Appendix

Volumetric-based Contact Point Detection for 7-DoF Grasping

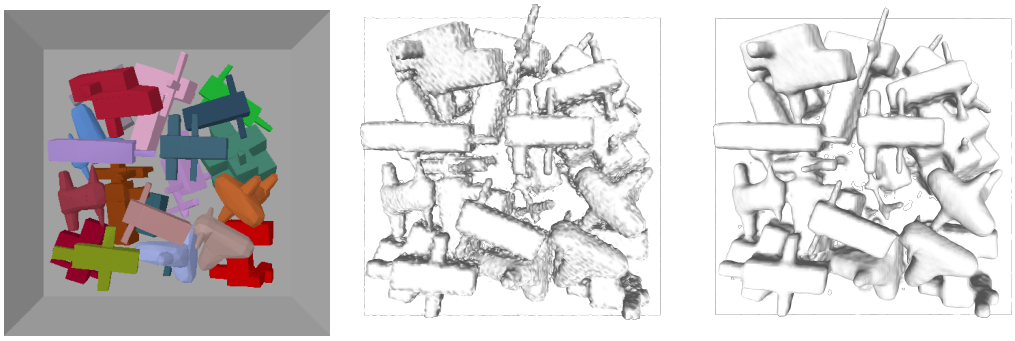
Junhao Cai[1,2] **Jingcheng Su**[3] **Zida Zhou**[3] **Hui Cheng**[3] **Qifeng Chen**[1] **Michael Yu Wang**[2,4]
[1]The Hong Kong University of Science and Technology
[2]HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen
[3]Sun Yat-sen University [4]Monash University

This appendix will describe the design and implementation of the proposed contact-based grasp pipeline in detail. We will first illustrate the details of contact point detection based on the TSDF volume and the proposed contact-point network (CPN). Next, we will introduce the implementation of the data generation process. Additional simulated experiments are conducted to have an in-depth understanding of our work. The subsequent section provides the details of real-robot experiments, including the setting of the workspace, control logic, and analysis of some failure cases. The visualization of the object sets we use in this work and the sampled scenarios are provided in the last section. The intermediate results for each module will also be visualized in this appendix to give a more intuitive understanding of this work.

1 Contact Point Detection

1.1 Contact Point Sampling

Gaussian smoothing on TSDF volume. Due to the noise introduced by the vision sensor, the reconstructed isosurface of the TSDF volume might be rugged, such that the surface normal cannot represent the local feature of the object geometry. Thus it cannot be regarded as the potential grasp vector. To eliminate this defect, we apply a spatial Gaussian filter to the TSDF volume before contact point sampling. The output isosurface of the filtered TSDF volume is much smoother than that of the original one. An example is shown in Figure 1.



(a) The stacked scene (b) TSDF without Gaussian filter (c) TSDF with Gaussian filter

Figure 1: Visualization of the TSDF volumes.

Details of sampling contact pairs. After the Gaussian smoothing operation, the Marching Cubes algorithm [1] is performed on the TSDF volume to obtain the surface vertices, which are regarded as one side of the contact pairs. The step size of Marching Cubes is set to 2, and the resulting number of vertices is around 1000 to 8000, which is sufficient for covering the entire scene. The corresponding surface normals are treated as the grasp vectors and used to guide the retrieval of

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

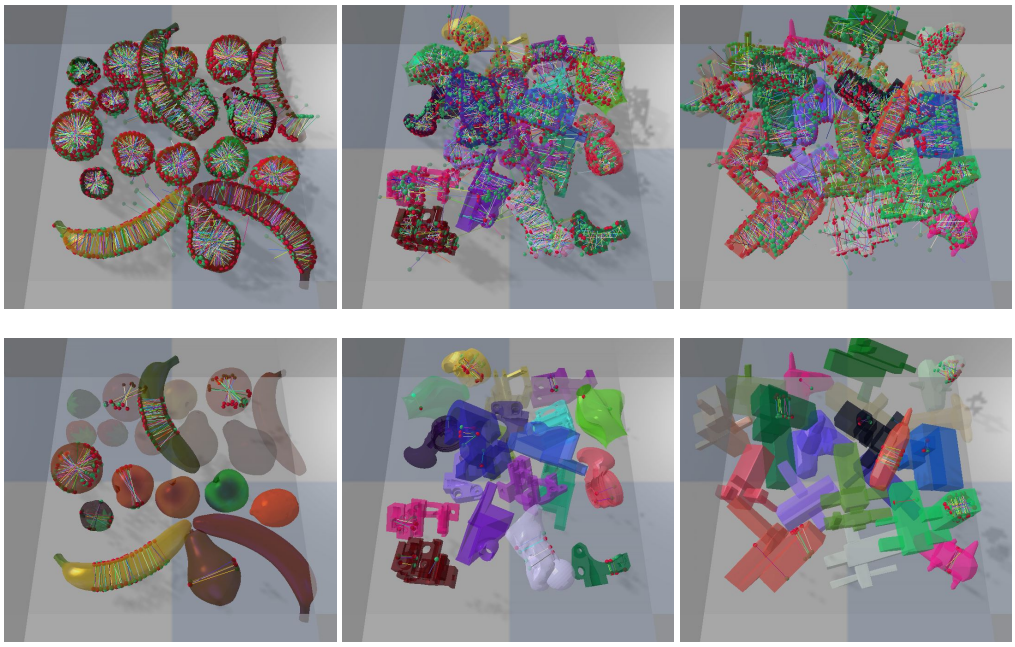


Figure 2: Visualization of the estimated contact pairs. The first row shows all the contact pairs sampled by the proposed method. And the second row visualizes the selected contact pairs after grasp quality evaluation and collision checking.

other contact points. We uniformly sample 50 points in the inverse direction of the surface normal in which the distance between the vertex and the last sampled point is $0.08m$, i.e., the maximal width of the gripper. Moreover, we select as the retrieved contact points those signed distance values are negative and the subsequent points turn to be positive. Examples are shown in Figure 2.

