IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Learning to Place Unseen Objects Stably using a Large-scale Simulation

Sangjun Noh*, Raeyoung Kang*, Taewon Kim*, Seunghyeok Back, Seongho Bak, Kyoobin Lee†

***Abstract*— Object placement is a fundamental task for robots, yet it remains challenging for partially observed objects. Existing methods for object placement have limitations, such as the requirement for a complete 3D model of the object or the inability to handle complex shapes and novel objects that restrict the applicability of robots in the real world. Herein, we focus on addressing the Unseen Object Placement (UOP) problem. We tackled the UOP problem using two methods: (1) UOP-Sim, a large-scale dataset to accommodate various shapes and novel objects, and (2) UOP-Net, a point cloud segmentation-based approach that directly detects the most stable plane from partial point clouds. Our UOP approach enables robots to place objects stably, even when the object's shape and properties are not fully known, thus providing a promising solution for object placement in various environments. We verify our approach through simulation and real-world robot experiments, demonstrating state-of-the-art performance for placing single-view and partial objects. Robot demos, codes, and dataset are available at https://gistailab.github.io/uop/**

I. Introduction

Robots need to have the ability to manipulate unseen objects to operate effectively in various environments, which are common in manufacturing, construction, and household tasks. While deep learning has progressed to recognize and handle unseen objects, most of the current research focuses on identifying [1], [2] or grasping [3], [4] them [5], [6]. However, it is important to note that when a robot picks up an object from a cluttered container or receives it from a human, the robot must be able to place the object stably. Thus, this study addresses the **U**nseen **O**bject **P**lacement (**UOP**) problem, which involves stably placing novel objects in a real-world environment.

Conventional approaches [7], [8] for stable object placement require full 3D models and analytical calculations. These methods involve sampling stable planes after calculating the center of mass of the object, which is not feasible for all real-world objects that may be encountered. One approach [9] combines analytical methods with a 3D object completion model that can reconstruct the full shape of an object from raw perception data. However, using this approach is difficult since the predicted point cloud may not be precise, resulting inaccuracies in determining stable planes. Our UOP method addresses these limitations by directly detecting stable planes of unseen objects from single views and partial point clouds, thus eliminating the need for a full 3D object model. This enables the robot to stably place the object even when the shape and properties of the object are not fully known.

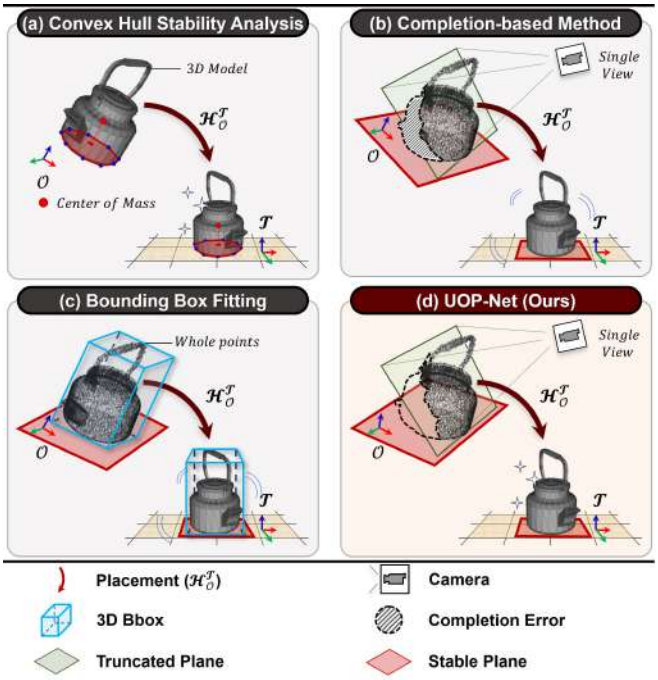

Fig. 1: **Comparison of UOP-Net (Ours) and previous methods.** Previous studies for object placement used (a) full-shape object models [7], [8], (b) completion modules [9], or (c) fitted primitive shapes [10], [11]. In contrast, (d) the proposed UOP-Net directly detects stable planes for unseen objects from partial observations.

In this paper, we propose a method for UOP that detects stable planes from complex shapes and novel objects. To achieve this, we generated a large-scale synthetic dataset called UOP-Sim, which contains various 3D objects and annotations of stable planes generated using a physics simulator. Unlike previous approaches [12], [13] that rely on heuristics to label the preferred placement configurations, we automatically annotate all feasible planes that can support stable object poses. Our dataset includes 17.4K objects and a total of 69K annotations. We propose a point cloud instance segmentation based network referred to as UOP-Net that predicts stable planes from partial point cloud and train it using only the UOP-Sim dataset. We compare the performance of our approach with three baselines and learning-based methods. We demonstrate that it achieves state-of-the-art (SOTA) performance in both the simulation and real-world experiments without any fine-tuning on real-world data.

The main contributions of this study are as follows:

* Equally contributed.

