 120 cameras (90 training + 30 testing), 300 jars (240 training + 60 testing), 300 guitars (240 training + 60 testing), and 220 mugs (180 training + 40 testing). The objects were converted to URDF format using the shapenet library. This is necessary as PyBullet requires this format to correctly perform physical interaction with the object. Moreover, meshes were converted to watertight surfaces to enable the collection of signed distances.

Because the models in ShapeNet are very big when loaded into a PyBullet simulator, their scale was reduced to 20% of their original size. This allows comparable sizes between the simulated and real environment.

Inspired by the method proposed by Smith et al. [11], we predicted local surfaces at touch location using a CNN conditioned on tactile images and sensor's pose. The ground truth data required to train the model are point clouds extracted at each touch location. To generate this point cloud, we define a ray casting approach from the simulated TacTip sensor to the object's surface. Specifically, 3600 points are sampled on the hemisphere characterising the sensor's skin. At a touch location, the sensor's geometry intersects the object's geoemtry, and a tactile image is rendered with its average intensity computed. The sensor is moved inwards until its average intensity exceeds a threshold equal to 1, when 3600 rays are cast from the centre of the hemisphere and through the 3600 points previously defined. Any intersection between the rays and the object's geometry is gathered. this procedure effectively produces a point cloud representing the contact geometry. However, not all contacts result in a local surface. If the number of rays intersecting the object's geometry is too small, i.e. fewer than 250, we do not collect any data. This choice was made as it is very challenging for our prediction model to predict such small local patches.

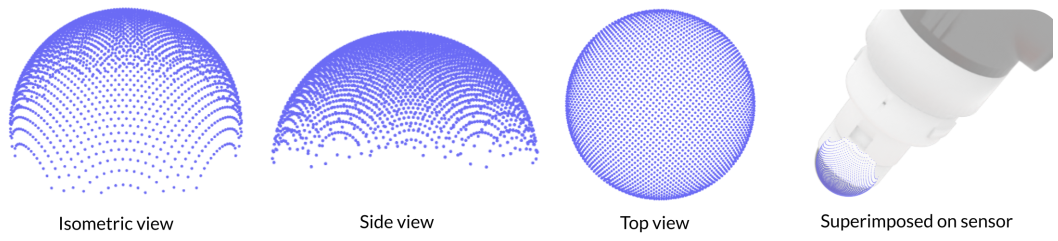

Figure 1: Points on tacTip surface for raycasting approach. Rays are cast from the centre of the hemisphere and through the 3600 points previously defined. Any intersection between the rays and the object's geometry is gathered.

Our data collection procedure is applied to the entire surface of the object and even extending regions that might be physically inaccessible in some real-world contexts. This approach is motivated by our pursuit of achieving robust generalisation across a spectrum of manipulation scenarios, so not only those settings where the object is places on a surface. For instance, in the scenario of a bottle positioned on a table, the lower region remains out of reach. However, should the bottle be gripped from its side by a grasper, the bottom surface becomes observable. At test time in the real world reconstruction, we only collect data that are physically accessible by our robot.

C.2 Collecting data for DeepSDF for ShapeNet

To ensure the consistency of our training data, we pre-processed the ShapeNet objects by removing gaps or holes in the geometry. This step is crucial in DeepSDF, as closed surfaces are required to retrieve signed distances. The DeepSDF model requires label pairs $\{(\mathbf{x_j}, s_j)\}_{j=1}^{N}$, where $\mathbf{x_j} \in \mathbb{R}^3$

denotes a spatial coordinate and $\mathbf{s_j} \in \mathbb{R}$ is a signed distance between $\mathbf{x_j}$ and the object geometry. All the objects in the ShapeNetV2 dataset are defined within a bounding box spanning the range of [-1, 1] along the x, y, and z axes. The original DeepSDF paper recommends mantaining a balanced ratio between positive and negative signed distances. To achieve this, we propose the following sampling strategy. We sample a total of 35000 points as follows: 5000 points are randomly sampled within the bounding box, 20000 points are sampled on object's surface, and 10000 points are chosen from inside a second bounding box that tightly encloses the object. This sampling approach ensures the collection of a comparable number of negative and positive signed distances, given that the object occupies a relatively small portion of the overall bounding box. For each of the 35000 points sampled per object, a signed distance function is computed using the Point Cloud Utils library [12].