1.2 Contact Point Network

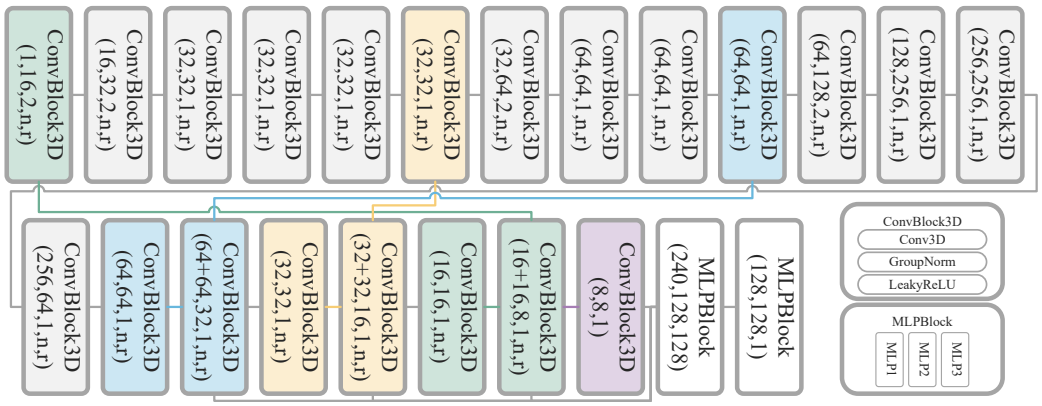


Figure 3: Contact-point network. The values in the tuple of the ConvBlock3D stand for the number of input and output channels, stride, and two flags specifying whether to use Group Normalization [2] and Leaky ReLU or not. The values in the tuple of the MLP block stand for the number of input dimensions for the layers in the block.

Network architecture. As shown in Figure 3, the proposed contact-point network is composed of two main modules, the 3D convolutional layer and the MLP layer. The input of the network includes the TSDF volume and the voxel indices of the contact pairs. To obtain the 3D features for the contact

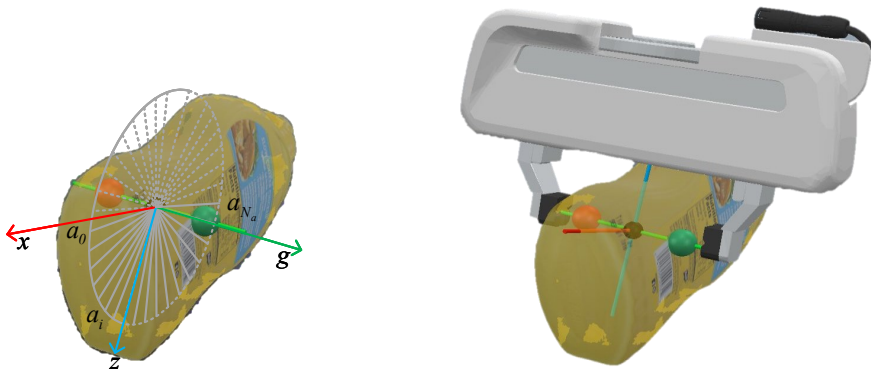


Figure 4: Potential approach vectors and the selected vector.

pairs, we use their indices to retrieve the corresponding feature vectors from the 3D feature maps extracted by the convolutional blocks. Since the indices can only be applied to the original TSDF volume, we should downsample them according to the accumulated stride used for each layer before extracting their features. Moreover, because the indices might not exactly point to a specific feature vector, trilinear interpolation is also leveraged to compute an approximate feature from the nearest ones. After feature extraction, the retrieved features of each contact pair will be sent to two MLP blocks. The final output is a single value for each contact pair representing the estimated grasp quality of the given contact candidate.

Hyper-parameters. During the course of training, we use the cross-entropy loss function and Adam optimizer with the learning rate and decay weight set to 0.0005 and 0.0005 to guide the training of the proposed CPN. The batch size is set to 24, and the number of epochs is 50. The training process is conducted on a server with two NVIDIA 3090 GPUs and an Intel Core i9-10920x CPU. The model is run on a PC with an NVIDIA 2080ti GPU and an Intel Core i9-12900K CPU during real-robot experiments.

1.3 Approach Vector Selection

Since the proposed CPN only predicts the grasp quality for the sampled contact pairs, there are no explicitly angle discretizations proposed by [3, 4]. Therefore, we can manually determine the number of approach vectors N_a and perform collision checking with the sampled poses, which is more flexible than previous methods. In the implementation, we first select the top 3‰ contact pairs with their grasp qualities estimated by the CPN. We then uniformly sample 32 approach vectors on the orthogonal subspace of each grasp vector. An example of approach vector sampling is shown in Figure 4. Finally, we select the collision-free poses by projecting the vertices of the gripper mesh model into the TSDF volume and checking the number of voxels having negative values. To avoid no solution being found, we select the top 1‰ of poses that have the maximal number of positive voxels.

1.4 Pipeline of Contact-based Grasping

Based on the TSDF volume and CPN, we can create a closed-loop grasping pipeline. At each step, the sampled contact points with grasp vectors are generated by executing Marching Cubes on the TSDF volume. To find contact points on the opposite side, we explore potential contacts in the inverse direction of the grasp vectors guided by the antipodal grasp rule. With the gripper mesh model and the TSDF volume, the collision-free gripper poses can be obtained by conducting collision checking on the sampled poses of the top 3‰ high-quality contact pairs. After NMS operation on the collision-free poses, an estimated optimal gripper pose can be obtained such that we can control the robot arm with the corresponding command. During the course of moving the robot, the vision sensor can capture different visual observations, enriching the information of the