All authors are with the School of Integrated Technology, Gwangju Institute of Science and Technology, Cheomdan-gwagiro 123, Buk-gu, Gwangju 61005, Republic of Korea. † Corresponding author: Kyoobin Lee kyoobinlee@gist.ac.kr

- We propose a task called UOP to place an unseen object stably from single views and partial point clouds.
- We provide a public, large-scale 3D synthetic dataset called UOP-Sim that contains a total of 69,027 annotations of stable planes for 17,408 different objects.
- We introduce a point cloud instance segmentation network named UOP-Net that predicts stable planes for partially observed unseen objects.
- We compare the performance of our approach with previous object placement methods and confirm that our method outperforms the SOTA methods without any fine-tuning in real-world environments.

II. Related Works

Stable object placement. Previous studies [7], [14]–[16] demonstrated that robots can stably place an object with known geometrical properties by analyzing the convex hull and sampling stable planes for the object. However, this approach requires precise object priors (e.g., CAD, mass), and it may not be available in real-world scenarios with partial observations (e.g., from an RGB-D camera). Several researchers attempted to address this limitation with deep learning-based completion methods that predict the invisible part of an object [9]; however, these approaches have limitations in generating the precise shapes of unseen objects. Our UOP method addresses these limitations by directly detecting stable planes from partial observations without the need for complete 3D object models. Unlike previous methods, the UOP method is more generalizable and adaptable to real-world scenarios with partial observations.

Unseen object placement. Previous studies on unseen object placement focused on the identification of stable placements that satisfy human preferences. For example, Jiang et al. [12] trained a classifier using a hand-crafted dataset to identify these placements; this approach relies on heuristic labels and requires complete observability. Cheng et al. [17] proposed a deep learning model based on simulations to address the issue of heuristic labels; however, this approach was limited to task-specific objects. Another common approach [18] for placing unseen objects is using bounding box fitting to determine the shape and orientation of the object. This method can be fast and effective; however, it ignores the geometry of the object and relies only on its bounding box. Although this approach can be applied to unseen objects, it may not stably place objects in all situations, and therefore, it may be less effective than methods that consider the geometry of the object. In contrast, our approach can stably place unseen objects on a horizontal surface using only a single partial observation. Our method can handle a broad range of objects instead of being limited to specific object types.

Robotic applications of object placements. Prior works on object placement for robotic applications focused on solving specific tasks such as constrained placement [18], upright placement [19], [20], and rearrangement [21], [22]. However, these methods have several limitations. For example, Mitash et al.'s approach [18] relies on multi-view shape fitting and requires access to object models that may not be available in some scenarios. The deep learning approach proposed in [22] is limited to determining the required rotation for stable placements of objects in an upright orientation. Li et al. [23] proposed a method that can only predict rotations that maintain objects in positions that maximize their height; these limitations restrict the applicability and potential of these methods for more general object placement tasks such as stacking and packing. In contrast, our approach addresses the fundamental problem of placing unseen objects on a horizontal surface and has the potential to be applied to a wider range of robotic applications.

III. Problem Statements

A. Assumptions

The suggested approach is used when a robot must retrieve unseen objects that are mixed up in a container, or when a person hands a novel object to a robot and the robot does not know its correct orientation. The camera pose is assumed to be known in the workspace. The robot begins by grasping an object and capturing the scene using a single-view RGB-D camera. The resulting partial point cloud of the object is fed into the model to predict the most stable plane.

B. Definitions

Point cloud: Let $X \in \mathbb{R}^{N\times 3}$ be point clouds obtained by capturing the manipulating scene in which the robot grasps the object from the camera.

Object instability and stable planes: Let $\mathcal{U}$ denote the instability of an object model. We define instability of an object model as the average of movements in a simulator over discrete time step L. Stable planes $\mathcal{S}$ are annotated for each object model that satisfies the condition $\mathcal{U} < \epsilon$. A stable plane $s \in \mathcal{S}$ is represented as normal vector $\vec{V} \in \mathbb{R}^3$, and threshold ϵ indicates that the object has stopped.

Dataset and deep learning model: The dataset $\mathcal{D} = \{(\mathcal{O}, \mathcal{S})_n\}_1^N$ represents the N set of object models $\mathcal{O}$ and corresponding stable planes $\mathcal{S}$ as the annotations. The function $\mathcal{F} : X \rightarrow s$ denote a deep learning-based model that considers point clouds X as the input and produces the most stable plane s as the output.

Seen and unseen objects: The set of object models used for training and testing the function $\mathcal{F}$ are denoted as $\mathcal{O}_{train}$, $\mathcal{O}_{test}$. If $\mathcal{O}_{train} \cap \mathcal{O}_{test} = \emptyset$, then objects $\mathcal{O}_{train}$ are considered seen objects while objects $\mathcal{O}_{test}$ are unseen objects for the model $\mathcal{F}$.

C. Objectives

Our objective is to detect the most stable plane for placing unseen objects from a single-view observation. We aim to develop a function $\mathcal{F} : X \rightarrow s$ that minimizes the instability of the object $\mathcal{U}$.

