2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

# Multi-Object RANSAC: Efficient Plane Clustering Method in a Clutter

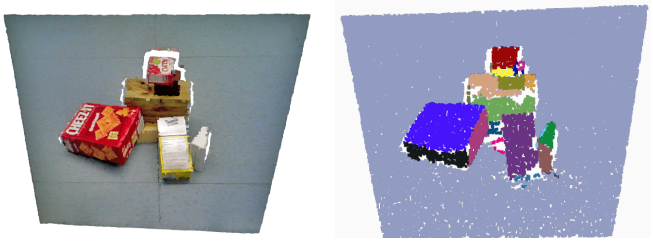
Seunghyeon Lim[1], Youngjae Yoo[2], Jun Ki Lee [3*] and Byoung-Tak Zhang[1,2,3*]
{shlim, yjyoo}@bi.snu.ac.kr, junkilee@snu.ac.kr, btzhang@bi.snu.ac.kr

***Abstract*— In this paper, we propose a novel method for plane clustering specialized in cluttered scenes using an RGB-D camera and validate its effectiveness through robot grasping experiments. Unlike existing methods, which focus on large-scale indoor structures, our approach—Multi-Object RANSAC emphasizes cluttered environments that contain a wide range of objects with different scales. It enhances plane segmentation by generating subplanes in Deep Plane Clustering (DPC) module, which are then merged with the final planes by post-processing. DPC rearranges the point cloud by voting layers to make subplane clusters, trained in a self-supervised manner using pseudo-labels generated from RANSAC. Multi-Object RANSAC demonstrates superior plane instance segmentation performances over other recent RANSAC applications. We conducted an experiment on robot suction-based grasping, comparing our method with vision-based grasping network and RANSAC applications. The results from this real-world scenario showed its remarkable performance surpassing the baseline methods, highlighting its potential for advanced scene understanding and manipulation.**

## I. INTRODUCTION

Plane clustering with point clouds plays a pivotal role in finding geometric cues essential for scene understanding in robotics. In particular, plane clustering in cluttered environments (Fig. 1a) is a significant challenge in robotic perception. This challenge is a key enabler for more sophisticated robotic grasping and improved manipulation capabilities. Although there are various methods for clustering planes with a point cloud or depth image [1]–[3], these techniques have mainly been designed for scenarios that involve larger objects such as desks, tables, and doors. Our approach—Multi-Object RANSAC (MO-RANSAC), shifts its focus towards cluttered environments encompassing diverse objects of varying scales. By achieving plane instance segmentation within such intricate settings, our method empowers robots to undertake complex grasping and manipulation tasks.

RANSAC, known as RANdom SAmple Consensus, offers a rapid and robust solution for clustering basic shapes, such as planes in our specific case [4]. This classical algorithm finds utility across various computer vision tasks, including fitting planes in 3D and lines in 2D. Through an iterative process, RANSAC selectively samples points where the distance from a plane hypothesis falls below a certain threshold (i.e. inlier points). Subsequently, it identifies the optimal parametric model with the greatest consensus among these sampled points. This simple yet efficient approach has gained widespread adoption such as [5]–[8].



(a) The cluttered environment (b) Plane clustering

Fig. 1: A plane clustering result of Multi-Object RANSAC (MO-RANSAC, our method).

We propose Multi-Object RANSAC which also takes point clouds from the RGB-D data as input. It is composed of two key components: Deep Plane Clustering (DPC) and Post-processing. DPC is trained in a self-supervised manner, generating pseudo-subplane labels through RANSAC. DPC first passes point clouds through a backbone network and subsequently through voting layers [9], which reorganize the point clouds to effectively group subplane clusters. Consequently, all points within subplane clusters lie on the same plane. The subsequent step, post-processing, combines multiple clusters that represent the same plane. Fig. 1 shows the results of the plane clustering using RGB-D data from the OCID dataset [10].

Through this approach, Multi-Object RANSAC learns the geometric attributes of planes. Our experiments have affirmed that using a naive clustering algorithm—an ablated version of our method—would result in the failure to cluster all inlier points located on the same plane. Additionally, such an approach would also lack scalability and adaptability to various environmental factors. On the contrary, Multi-Object RANSAC is trained using point clouds from various environments [10], allowing it to effectively mitigate these challenges.

In this paper, we make qualitative and quantitative evaluations of plane clustering. Our experiment is conducted using two datasets: OCID [10] and OSD [11] which encompass RGB-D data captured in various cluttered scenes. To measure the effectiveness of our instance segmentation, we create ground-truth labels by generating plane segments for each object and the background. We compared our approach with

This work was partly supported by the IITP (2021-0-02068-AIHub/15%, 2021-0-01343-GSAI/20%, 2022-0-00951-LBA/25%, 2022-0-00953-PICA/25%) and NRF (RS-2023-00274280/15%) grant funded by the Korean government.