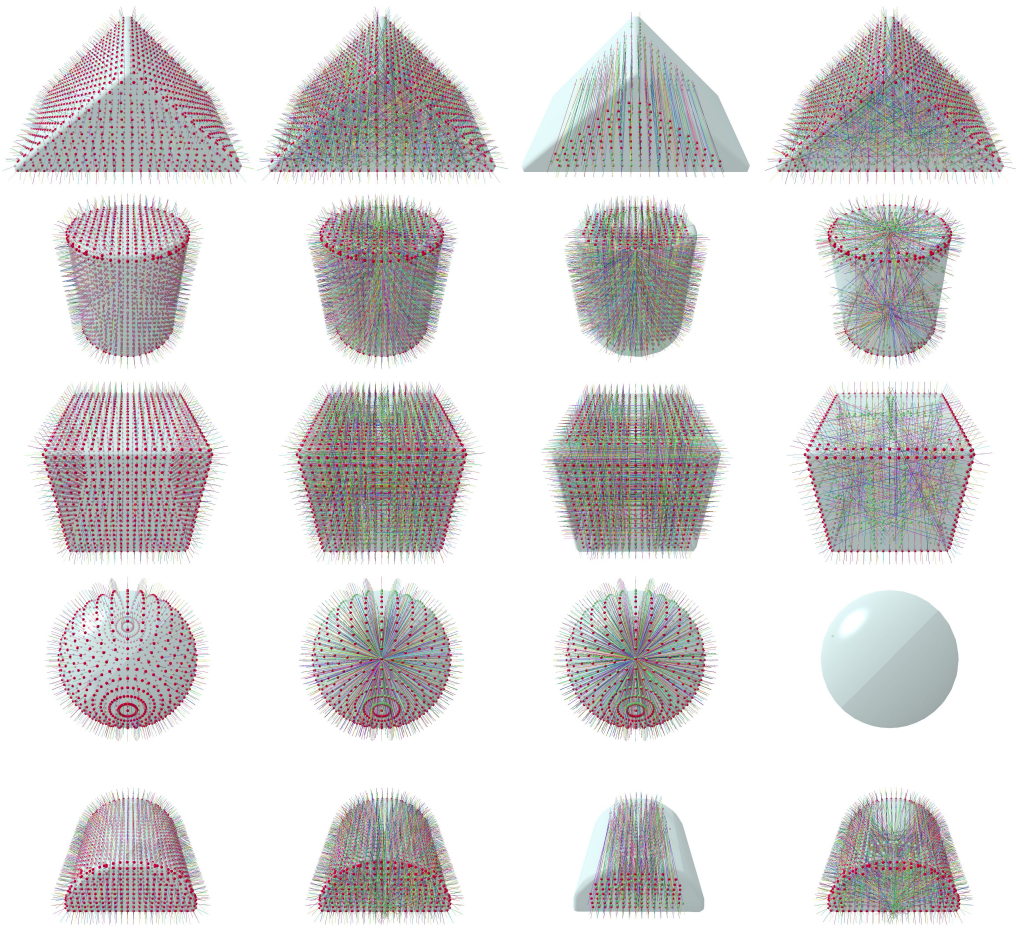


Figure 5: Visualization of grasp analysis on a single mesh. Column 1: contact points with normal. Column 2: all of the sampled contact point pairs. Column 3: positive contact pairs. Column 4: negative contact pairs.

scene in the TSDF volume and thus improving the performance of contact point detection. All these modules can be implemented with GPU acceleration such that the entire pipeline can be executed efficiently in a closed-loop manner.

In implementation, all the modules are performed on robot operating system (ROS). Specifically, TSDF fusion, CPN inference, and velocity controller are executed on ROS nodes individually, and each module requests the messages based on ROS service or message. For example, the TSDF node not only continuously updates the TSDF volume by integrating the latest frame of depth map with the current camera pose into the volume, but also provides a ROS service by which the TSDF node will send the volume data to the CPN node when it requests a volume. Similarly, each time the CPN node obtains a volume data, pose sampling, detection, and post-processing are conducted instantly. And the output pose will be published. For the velocity controller, it will keep updating the estimated pose and generating the velocity command based on the latest target.

2 Data Generation

2.1 Grasp Analysis on Single Mesh

In this work, we perform antipodal grasp analysis and label validation on 191 primitive-shaped objects. The objects include spheres, cubes, ellipsoids, cylinders, and building blocks released by [5].

An overview of the training objects is shown in Figure 13. Since the number of vertices of the meshes might be sparse (e.g., a square face of a cube mesh may contain only two triangle faces and four vertices, which is not enough for this task. To have a sufficiently large number of vertices that are uniformly located at the object surface, we first perform subdivision on the meshes to increase the number of vertices without changing the original shape. The output meshes with uniformly dense vertices will be leveraged for antipodal grasp analysis. Visualization of grasp labels of some objects is shown in Figure 5.

2.2 Scene Construction

Noise model for rendered depth map. In this paper, we use the noise model proposed in [6] to add axial and lateral noise to the synthesized depth maps. Given each pixel containing its image position p_i, range value z_i and the corresponding normal vector n_i, the lateral shift of each pixel is first sample from a Gaussian distribution, which is denoted as $\Delta_{l,i} \sim \mathcal{N}(0, \sigma_{l,i}^2)$, where $\sigma_{l,i} = 0.8 + 0.035 \cdot \theta_i/(\pi/2 - \theta_i)$, and θ_i represents the angle between the camera optical axis and the surface normal of the captured point at pixel i. Next, we shift each pixel by adding the sampled lateral noise to the original pixel position, which obtains $\hat{p}_i = p_i + \Delta_{l,i}$. Similarly, we then sample axial noise based on the axial noise model proposed in [6], which is defined as $\Delta_{a,i} \sim \mathcal{N}(0, \sigma_{a,i}^2)$, where $\sigma_{a,i} = 0.0012 + 0.0019(z_i - 0.4)^2$ and z_i is the range of the current pixel i. And thus the noisy range data will be $\hat{z}_i = z_i + \Delta_{a,i}$. Finally, we randomly set the range value to 0 with the probability of $(0, 0, 1)^T n_i = n_{i,3}$, which is exactly the cosine of the angle between the z axis and the normal vector. Sample noisy depth maps are shown in Figure 6.