IV. Learning for Unseen Object Placement

We address the challenges of the UOP task, which is difficult to solve because of the need for a large-scale dataset for approximating stable planes and the complexity of

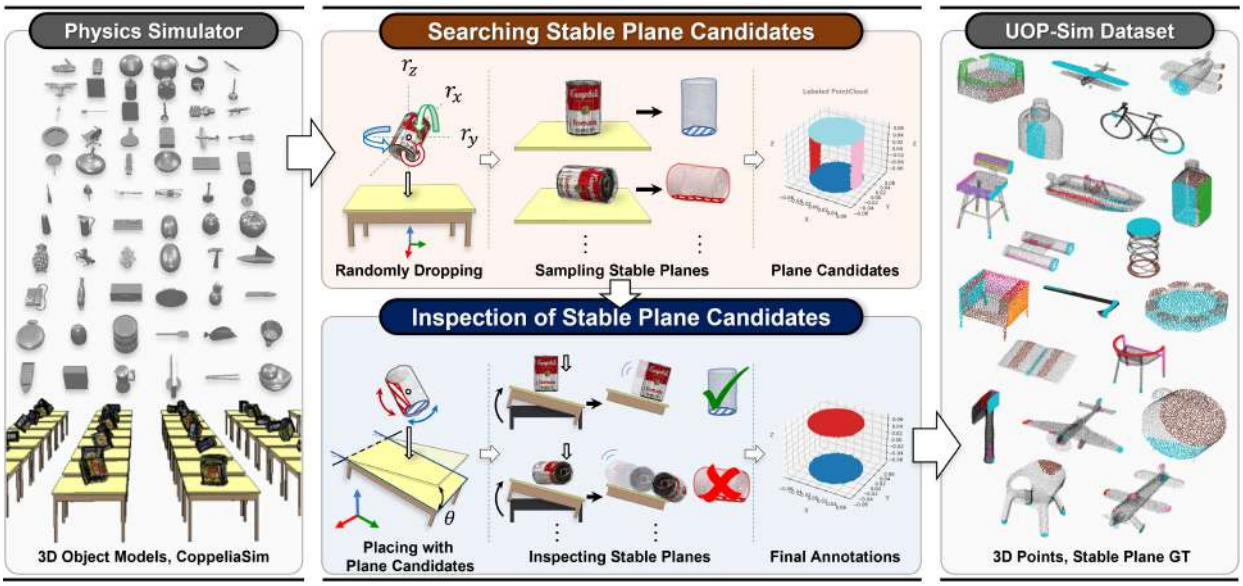

Fig. 2: **UOP-Sim dataset generation pipeline.** The UOP-Sim dataset is a large-scale synthetic dataset that contains 3D object models (17.4K) and annotations of stable planes (69K). The dataset is generated by dropping each object on a table in 512 different configurations and by sampling stable planes that satisfy Eq.2. The stable plane candidates are verified using a tilted table.

the relationship between point clouds and annotated planes. We present a novel approach by introducing the UOP-Sim dataset to mitigate these challenges; this dataset includes 17K 3D object models and 69K labeled stable planes, and a UOP-Net neural network that can detect robust stable planes from partial point clouds. We propose a general and adaptable approach to the UOP task using these tools, which enables robots to accurately place unseen objects in real-world scenarios.

A. UOP-Sim Dataset Generation

Stable plane annotation. We defined the movement of the object at time step i as $\mathcal{M}_i$ in terms of its translation and rotation change in the world coordinates to evaluate the instability of the pose of the object in dynamic simulation. The pose can be represented as $\mathbf{H} = [\mathbf{R}|\mathbf{T}] \in \mathbb{SE}(3)$, where $\mathbf{R}, \mathbf{T}$ are rotation and translation matrix. We tracked the pose of the object at each time step and calculated the difference between the consecutive poses (Eq.1). Then, we took the average of these differences over a certain period L to estimate the instability of the object at a time step i (Eq.2). To ensure robust annotation, we consider a range of discrete time step L rather than only single time step i.

$$\mathcal{M}_i = ||\mathbf{H}_i - \mathbf{H}_{i-1}||_2 \tag{1}$$

$$\mathcal{U}_i = \begin{cases} \frac{1}{L}\sum_{j=i-L+1}^{i} \mathcal{M}_j, & \text{if } i >= L \\ \frac{1}{i}\sum_{j=1}^{i} \mathcal{M}_j, & \text{otherwise} \end{cases} \tag{2}$$

We generated 512 orientations by dividing the roll, pitch, and yaw into eight intervals to explore a wide range of possible poses for the object. The object was then placed on a table with a random pose along the normal direction of the table. We dropped the object on the table and recorded all poses in which it remained stable ($\mathcal{U}_i < \epsilon_1$) to identify stable planes that support the object.

We then used the density-based spatial clustering of applications with the noise algorithm [24] to cluster the sampled poses. This allowed us to identify stable planes by clustering the poses along the z-axis; this represents the normal vector at the contact points of a stable plane with a horizontal surface. Subsequently, we masked the bottom 5% of height along the surface normal of table to indicate areas that could support the stability of the object.

Specific planes may not be easily generalized because real-world environments cannot be perfectly simulated. This can be a problem for spherical model planes or the sides of a cylinder. We placed each object on a flat table with a normal vector of the plane candidates and tilted the table by 10° to address the issue. We then estimated the movement of the object across each time step and eliminated any planes that did not satisfy the condition $\mathcal{U}$ less than ϵ_2. This allowed us to label stable planes that were robust for application to a horizontal surface, as shown in the samples of the UOP-Sim dataset in Fig.2. The UOP-Sim contains a total of 17,408 3D object models and 69,027 stable plane annotations. Furthermore, our dataset contains both explicit and implicit planes, such as a flat surface formed by four chair legs. Supplementary Figure.1 contains additional sample images from UOP-Sim (YCB objects).