D Training details

D.1 Training the CNN for local surface prediction

A CNN is trained to map a collected simulated tactile image into a set of vertices defining a mesh at touch location. This approach follows the methodology propose by [11]. The architecture of the CNN used for local surface prediction is shown in Table 1.

Input shape	Layer	Stride	Padding	Output shape
1 x 256 x 256	Conv2D(5 x 5) + BN + ReLU	2	2	16 x 128 x 128
128 x 128	Conv2D(5 x 5) + BN + ReLU	2	2	16 x 64 x 64
16 x 64 x 64	Conv2D(5 x 5) + BN + ReLU	0	2	16 x 64 x 64
16 x 64 x 64	Conv2D(5 x 5) + BN + ReLU	2	2	16 x 32 x 32
16 x 32 x 32	Conv2D(5 x 5) + BN + ReLU	2	2	16 x 16 x 16
16 x 16 x 16	Conv2D(5 x 5) + BN + ReLU	0	2	16 x 16 x 16
16 x 16 x 16	Conv2D(5 x 5) + BN + ReLU	2	2	32 x 8 x 8
32 x 8 x 8	Conv2D(5 x 5) + BN + ReLU	2	2	32 x 4 x 4
32 x 4 x 4	Conv2D(5 x 5) + BN + ReLU	0	2	32 x 4 x 4
512	FC + ReLU	-	-	256
256	FC + ReLU	-	-	128
128	FC + ReLU	-	-	75
75	FC	-	-	25 x 3

Table 1: Architecture of the CNN model to predict vertex displacements from tactile images.

The model was trained for 1000 epochs using the Adam optimiser [13], a batch size of 16, and a learning rate of 0.0001. The learning rate was adapted using a scheduler with a decaying value of 0.9. The scheduler was set to reduce the learning rate after 100 epochs with no improvement in the validation loss. The loss values were multiplied by a 1000 to avoid vanishing gradients. Our model was trained for 10 hours on a NVIDIA RTX 3090.

D.2 Training DeepSDF

Our dataset contains a total of 45 million SDF pairs $\{\mathbf{x}, \mathbf{z}\}$, as 35000 spatial coordinates $\mathbf{x}$ were sampled per shape. At training time, an initial set of per-shape latent vectors are sampled from a Gaussian distribution $z \sim \mathcal{N}(0, 0.01)$. The latent vectors are jointly optimised with the network parameters to learn a more expressive and compact representation of the shapes in the latent space.

As mentioned in the main paper, we optionally allow to capture high frequency information in the spatial domain. Specifically, this encoding produces a Fourier feature representation where each spatial coordinate $\mathbf{x_j}$ is transformed into a set of high-frequency sinusoidal functions

$$\gamma(\mathbf{x_j}) = [\gamma_1(\mathbf{x_j}), \gamma_2(\mathbf{x_j}), .., \gamma_m(\mathbf{x_j})]$$

where:

$$\gamma_i(\mathbf{x_j}) = [\sin(2^{i-1}\pi\mathbf{x_j}), \cos(2^{i-1}\pi\mathbf{x_j})]$$

Using these sinuosoidal embeddings of length 10, a spatial coordinate $\mathbf{x_j} \in \mathbb{R}^3$ is mapped into $\gamma(\mathbf{x_j}) \in \mathbb{R}^{60}$. This feature vector is concatenated to a 128-dimensional latent vector $\mathbf{z_i}$ to produce a 188-dimensional input. Note that the original DeepSDF paper does not leverage positional encoding, which therefore defines the input space as the concatenation of a latent vector and a three-dimensional spatial coordinate.

