Supplementary Material for NeuralGrasps: Learning Implicit Representations for Grasps of Multiple Robotic Hands

Ninad Khargonkar[1] **Neil Song**[2] **Zesheng Xu**[1] **Balakrishnan Prabhakaran**[1] **Yu Xiang**[1]
[1]The University of Texas at Dallas [2]St. Mark's School of Texas
{ninadarun.khargonkar,zesheng.xu,bprabhakaran,yu.xiang}@utdallas.edu
23songn@smtexas.org

1 Details about the Multi-Hand Grasping Dataset

We use the Graspit! [1] simulation software to generate a set of initial grasps for each gripper-object pair in a similar fashion as the ObMan dataset [2]. Graspit assumes setting contact points on the gripper links which we manually set for grippers not present by default in Graspit (Franka Panda and Allegro grippers). We also ensure to not select a proposed grasp if there is no contact with the object. On the initial set of grasps generated by Graspit, we use farthest point sampling to obtain a final, fixed set of diverse grasps. In our experiments, the multi-hand grasping dataset has 60 grasps for each gripper-object pair i.e across the five grippers and the seven objects. The relevant grasp information (griper pose and configuration for the joints) generated by Graspit were stored as a json file which was later utilized in the PyBullet rendering. Once the grasps were generated, the object and gripper models were loaded in PyBullet environment with the pose and the joint configuration corresponding to the grasp applied on the gripper. We used 128 virtual Pybullet camera viewpoints around the object to ensure complete scene information and render a dense point cloud for each grasping scene.

Upon generation of the point clouds, the SDF values for the gripper and the object were generated using the scheme outlined in [3] with the caveat that we computed the SDFs over surface point clouds instead of 3D meshes as we had access to the surface points for the gripper and object via the dense point cloud rendering. For each object, we ensured that every grasping scene is scaled to a unit sphere using a constant scaling factor across all grippers before the computing the SDF training data to ensure that the relative differences in gripper sizes stay the same. Given a desired number of points for the SDF generation (we used 40,000 points), there was an equal division between points having (at least one of) *positive* and *negative* SDF values to either the gripper or the object. As seen in other related implicit representation works, a majority (95%) of the sampled points were close to the gripper and object surface to encourage learning of SDF near their surfaces. The SDF values were also utilized in the contact map generation to find the object contact points as all the points within 5cm of a gripper surface point.

2 Network Architecture and Training Details

The auto-decoder architecture shown in Fig. 1 is used in all of our experiments. The decoder component consists of 12 MLP layers, each having 512 dimensions and having the two SDF values (for gripper and object) as the output. We use Batch Normalization [4] and Dropout [5] after each layer with a dropout probability of 0.2. The query point and latent code for a specific grasping scene are concatenated and passed as input to the auto-decoder. The latent code size (d) for all experiments was 128 and the latent code distribution had a zero-mean Gaussian prior $\mathcal{N}(0, \sigma^2 \mathcal{I}_d)$ with $\sigma = 0.01$. We used the Adam [6] optimizer to the train the network parameters and the latent codes for a total of 2,000 epochs with a step learning rate schedule for the optimizer. From the 60 grasps for every object-gripper pair in the dataset, we choose 50 of them across each object to construct the training set for the neural network model training. For the loss function on the SDF values, we have equal weight between the losses on the gripper and object SDFs and the triplet loss with each component

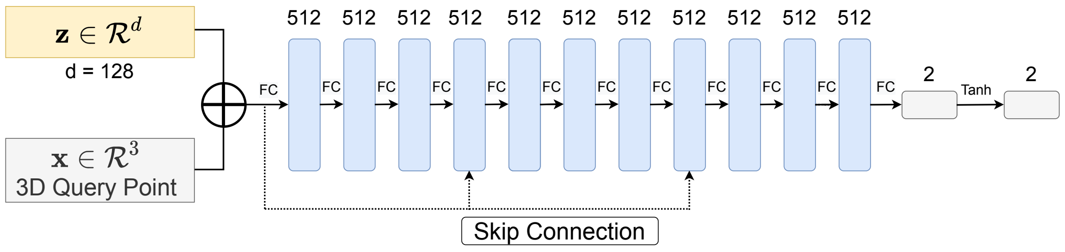

$\mathbf{z} \in \mathcal{R}^d$
d = 128
$\mathbf{x} \in \mathcal{R}^3$
3D Query Point
FC
512
2
Tanh
Skip Connection

Figure 1: Network architecture used in our experiments. The 12 MLP blocks are shown along with the final layer tanh activation on the 2 output SDF values.

being given a weight of 1.0. The SDF clamping distance δ is set to be 0.05 and for each grasping scene, we sample 20,000 points from the corresponding SDF samples in the dataset.