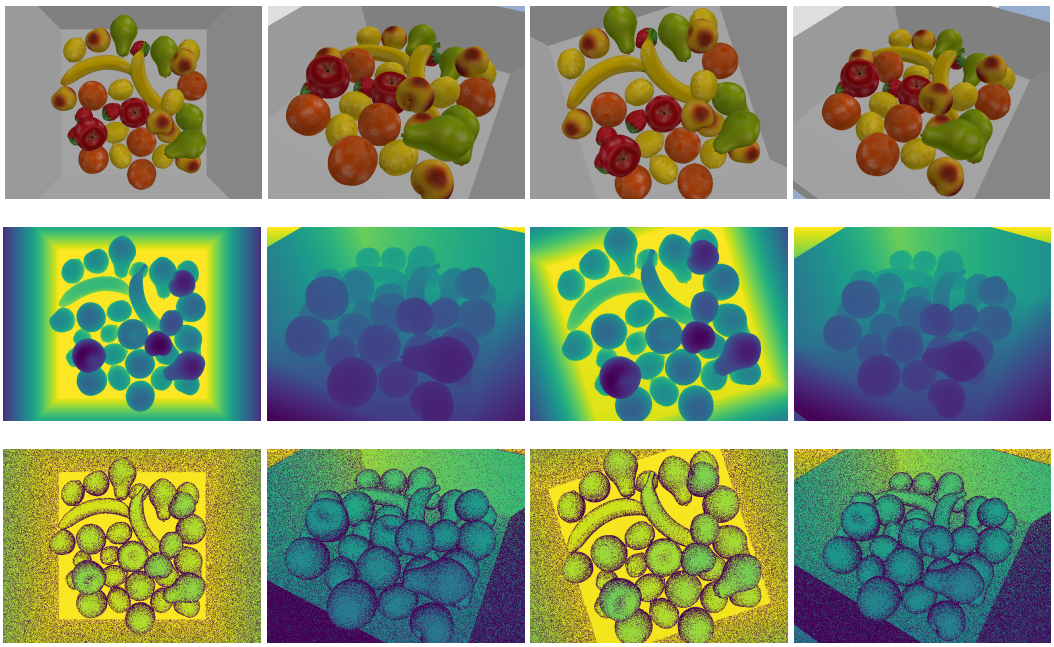


Figure 6: Visualization of noisy depth maps. First row: color maps. Second row: colorized depth maps without noise. Third row: noisy depth maps.

Visualization of grasp labels. Figure 7–10 visualize the positive (row 1) and negative (row 2) contact pairs on scenes that have different numbers of objects stacked together. The contact pairs are generated from the grasp analysis step, and only the pairs whose points are sufficiently close to the reconstructed surface of the TSDF volume will be used as the ground truth.

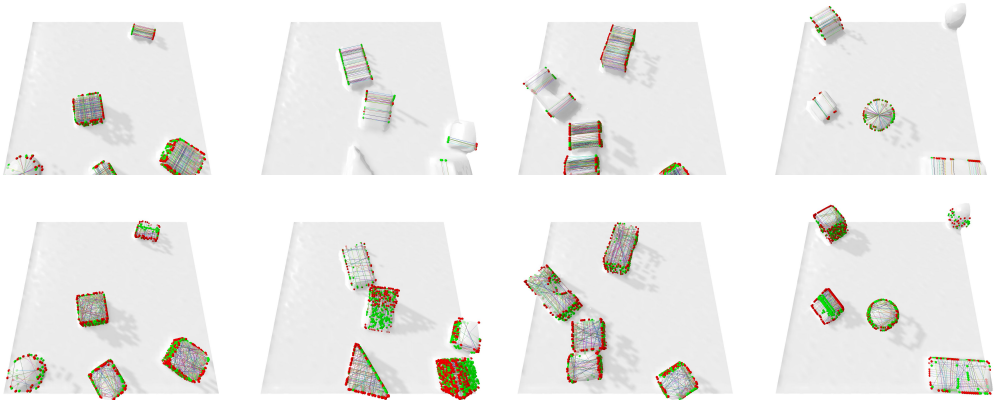


Figure 7: Reconstructed scene with labelled contact points (5 objects).

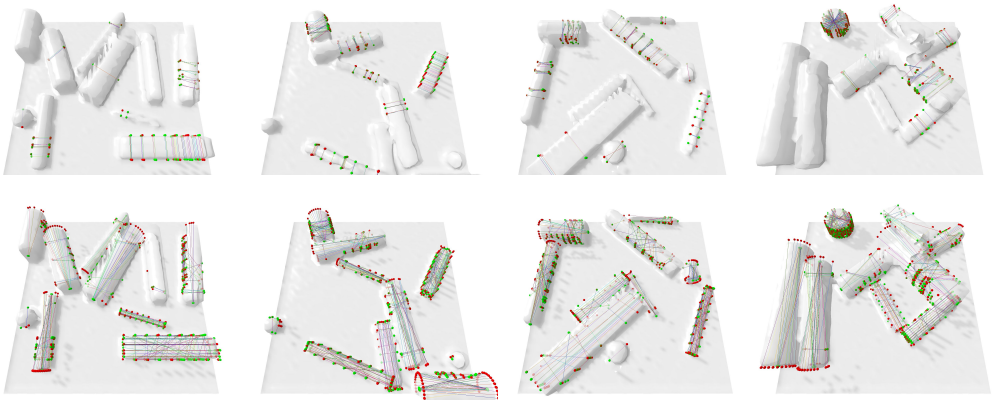


Figure 8: Reconstructed scene with labelled contact points (10 objects).

