Planar Manipulation via Learning Regrasping

Peng Xu1, Zhiyuan Chen1, Jiankun Wang∗, Senior Member, IEEE, and Max Q.-H. Meng∗, Fellow, IEEE

Abstract—Regrasping is important for robots to reorient objects in planar manipulation tasks. Different placements of objects can provide robots with alternative grasp configurations, which are used in complex planar manipulation tasks that require multiple pick-rotate-and-place steps due to the constraints of the environment and robot kinematics. In this work, our goal is to generate diverse placements of objects on the plane using deep neural networks. We propose a pipeline with the stages of orientation generation, position refinement, and placement discrimination to obtain accurate and diverse stable placements based on the perception of point clouds. A large-scale dataset is created for training, including simulated placements and contact information between objects and the plane. The simulation results show that our pipeline outperforms the start-of-the-art, achieving an accuracy rate of 90.4% and a diversity rate of 81.3% in simulation on generated placements. Our pipeline is also validated in real-robot experiments. With the generated placements, sequential pick-rotate-and-place steps are calculated for the robot to reorient objects to goal poses that are not reachable within one step. Videos and dataset are available at https://sites.google.com/view/pmvlr2022/.

Note to Practitioners—This paper aims to facilitate planar manipulation tasks with regrasping, which often requires generating intermediate placement locations for the object being manipulated. We propose a learning-based approach composed of stages of orientation generation, position refinement, and placement discrimination to obtain diverse, stable planar placements of a variety of objects, using point clouds. To enable the learning, the paper proposes a dataset of stable placements constructed by dropping objects from random poses in PyBullet, and the placements are subsequently clustered. The experiment results demonstrate that our approach achieves state-of-the-art compared with the ablation and baseline methods. The generated placements are also demonstrated in manipulation tasks involving multiple pick-rotate-and-place operations to flip novel objects in the simulation and real environment. Generating diverse placements of objects is important for manipulation tasks with regrasping, and our approach could be used as a module in more complex frameworks. Feasible and collision-free grasp configurations are important for robots to realize the transformation between placements. In the future, we will attempt to generate smoother and safer trajectories for the robot than the current grasp configurations from the sampling-based algorithm.

Index Terms—Regrasping, Deep learning, Planar manipulation.

Fig. 1. The example of flipping a T-nut. The robot can not flip the object in its initial state (blue border) with one pick-rotate-and-place step (red border) due to kinematic constrains and environmental occlusion. However, based on the stable placement generated by our pipeline, the robot can flip the object with sequential steps (green border).

I. INTRODUCTION

PLANAR manipulation means manipulating objects on a plane [1], [2]. This process involves object reorientation, which is vital for robots to rotate objects to desired poses. Typical tasks, such as packing objects, assembly, and using tools [3], require robots to arrange objects in specific poses. When the robot cannot reorient an object to the goal pose with a single pick-rotate-and-place step, regrasping is necessary for reorienting objects. Mimicking the regrasping manipulation of humans, robots rotate the object to intermediate placements to access the grasp points impeded by the initial poses. As illustrated in Fig. 1, we take flips of an industrial part as an example. Due to the constraints of the environment and robot kinematics, the robot is unable to flip the object with one pick-rotate-and-place step. Therefore, regrasping is effective for reorientating objects by noticeable angles.

The key components of regrasping objects are the placements of objects and the motions of robots. First, diverse placements of objects make different grasp points reachable for robots. Second, an appropriate pick-rotate-and-place motion of the robot produces a successful state transition of an object from one placement to another.

1 Peng Xu and Zhiyuan Chen contributed equally to this work.
∗Corresponding authors: Jiankun Wang; Max Q.-H. Meng.
Peng Xu is with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong peterxu@link.cuhk.edu.hk
Zhiyuan Chen and Jiankun Wang are with Shenzhen Key Laboratory of Robotics Perception and Intelligence, and the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China 12132111@mail.sustech.edu.cn, wangjk@sustech.edu.cn
Max Q.-H. Meng is with Shenzhen Key Laboratory of Robotics Perception and Intelligence, and the Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen 518055, China, on leave from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute of The Chinese University of Hong Kong, Shenzhen 518057, China max.meng@ieee.org
Plenty of approaches \cite{3,4,5} calculate placements of objects using their given meshes, which do not apply to the placement calculation of novel objects. In order to obtain placements of long and thin objects, \cite{6,7} develop learning-based approaches to predict placements afforded by the supporting items, such as containers and holders. However, using extra supporting objects for placements is unsuitable for many objects. On the one hand, matching appropriate supporting items for certain types of objects is complex. On the other hand, various placements on the plane can be generated depending on the geometry of objects. For instance, manipulating the T-nut shown in Fig. 1 on the plane is more proper than placing that in a container.