3 Experiment Details

3.1 Shape Reconstruction and Grasp Retrieval

For the experiments on unseen test grasps mentioned in the paper, we pick a set of held-out grasps not used during the model training. For an object, we select 5 grasping scenes from every gripper as the test set. The latent code for each test grasp is inferred using the SDF samples prepared during dataset construction and doing the MAP optimization for 800 iterations. Once the latent code is inferred, we try to predict the SDF values (query it via the model) on a randomly sampled set of query points inside the unit sphere. For all our experiments, we sample 100,000 points inside the unit sphere. Using the predicted SDF values on the query points, we obtain the estimated point clouds for the object and the gripper as the query points having the SDFs less than zero for object and gripper respectively. Chamfer distance is then computed between the inferred and ground truth point clouds for the object and gripper in the specific test grasping scene.

In the grasp retrieval experiments, for an input query grasp, the L_1 distance matrices for 1) the inferred latent code of the query grasp and training grasp latent codes and 2) the ground truth contact map of the query grasp and training set contact maps, each has the distances normalized to be between 0 and 1 so that the similarity score (computed as 1-distance) also remains between 0 and 1. We opted to use the simple 1-distance measure since any monotonically decreasing function of the distance works as a notion of similarity. Once we have the distances between the query and training set grasps, we simply sort them to obtain the *closest* and *farthest* examples.

We present the Chamfer distances and grasp retrieval similarity scores on the grasps in the training set in Table 1. The training set experiments were used as a sanity check on the model for representing grasps it had already seen. Although reconstruction is not the primary goal, we see low values for Chamfer distance and a high degree of match between the ground truth contact map-based and latent code-based grasp retrieval results. The Barnes-Hut t-SNE visualizations [7] of the learned embeddings of the different grasps from five robotic hands across the different objects are shown in Figs. 2, 3, 4.

Table 1: Chamfer distance (x1e-3) for shape reconstruction and contact map similarity for grasp retrieval on training set.

Ycb Model	Shape Reconstruction						Grasp Retrieval			
	Object	Fetch	Barrett	Human	Allegro	Panda	Near Z	Near GT	Far Z	Far GT
Cracker Box	1.28	4.06	5.89	6.36	4.11	7.84	0.87	0.88	0.26	0.24
Soup Can	1.29	3.77	4.84	2.91	3.30	3.12	0.78	0.79	0.12	0.10
Mustard Bottle	0.99	1.91	3.22	2.14	1.98	2.48	0.83	0.84	0.19	0.16
Pudding Box	1.22	5.14	8.16	6.55	5.25	6.11	0.83	0.84	0.24	0.23
Gelatin Box	1.12	5.53	7.73	4.66	4.99	4.73	0.80	0.81	0.23	0.21
Potted Meat Can	1.96	3.90	5.23	3.78	3.77	3.05	0.79	0.80	0.14	0.10
Bleach Cleanser	7.09	4.87	6.12	7.52	5.50	8.85	0.72	0.73	0.18	0.17

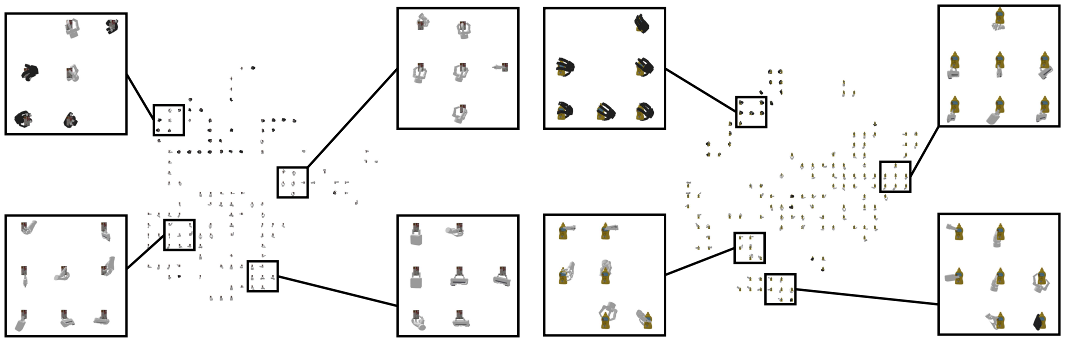


Figure 2: t-SNE visualization of the grasp embeddings on tomato soup can (left) and mustard bottle (right).