3 Simulation Experiments

Grasp success rate on EGAD [7]. To study the proposed method in-depth, we further perform grasp experiments on the EGAD object dataset. The EGAD object sets, generated by evolutionary algorithms, are geometrically diverse in terms of grasp difficulty and shape complexity, which allows them to better reflect the features and limitations of the grasping methods. Similar to EGAD [7], we set the maximal width of the gripper to $0.07m$, and there is only a single object randomly placed at the workspace for each grasp attempt. We perform 1000 grasp trials for each evaluation object on the Isaac Gym simulator. A snapshot of the 3D-printed evaluation object set provided in the EGAD paper [7] is shown in Figure 11. The average success rate is also reported in Figure 11. These experiments reveal the following characteristics of the proposed pipeline: 1) Regardless of the levels of grasp difficulty and shape complexity, our method can achieve significant performance on most objects. 2) From the results of objects F0, G2–G4, and G6, we can see that it is difficult for the contact-based method to handle flat or slim objects. Due to the large voxel length and the truncated margin set in the TSDF volume, the shape information may be missing during the TSDF fusion, which leads to no contact points being found. Such cases happen more frequently when performing real-robot experiments, and we will discuss this problem in the subsequent section.

Time efficiency. We evaluate the time efficiency by counting the average inference time of the CPN and the average planning time of the entire pipeline on scenes with 5, 10, 15, and 20 objects piled in the workspace. The definition of inference time and planning time are the same as those given in [4]. The experiments are run on a PC with an NVIDIA 2080ti GPU and an Intel Core i9-12900K

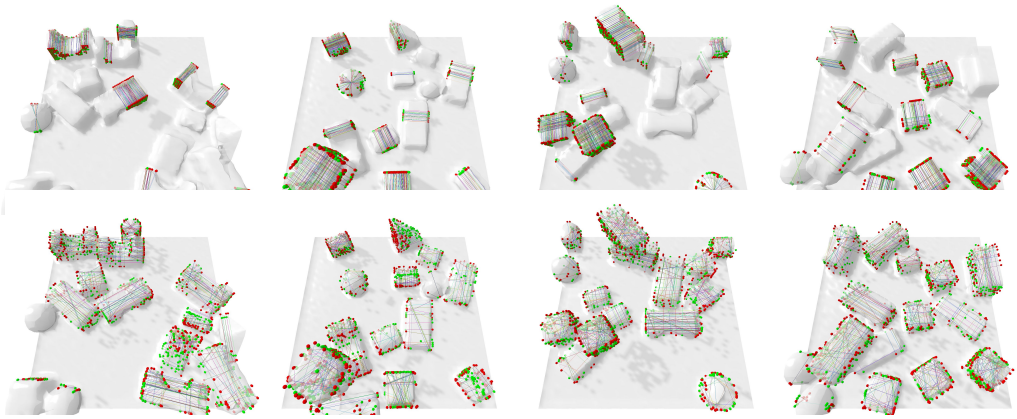


Figure 9: Reconstructed scene with labelled contact points (15 objects).

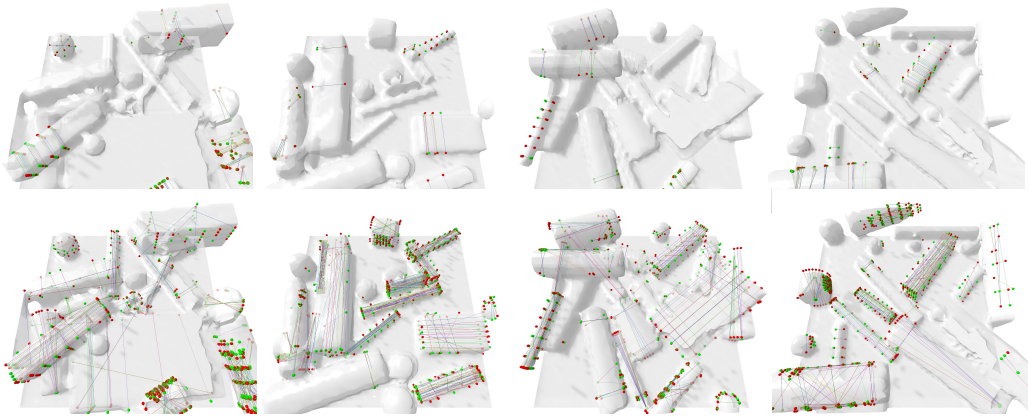


Figure 10: Reconstructed scene with labelled contact points (20 objects).

CPU. The results are listed in Table 1. Though using the 3D FCN and the TSDF volume with large resolution, the CPN can still achieve around 100 Hz during inference. Moreover, executing a pipeline once has an efficiency of at least 20 Hz, which is fast enough for closed-loop grasping.

Table 1: Computation Times

	5 objects	10 objects	15 objects	20 objects
Inference time (ms)	10.02	10.11	10.06	9.92
Planning time (ms)	38.71	45.72	44.03	46.80

4 Real-robot Experiments

Setting of the workspace. To demonstrate the effectiveness of the proposed method in a physical environment, we perform clutter removal experiments on four types of objects using the Franka Panda robot arm. We use the RealSense D435 as the vision sensor to capture the depth maps, and attach it to the end-effector using the mounter released by [8]. Sample cluttered scenes of the experiments are shown in Figure 17.

Control logic. To achieve smooth closed-loop control, all modules, including TSDF fusion, network inference, and velocity control, are implemented in separate ROS nodes. In particular, we divide an entire grasp attempt into three parts. We first move the robot based on the commands generated by

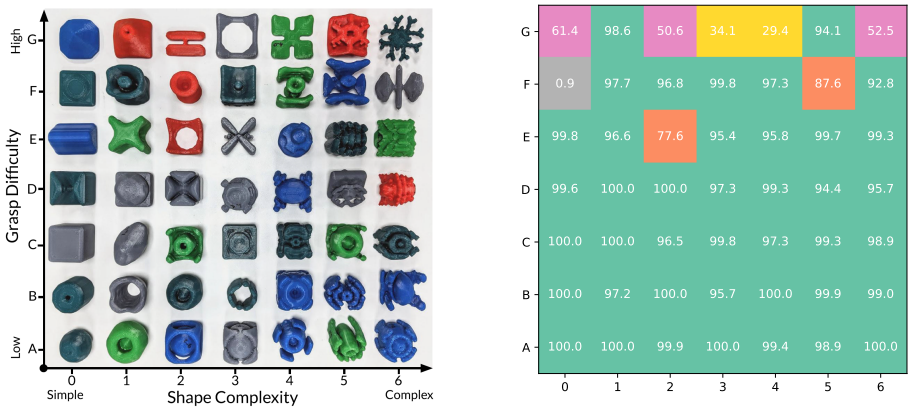


Figure 11: Snapshot of 3D-printed EGAD object set [7] and the average grasp success rates.