Simulation environment setting We used PyRep [25] and CoppeliaSim [26] to build a simulation environment for computing object instability ($\mathcal{U}$). For the physics engine, we employed the bullet engine. Further, we used 3D object mod-

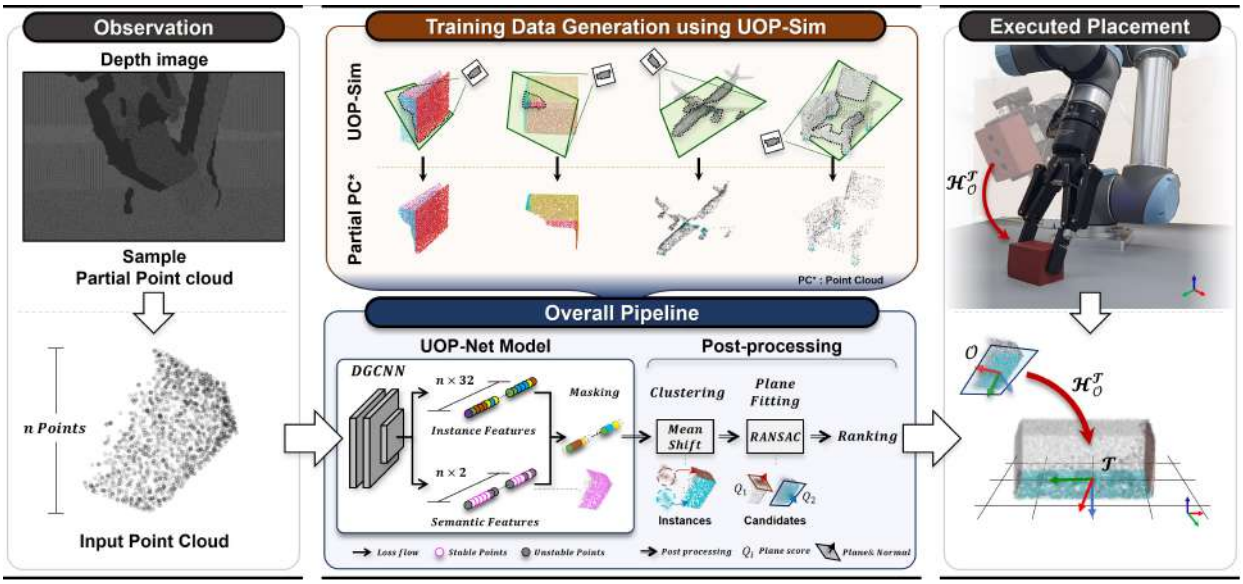

Fig. 3: **Overall pipeline of UOP Method**. UOP detects the most stable plane directly from single-view and partial point cloud. UOP-Net is trained on the UOP-Sim dataset, and takes in a partial point cloud to predict the stable plane. The estimated stable plane is used to execute object placement based on the angle difference between the normal vector of the plane and the negative gravity vector.

els from three benchmark datasets (3DNet [27], ShapeNet [28], and YCB [29]), which yielded a total of 17,408 models. We built 64 table models in the simulation environment to facilitate the annotation process.

B. UOP-Net

Network Architecture. The UOP-Net is based on DGCNN [30] architecture and JSIS3D [31] model. The network architecture includes three EdgeConv layers which are used to extract geometric features. These three EdgeConv layers use three shared fully-connected layers with sizes 64, 128, and 256. A shared fully-connected layer with size 1024 was then used to aggregate information from the previous layers. The global feature of point cloud was obtained using the Max-pooling operation, and two branches are used to transform the global features: one branch for semantic segmentation (which predicts whether a point is stable or unstable), and another branch for embedding instance features of stable planes. Both branches use fully-connected layers with sizes of 512 and 256. Before the two branches, LeakyReLU and batch normalization are applied to all layers.

A mean-shift clustering algorithm [32] is applied to the predicted stable points for identifying the stable points, and RANSAC [10] is used to fit planes onto the clustered points [11]. Stability scores for each plane are calculated by the element-wise multiplication of semantic logits, predicted instance labels, and number of points composing each plane. Then, the plane with the highest score is output after fitting the planes and assigning stability scores based on the number of inliers that constitute the planes. The rotation matrix $\mathbf{R}$ is then determined by estimating the angular difference between the predicted normal vector of the stable plane and the gravity vector (negative table surface normal).

C. Loss Function

Our loss function comprise two terms; stability loss as $\mathcal{L}_{stability}$ and plane loss $\mathcal{L}_{plane}$,

$$\mathcal{L} = \lambda_1 * \mathcal{L}_{stability} + \lambda_2 * \mathcal{L}_{plane}, \tag{3}$$

where λ_1 and λ_2 are hyper-parameters, setting as $\lambda_1 = 10$ and $\lambda_2 = 1$ respectively. The stability loss $\mathcal{L}_{stability}$ is defined by the binary cross-entropy loss to encourage predicted point label to match with the ground truth label. We adopted the discriminative function for 2D images [33] and 3D point cloud [31] to embed instance features for the plane loss $\mathcal{L}_{plane}$.