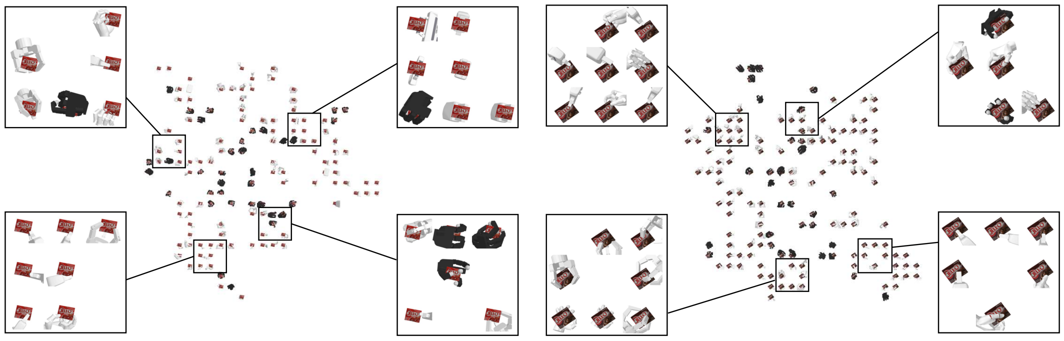


Figure 3: t-SNE visualization of the grasp embeddings on pudding box (left)and gelatin box (right).

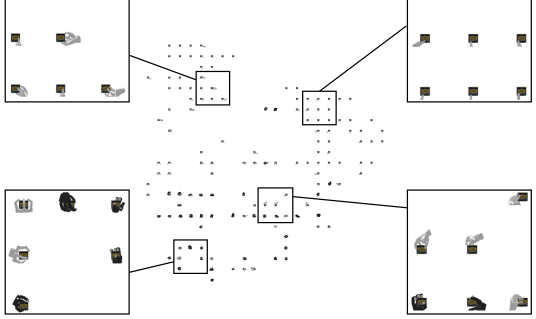


Figure 4: t-SNE visualization of the grasp embeddings on potted meat can.

3.2 Robotic Grasping

For the object pose estimation and grasping experiments, we assume an uncluttered tabletop scene with a single object placed on the table. Since the goal of the experiments is not to showcase detection or segmentation on object point clouds, we adopt a simple heuristic of using the table height to segment out the real world point cloud of the target object. We only use a single-view (partial) point cloud of the object in the real-world experiments.

Baseline Model for 6D Object Pose Estimation. The pose optimization assumes that the given points lie on the object surface and hence the predicted SDF values for them should be close to zero (first term of Equation 5 in the paper). We solve the following optimization problem to estimate the 3D rotation R and the 3D translation $\mathbf{t}$ of the object pose:

$$R^*, \mathbf{t}^* = \arg\min_{R,\mathbf{t}} \Big[\sum_{j=1}^{M} |f^o_{\text{SDF}}(R\mathbf{x}^j_c + \mathbf{t}, \mathbf{z}; \theta_o)| \Big], \tag{1}$$

where $\{\mathbf{x}^j_c\}^M_{j=1}$ denotes the points in the camera frame. $\mathbf{z}$ is the latent code of an arbitrary grasp in the training set since we do not use the SDF of the gripper in this optimization. We name this model the baseline model for object pose estimation since it does not utilize the grasp contact. To start the optimization with a good initial pose, we initialize the 3D translation with the center of the point cloud. For 3D rotation, we discretize the SO(3) space by yaw, pitch and roll angles with 45-degree intervals, then evaluate and randomly select a 3D rotation within the top-5 minimum objective values according to Eq. (1) as the initial rotation. After the optimization, we accept the estimated pose if the objective value is smaller than a pre-defined threshold. Otherwise, we re-run the optimization until a good pose is obtained. These steps are useful to avoid local minimums in SDF-based optimization for object pose estimation.