Nevertheless, enumerating the placements of objects remains challenging in planar manipulation. In this work, we present a data-driven approach for obtaining stable placements on the plane with the visual perception of point clouds. Our pipeline for placement proposals is comprised of three stages. With point clouds and parameterized noise as input, a neural network is trained in the first stage to generate approximate placement poses, which aims to enumerate placements on the plane. Next, the initial point cloud is transformed into new point clouds using the generated poses, which are then refined with PointNet++ \cite{8} networks predicting the poses of the plane. Finally, a classifier model discriminates between stable and unstable placements from previous stages.

In order to address the issue of long-horizon manipulation, manipulation graphs are constructed, where the placements are nodes for providing alternative grasp configurations, and the shared grasp configurations are edges for connecting transferable placements. Previous works \cite{3,4,5} construct regrasping graphs to search for pick-and-place steps to reorient the known placements. A new algorithm to construct manipulation graphs using point clouds, which generalizes to novel objects. We also employ this approach to flip novel objects with placements generated by our pipeline. Again illustrated in Fig. 1, sequential steps from the manipulation graph should be performed if it is impossible to reorient a part within one step due to the kinematic constraints of the robot and the occlusion by the environment.

To evaluate our pipeline, we compare different loss functions implemented in the generation stage, conduct ablation experiments and contrast various discriminators. The experiments demonstrate that our approach outperforms others in terms of placement accuracy and types of generated placements. The main contributions of our work are as follows: (1) To our knowledge, this work is the first to enumerate objects’ placements on the plane with neural networks. (2) We construct a large-scale dataset containing stable placements of objects with diverse geometry and contact information between objects and the plane. (3) We show that our pipeline can generate accurate and diverse stable placements for sequential planar manipulation operations.
discriminating between stable and complex unstable placements, such as the unstable placements contacting with the supporting surface. To mitigate this problem, [6] proposes the joint training of placement classification and contact point regression to improve feature mapping. Xu et al. [7] use two PointNet++ networks supervised by disjoint unstable placements to classify placements in consideration of two classification boundaries. Recent work [17] demonstrates that structured local feature representation projected from 3D features improves joint learning of tasks. Inspired by this, we apply the feature projection approach to obtain local features for placement classification.

III. APPROACH

The goal of our work is to predict diverse placements of novel objects on the plane from point clouds. The point clouds are obtained by fusing multi-view images [18] captured with the camera mounted on a robot. The single robot arm with a two-finger gripper uses the placement proposals to perform planar manipulation.

The placements collected in Pybullet [19] are categorized in accordance with decisive dimensions of positions, which consist of the orientation positions. Different placements of objects vary in these dimensions of positions. Therefore, we propose a three-stage pipeline to predict different stable placements. The first stage is orientation generation, where Euler angles in the world coordinate frame are generated for diverse preliminary positions. Given one transformed point cloud above, the second stage predicts the plane position for the refinement of the orientation and height positions. The final discrimination stage is to determine stable placements after the refinement. The neural network model in each stage is trained separately.

As illustrated in Fig. 2, the input point cloud is transformed to new positions with the generated orientation from the first stage and the refined position relative to the plane from the second stage. The point indices of the placements are kept stationary. The finally selected placements after the third stage are used to construct the manipulation graph. The generated diverse placements can provide respectable grasp configuration space for the robot [9].

A. Orientation Generation

In the first stage, our model takes as input the point clouds and the samples from a multivariate Gaussian distribution. The purpose of including noise vectors in the input space is to prevent the generated results from converging to a single position. Inspired by [7], we adopt a parameterized Gaussian distribution with selected mean and variance parameters to take random samples. The random samples \( N_{in} \) are passed to a CNN encoder to obtain the feature \( F_N \).

PointNet++ networks are widely used to encode point clouds and map them to feature embeddings [6], [7], [11]. Hence, we are motivated to employ PointNet++ networks to extract features of input point clouds. In the orientation generation, the point cloud \( P_{in} \) is passed to a PointNet++ network with set abstraction layers to obtain the feature \( F_P \). Then, we concatenate the features \( F_N \) and \( F_P \) to form \( F_G \), which is passed to a CNN decoder to output Euler orientations in the world coordinate frame.