the velocity controller, which is defined as

$$v_t = \lambda (J_t^T J_t)^{-1} J_t^T [(\widetilde{p}_t^g - p_t)^T,\ r_t^T]^T, \tag{1}$$

where $\widetilde{p}_t^g$ and p_t are the estimated target position and current position of the gripper, and r_t is the difference between the target and current orientations. To make the sensor focus on the workspace during the movement of the robot arm, we make the z-axis of the target orientation point towards the bottom center of the workspace. When the end-effector reaches the pre-defined minimal height, the velocity controller keeps working while the target position and orientation switches to a pre-approach pose which is $7cm$ upwards of the target grasp pose in the approaching direction. When the gripper reaches the pre-approach pose, we set $r = 0$ and move the robot to the target grasp pose and execute the grasp attempt. Through this piecewise setting, the generated trajectory rarely collides with the objects during the motion planning. However, the controller cannot avoid the problem of exceeding the joint limits. When such cases happen, we have to neglect the grasp attempt and conduct another trial.

Failure cases. As mentioned in the paper, the performance of the proposed methods highly depends on the precision of the TSDF volume. Due to the measurement error and the noise introduced by RealSense D435, and the sharp-edge effect of the TSDF volume, the reconstructed scene might be distorted, especially on flat or slim objects. Some failure cases are presented in Figure 12, where we can see that part of the cups wall is missing during the reconstruction. One reason might be that the truncated margin is larger than the thickness of the wall. Moreover, as shown in the third instance, there is only a tiny discrepancy in the depth between the background and the object. Such difference is unclear for the TSDF volume due to its low resolution, and it is difficult for the consumer-level sensor to capture this difference, which leads to no contact candidates being found. The failure cases imply that the proposed method currently cannot handle tiny or flat objects. A direct solution is to use the TSDF volume with a higher resolution or propose heuristics to guide the movement of the robot arm to have a more comprehensive view of the target grasp region. Alternatively, we could replace the Marching Cubes module with learning-based methods to infer the contact points with their contact normals. We treat this possible as future work.

Figure 12: Failure cases.

5 Object Sets and Sampled Scenes

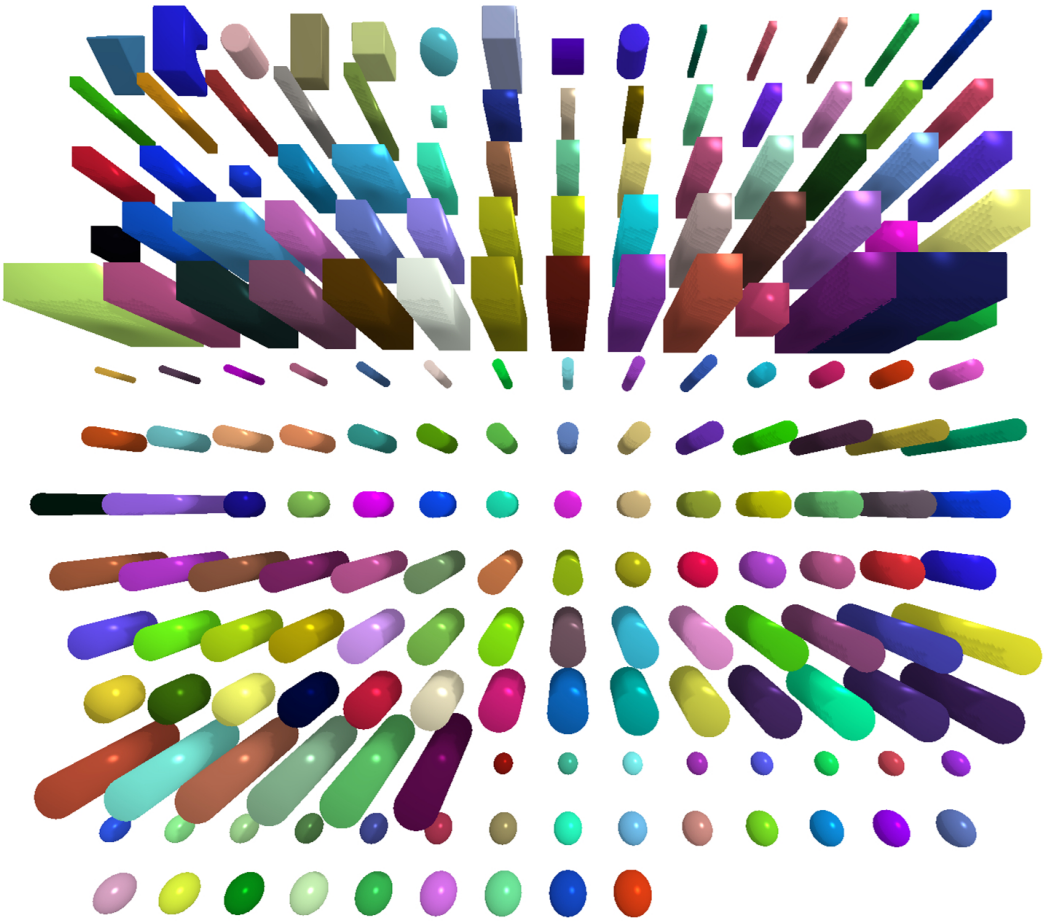


Figure 13: 191 primitive-shaped objects.