6D Object Pose Estimation with Contact. Let's denote the estimated object pose using the baseline model as $(R_0, \mathbf{t}_0)$. Using this initial pose estimation, we find a Fetch gripper grasp from the training set closest to the current gripper position for optimization. Because all the grasps are defined in the coordinate frame of the object. We need to use the object pose to transform the grasps to the camera frame, and use the pose between camera and robot base to transform them into the robot base frame. After that, we can compare them with the current gripper pose in the robot base frame. Assume the select grasp has a latent code $\mathbf{z}^*$ and contact map ϕ. We can first find a set of contact points on the object surface $\{\mathbf{x}^k_o\}^L_{k=1}$ using the contact map ϕ. Note that these points are defined in the object coordinate frame. Using the initial pose estimation $(R_0, \mathbf{t}_0)$, we can transform these points to the camera frame and then find the nearest neighbors of the transformed points from the object point cloud in the camera frame. Let's denote these nearest neighbor points in the camera frame as $\{\mathbf{x}^k_g\}^L_{k=1}$. We consider these points to be the contact points in the camera frame for the grasp encoded by $\mathbf{z}^*$. Therefore, we can perform the following optimization to refine the initial pose:

$$R^*, \mathbf{t}^* = \arg\min_{R,\mathbf{t}} \Big[\sum_{j=1}^{M} |f^o_{\text{SDF}}(R\mathbf{x}^j_c + \mathbf{t}, \mathbf{z}^*; \theta_o)| + \sum_{k=1}^{L} |f^g_{\text{SDF}}(R\mathbf{x}^k_g + \mathbf{t}, \mathbf{z}^*; \theta_o)| \Big], \tag{2}$$

This is our algorithm for 6D object pose estimation with contact using our implicit representations.

Grasp Execution. In order to execute the chosen grasp with optimized object pose, we first compute a standoff position and then sample 10 way points from the the standoff to the final grasp position for smooth execution. In the videos, it can be seen that sometimes the motion from standoff to final pose is still very fast resulting in some momentum transferred to the object being grasps. We add the table on which the object is placed as an obstacle and then utilize the Moveit [8] package to do the motion planning from an initial stowed position for the Fetch gripper. Lastly, on arriving in the final position, the Fetch gripper attempts to grasp the object by closing its fingers and then lifting the object from the table. The pose estimation method is not fail-proof and as such is seen from the results of Table 2 in the paper and attached video containing the grasping clips. A few failures are a consequence of the pose estimation getting stuck in a local minima. In some cases, the pose estimation for the *"with contact"* method actually performed worse due to an incorrect closest point matching, also likely resulting from a bad initial pose estimate.

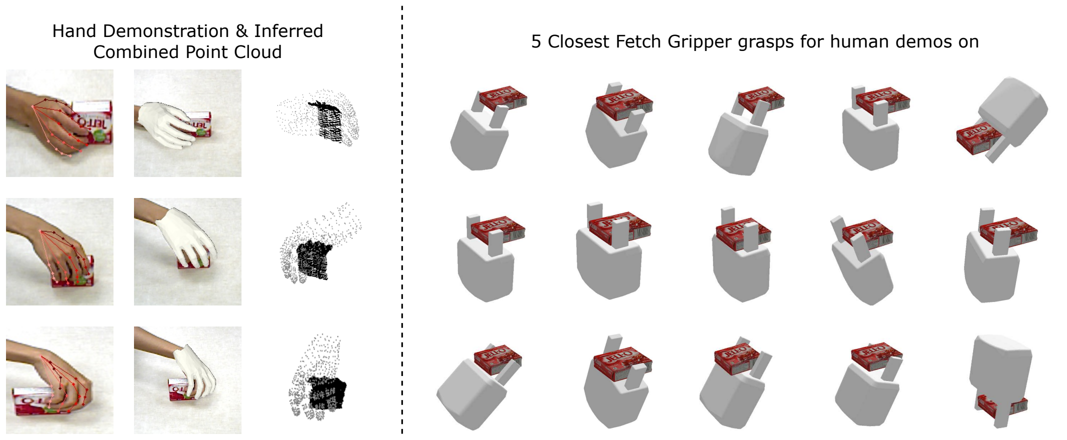

Hand Demonstration & Inferred Combined Point Cloud
5 Closest Fetch Gripper grasps for human demos on

Figure 5: Closest Fetch gripper grasps for the human demonstrations on the gelatin box (009). The grasps in the dataset generated from Graspit are not similar to a pincer like grasp.