The advantage of using a network to generate orientations instead of sampling orientations is better fitting the density distribution of the stable placements through the learning process and obtaining the orientations closer to the stable placements.

For the loss function, we improve the Chamfer distance [20] by substituting the Euclidean distance with the Geodesic distance [21]. The Geodesic distance can precisely represent the angular distance between two orientations and is formulated as:

\[
d_{geo}(R_g, R_T) = \arccos \left( \frac{\text{tr}(R_g R_T^{-1}) - 1}{2} \right),
\]
where $R_g$ and $R_T$ are rotation matrices equivalent to Euler angles. However, $d_{geo}(R_g, R_T)$ is not completely differentiable in back-propagation during training. Therefore, we modify the expression of the original $d_{geo}(R_g, R_T)$ in the form of a polynomial. A tenth order polynomial $d_{geo}(R_g, R_T)$ is used to fit $d_{geo}(R_g, R_T)$, which is defined as:

$$
\tilde{d}_{geo}(tr) = (tr - 3) \sum_{i=0}^{9} a_i tr^i,
$$

(2)

where $tr$ is short for $tr(R_g R_T^{-1})$ in Eqn. [1] and $\tilde{d}_{geo} = d_{geo} = 0$ at $tr(R_g R_T^{-1}) = 3$. Ultimately, the loss of training is:

$$
L_{geo} = \sum_{R_g \in S_g} \min_{R_T \in S_T} \tilde{d}_{geo}(R_g, R_T)
+ \sum_{R_T \in S_T} \min_{R_g \in S_g} \tilde{d}_{geo}(R_g, R_T),
$$

(3)

where $S_g$ and $S_T$ denote the generated and ground-truth orientations, respectively. Based on a generated orientation, we can obtain the rough placement $P_g$ from $P_{in}$.

B. Position Refinement

Given the transformed point clouds, our trained model predicts the positions of the supporting surface to construct stable placements. As shown in Fig. [2], a surface is represented as a vector (the arrow $v_R$) with a specific length, which is perpendicular to the surface. Mathematically, a surface can be expressed as:

$$
a x + b y + c z = a^2 + b^2 + c^2,
$$

(4)

where $v_R = (a, b, c)$ is a vector in the world coordinate frame. We utilize a PoinNet++ network with set abstraction layers and feature propagation layers to process the transformed point cloud $P_g$ with $M$ points for obtaining a feature representation of each point $p_i \in P_g$. This network is followed by fully connected layers with ReLU activation and batch normalization. Accordingly, the size of each feature $v_i^d$ goes down to 3. We let each feature vector represent the difference from a point vector $p_i$ to the surface vector $v_R$.

Given different transformed point clouds of an object, our model infers different positions of the supporting surface. The ground-truth plane position of the point cloud for training is denoted as $v_{GT}$. The previous work [6] uses points in a specific region on the object to regress a target point. In contrast, our model uses all points of the transformed point cloud to infer coordinate values of a vector. The more points considered in the objective, the more accurate the positions of the supporting surface will be. We thus define the training objective as:

$$
L_{\text{refine}} = L_{\text{field}} + L_{\text{var}}
= \alpha \frac{1}{M} \sum_{i=1}^{M} \text{smoothl1loss}(v_{GT} - p_i, v_i^d)
+ \beta \frac{1}{M} \sum_{i=1}^{M} (p_i + v_i^d - \frac{\sum_{i=1}^{M} p_i + v_i^d}{M})^2,
$$

(5)

where $L_{\text{field}}$, $L_{\text{var}}$ are mentioned in [6] as the field loss and the variance loss, and $\alpha$, $\beta$ are weights.

Given the predicted vector $v_R = \sum_{i=1}^{M} p_i + v_i^d$, we obtain the point cloud $P_r$ with the refined plane position. Then, $P_r$ is transformed to $P_o$ to ensure the plane and the table surface are in the same position in the world coordinate frame. This plane is aligned with the plane of the table and the point cloud placement is further transformed, refining it. The calculation of transformation is:

$$
P_o = R \cdot (P_r - v_R),
$$

(6)

where $R$ is a rotation matrix transforming the object points from $P_r$ to $P_o$. $v_R$ is obtained by matching the surface vectors of the refined plane and the table surface through the shortest path [22].

C. Placement Discrimination

A challenge in placement prediction is discriminating between stable and unstable placements. We design a neural network model as a classifier to select stable placements obtained from previous stages. In previous work [6], [7], [11], PointNet++ networks with several set abstraction layers are used to find stable placements. However, a single PointNet++ network, which performs well in classifying point clouds of objects with different geometry [6], falls short of discriminating point clouds of the same object with different poses. For placement discrimination, local contact information is important. Therefore, we adopt the feature projection process in GIGA [17] to construct local features.