*Corresponding authors.
[1]Interdisciplinary Program in Cognitive Science, Seoul National University, Seoul, Korea
[2]Department of Computer Science and Engineering, Seoul National University, Seoul, Korea
[3]AIIS, Seoul National University, Seoul, Korea

recent implementations of RANSAC [5], [7], [8] and a multi-plane clustering method [1]. In the experiment, Multi-Object RANSAC outperforms the baseline methods across most scenarios. It also shows qualitative improvements in clustering performances on the datasets and real-world cluttered environments.

In real-world scenarios, we performed suction grasp experiments using an UR5 robot arm. The robot was equipped with a RealSense camera at its end effector, receiving RGB-D images. In particular, Multi-Object RANSAC showed a superior grasp accuracy rate compared to existing vision-based suction grasping [12] and plane clustering methods [5], [7], [8]. Ultimately, the results underline the possibilities of numerous applications for advanced robotic tasks, such as scene understanding and robotic manipulation.

## II. Related Works

### A. Plane Clustering

Plane clustering is of significant importance in the field of robotics. Some studies have employed plane extraction for 3D indoor mapping and localization [2], [13]. Oriented point sampling [1] uses an unorganized point cloud with estimated normals to fit planes. However, these methods primarily target expansive indoor environments [14]–[17], rendering them unsuitable for complex settings with multiple objects. Many studies [18]–[20] use the Hough voting to find basic shapes. We developed this strategy using VoteNet voting layers [9], which represents a recent integration of Hough voting into neural networks.

RANSAC-based shape fitting techniques are versatile and robust approaches with low computational costs. Efficient RANSAC [21] can identify multiple geometric shapes by random sampling. CC-RANSAC [22] and NCC-RANSAC [23] address RANSAC's limitation in distinguishing between adjacent planes by identifying suitable groups of connected inlier points. In addition, some studies [6], [24] use neural networks to efficiently sample inlier points, using self-supervised or supervised methodologies. However, their studies do not emphasize plane fitting or other 3D geometric shapes and focus on computer vision tasks such as estimation of homography and epipolar geometry.

### B. Suction Grasp in a Dense Clutter

Robotic suction grasping is known as an efficient method to handle objects on various scales without damage, widely used in many industrial applications. Recent works focus on methods that directly find grasp points by assigning grasp quality scores in images or 3D spaces.

SuctionNet 1-Billion [12] is an example of a vision-based suction grasping network. It operates by generating heat maps that show the feasibility of suction grasps based on the shapes and physical characteristics of the object. Dex-Net 3.0 [25] and Shao et al. [26] employ specialized deep learning frameworks designed to extract features to identify graspable suction points.

**Algorithm 1** Multi-Object RANSAC

**Require:** $X=\{x_i\}_{i=1}^{N}$, RGB data $\{c_i\}_{i=1}^{N}$, normals $\{u_i\}_{i=1}^{N}$
1: ▷ $x_i, c_i, u_i \in \mathbb{R}^3$
2: Sample $K$ points from $X$ by Farthest Point Sampling.
3: Get $\tilde{X}$ by concatenating RGB data and normals to $X$.
4: ▷ $\tilde{X} = \{\tilde{x}_i\}_{i=1}^{N}$ s.t. $\tilde{x}_i \in \mathbb{R}^9$
5: $\mathcal{F} \leftarrow$ BackboneNetwork($\tilde{X}$) ▷ $\mathcal{F} \in \mathbb{R}^{N\times 128}$
6: $\Delta X \leftarrow$ VotingLayers($\mathcal{F}$)
7: ▷ $\Delta X = \{\Delta x_i\}_{i=1}^{N}$ s.t. $\Delta x_i \in \mathbb{R}^3$
8: Define $P = \{p_i\}_{i=1}^{N}$ s.t. $p_i = x_i + \Delta x_i$.
9: Define $S \subset P$ that belongs to the sampled $K$ points.
10: ▷ $|S| = K$
11: Align all points to the nearest $p_s$ s.t. $p_s \in S$.
12: ▷ Making subplane clusters.
13: **for** each cluster **do**
14: Apply RANSAC to determine the plane equation and identify inlier points.
15: **end for**
16: Apply contrastive loss to gather inliers and push against outliers towards the $p_s$ in line 11.
17: ▷ pseudo-labels from RANSAC in lines 13-15.
18: Post-processing by merging clusters that lie on the same plane during the inference.
19: ▷ See Algorithm 2.
20: **return** clusters

This paper introduces the application of MO-RANSAC to robotic tasks by suction grasping tasks. Note that MO-RANSAC does not require one to learn grasp points unlike the above methods, but still maintains high accuracy in cluttered environments.