V. Experiments

A. Comparison with Traditional Method in Simulation

Datasets. We obtained a total of 152, 57, and 63 object categories in the 3DNet [27], ShapeNet [28], and YCB [29] datasets, respectively. We labeled the YCB object models in the simulation, but they were excluded from the training set to allow us to use the test set in both the simulation and real-world experiments. We excluded objects that had no stable planes (e.g., spherical objects) to ensure the quality of our dataset. Then, we splited the dataset into training and validation sets in a 8:2 ratio. The training set contained 13,926 objects and 55,261 annotations, while the validation set contained 3,482 objects and 13,766 annotations.

Training Details. We trained UOP-Net using partial point clouds sampled from the UOP-Sim dataset. During training, 2,048 points randomly sampled for each object and they underwent various types of data augmentation techniques such as rotation, sheering, point-wise jitter, and adding Gaussian noise to improve the performance of the model

TABLE I: UOP Performance of UOP-Net and other baselines on the three benchmark objects (partial shape) in the simulation.

Object type & Dataset		Object stability (OS)								Success rate of object placement (SR, %)			
		Rotation (R, °) ↓				Translation (T, cm) ↓							
		CHSA [7]	BBF [18]	RPF [10]	UOP (**Ours**)	CHSA [7]	BBF [18]	RPF [10]	UOP (**Ours**)	CHSA [7]	BBF [18]	RPF [10]	UOP (**Ours**)
Partial Point cloud	3DNet [27]	28.81	33.10	25.40	**8.68**	3.02	3.45	2.49	**0.77**	52.60	36.12	64.34	**65.24**
	ShapeNet [28]	31.65	38.14	21.83	**7.91**	3.97	4.53	2.46	**0.86**	51.05	29.95	69.70	**72.82**
	YCB [29]	34.85	41.69	22.41	**5.92**	5.07	5.89	2.55	**0.56**	42.48	30.41	62.13	**73.32**
	Total avg.	31.18	36.72	23.69	**7.73**	3.82	4.39	2.50	**0.74**	49.43	33.01	65.07	**69.35**

in real-world scenarios. The model was implemented using PyTorch [34] and trained on an NVIDIA Titan RTX GPU with a batch size of 32 and a total of 1,000 epochs. We employed early stopping with a patience of 50 and used the Adam optimizer at a learning rate of 1e-3 to prevent overfitting.

Baselines. We compared the performance of our method with those of the following baselines:

- **Convex hull stability analysis (CHSA)** [7], [8]: The baseline method for determining stable object poses involves the calculation of the rotation matrix to allow an object to rest stably on a flat surface. The center of mass of the object is sampled, and the stable resting poses of the object on a flat surface are determined using the convex hull of the object. Then, the probabilities of the object landing in each pose were evaluated, and the pose with the highest probability was output.
- **Bounding box fitting (BBF)** [11], [18]: The method involves fitting an oriented bounding box to the convex hull of the object using principal component analysis (PCA) to minimize the difference between the volume of the convex hull and that of the bounding box. The object was placed on a planar workspace with the largest area facing down.
- **RANSAC plane fitting (RPF)** [10], [11]: The approach segments planes in a point cloud by fitting a model of the form $ax + by + cz + d = 0$ to each point (x, y, z). Then, it samples several points randomly and uses them to construct a random plane while repeating this process iteratively to determine the plane that appears most frequently.

TABLE II: UOP Performance of UOP-Net and other baselines in the simulation (whole shape).The best and second-best results are indicated in **bold** and underline, respectively.

Object type & Dataset		Success rate of object placement (SR, %)			
		CHSA [7]	BBF [18]	RPF [10]	UOP-Net (Ours)
Whole Point cloud	3DNet [27]	**83.28**	55.48	70.89	80.16
	ShapeNet [28]	**92.37**	49.09	79.00	86.14
	YCB [29]	**86.08**	45.86	78.11	84.32
	Total avg.	**86.31**	51.24	74.90	82.79

Evaluation metrics. We used two metrics to evaluate the performance of UOP-Net: object stability *(OS)* and success rate of object placement *(SR)*. We placed an object on a flat table and used the output of the model to estimate its stability for measuring *OS*. We considered only rotational motion when we evaluated object stability because rotational motion is more common than translational motion when an object placed in an unstable state falls due to the vibrations. We evaluated the performance of the model by placing the object 100 times; we considered as a failure case if no planes were detected.

- **Object Stability (OS)**: The metric quantifies the movement of the object during a discrete time step when it is placed on a horizontal surface using the predicted plane.
- **Success Rate (SR)**: The metric indicates the percentage of placements where the object remains stationary for a minute with accumulated rotation under $10°$.

TABLE III: UOP performance for number of objects (500, 2,500, 5,000, and overall UOP-Sim dataset).