The DeepSDF architecture using positional encoding is represented in Table 2. The model was trained for 150 epochs using the Adam optimiser [13], a batch size of 20864, a learning rate for the network parameters of 10^{-5}, a learning rate for latent vector optimisation of 10^{-3}. Both learning rates were adapted using a scheduler with a decaying value of 0.9. The schedulers were set to decay the learning rate after 10 epochs with no improvement in the validation loss. Our model was trained for 30 hours on a NVIDIA RTX 3090.

Input shape	Layer	Output shape
188	FC + ReLU	512
512	FC + ReLU	512
512	FC + ReLU	512
512	FC + ReLU	384
128 + 384	FC + ReLU	512
512	FC + ReLU	512
512	FC + ReLU	512
512	FC + ReLU	512
512	FC + Tanh	1

Table 2: Architecture of the CNN model to predict vertex displacements from tactile images.

E Test-time optimisation

To reconstruct a mesh after a partial point cloud is predicted, we employ test-time optimisation of the latent vector by conditioning the DeepSDF model on concatenated coordinates and an initial random latent vector. However, optimizing the latent vector based on the partial point cloud alone is prone to convergence to local minima. To mitigate this issue, we adopt a strategy of optimising for 1000 epochs with a learning rate of 0.0005. Additionally, we incorporate a learning scheduler to regulate the optimisation process and apply a clipping threshold of 0.1 to the SDF predictions before backpropagating the loss.

F Ablation analysis

F.1 Pose sensitivity on unseen objects

The DeepSDF model used for shape completion is known to be sensitive to pose perturbation. This is due to the fact that DeepSDF captures a pose-specific object representation, which is encoded within a latent vector. One potential approach to mitigate this sensitivity involves training on objects in diverse poses. However, given the constraints posed by computational resources due to the size of the employed dataset, our emphasis in the main paper centred on objects in their canonical poses. This section offers an analysis of the impact of orientation perturbations along the x-, y-, and z-axes on Chamfer Distance, accompanied by a corresponding analysis of performance degradation.

Simulation

Table 3 shows how the reconstruction quality is affected by rotations along the three axes for 300 unseen objects (ShapeNet dataset). The highest degradation corresponds to rotations around the y-axis. This can likely be attributed to an inherent lack of geometrical symmetry along this axis. In contrast, the impact of rotations around the z-axis on reconstruction quality is relatively minor, due to the model's capability of leveraging symmetrical features along this dimension. As a reminder,

Category	Canonical pose	*x*-axis		*y*-axis		*z*-axis	
	Avg. CD (↓)	5 deg	10 deg	5 deg	10 deg	5 deg	10 deg
Camera	0.011	+0	+0	+0.001	**+0.002**	+0	+0.001
Bottle	0.004	+0	**+0.004**	+0	**+0.004**	+0	+0
Bowl	0.002	+0.001	+0.001	+0.001	**+0.003**	+0.001	+0
Mug	0.003	+0	+0.001	**+0.002**	**+0.002**	+0.0	**+0.002**
Guitar	0.002	+0.004	**+0.009**	+0.002	+0.006	+0.001	+0.001
Jar	0.005	+0	**+0.001**	+0	**+0.001**	+0	+0
Average	0.005	+0.0008	+0.0027	+0.001	**+0.003**	+0.0004	+0.0006

Table 3: Degradation in performance due to pose perturbation along the *x*-, *y*-, and *z*-axis. Perturbations are expressed in degrees and have been applied clockwise. The reported values express the relative Chamfer Distance between the perturbed pose and the canonical pose. Because lower CD is better, positive relative values imply degradation in performance.

the z-axis corresponds to the upward-facing axis within our defined setting. To illustrate this, a bottle exhibits symmetry relative to the z-axis, but this symmetry is not mirrored along the x- and y-axes.

The highest degradation can be observed for the guitar object. This can be attributed to highly non symmetrical features characterising these objects, thus the model cannot correctly generalise to perturbations in the objects' orientations. Cameras do not display significant degradation in reconstruction quality. However, it's crucial to note that this category demonstrates the poorest performance in terms of Chamfer Distance across all categories due its broad set of diverse geometries. This aspect is further analysed in Appendix G. Therefore, alterations in the orientation of objects in this category do not penalise reconstruction quality to the same extent as observed in other categories. Fig. 2 shows an instance of perturbed orientation for an object in the *bottle* category.