## III. Methods

### A. Overview

Alg. 1 provides an overview of the Multi-Object RANSAC (MO-RANSAC) procedure. MO-RANSAC takes a 3D point cloud denoted as $X = \{x_i\}_{i=1}^{N}$ s.t. $x_i \in \mathbb{R}^3$ and utilizes Open3D [27] for normal estimation, determining the principal axis of nearby points selected through KDTreeSearch. We align RGB and depth images with identical resolutions and transform these 2D images into 3D point clouds using the intrinsic parameters of the camera. As a result, an input point cloud is concatenated with RGB data $\{c_i\}_{i=1}^{N}$ and 3D normal vectors $\{u_i\}_{i=1}^{N}$, forming $\tilde{X} = \{\tilde{x}_i\}_{i=1}^{N}$ s.t. $\tilde{x}_i \in \mathbb{R}^9$ (Alg. 1 lines 2-4). It establishes subplane clusters within the Deep Object Clustering (DPC) phase, where all points belonging to the same subplane cluster are situated on the same plane (Alg. 1 lines 1-17). However, these subplane clusters may not represent optimal solutions for plane clustering, as it is not guaranteed that multiple subclusters correspond to the same planar surfaces as shown in Fig. 2 (Clusters after DPC). For this reason, we add the post-processing (Alg. 1 lines 18-19) to effectively merge them into final clusters.

### B. Deep Plane Clustering

Deep Plane Clustering (DPC) serves as a self-supervised deep learning framework designed to cluster points that re-

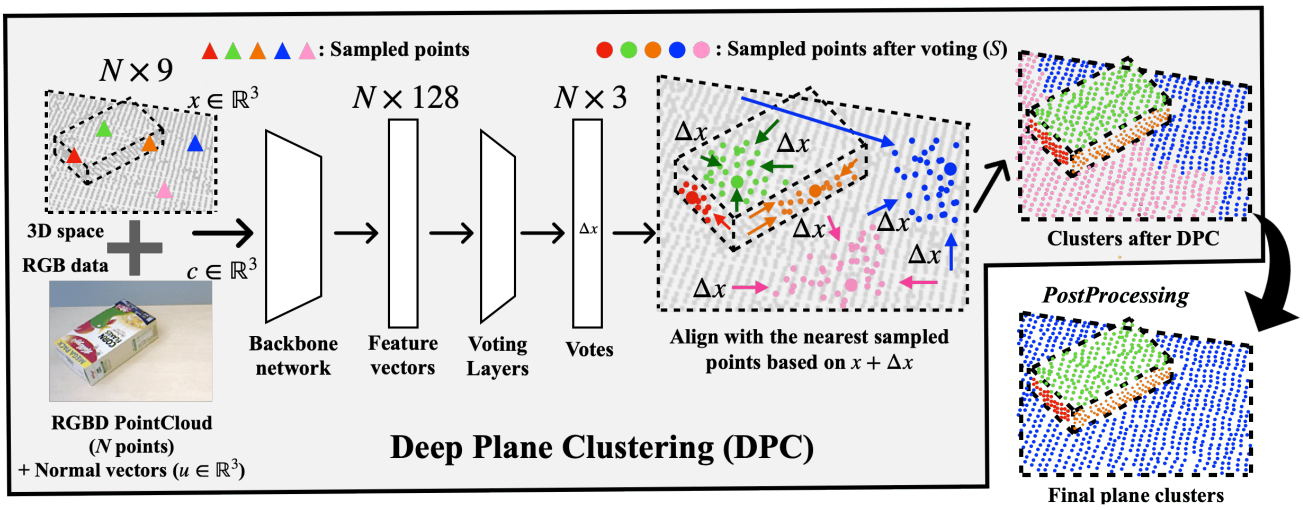



Fig. 2: **Overall framework of Multi-Object RANSAC.** Deep Plane Clustering (DPC) takes a 9D point cloud and produces votes in the form of $\{\Delta x\}_{i=1}^{N}$ s.t. $\Delta x \in \mathbb{R}^3$. Meanwhile, DPC initially samples $K$ points (triangle-shaped points), to individually represent clusters after voting (large circle-shaped points, S). Each point is then associated with the nearest sampled point, resulting in the creation of $K$ subplane clusters. These subplane clusters undergo further refinement through post-processing following DPC.

side on the same plane. This is achieved by taking advantage of the principles of the RANSAC algorithm.

In this framework, RANSAC is employed individually for each cluster, generating pseudo-labels that distinguish inliers—points close to the plane—from outliers—points farther away. Fig. 2 shows the framework of our model. Before using neural networks, an initial step involves Farthest Point Sampling (FPS) on the 3D point cloud to select $K$ points (Alg. 1 line 2). After being processed by the network, each of these sampled points will independently form its subplane cluster (Alg. 1 lines 9-12).

The DPC framework comprises a backbone network and voting layers. Taking inspiration from VoteNet [9], we have changed the network to fit our self-supervised plane clustering problem. The backbone network propagates the feature vectors for each point. We use PointNet++ [28] for the backbone network and produce a 128-dimensional feature vector for each point in our settings.

The voting layers consist of two fully connected linear networks that produce 3-dimensional features for every point. These features are termed votes, since they are subsequently added to each $x_i$ (i.e., $x_i + \Delta x_i$). Each voting layer is followed by a non-linear network, and the final nonlinear network is succeeded by a 1D BatchNorm [29] layer. Obtaining $\Delta X = \{\Delta x_i\}_{i=1}^{N}$ from the voting layers, DPC uses $\{x_i + \Delta x_i\}_{i=1}^{N}$ to make subplane clusters, denoted by $P$ (Alg. 1 line 8).