3.3 Human Demonstrations and Grasp Transfer

We utilize existing methods for hand pose estimation in our human-to-robot grasp transfer experiments. The same RGB-D robot head camera is used to observe the human grasp demonstrations and infer the hand pose and 3D mesh for it on a per frame basis. Following a demo of the desired grasp by a human hand, we estimate the human hand grasps with the following pipeline. Given a RGB-D video of the human grasp demo, the Region of Interest (RoI) around the hand is detected using the Fully Convolutional One-Stage Object Detector (FCOS [9]) trained on 100 Days of Hands [10] dataset. Next, the RoI of the hand is cropped from the depth image, and the 3D hand joints are estimated using the Anchor to Joint (A2J [11]) model trained on the DexYCB [12] dataset. Lastly, we use the Pose2Mesh [13] model trained on FreiHAND [14] dataset to recover a 3D mesh of the hand from the predicted 3D hand joints. The 3D mesh consequently also gives the hand surface point cloud required for the latent code inference (here the hand is considered as a *gripper*). The supplementary video also shows how the grasp transfer process looks like with demos for each object. Once we have the closest Fetch gripper grasps, the Moveit [8] motion planning package is used as before to find a motion plan to the closest grasp. We note that the closest retrieved Fetch grasp might not have a feasible motion plan associated with it and hence we iterated along the ordered list of closest grasps and execute the one for which a successful motion plan is found. This set of executed grasps are shown in the video and hence for a few cases, the execution slightly differs from the demonstration.

The grasp transfer for the gelatin box (009) demo was not successful due to the motion planning finding no feasible plan to the inferred closest Fetch gripper grasps. This is likely due to the fact that the inferred closest Fetch grippers grasps have the gripper to grasp the box using the top and bottom rather than the corner as see in Fig. 5. Additional qualitative results on closest inferred grasps across the entire gripper set are shown in Fig. 6, and 7. It is possible that the transfer process is sometimes not feasible for such scenarios where the mapping to the Fetch gripper is not perfect. It can be a consequence of the dataset not containing a specific type of Fetch gripper grasp and the existing grasps being very different from the human demo even though the contact regions may be close on a small object relative to the grippers like the gelatin box. Another avenue to consider in the grasp transfer process is utilizing reliable object pose estimates of real world scans. For example in Fig. 6, for the first row of cracker box demonstration, the closest Panda gripper grasp is diagonally opposite to others. This result is due to the fact that the latent code inference does not take texture information into account and hence we have a flipped result.