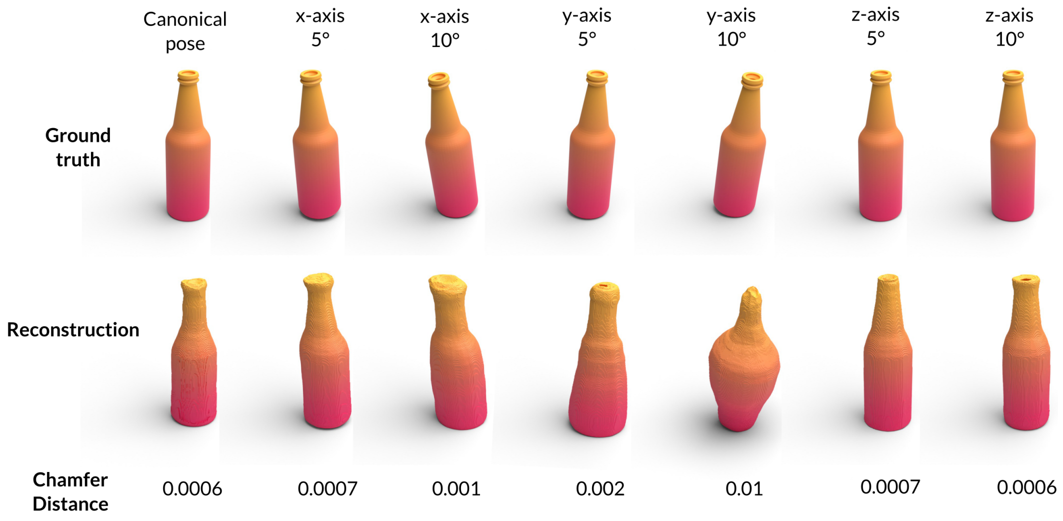

Figure 2: Instance of perturbed orientation for an object in the *bottle* category. Rotations around the x- and y-axes are more challenging to reconstruct. The difference in CD between the perturbation on the x- and y-axes can be attributed to the stochasticity introduced by the sampling method, as the robot randomly samples locations on the object's surface.

Real world

Because the 3D printed objects were designed with mounting sockets for secure attachment to a board during manipulation, the rotation of these objects was not practical. As an alternative approach, we introduced synthetic adjustments to the pose of the touch locations after data collection in the real world, thereby simulating perturbations in the object's pose. From a reconstruction standpoint, this effectively mirrors perturbations in the actual object pose.

Table 4 provides a visual representation of the degradation in Chamfer Distance (CD) for the 3D printed objects. Analogous to the simulation scenario, a decrease in performance can be observed for rotations along the x- and y-axes. Conversely, the decrease in performance during rotations around the z-axis is relatively minor. This can be attributed to the exploitation of symmetries by our model.

Object	Canonical pose	*x*-axis		*y*-axis		*z*-axis	
	CD (↓)	5 deg	10 deg	5 deg	10 deg	5 deg	10 deg
Camera	0.004	+0.003	**+0.004**	+0.001	+ 0.003	+0.002	+0.003
Bottle$_A$	0.002	+0.001	**+0.005**	+0	**+0.005**	+0	+0
Bottle$_B$	0.001	+0	+0.002	+0.001	**+0.005**	+0	+0
Bowl	0.002	+0.002	+0.002	+0.001	**+0.003**	+0.001	+0.001
Average	0.003	+0.0015	+0.0033	+0.0008	**+0.004**	+0.0008	+0.001

Table 4: Degradation in performance due to pose perturbation along the *x*-, *y*-, and *z*-axis to real world objects. Perturbations are expressed in degrees and have been applied clockwise. The reported values express the relative Chamfer Distance between the perturbed pose and the canonical pose. Because lower CD is better, positive relative values imply degradation in performance.