Once the vote is complete, each point $p_i = x_i + \Delta x_i$ is grouped to create a different subplane cluster $\mathcal{C}_k$. Also, let $S$ denote the set of $K$ sampled after voting. All points in $P$ select the cluster with the nearest L2 distance to the "sampled point" among $K$ points (Alg. 1 lines 9-12). For better understanding, we re-assign the sampled point after votes $p_s \in S$ in the $k$th cluster as $p^k$. This allows us to describe the process as follows.

$$\text{if} \quad p \in \mathcal{C}_l, \quad \text{then} \quad l = \underset{k=1,\dots,K}{\arg\min} \|p - p^k\|_2 \tag{1}$$

After aligning the points with the $K$ clusters, we utilize the RANSAC algorithm with the 3D point cloud $X$ to extract plane equations along with the inlier points within each plane (Alg. 1 lines 13-15). This step produces sets of inlier points $I_k$, outlier points $O_k$, which serve as pseudolabels, and 3D plane equations $eq_k$ for the $k$th cluster.

We define a self-supervised contrastive loss function that guides DPC to encourage points in $I_k$ to move closer to $p^k$ and points in $O_k$ to move away from $p^k$ (Alg. 1 lines 16-17). In this way, the network learns to group all the points belonging to the ground truth subplane clusters. Let the $j$th point in $\mathcal{C}_k$ as $p_{kj}$, then this is accomplished using a simple contrastive loss $L_k$ with the L1 norm for the $k$th subplane cluster:

$$L_k = \frac{1}{|I_k|} \sum_{p_{kj} \in I_k} \|p_{kj} - p^k\| + \frac{1}{|O_k|} \sum_{p_{kj} \in O_k} [\alpha - \|p_{kj} - p^k\|]_+ \tag{2}$$

Here, $[a]_+ = \max(0, a)$ and $\alpha$ is set to 3. The overall loss function $L$ across the $K$ clusters is defined as:

$$L = \frac{1}{K} \sum_{k=1}^{K} L_k \tag{3}$$

During training, the cluster number $K$ is randomly selected between 8 and 196 for each prediction.

Furthermore, we have made a slightly different method in the inference process. Unlike the training process, a 9D point cloud $\tilde{X}$ is initially divided into 3 clusters using K-means clustering, which are then separately fed into DPC, in the steps corresponding to lines 3 to 4 of Alg. 1. This allows DPC to concentrate more on points with similar normal vectors and locations.

**Algorithm 2** Merge Process in Post-processing

**Require:** $cluster\_dict$
**Ensure:** $\mathcal{X}_k = \{x_{k1}, x_{k2}, ..., x_{k(n-1)}, x_{kn}\}$ s.t. $|\mathcal{X}_k| = n$
1: ▷ $x_{kj} \leftarrow$ the $j$th 3D point w/o voting in the $k$th cluster
2: $\mathcal{X}_{cluster\_id}, eq_{cluster\_id} \in cluster\_dict$
3: Define center points of each cluster as $cluster\_center$
4: Get KNN graph for $cluster\_center$ as $G$ which consists of a set of vertices (clusters) and edges $(V, E)$
5: **while** not all vertices of $G$ are visited **do**
6: Define an unvisited vertex $\mathcal{X}_a$ that meets $\mathcal{X}_a \in V$
7: **for** $\mathcal{X}_b \in \{\mathcal{X}_b | (\mathcal{X}_a, \mathcal{X}_b) \in E\}$ **do**
8: $m_a = |\{x | x \in \mathcal{X}_a, d(x, eq_b) < \delta\}| / |\mathcal{X}_a|$
9: $m_b = |\{x | x \in \mathcal{X}_b, d(x, eq_a) < \delta\}| / |\mathcal{X}_b|$
10: $min_d(\mathcal{X}_a, \mathcal{X}_b) \leftarrow \min(x_{ai}, x_{bj})$ for all $i$ and $j$
11: **if** $min_d(\mathcal{X}_a, \mathcal{X}_b) < \beta$ and $\max(m_a, m_b) > \gamma$ **then**
12: Merge $\mathcal{X}_a, \mathcal{X}_b$ into single $\mathcal{X}_a$
13: Update $cluster\_dict$
14: **end if**
15: **end for**
16: **end while**

### C. Post-processing Module

Once the DPC module has performed a clustering on a set of $N$ points, the post-processing module performs a further refinement by merging clusters that are located on the same planar surface. It is important to note that post-processing is required only during the inference phase and is not applied during the training. The post-processing module continuously combines two other nodes by the merge process. The details are explained in Alg. 2.