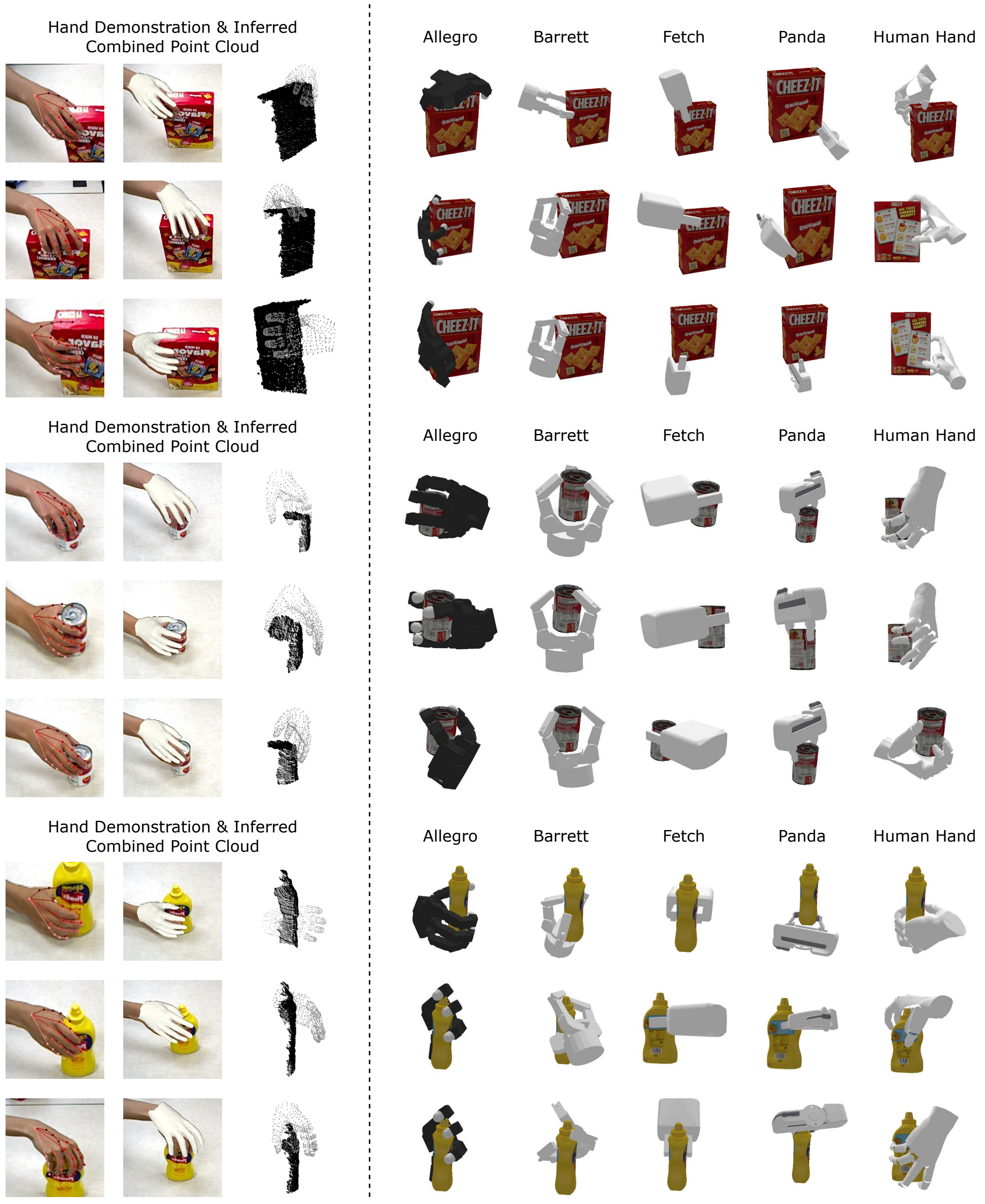
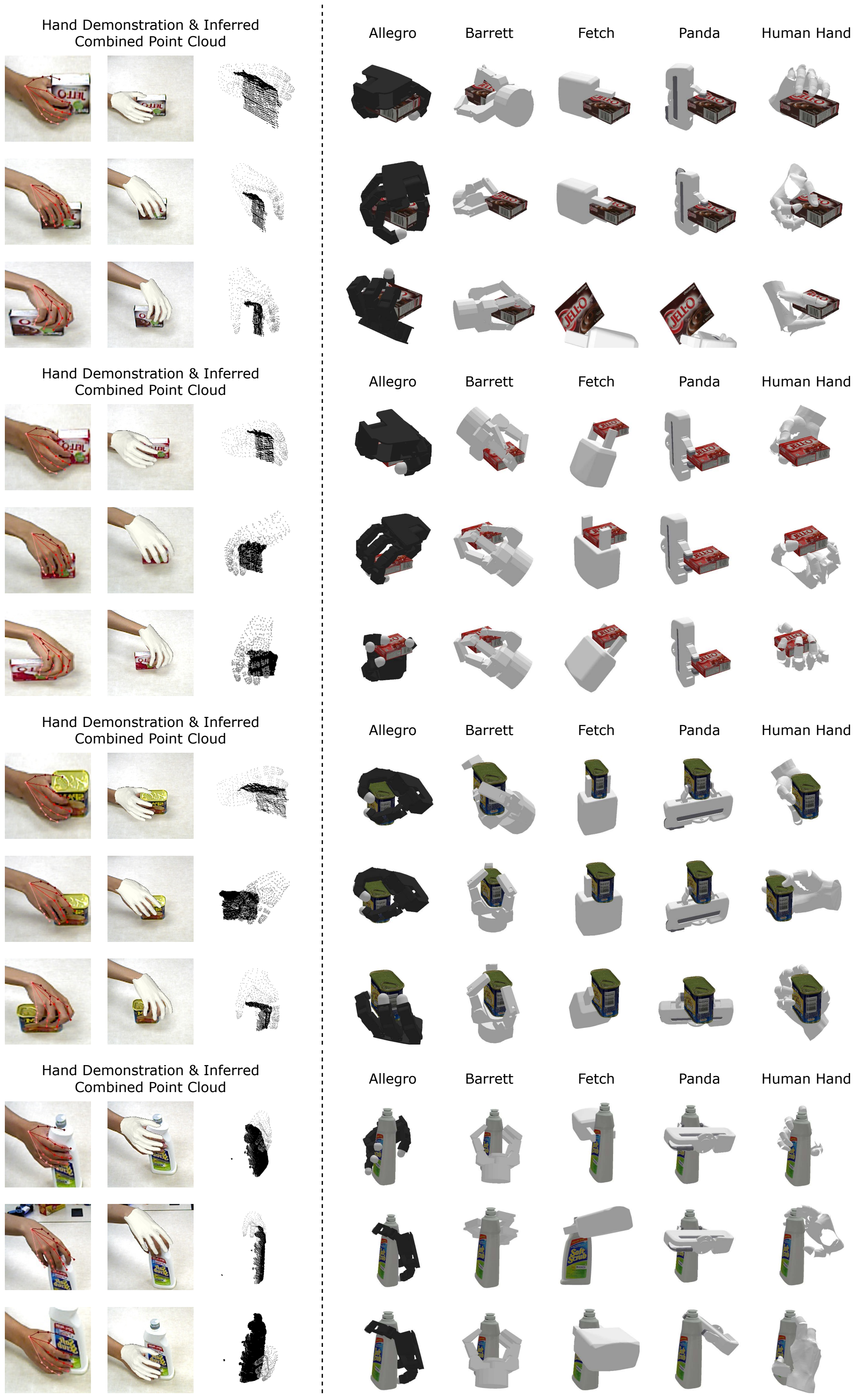