Our discriminator takes the transformed point cloud $P_r$ as input and extracts a global feature embedding $F_D$ using a PointNet++ network with one set abstraction layer. Given the global features, we construct orthogonal feature planes ($F_{xy}$, $F_{yz}$, and $F_{xz}$), which contain pixel-wise local features in the feature space. The feature vectors are projected vertically to the pixel cells of feature planes. We then use a U-Net [23] network to process each feature plane for local feature inpainting as introduced in [17]. These local features are concatenated as the input to a ResNet18 [24] network and then mapped to the 2d output of predicates. We adopt the cross-entropy loss [7] as the training objective. The binary classification is supervised by the ground-truth labels in Sec. [IV]. In order to guarantee the accuracy and diversity of placements, the predicted placements with scores $s_D$ above 0.92 are sorted into different types using our clustering approach in Sec. [IV].

Then, the classified placements are used to calculate the grasp configurations of the robot. Contact points on the generated placements are inferred with a data-driven method [25]. Shared grasp configurations between the generated placements are obtained after calculating the shared contact points on the placements and the robot’s feasible inverse kinematics. With the generated placements and calculated grasp configurations, manipulation graphs are constructed using the algorithm in [7].

IV. DATASET AND TRAINING

We build a large and accurate dataset using the objects from the Mechanical Components Benchmark (MCB) [26] comprising 3D objects with diverse geometric shapes. A total
Fig. 3. Models of part of the objects in our dataset. Objects in the first two rows are in the training set. Objects in the last row are in the test set and used for experiments.

Fig. 4. Collected stable placements of several objects in the training set. The stable placements are classified according to the orientations.

of 374 objects of 25 categories are contained in our dataset. The training set contains 275 objects, and the test set contains 99 objects with different shapes from those for training. As depicted in Fig. 3, we display part of the objects in our dataset to show the shape variations.

Objects in our dataset are loaded into PyBullet [19], and the accurate states of objects in the simulation process are recorded. In order to obtain diverse placements, every object is simulated 1000 times in PyBullet. Each time an object starts at a random pose above the table surface, we run the simulation process for at least 10s and stop it when the object’s velocity is under 0.1 mm/s to ensure every object placement is finally stable under gravity. The 6D positions of 374,000 placements in the world coordinate frame are then recorded. According to the orientation positions in continuous representation [21], the placements are sorted into different types using our clustering approach that includes MeanShift [27]. The number of placement clusters can be set automatically according to the distribution of the orientations. As shown in Fig. 4, placements of objects in the training set are classified. We assume that implementing rotations around a vertical axis and translations along the \(xy\) plane on a classified placement results in placements of the same type as this classified placement [3]. Thus, we use the classified orientation set to supervise the training of the first stage, where the numbers of different placement types for each object are identical. In the first stage, a multivariate Gaussian distribution with the mean of 1000 and the variance of 10 is adopted.

Apart from each stable placement, contact points on the object in a stable placement and the unstable pose of the object in the air are also recorded during each simulation. In particular, we select three non-collinear points from the contact points, which are used to describe the position of the supporting surface. Based on the transformation between the stable placement and the unstable pose, we can obtain the coordinates of the three points corresponding to the unstable pose and calculate a vector representing the supporting surface in the world coordinate frame. The unstable pose of the object and its associated position of the supporting surface are for training the model in the second stage. Finally, unstable placements that contact the table are also collected to train our discriminator. Neural network models in our pipeline are implemented in PyTorch and trained using an NVIDIA GeForce GTX 3090. We utilize Adam optimizer in training. All training loss curves are converged.

V. Experiments

We conduct experiments to verify that our approach can predict diverse stable placements of novel objects with different geometry. The performances of the orientation generation, position refinement, and placement discrimination stages are investigated with comparison and self-ablation experiments. Real-robot experiments are also conducted to validate our approach.

A. Simulation Experiments

We build our simulated environment with PyBullet [19]. Objects used in simulation experiments are from the test set unseen in training. The ground-truth labels of placements for objects in the test set are introduced in Sec. IV, including the positions of stable placements and unstable placements.
In contrast, predicted placements of objects in the test set are generated with different approaches for comparison. When comparing different approaches on an object, the point cloud passed to the approaches is the same, and the number of input random samples is set to 1024. The objects in predicted placements are then loaded into PyBullet for records of the final stabilized positions after the simulation.