F.2 Comparing Single Across-Categories Model to Per-Category Models

In the main paper, we presented results obtained through training single model trained across multiple distinct categories and 1100 objects. In this section, we assess how the across-categories and per-category models compare when evaluating on unseen objects.

As indicated in Table 1 of the main paper, objects belonging to the *camera* category exhibit suboptimal reconstructions, obtaining a much higher Chamfer Distance compared to the objects in other categories. Appendix G expands on the reason underlying this difference. In this section, we compare the performance achieved by a per-category model and the across-categories model on three distinct categories: *camera*, *bottle*, and *bowl*.

Category	Across-categories	Per-category
Camera	0.011 $\pm$ 0.009	**0.007** $\pm$ **0.006**
Bowl	0.002 $\pm$ 0.003	0.002 $\pm$ 0.002
Bottle	0.004 $\pm$ 0.007	0.004 $\pm$ 0.005

Table 5: Comparison between across-categories and per-category models on unseen objects. Reconstructions correspond to 20 touches. Noticeable improvements can be observed for the *camera* category, which is challenging to reconstruct for the across-category model due to the broad variation in geometries belonging to this category.

The findings presented in in Table 5 highlight a substantial reduction in Chamfer Distance when employing the per-category model for the *camera* category. This improvement can be attributed to a more robust latent space. The use of a single category mitigates the risk of the latent vector inference procedure converging towards a local minima associated with an incorrect object category. No noticeable improvements was observed for the remaining two categories, that could already be correctly reconstructed when using the single across-categories model.

Despite the good performance achieved by the per-category model, its use is not practical due to our testing setup. During testing, the model lacks awareness of the specific object category, thus making the selection of the appropriate per-category model not feasible. As a consequence, using a single model becomes necessary. Addressing this limitation in future research could involve the integration of a category classification model, which would facilitate the selection of the appropriate per-category DeepSDF model to utilise.

G Supplementary results

In this section, additional results are presented.

G.1 Tactile images to point cloud

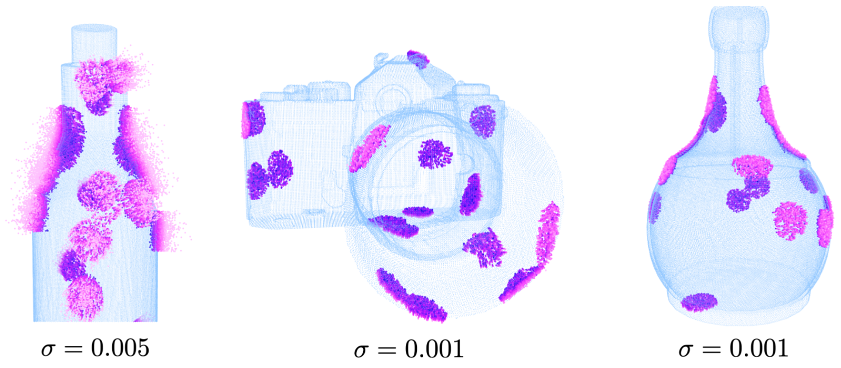

Figure 3: Predicted point clouds overlaid on the respective objects from which the tactile images were obtained.

The accurate prediction of local point clouds from tactile images is essential for achieving successful 3D reconstructions. In Fig. 3, we provide a visual representation of the predicted point clouds overlaid on the respective objects from which the tactile images were obtained. The predicted point clouds closely aligns with the contact geometry, demonstrating the effectiveness of the point cloud prediction model. The parameter σ denotes the standard deviation used to sample points along the surface normals estimated by the predicted point clouds. Increasing σ results in more scattered point clouds, whereas decreasing yields a denser and more compact point cloud representation.

G.2 Reconstruction in simulation

The main report indicates that reconstruction quality improves with the number of touches for all categories. However, when comparing the categories, soe categories (i.e. *camera* and *guitar*) result in higher Chamfer Distance scores. This can be attributed to the diverse range of geometries observed in these categories, which pose challenges for the optimisation of a robust latent code within the auto-decoder framework. This issue is visualised in Fig 4, which displays a projection of the 128-dimensional per-shape latent vectors $\mathbf{z_i}$ to a 2-dimensional plane. Notably, the latent codes associated with objects *camera* and *guitar* objects exhibit a scattered distribution throughout the latent space, showing why inference at test-time optimisation is challenging.