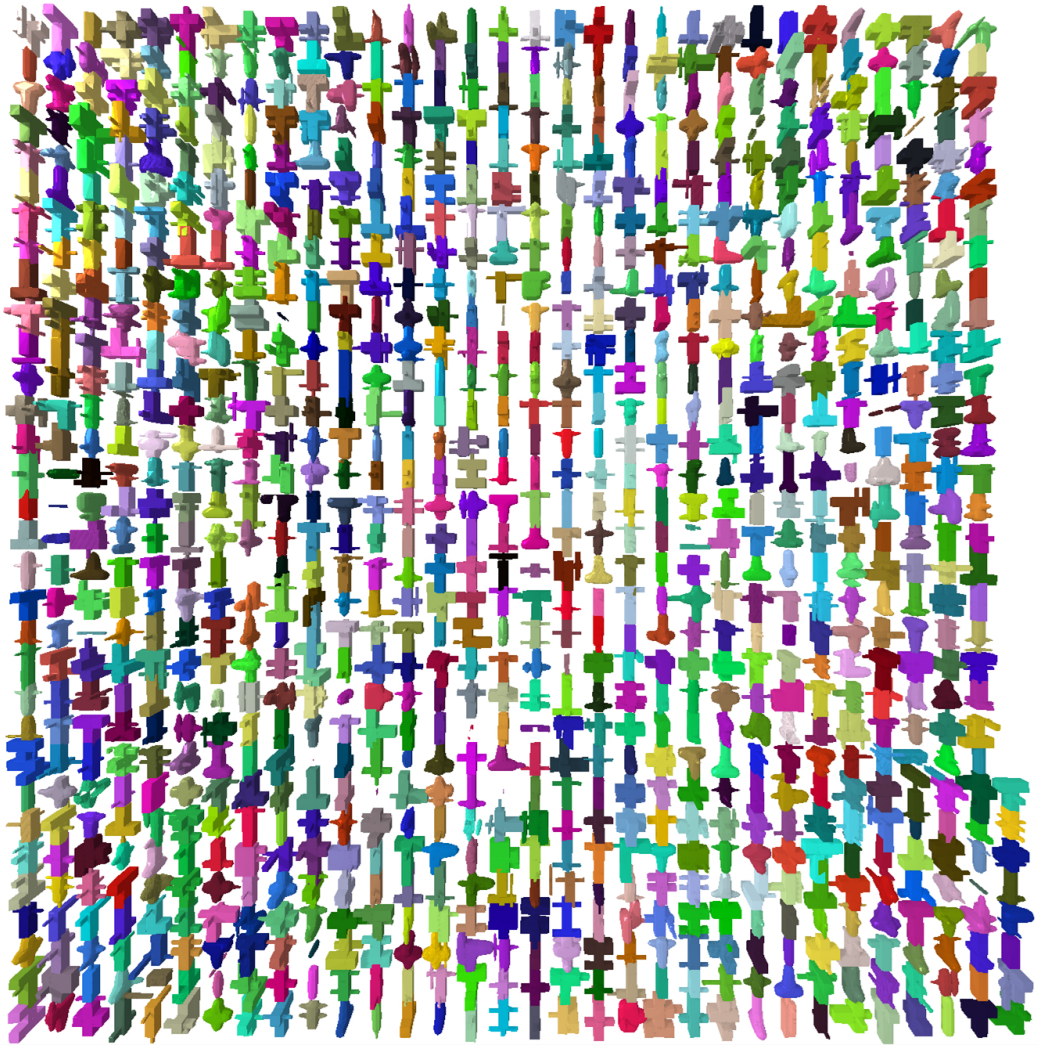


Figure 14: 1000 Procedual objects.

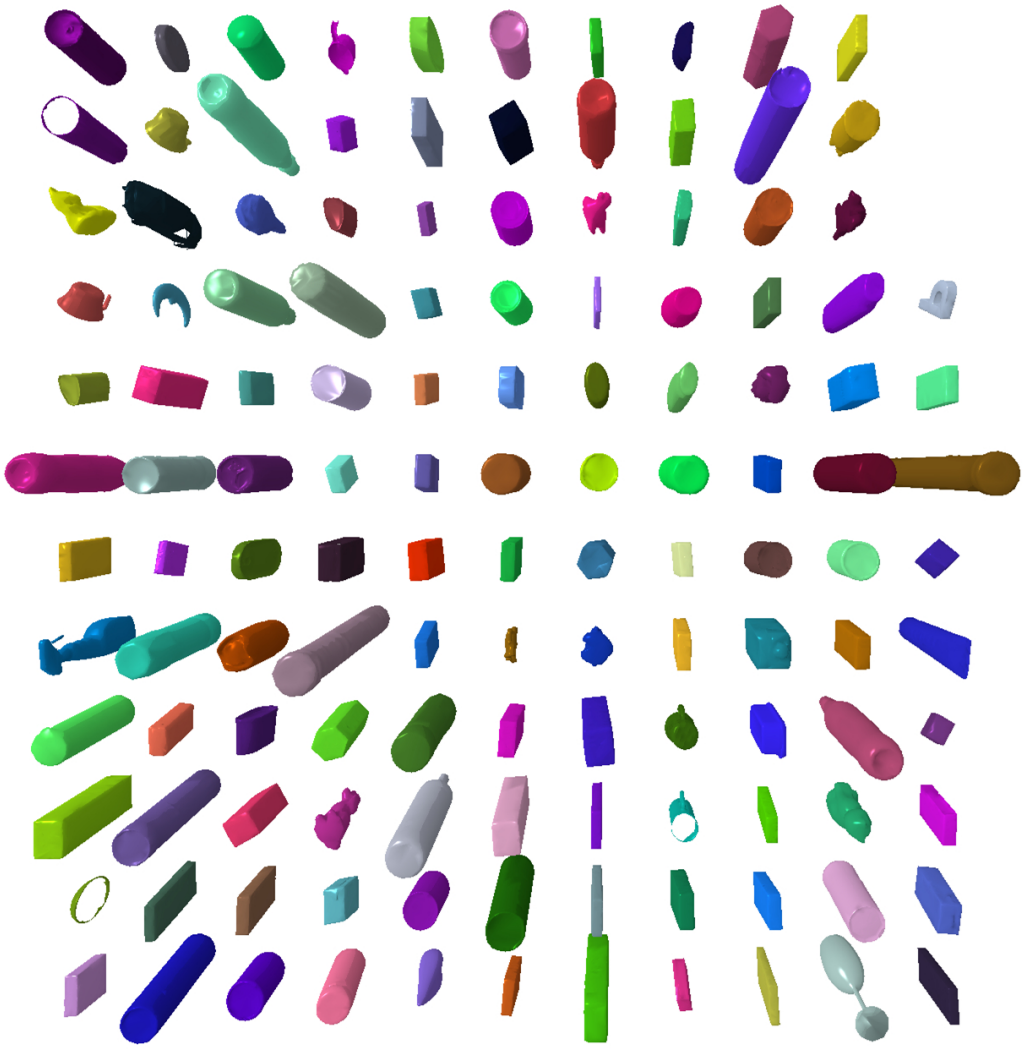


Figure 15: 129 Kit objects.