The following metrics are used for approach evaluation:

- The accuracy of predicted placements: the proportion of verified stable placements among the predicted placements. We define the criterion to determine whether a predicted placement is stable or not. Suppose a predicted placement’s orientation and height position is \((O_{\text{start}}, h_{\text{start}})\) and changes to \((O_{\text{end}}, h_{\text{end}})\) after the simulation. We assume that the predicted placement is stable if \(\Delta D = d_{\text{geo}}(O_{\text{start}}, O_{\text{end}}) \times \times 180^\circ \leq 10^\circ\) and \(\Delta H = \|h_{\text{start}} - h_{\text{end}}\| \leq 2cm\).
- The diversity of predicted placements: the proportion of the predicted placement types to the ground-truth placement types. Using the approach introduced in [14], we also classify the predicted placements according to the orientation positions. Regarding whether a ground-truth type \(O_{gt}\) is covered by a predicted type \(O_{pred}\), we set a threshold \(\Delta d = d_{\text{geo}}(O_{pred}, O_{gt}) \times \times 180^\circ \leq 15^\circ\). Suppose a tested object has \(n\) types of stable placements. An approach predicts \(m\) types meeting the threshold condition. Then the diversity of predicted placements can be calculated as \(\frac{m}{n-1}\), where the initial placement type is excluded.

1) Generator Performance: To demonstrate the performance of our orientation generation stage, we compare our approach \((G_{ours} + R_{ours} + D_{ours})\) with the baseline. In \(G_{ours}\), \(R_{ours}\) denotes our generator of orientation generation stage, \(R_{ours}\) denotes our model of refinement, and \(D_{ours}\) denotes our discriminator with the following variations of our approach:

- \(G_{ours} + R_{ours} + D_{ours}\): the generator in our approach is replaced by \(G_{ours}\) in [8]. The loss function in \(G_{ours}\) computes the Chamfer Distance [20] between the transformed point clouds in predicted poses and the ground-truth point clouds. The \(G_{ours}\) is re-trained on our training set. The hyperparameters used in training the previous model [6] remain the same.
- \(G_{ours} + R_{ours} + D_{ours}\): the generator in our approach is replaced by the generator \(G_{ours}\) in [7]. The loss function in \(G_{ours}\) computes the Chamfer Distance between the predicted placement poses and the ground-truth poses. We re-train \(G_{ours}\) with the training data used for training our generator \(G_{ours}\).
- \(G_{ours} + R_{ours} + D_{ours}\): the orientation positions in the first stage are randomly generated. Specifically, we randomly generate the Euler angles with values between

| Approaches          | Square Nut | Standard Fitting | Conventional Fitting | Collar | Tapping Screw | Flanged Block Bearing | Cylindrical Pin | Clamp | Bush | Locknut | T-nut | Key | Average |
|---------------------|------------|------------------|----------------------|--------|---------------|-----------------------|-----------------|-------|------|---------|------|-----|---------|
| \(G_{ours} + R_{ours} + D_{ours}\) | 0.000 | 0.029 | 0.091 | 0.000 | 0.143 | 0.000 | 0.429 | 0.000 | 0.000 | 0.000 | 0.571 | 0.105 |
| \(G_{ours} + R_{ours} + D_{ours}\) | 1.000 | 0.477 | 0.773 | 0.600 | 0.621 | 0.382 | 0.900 | 0.961 | 0.630 | 0.810 | 0.971 | 0.907 | 0.753 |
| \(G_{ours} + R_{ours} + D_{ours}\) | 1.000 | 0.857 | 0.759 | 0.955 | 0.857 | 0.857 | 0.571 | 1.000 | 1.000 | 0.714 | 1.000 | 0.857 | 0.869 |
| \(G_{ours} + R_{ours} + D_{ours}\) | 1.000 | 0.857 | 0.857 | 1.000 | 0.786 | 1.000 | 0.875 | 1.000 | 0.571 | 0.905 | 1.000 | 1.000 | 0.904 |
| \(G_{ours} + R_{ours} + D_{ours}\) | 0.619 | 0.400 | 0.692 | 0.533 | 0.609 | 0.429 | 1.000 | 0.700 | 0.381 | 0.458 | 0.643 | 0.429 | 0.574 |
| \(G_{ours} + R_{ours} + D_{ours}\) | 0.857 | 0.826 | 0.571 | 0.500 | 0.714 | 0.571 | 0.429 | 0.429 | 1.000 | 0.714 | 0.524 | 0.143 | 0.562 |
| \(G_{ours} + R_{ours} + D_{ours}\) | 0.300 | 0.350 | 0.350 | 0.050 | 0.250 | 0.150 | 0.250 | 0.350 | 0.400 | 0.200 | 0.250 | 0.100 | 0.250 |
| \(G_{ours} + R_{ours} + D_{ours}\) | 0.857 | 0.857 | 1.000 | 0.786 | 1.000 | 0.875 | 1.000 | 0.571 | 0.905 | 1.000 | 1.000 | 0.904 |