Moreover, in Fig. 5 and Fig. 6 the predicted point clouds used for the shape reconstruction reported in the main paper are displayed. Contact geometries are correctly predicted both for seen and for unseen objects and poses. The resolution of the Marching Cubes algorithm used to extract the meshes is set to 256^3.

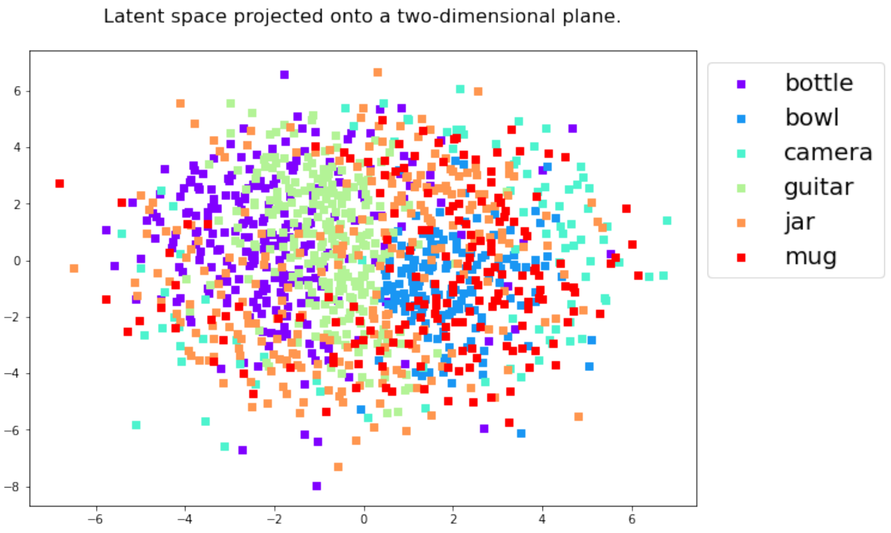

Figure 4: Projection of the 128-dimensional optimised latent vectors $\mathbf{z_i}$ to a 2-dimensional plane. The plot illustrates the distribution of latent vectors for all 1300 objects present in the training and validation sets. The latent codes associated with Camera objects exhibit a scattered distribution throughout the latent space.