Figure 16: 32 real objects.

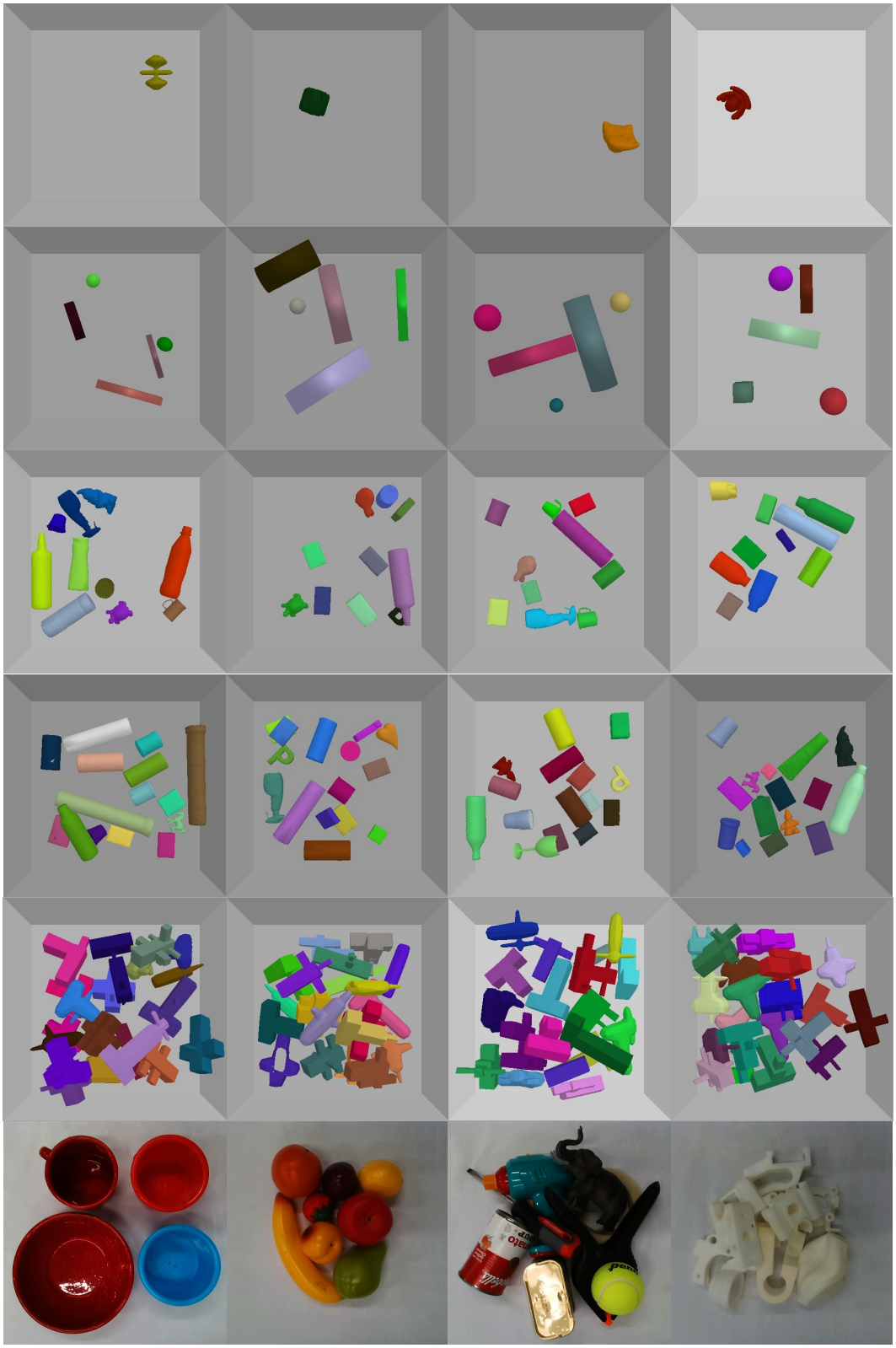


Figure 17: Sampled scenes. Row 1: single object. Row 2: 5 objects. Row 3: 10 objects. Row 4: 15 objects. Row 5: 20 objects. Row 6: real objects.

References

[1] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Siggraph Computer Graphics*, 1987.

[2] Y. Wu and K. He. Group normalization. In *The European Conference on Computer Vision (ECCV)*, 2018.

[3] M. Gou, H.-S. Fang, Z. Zhu, S. Xu, C. Wang, and C. Lu. Rgb matters: Learning 7-dof grasp poses on monocular rgbd images. In *The International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[4] J. Cai, J. Cen, H. Wang, and M. Y. Wang. Real-time collision-free grasp pose detection with geometry-aware refinement using high-resolution volume. *IEEE Robotics and Automation Letters*, 2022.

[5] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018.

[6] C. V. Nguyen, S. Izadi, and D. Lovell. Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *The International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE, 2012.

[7] D. Morrison, P. Corke, and J. Leitner. Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation. *IEEE Robotics and Automation Letters*, 2020.

[8] D. Morrison, P. Corke, and J. Leitner. Multi-view picking: Next-best-view reaching for improved grasping in clutter. In *The International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.