| Approaches          | Square Nut | Standard Fitting | Conventional Fitting | Collar | Tapping Screw | Flanged Block Bearing | Cylindrical Pin | Clamp | Bush | Locknut | T-nut | Key | Average |
|---------------------|------------|------------------|----------------------|--------|---------------|-----------------------|-----------------|-------|------|---------|------|-----|---------|
| \(G_{ours} + R_{ours} + D_{ours}\) | 1.000 | 0.857 | 0.857 | 1.000 | 0.786 | 1.000 | 0.875 | 1.000 | 0.571 | 0.905 | 1.000 | 1.000 | 0.904 |

The comparison results of generator performance, refinement performance, discriminator performance and the baseline. In each part, the results of our approach are reported in the bottom row and the best results are shown in bold.
First, we evaluate the ability of the learning-based models to generalize to novel objects in the test set. We use generator models trained for 500 epochs, and all models converge. The distance between the generated and ground-truth orientations \((S_g, S_T)\) for each model is calculated by

\[
d_{d}(R_T) = \min_{R_g \in S_g} d_{geo}(R_g, R_T),
\]

where \(R_T \in S_T\). The distribution histogram of \(d_{d}(R_T)\) is visualized in Fig. 5. The figure shows that the distance distribution of \(G_{ours}\) is more concentrated on small distances than \(G_{CD}\) and \(G_{L2G}\). The proportion of small distances is also larger than \(G_{CD}\) and \(G_{L2G}\). In addition, the mean distances of \(G_{ours}\), \(G_{CD}\) and \(G_{L2G}\) are 0.24, 0.32, and 0.94, respectively, which indicate that the generated orientations of \(G_{ours}\) are the closest to the ground-truth orientations.

We further compare the results of the above approaches on test objects of different categories displayed in Fig. 3. As reported in the bottom rows of each part in Tab. I and II, our approach gives the best performance compared to others in terms of the average accuracy and diversity of predicted placements, achieving 90.4% accuracy and 81.3% diversity. \(R_{ours}\) and \(D_{ours}\) remaining constant across all approaches highlights \(G_{ours}\)’s ability to generate accurate and diverse placements. We attribute the results to the loss design in \(G_{ours}\) to bring the generated orientations of the trained model closer to the ground-truth orientations. For most objects in tables, the accuracy and diversity of placements predicted by our approach exceed or are on par with the best results of the rest approach. The first approach does not perform well on both accuracy and diversity. Taking a close look at its outputs, we find that \(G_{L2G}\) tends to generate limited types of placements for each object. Therefore, its results lack diversity. Since the proportion of sorted stable placements is small, the first approach is deficient in accuracy.

2) Refinement Performance: We carry out self-ablation experiments to validate the effectiveness of the position refinement stage. The comparison results between our approach \((G_{ours} + R_{ours} + D_{ours})\) and the two-stage approach \((G_{ours} + D_{ours})\) are shown in the second part in Tab. I and II. As shown in Tab. I, our approach predicts placements with higher accuracy for all the objects except the cylindrical pin compared with \((G_{ours} + D_{ours})\). Moreover, our approach significantly improves the average accuracy of the predicted placements from 57.4% of \((G_{ours} + D_{ours})\) to 90.4%. These results indicate that the position refinement stage can refine the positions of generated placements from the first stage, resulting in unstable placements becoming stable. From Tab. II, we can see that the refinement stage influences the placement diversity of only half of the test objects and our approach performs better than \((G_{ours} + D_{ours})\) by 2.1% for the average diversity of predicted placements.