The merge process itself is a graph-based algorithm whose graph is constructed by establishing connections between nearest neighbors based on the centroids' distances between clusters. In the context of Alg. 2, a data structure named $cluster\_dict$ is defined. This structure stores cluster identifiers ($cluster\_id$), plane equations ($eq_{cluster\_id}$) obtained in Alg. 1 lines 13-15, and the sets of points without voting values associated with these clusters ($\{\mathcal{X}_k\}_{k=1}^{K}$). After the graph is created successfully, we compare two connected vertices (clusters). Two vertices are merged (a) if the minimum distance between points in set A ($\mathcal{X}_a$) and set B ($\mathcal{X}_b$) is less than $\beta$, and (b) if the portion of points in $\mathcal{X}_a$ whose distances from plane equations $eq_b$ fall below $\delta$ exceeds the threshold $\gamma$, and vice versa (as shown in Alg. 2 lines 8-14). We have set $\beta$, $\gamma$, and $\delta$ at 0.2, 0.9, and 0.005, respectively.

At inference, $K$ subplane clusters ($cluster\_dict$) are generated for each of the three large point sets from K-means clustering in DPC. Initially, the merge process is applied to individual sets of $cluster\_dict$ for each $K$, followed by a secondary merge process for all $cluster\_dict$ after the first merge.

The primary bottleneck in terms of computational cost occurs in line 11 of Alg. 2, where we need to compare all points between two clusters in each iteration. Assuming that each cluster has $\mathcal{U}$ nearest neighbors and there are a total of $N$ points, the computational cost becomes $O(\mathcal{U}KN^2)$.

## IV. Experiments

### A. Experiments Setup

We used the OCID [10] dataset for training DPC. We randomly sampled the dataset and split it into the train and test sets. MO-RANSAC was trained with the 2,080 scenes and tested with the 300 scenes. We also used the OSD [11] dataset for the test dataset, which contains cluttered scenes from table views. We trained MO-RANSAC on an NVIDIA TITAN RTX for 10 epochs with a learning rate of $10^{-5}$ and weight decay of $10^{-5}$. Each point cloud from a single image is downsampled to 32768 points.

We conducted experiments with three recently developed RANSAC methods, GC-RANSAC [5], MAGSAC [7], and MAGSAC++ [8], as our baselines. We implemented multi-plane clustering iteratively since they made a single model hypothesis per execution. The methods continuously identify a plane hypothesis, extracting inlier points, and then proceed with the same process for the remaining points in the point cloud. We also compared MO-RANSAC with Oriented Point Sampling (OPS) [1]—a multiple plane clustering algorithm with point clouds designed for robotic applications.

### B. Qualitative Results

Fig. 3-5 illustrates the outcomes of plane clustering applied to images from the OCID Dataset and the real-world. Clusters are depicted in distinct colors and projected onto 2D images using camera intrinsic parameters.

Fig. 3 shows the comparison of the segmentation results with other RANSAC applications. It is evident that the baseline methods mostly failed to accurately segment surfaces from individual objects. In contrast, MO-RANSAC consistently exhibits superior segmentation performance in all scenarios. Although MO-RANSAC may encounter challenges in distinguishing surfaces amidst closely positioned objects indicated by blue circles, it effectively shows its capabilities by accurately segmenting planar surfaces on smaller objects indicated by red circles.

In Fig. 4, we made a comparison with an ablated version that makes subclusters without using DPC (FPS + PostProcess). Specifically, it excludes the use of votes ($\Delta x$), opting to create subclusters by $K$ sampled points (Alg. 1 line 11) based solely on the $x$ values instead of $p$ (Equation 1). The ablated version produces less refined clustering outcomes, as the points within the same subcluster do not lie on a common plane. This poses a significant risk during the subsequent merge processes in the post-processing.

Additionally, experiments were conducted on real-world objects. Fig. 5 shows examples of plane clustering outcomes. We conducted a suction grasping experiment on these objects. The details are given in Section IV.D.

### C. Plane Instance Segmentation Result

We conducted an instance segmentation experiment on the planar clustering results, utilizing the experimental settings detailed in Section IV.A. Consistent with previous studies, we employed Variation of Information (VOI), Rand Index (RI), and Segmentation Covering (SC) as evaluation metrics

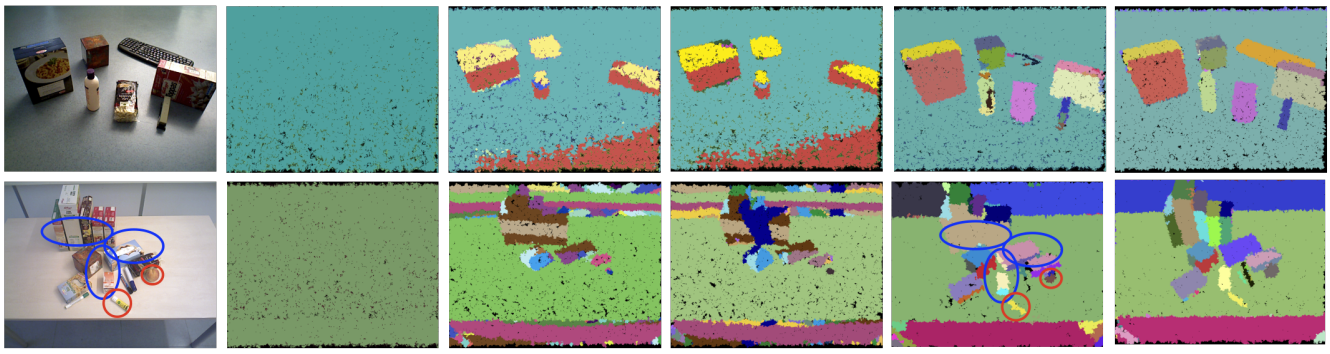



Fig. 3: Segmentation results compared to plane clustering baselines: red circles represent successful segmentation outcomes for small objects, while blue circles denote segmentation failures for nearby objects.