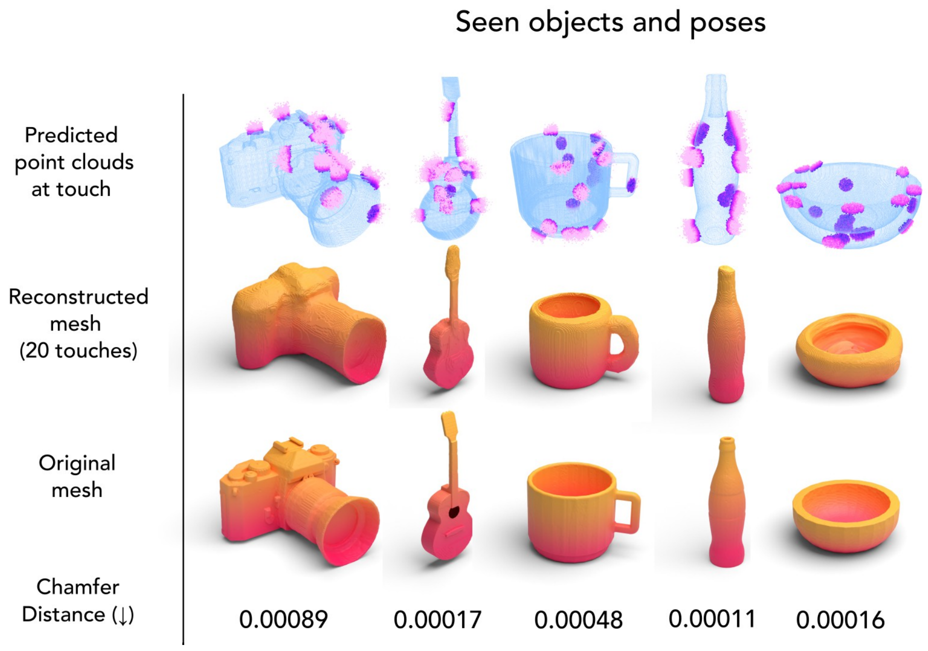

Figure 5: Reconstructed meshes after 20 touches, corresponding predicted point clouds, and Chamfer Distance between reconstructed and ground truth meshes. Ground truth meshes belong to training objects.

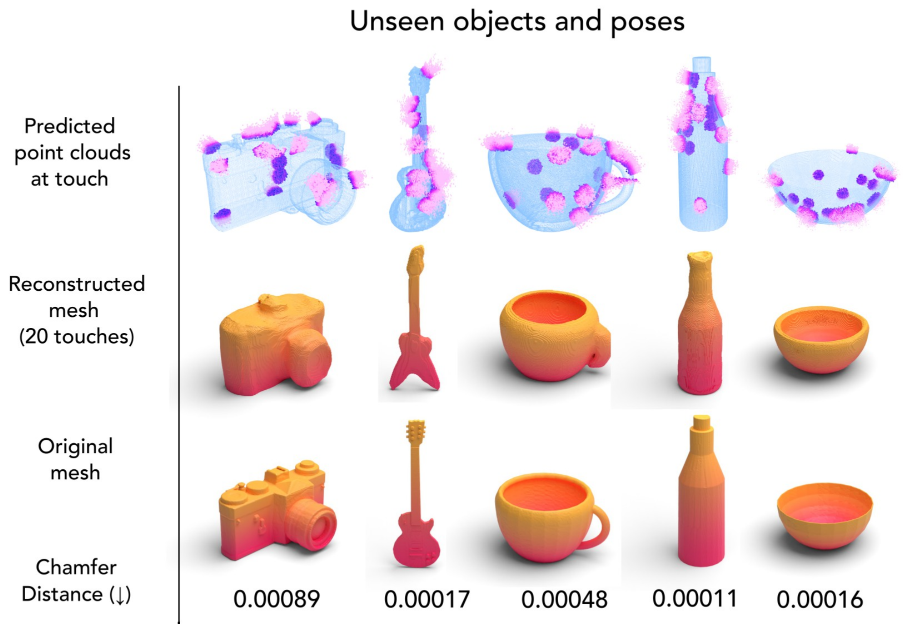

Figure 6: Reconstructed meshes after 20 touches, corresponding predicted point clouds, and Chamfer Distance between reconstructed and ground truth meshes. Ground truth meshes belong to test objects, therefore they are unknwon to the model.

References

[1] C. Chen, U. Jain, C. Schissler, S. V. A. Gari, Z. Al-Halah, V. K. Ithapu, P. Robinson, and K. Grauman. Soundspaces: Audio-visual navigation in 3d environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 17–36. Springer, 2020.

[2] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[3] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[4] Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

[5] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481. IEEE, 2021.

[6] 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*, 7(2):3930–3937, 2022.

[7] N. F. Lepora. Soft biomimetic optical tactile sensing with the tactip: A review. *IEEE Sensors Journal*, 21(19):21131–21143, 2021.

[8] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, et al. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5(3):3838–3845, 2020.

[9] Y. Lin, J. Lloyd, A. Church, and N. F. Lepora. Tactile gym 2.0: Sim-to-real deep reinforcement learning for comparing low-cost high-resolution robot touch. *IEEE Robotics and Automation Letters*, 7(4):10754–10761, 2022.

[10] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[11] E. Smith, D. Meger, L. Pineda, R. Calandra, J. Malik, A. Romero Soriano, and M. Drozdzal. Active 3d shape reconstruction from vision and touch. *Advances in Neural Information Processing Systems*, 34:16064–16078, 2021.

[12] F. Williams. Point cloud utils, 2022. https://www.github.com/fwilliams/point-cloud-utils.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.