3) Discriminator Performance: To demonstrate the performance of our discrimination stage, we compare our approach with an ablated version of our approach and a variation with a discriminator of PointNet++ [8] networks. The approaches mentioned for comparison are listed below:

- \(G_{ours} + R_{ours} + D_{PN2}\): the discriminator in our approach is replaced by \(D_{PN2}\) [11], which is a PointNet++ network with abstraction layers. The discriminator \(D_{PN2}\) is trained on our training set, and the same data is also used to train our discriminator \(D_{ours}\). The score threshold in \(D_{PN2}\) for selecting stable placements is set to 0.92, the same threshold for our discriminator \(D_{ours}\).
- \(G_{ours} + R_{ours} + D_{rand}\): the discriminator in our approach is blocked. The object’s placements predicted after our generator and refinement stage are randomly selected as the final predicted placements.

First, we evaluate the generalization performance of the two learning-based discriminators \((D_{PN2} \text{ and } D_{ours})\) on the test set. The results of the precision-recall curves tested on 2,000 collected placements of 99 test objects are shown in Fig. 6. The curves illustrate the relationship between the precision of classifying all placements and the recall of stable place-
ments. Our discriminator (the blue curve) performs better than $D_{PN2}$ (the green curve) in precision and recall, respectively. Next, we compare the results of approaches with different discriminators on test objects. The accuracy and diversity of predicted placements for comparison are reported in the third part in Tab. I and II, respectively. Our discriminator outperforms others for almost all the test objects in terms of placement accuracy. We hypothesize that the operation of obtaining features of local contact situations facilitates feature extraction for placement classification. The performance of our discriminator for predicting diverse placements for almost all the objects is on par with the best results from others. In addition, our approach achieves an average diversity of 81.3%, the best among the above approaches.

4) Comparison with Baseline: The baseline method [6] predicts object placements on the supporting objects with point clouds. We use the table plane as the support and select the top 100 generated results with the highest scores from the baseline. We compare the accuracy and diversity of the generated placements between our approach and the baseline. The comparison results are shown in the last two rows of Tab. I and II. For most test objects, the baseline fails to predict stable placements. The baseline method only predicts a few stable positions for a few objects, such as the cylindrical pin. The accuracy is 5/100=5.0%, and these placements cover half the placement types. Our approach outperforms the baseline on all test objects. The predicted placements of the baseline and our approach are shown in Fig. 7. For the displayed test objects, the generated placements of the baseline are unstable. In contrast, the placements generated by our approach are stable.

B. Manipulation Tasks

Fig. 8 shows constrained scenarios that require generated placements necessitating diversity. In these scenarios, the initial and goal pose of the objects are given. However, the shared grasp configurations between the initial and the goal placements are unavailable due to the constraints of the objects’ sizes and the fixed base position of the robot. For example, the robot can grasp the bearing and the T-nut (the first and last objects in Fig. 8) in the initial placements. However, it cannot flip these objects with one pick-rotate-and-place step because of the constraints of the object’s geometry. Since the robot base is fixed in the environment, the robot is unable to grasp the knob (the second object in Fig. 8) in the initial placement and flip it with one pick-rotate-and-place step. Based on the generated placements obtained from our approach, the robot is able to perform sequential pick-and-place steps to flip novel objects with different shapes. These manipulation tasks in Fig. 9 emphasize that diversity...
Fig. 9. Solutions for flipping objects. The robot can flip novel objects with multiple pick-and-place steps based on the generated placements.
in placement generation is critical. At last, we validate our approach on the real platform of a Franka robot arm and a table. The object is a 3D-printed object in the test set, which vary in shape from the training objects. We show the scenario of planar manipulation in Fig. 10. Initially, the object is placed on the table in a random pose. We pass the point cloud generated from multiple views to our trained models and then construct the manipulation graph with the generated placements. The robot performs sequential pick-rotate-and-place steps in the manipulation graph to reorient the object. The robot cannot flip the T-nut with one step due to the collision between the table and the robot. While as the example shown in Fig. 10, the flipping of the T-nut is achieved by sequential collision-free steps of the robot based on our generated placement.

VI. Conclusion

In this work, we propose a three-stage pipeline to generate accurate and diverse stable placements of novel objects based on point clouds. A large-scale dataset containing collected placements and contact information between objects is also created to train our neural network models. The comparison experiments demonstrate that our approach outperforms the start-of-the-art and achieves the average accuracy rate of 90.4% and the average diversity rate of 81.3% on the generated placements. Meanwhile, the real-robot experiments validate that our approach can generate stable placements for planar manipulation tasks. Feasible and collision-free grasp configurations are important for robots to realize the transformation between placements. In the future, we will attempt to improve the robot’s trajectory with learning-based algorithms to ensure that the robot’s trajectory is smoother and safer than the currently used sampling-based algorithm.