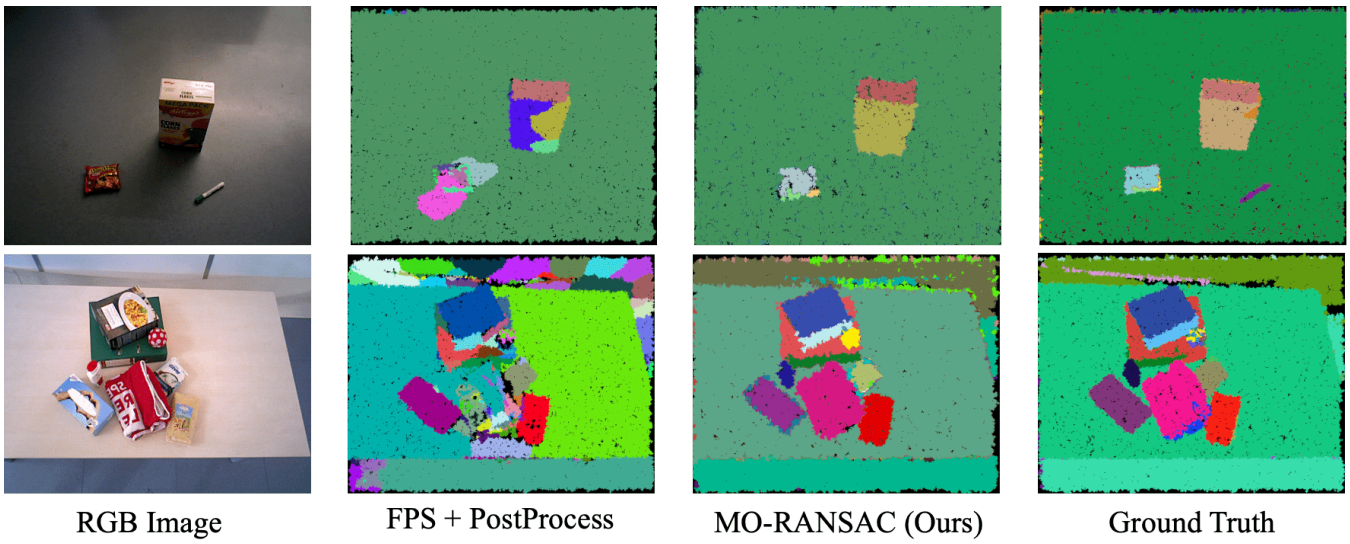
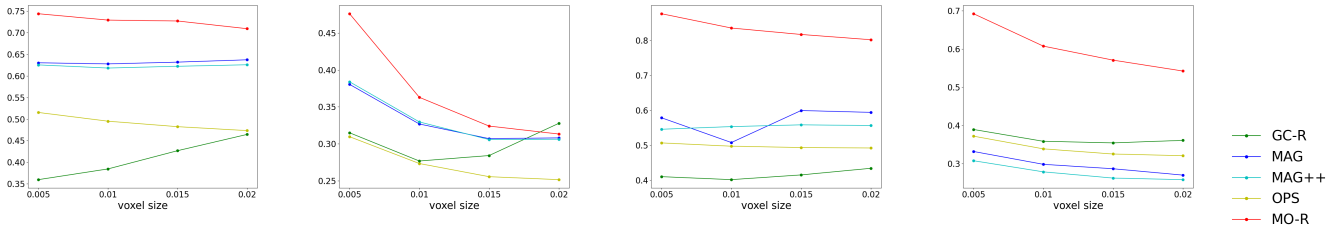



Fig. 4: Ablation studies of MO-RANSAC and the ground truth.

Fig. 5: Real-world examples.

TABLE I: Segmentation performance comparisons over OCID and OSD datasets.

| Method | Datasets | | | | | |
|---|---|---|---|---|---|---|
| | OCID [10] | | | OSD [11] | | |
| | VOI ↓ | RI | SC | VOI ↓ | RI | SC |
| GC-RANSAC [5] | **2.584** | 0.360 | 0.315 | 2.181 | 0.411 | 0.389 |
| MAGSAC [7] | 3.004 | 0.630 | 0.380 | 3.392 | 0.579 | 0.332 |
| MAGSAC++ [8] | 2.939 | 0.626 | 0.384 | 3.324 | 0.546 | 0.307 |
| OPS [1] | 2.989 | 0.515 | 0.310 | 2.594 | 0.507 | 0.372 |
| MO-RANSAC (Depth) | 2.941 | 0.738 | 0.457 | 1.980 | 0.871 | **0.694** |
| MO-RANSAC (RGB + Depth) | 2.845 | **0.744** | **0.476** | **1.974** | **0.876** | 0.692 |




(a) OCID - RI (b) OCID - SC (c) OSD - RI (d) OSD - SC

Fig. 6: Performance change of MO-RANSAC based on the **voxel size** (m) to sample down for evaluation.