Number of objects	Object stability (OS)		Success rate of object placement (SR*, %)
	R (°) ↓	T (cm) ↓	
500	14.71	1.63	76.53
2,500	11.98	1.34	79.36
5,000	11.28	1.26	80.05
13,926	3.98	0.39	90.08

TABLE IV: UOP-Net performance for two different backbones (PointNet [35] and Dgcnn [30])

Backbone	Object stability (OS)		SR (%)	Modal params (MB) ↓	Flops (GFLOPs) ↓	Inference time (ms) ↓
	R (°) ↓	T (cm) ↓				
PointNet [35]	**3.76**	**0.37**	70.10	28.05	10.219	732.09
DGCNN [30]	3.98	0.39	**73.32**	**8.96**	**6.834**	**34.29**

Discussion. Our simulation experiments compared UOP-Net's performance with baseline methods on partial shapes from the 3DNet, ShapeNet, and YCB datasets. The results, detailed in Table I, demonstrate UOP-Net's superior performance in scenarios involving single partial observations. This is further illustrated in Fig. 4, where the blue line indicates the enhanced reliability of object placement using UOP-Net compared to other methods.

For fully visible objects, the CHSA method showed optimal results for stable placements, as shown in Table II. However, UOP-Net outperforms CHSA when dealing with only partial point clouds. This effectiveness is due to the limitations of CHSA in handling incomplete data, where it struggles with the determination of an object's center of mass and often misidentifies truncated planes as stable placement options (Fig.4).

The BBF method underperformed as it neglects the geometric properties of objects and relies on placing objects on

the largest plane within the bounding box. This approach is unsuitable for objects with complex geometries. While the RPF method outperforms BBF, it still falls short in reliably detecting stable planes due to its limited approach in selecting frequently sampled planes, which often leads to unsuccessful object placements. UOP-Net, by contrast, demonstrates its ability to discern the most stable planes from partial observations. This capability is attributed to its training approach, which emphasizes predicting planes based on visible parts.

Furthermore, UOP-Net's versatility extends to both explicit and implicit planes detection. It can identify planes that are directly visible in the point cloud and those that are implied, such as a plane formed by the four legs of a chair, contributing to the overall stability of the object. This versatility stems from the diverse training set of objects and planes UOP-Net was exposed to, enabling it to generalize and accurately predict unseen objects and planes, as depicted in Figure 5.

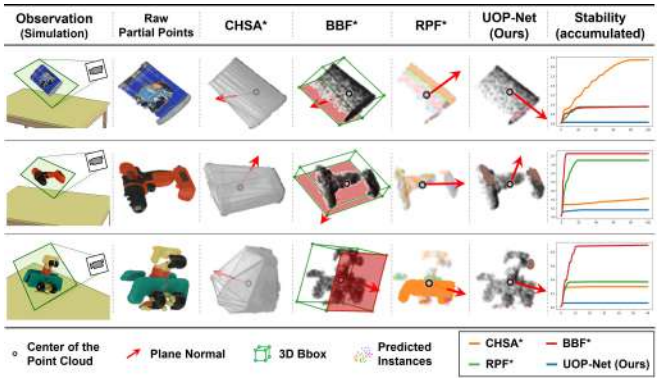

Fig. 4: Visualization of simulation results on YCB [29] objects (Red arrows are predicted normal vectors). The right graph depicts object stability; lower values are better.

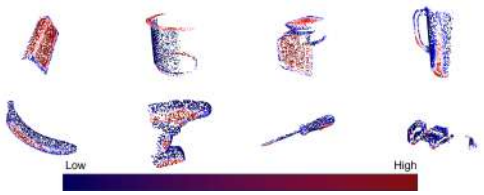

Fig. 5: Visualization of the shape features learned by the last layer of UOP-Net backbone on YCB objects.

B. Comparison with Learning-based Method in Simulation

We compared the performance of UOP-Net, the Upright-Net [20] and the CHSA method alongside the point cloud completion method [36]. To ensure a fair comparison with Upright-Net, we conducted the experiments on a subset of UOP-Sim, where we used the same categorization scheme of Upright-Net. The dataset was divided into three splits: seen objects for training, seen objects for evaluation, and unseen objects for evaluation. Each category included 40 objects for training and 10 for evaluation. The categories are the following:

- ***Seen***: Bed, Bench, Bottle, Bowl, Bus, Cabinet, Camera, Cap, Car, Chair, Jar, Laptop, Mug, Printer, and Table
- ***Unseen***: Basket, Helicopter, Lamp, Pot, Skateboard, Sofa and Tower

For the completion method, we used the Pointr [36] that was trained on our subset of UOP-Sim; we confirmed that this model performs better than their pretrained model due to the difference in input preprocessing. Table V shows each method's object placement success rate in simulations. The rate is the ratio of successful placements to the total number of inference trials. Each object underwent 60 trials.

TABLE V: The placement success rates of UOP-Net, Upright-Net, CHSA with and without completion (C: completion [36])

	Data type	Method			
		CHSA w/o C	CHSA w/ C	Upright-Net [20]	UOP-Net (Ours)
Whole	Seen	**92.27**	-	77.97	87.59
	Unseen	86.76	-	84.26	**87.77**
Partial	Seen	57.04	67.06	17.04	**76.75**
	Unseen	51.68	63.92	25.21	**73.78**

Comparison with Upright-Net. Our proposed UOP-Net consistently outperformed Upright-Net [20] under all conditions. Upright-Net demonstrates comparable performance with the whole point cloud input; however, its performance was significantly lower than ours when the partial point cloud was input. This is due to Upright-Net's design, specifically constructed to predict only the upright orientation from the whole point cloud input. Consequently, it tends to fail when upright planes are invisible within the input partial point cloud. On the contrary, our UOP-Net was designed to detect all stable planes from the partial point cloud, thus leading to better performance.