Acknowledgement

The work is supported by the Shenzhen Key Laboratory of Robotics Perception and Intelligence, Southern University of Science and Technology, Shenzhen 518055, China, under Grant ZDSYS20200810171800001.

References

[1] J. Zhou, J. A. Bagnell, and M. T. Mason, “A fast stochastic contact model for planar pushing and grasping: Theory and experimental validation,” arXiv preprint arXiv:1705.10664, 2017.
[2] Y. Hou, Z. Jia, A. M. Johnson, and M. T. Mason, “Robust planar dynamic pivoting by regulating inertial and grip forces,” in Algorithmic Foundations of Robotics XII, pp. 464–479, Springer, 2020.
[3] 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.
[4] W. Wan, K. Harada, and F. Kanehiro, “Preparatory manipulation planning using automatically determined single and dual arm,” IEEE Transactions on Industrial Informatics, vol. 16, no. 1, pp. 442–453, 2019.
[5] W. Wan and K. Harada, “Achieving high success rate in dual-arm handover using large number of candidate grasps, handover heuristics, and hierarchical search,” Advanced Robotics, vol. 30, no. 17-18, pp. 1111–1125, 2016.
[6] S. Cheng, K. Mo, and L. Shao, “Learning to regrasp by learning to place,” arXiv preprint arXiv:2109.08817, 2021.
[7] P. Xu, H. Cheng, J. Wang, and M. Q.-H. Meng, “Learning to reorient objects with stable placements afforded by extrinsic supports,” arXiv preprint arXiv:2205.00970, 2022.
[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” Advances in neural information processing systems, vol. 30, 2017.
[9] J. Ma, W. Wan, K. Harada, Q. Zhu, and H. Liu, “Regrasp planning using stable object poses supported by complex structures,” IEEE Transactions on Cognitive and Developmental Systems, vol. 11, no. 2, pp. 257–269, 2018.
[10] K. Wada, S. James, and A. J. Davison, “Reorientbot: Learning object reorientation for specific-posed placement,” arXiv preprint arXiv:2202.11092, 2022.
[11] C. Paxton, C. Xie, T. Hermans, and D. Fox, “Predicting stable configurations for semantic placement of novel objects,” in Conference on Robot Learning, pp. 808–815, PMLR, 2022.
[12] Y. Hou, Z. Jia, and M. T. Mason, “Reorienting objects in 3d space using pivoting,” arXiv preprint arXiv:1912.02752, 2019.
[13] Y. You, L. Shao, T. Migimatsu, and J. Bohg, “Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 5921–5927, IEEE, 2021.
[14] 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, pp. 14911–14919, 2022.
[15] Y. Shi, J. Huang, X. Xu, Y. Zhang, and K. Xu, “Stablepose: Learning 6d object poses from geometrically stable patches,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15222–15231, 2021.
[16] 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.
[17] Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, and Y. Zhu, “Synergies between affordance and geometry: 6-dof grasp detection via implicit representations,” arXiv preprint arXiv:2104.01542, 2021.
[18] M. Breyer, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto, “Volumetric grasping network: Real-time 6 dof grasp detection in clutter,” arXiv preprint arXiv:2101.01132, 2021.
[19] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” 2016.
[20] H. Fan, H. Su, and L. J. Guibas, “A point set generation network for 3d object reconstruction from a single image,” in \textit{Proceedings of the IEEE conference on computer vision and pattern recognition}, pp. 605–613, 2017.

[21] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” in \textit{Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition}, pp. 5745–5753, 2019.

[22] J. Á. Cid and F. A. F. Tojo, “A lipschitz condition along a transversal foliation implies local uniqueness for odes,” \textit{arXiv preprint arXiv:1801.01724}, 2018.

[23] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in \textit{International Conference on Medical image computing and computer-assisted intervention}, pp. 234–241, Springer, 2015.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in \textit{Proceedings of the IEEE conference on computer vision and pattern recognition}, pp. 770–778, 2016.

[25] L. Shao, F. Ferreira, M. Jordà, V. Namdar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, “Unigrasp: Learning a unified model to grasp with multifingered robotic hands,” \textit{IEEE Robotics and Automation Letters}, vol. 5, no. 2, pp. 2286–2293, 2020.

[26] S. Kim, H.-g. Chi, X. Hu, Q. Huang, and K. Ramani, “A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks,” in \textit{Proceedings of 16th European Conference on Computer Vision (ECCV)}, 2020.

[27] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” \textit{IEEE Transactions on Information Theory}, vol. 21, no. 1, pp. 32–40, 1975.