[30], [31]. For evaluation, we transformed the point cloud into voxels with a voxel size of 0.5cm. The experimental results are given in Table I. MO-RANSAC outperforms other clustering algorithms in most scenarios. Note that while GC-RANSAC exhibits a relatively high VOI, it shows poor clustering performance as in Fig. 3. This is because it tends to group all the points into a single cluster, unlike other methods which tend to produce multiple smaller false-positive clusters. We also conducted a comparison involving MO-RANSAC using solely depth information as input (MO-RANSAC (Depth)). While MO-RANSAC with RGB-D data exhibits slight improvements in performance except for SC on the OSD datasets, RGB inputs provide additional visual features that aid in comprehending the object's geometric attributes.

Figure 6 shows the variations in RI and SC based on the voxel sizes set for evaluation. Here, it becomes evident that MO-RANSAC shows significant enhancements in plane segmentation for smaller voxel sizes, whereas other methods display relatively weak correlations. As smaller voxel sizes encapsulate more intricate object geometric information, we can conclude that MO-RANSAC exhibits robustness in cluttered environments with smaller objects than the others.

### *D. Suction Grasp Performance in the Real World*

**Baselines** We conducted two distinct experiments to evaluate the effectiveness of our suction grasping approach. The first experiment compared MO-RANSAC with the plane clustering baselines (Table II). In the second experiment, we pit MO-RANSAC against the SuctionNet Baseline [12], a vision-based suction grasping network that infers graspable points from an RGB-D camera (Table III). By conducting these experiments using different baseline methods, we aimed to show the usefulness of MO-RANSAC in robotic grasping.

**Setup** We use an UR5 robot arm equipped with ROBOTIQ one-cup suction gripper on its end effector. An Intel Realsense D435 RGB-D camera was also affixed to the end effector, and objects were on a flat table within the camera's observable range. We collected 21 objects from the YCB [32], HOPE [33] datasets, and the local grocery items. To ensure fairness, the experiment in Table III exclusively employed 6 objects (cracker box, tomato soup can, sugar box, mug, cup, mustard), which were the same as or similar to those used for training the SuctionNet Baseline [12].

**Rules** Each experiment consists of 10 rounds. In the experiment in Table II, each round includes 7 objects: 5 with planar surfaces (e.g., boxes) and 2 with curved surfaces (e.g., cans). In the other experiment (Table III), 6 objects are present in every round for the experiment. The objective is to successfully grasp and relocate all objects to a predefined location. A round ends when the robot effectively moves all objects to the predefined location. If the robot cannot grasp an object twice consecutively, the object is manually removed and the round continues.

TABLE II: Comparisons of suction grasp accuracy with other plane clustering methods. (S: Success, C: Clearance)

| Method | Plane | | Curved | | Overall | |
|---|---|---|---|---|---|---|
| | S | C | S | C | S | C |
| GC-RANSAC [5] | 0.302 | 0.520 | 0.452 | 0.700 | 0.342 | 0.571 |
| MAGSAC [7] | 0.561 | 0.740 | 0.464 | 0.650 | 0.532 | 0.714 |
| MAGSAC++ [8] | 0.551 | 0.760 | 0.364 | 0.600 | 0.490 | 0.714 |
| MO-RANSAC | **0.717** | **0.860** | **0.536** | **0.750** | **0.659** | **0.829** |

TABLE III: Comparison of suction grasp accuracy with a vision-based suction grasping method.

| Method | Success | Clearance |
|---|---|---|
| SuctionNet Baseline [12] | 0.248 (26 / 105) | 0.433 (26 / 60) |
| MO-RANSAC | **0.482 (42 / 87)** | **0.700 (42 / 60)** |

**Grasp Point Inference** To determine the grasp point, the robot selects the centroid of the cluster farthest from the floor underneath the clustered objects.

**Evaluation Metrics** We utilize two key metrics: **success rate** and **clearance rate** [34]. The success rate is the ratio of successful grasps to the total attempts. The clearance rate represents the proportion of objects successfully cleared by robotic grasping.

**Results** MO-RANSAC demonstrates significant enhancements in suction-based grasping in both experiments. In Table II, MO-RANSAC achieves 16% higher success rates for planar surface objects and 7% higher for curved surface objects compared to MAGSAC [7], which showed the most favorable baseline performance. In particular, the comparative performance of suction grasp is closely correlated with the overall instance segmentation results in Fig. 3 and Table I. This emphasizes the critical role of plane instance segmentation in the effectiveness of robotic grasping.