Figure 6: Closest grasps for all grippers for the human demonstrations on the cracker box (first row), tomato soup can (second row), and mustard bottle (third row).



Hand Demonstration & Inferred Combined Point Cloud
Allegro
Barrett
Fetch
Panda
Human Hand
Hand Demonstration & Inferred Combined Point Cloud
Allegro
Barrett
Fetch
Panda
Human Hand
Hand Demonstration & Inferred Combined Point Cloud
Allegro
Barrett
Fetch
Panda
Human Hand
Hand Demonstration & Inferred Combined Point Cloud
Allegro
Barrett
Fetch
Panda
Human Hand

Figure 7: Closest grasps for all grippers for the human demonstrations on the pudding box (first row), gelatin box (second row), potted meat can (third row) and bleach cleanser (fourth row).

References

[1] A. T. Miller and P. K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[2] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid. Learning joint reconstruction of hands and manipulated objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11807–11816, 2019.

[3] 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.

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15 (1):1929–1958, 2014.

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

[7] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[8] S. Chitta, I. Sucan, and S. Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.

[9] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[10] D. Shan, J. Geng, M. Shu, and D. F. Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9869–9878, 2020.

[11] F. Xiong, B. Zhang, Y. Xiao, Z. Cao, T. Yu, J. T. Zhou, and J. Yuan. A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 793–802, 2019.

[12] Y.-W. Chao, W. Yang, Y. Xiang, P. Molchanov, A. Handa, J. Tremblay, Y. S. Narang, K. Van Wyk, U. Iqbal, S. Birchfield, et al. Dexycb: A benchmark for capturing hand grasping of objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9044–9053, 2021.

[13] H. Choi, G. Moon, and K. M. Lee. Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose. In *European Conference on Computer Vision*, pages 769–787. Springer, 2020.

[14] C. Zimmermann, D. Ceylan, J. Yang, B. Russell, M. Argus, and T. Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 813–822, 2019.