Comparison with CHSA + Completion. Our UOP-Net outperforms CHSA in all cases except for seen objects provided with a complete point cloud. While the completion method does enhance CHSA's performance, it still tends to underperform compared to our method. This is primarily because the completion method often struggles to generate an accurate and detailed point cloud (Fig6).

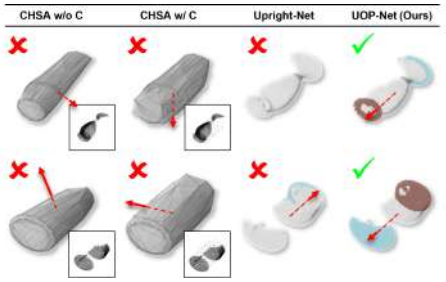

Fig. 6: Comparison with learning-based method (only CHSA, CHSA + completion, Upright-Net and UOP-Net).

C. Additional Experiments

We conducted experiments to evaluate the impact of

the UOP-Sim dataset size on UOP-Net's performance. We compared its performance when trained on datasets of 500, 2,500, 5,000 objects, and the complete UOP-Sim dataset. We utilized Object Stability (OS) and Success Rate* (SR*) as evaluation metrics, where SR* is defined as the number of successful stable object placements divided by the number of successful inferences. As shown in the table III, we observe a proportional increase in both S and SR* as the number of training objects increases. In other words, the stability detection performance of UOP-Net improves with an increasing amount of training data.

Additionally, we conducted evaluation of UOP-Net with different backbones, PointNet [35] and DGCNN [30]. In Table IV, though PointNet-based UOP-Net is a slight improvement in Object Stability, Success Rate is better when employing the DGCNN. Considering model efficiency, it shows that DGCNN-based UOP-Net significantly outperforms PointNet.

D. Robot Experiments

Real environment setting. We conducted experiments using a universal robot (UR5) manipulator and an Azure Kinect RGB-D camera to evaluate the performance of our object placement method in a real-world scenario. We used the MANet [37] object segmentation method with a DenseNet121 [38] backbone to segment the target object and the gripper. We segmented the visible region of the target object from the RGB image and cropped the depth image using a mask. Then, we sampled the point cloud from the depth image using voxel-down sampling [11] and fed it to UOP-Net. The model predicted the most stable plane and calculated the rotation value between the plane and the table. Then, the UR5 robot placed the target object on the table. We utilized the BiRRT algorithm [39] implemented with PyBullet [40] and integrated it with collision checking in a physics engine to ensure smooth planar motion.

Evaluation metrics. To ensure a fair comparison, we strived to standardize the object grasp configurations across various methodologies. Subsequently, we adopted the SR metric to quantitatively assess the efficacy of our proposed approach in real-world scenarios. Throughout the experiments, we instructed the robotic system to execute object placements onto the surfaces predicted by the UOP-Net model. The successful placement was predicated based on visual confirmation of the object maintaining a stable, non-sliding position on the predicted plane. Conversely, if the model failed to identify any viable stable planes, the trial was classified as unsuccessful. For each distinct object, we conducted a series of 10 placement trials to ensure a comprehensive evaluation outcome.

Results. We selected 12 objects from the YCB dataset. Objects with spherical shapes (e.g., apples), dimensions that were extremely small, or small depth values were excluded from the test set. Table VI indicates that our method outperforms other baselines in terms of the success rate across all objects. Although real-world perceptions are noisy, UOP-Net provides a stable plane that can be attributed to our model learning from partial point clouds captured by a depth camera and corrupted by noise. Other benchmarks (CHSA and primitive shape fitting) performed extremely poorly because they could not obtain the complete shape of the object in the real world and were unable to respond to sensor noise. We evaluated our method on completely new objects that did not have an available CAD model (a dinosaur figurine and an ice tray, as shown in Fig.7) to verify further that our model can perform on unseen objects. UOP-Net detected implicit planes (e.g., the four legs of the dinosaur) even though the object shapes were complex.

TABLE VI: UOP performance of the UOP-Net and baselines on YCB [29] in the real world. Each object attempted 10 times per trial.

YCB	CHSA [7]	BBF [18]	RPF [10]	UOP-Net (Ours)
Coffee can	0	0	4	**10**
Timer	0	1	**6**	**6**
Power drill	0	1	**5**	**5**
Wood block	1	1	**10**	**10**
Metal mug	0	0	6	**9**
Metal bowl	**10**	5	**10**	**10**
Bleach cleanser	3	3	**9**	**9**
Mustard container	2	0	5	**10**
Ariplane toy	0	3	0	**4**
Sugar box	2	3	**10**	**10**
Chips can	2	0	8	**10**
Banana	5	5	**9**	**9**
Average	2.1	1.8	6.8	**8.5**