For the experiment in Table III, MO-RANSAC shows a 23% increase in success rate and a 26% increase in clearance rate. SuctionNet 1-Billion shows a poor grasp accuracy rate due to the mismatch between the experimental environment and the training dataset conditions. This discrepancy exists even though the method uses objects from their list that reported superior performance in their paper [12]. From this perspective, MO-RANSAC emerges as a possible substitute for robotic grasp networks, capable of adapting to various cluttered environments.

## V. CONCLUSIONS

This paper introduces MO-RANSAC, a novel plane clustering technique specially designed for cluttered environments involving multiple objects, leveraging data from an RGB-D camera. MO-RANSAC efficiently rearranges points via voting layers for plane clustering. MO-RANSAC excels in complex plane clustering scenarios and shows promise for real-world robotics applications, including suction grasping.

## ACKNOWLEDGMENT

We extend our heartfelt gratitude to Jae-in Kim and Hee-bin Yoo for their invaluable insights and help with the revision of this manuscript.

## REFERENCES


[1] B. Sun and P. Mordohai, "Oriented point sampling for plane detection in unorganized point clouds," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2917–2923.

[2] T. T. Pham, M. Eich, I. Reid, and G. Wyeth, "Geometrically consistent plane extraction for dense indoor 3d maps segmentation," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 4199–4204.

[3] R. Hulik, V. Beran, M. Spanel, P. Krsek, and P. Smrz, "Fast and accurate plane segmentation in depth maps for indoor scenes," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1665–1670.

[4] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[5] D. Barath and J. Matas, "Graph-cut ransac," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6733–6741.

[6] E. Brachmann and C. Rother, "Neural-guided ransac: Learning where to sample model hypotheses," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4322–4331.

[7] D. Barath, J. Matas, and J. Noskova, "Magsac: marginalizing sample consensus," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 197–10 205.

[8] D. Barath, J. Noskova, M. Ivashechkin, and J. Matas, "Magsac++, a fast, reliable and accurate robust estimator," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1304–1312.

[9] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[10] M. Suchi, T. Patten, D. Fischinger, and M. Vincze, "Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic RGB-D datasets," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6678–6684. [Online]. Available: https://doi.org/10.1109/ICRA.2019.8793917

[11] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4791–4796.

[12] H. Cao, H.-S. Fang, W. Liu, and C. Lu, "Suctionnet-1billion: A large-scale benchmark for suction grasping," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8718–8725, 2021.

[13] P. Kim, B. Coltin, and H. J. Kim, "Linear rgb-d slam for planar environments," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 333–348.

[14] S. Song, S. P. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 567–576.

[15] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[16] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[17] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, May 2014.

[18] B. Drost and S. Ilic, "Local hough transform for 3d primitive detection," in *2015 International Conference on 3D Vision*. IEEE, 2015, pp. 398–406.

[19] E. Vera, D. Lucio, L. A. Fernandes, and L. Velho, "Hough transform for real-time plane detection in depth images," *Pattern Recognition Letters*, vol. 103, pp. 8–15, 2018.

[20] C. Sommer, Y. Sun, E. Bylow, and D. Cremers, "Primitect: Fast continuous hough voting for primitive detection," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8404–8410.

[21] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, no. 2. Wiley Online Library, 2007, pp. 214–226.

[22] O. Gallo, R. Manduchi, and A. Rafii, "Cc-ransac: Fitting planes in the presence of multiple surfaces in range data," *Pattern Recognition Letters*, vol. 32, no. 3, pp. 403–410, 2011.

[23] X. Qian and C. Ye, "Ncc-ransac: A fast plane extraction method for 3-d range data segmentation," *IEEE transactions on cybernetics*, vol. 44, no. 12, pp. 2771–2783, 2014.

[24] F. Kluger, E. Brachmann, H. Ackermann, C. Rother, M. Y. Yang, and B. Rosenhahn, "Consac: Robust multi-model fitting by conditional sample consensus," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4634–4643.

[25] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *2018 International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5620–5627.

[26] Q. Shao, J. Hu, W. Wang, Y. Fang, W. Liu, J. Qi, and J. Ma, "Suction grasp region prediction using self-supervised learning for object picking in dense clutter," in *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*. IEEE, 2019, pp. 7–12.

[27] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[29] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[30] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz, "Planercnn: 3d plane detection and reconstruction from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4450–4459.

[31] Y. Xie, M. Gadelha, F. Yang, X. Zhou, and H. Jiang, "Planarrecon: Real-time 3d plane detection and reconstruction from posed monocular videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6219–6228.

[32] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[33] S. Tyree, J. Tremblay, T. To, J. Cheng, T. Mosier, J. Smith, and S. Birchfield, "6-dof pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark," in *International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[34] M. Breyer, J. J. Chung, L. Ott, S. Roland, and N. Juan, "Volumetric grasping network: Real-time 6 dof grasp detection in clutter," in *Conference on Robot Learning*. PMLR, 2021, pp. 1602–1611.