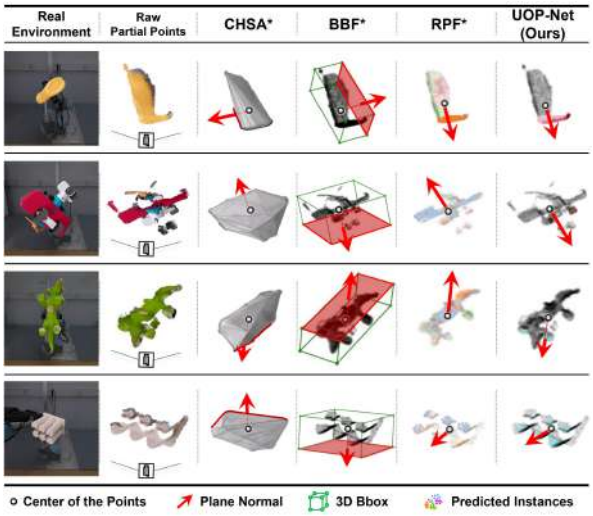

Fig. 7: Visualization of the real world results. (rows 1, 2: YCB objects, rows 3, 4 : novel objects)

VI. Conclusion

In this study, we presented UOP-Net, a novel method developed to detect stable planes of unseen objects. We also introduced an approach to annotate automatically stable planes for various objects, and the large-scale synthetic dataset, called UOP-Sim, was generated. Our dataset contains 17.4K 3D objects and 69K stable plane annotations. The effectiveness of UOP-Net was demonstrated by achieving SOTA performance on objects from three benchmark datasets, thus indicating its accuracy and reliability in detecting stable

planes from unseen and partially observable objects.

ACKNOWLEDGMENT

This research was completely supported by a Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (Project Name: Shared autonomy based on deep reinforcement learning for responding intelligently to unfixed environments such as robotic assembly tasks, Project Number: 20008613). This research was partially supported by an HPC Support project of the Korea Ministry of Science and ICT and NIPA.

REFERENCES

[1] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "Unseen object instance segmentation for robotic environments," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1343–1359, 2021.

[2] S. Back, J. Lee, T. Kim, S. Noh, R. Kang, S. Bak, and K. Lee, "Unseen object amodal instance segmentation via hierarchical occlusion modeling," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5085–5092.

[3] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[4] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2901–2910.

[5] H. Tian, K. Song, S. Li, S. Ma, J. Xu, and Y. Yan, "Data-driven robotic visual grasping detection for unknown objects: A problem-oriented review," *Expert Systems with Applications*, vol. 211, p. 118624, 2023.

[6] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic *et al.*, "Deep learning approaches to grasp synthesis: A review," *IEEE Transactions on Robotics*, 2023.

[7] J. A. Haustein, K. Hang, J. Stork, and D. Kragic, "Object placement planning and optimization for robot manipulators," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7417–7424.

[8] Dawson-Haggerty et al., "trimesh." [Online]. Available: https://trimsh.org/

[9] M. Gualtieri and R. Platt, "Robotic pick-and-place with uncertain object instance segmentation and shape completion," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1753–1760, 2021.

[10] 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.

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

[12] Y. Jiang, C. Zheng, M. Lim, and A. Saxena, "Learning to place new objects," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 3088–3095.

[13] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1021–1043, 2012.

[14] P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," in *International Conference on Robotics and Automation (ICRA)*, vol. 4. IEEE, 1987, pp. 1924–1928.

[15] W. Wan, H. Igawa, K. Harada, H. Onda, K. Nagata, and N. Yamanobe, "A regrasp planning component for object reorientation," *Autonomous Robots*, vol. 43, no. 5, pp. 1101–1115, 2019.

[16] P. Lertkultanon and Q.-C. Pham, "A certified-complete bimanual manipulation planner," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1355–1368, 2018.

[17] S. Cheng, K. Mo, and L. Shao, "Learning to regrasp by learning to place," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 277–286.

[18] C. Mitash, R. Shome, B. Wen, A. Boularias, and K. Bekris, "Task-driven perception and manipulation for constrained placement of unknown objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5605–5612, 2020.

[19] R. Newbury, K. He, A. Cosgun, and T. Drummond, "Learning to place objects onto flat surfaces in upright orientations," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4377–4384, 2021.

[20] X. Pang, F. Li, N. Ding, and X. Zhong, "Upright-net: Learning upright orientation for 3d point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 14 911–14 919.

[21] K. Wada, S. James, and A. J. Davison, "Reorientbot: Learning object reorientation for specific-posed placement," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8252–8258.

[22] C. Paxton, C. Xie, T. Hermans, and D. Fox, "Predicting stable configurations for semantic placement of novel objects," in *Conference on Robot Learning*. PMLR, 2022, pp. 806–815.

[23] R. Li, C. Esteves, A. Makadia, and P. Agrawal, "Stable object reorientation using contact plane registration," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6379–6385.

[24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[25] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.

[26] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1321–1326.

[27] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 5384–5391.

[28] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[29] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517.

[30] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[31] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung, "Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8827–8836.

[32] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[33] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[36] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, "Pointr: Diverse point cloud completion with geometry-aware transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 12 498–12 507.

[37] T. Fan, G. Wang, Y. Li, and H. Wang, "Ma-net: A multi-scale attention network for liver and tumor segmentation," *IEEE Access*, vol. 8, pp. 179 656–179 665, 2020.

[38] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[39] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, vol. 68, pp. 1–11, 2015.

[40] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.