Grasp Planning for Occluded Objects in a Confined Space with Lateral View Using Monte Carlo Tree Search

Minjae Kang, Hogun Kee, Junseok Kim, and Songhwai Oh

***Abstract*— In the lateral access environment, the robot behavior should be planned considering surrounding objects and obstacles because object observation directions and approach angles are limited. To safely retrieve a partially occluded target object in these environments, we have to relocate objects using prehensile actions to create a collision-free path for the target. We propose a learning-based method for object rearrangement planning applicable to objects of various types and sizes in the lateral environment. We plan the optimal rearrangement sequence by considering both collisions and approach angles at which objects can be grasped. The proposed method finds the grasping order through Monte Carlo tree search, significantly reducing the tree search cost using point cloud states. In the experiment, the proposed method shows the best and most stable performance in various scenarios compared to the existing TAMP methods. In addition, we confirm that the proposed method trained in simulation can be easily applied to a real robot without additional fine-tuning, showing the robustness of the proposed method.**

I. Introduction

Robotic manipulation in clutter is challenging due to object occlusions and collisions. In addition, the cluttered object manipulation task requires an agent to handle preliminary tasks (e.g., object singulation or clutter removal) [1]–[4]. Especially, in the lateral access environment such as a shelf, cabinet, and refrigerator, approaching directions to the object are limited, which requires the robot behavior to be planned considering surrounding objects as shown in Figure 1.

Task and motion planning (TAMP) methods have been proposed to perform manipulation tasks in a lateral and confined environment [5]–[7]. TAMP-based methods successfully solve tasks through the elaborately organized algorithm. However, existing TAMP methods are designed for well-defined and known environments. For example, in [5], [6], they assume that all objects are the same type, and the agent knows the location, size, and shape of all objects.

In recent studies, learning-based methods have been studied for robot manipulation tasks. Unlike the above TAMP methods, learning-based models can solve manipulation tasks with objects of various types and sizes and even unknown objects [8], [9]. In the case of lateral environments, there are learning-based methods for solving tasks such as searching a completely occluded target [10], [11] or reaching a target in the clutter [12]. However, these methods are implemented using non-prehensile actions, such as push, which is difficult to apply in dense environments where objects are fragile and prone to fall. Therefore, in this paper, we propose a learning-based manipulation planning method to retrieve a target in the lateral environment composed of prehensile objects with various types and sizes. To be similar to the actual situation, we do not use ground truth information nor the top view of objects. We solve the retrieval task using only lateral view which is accessible to the agent.

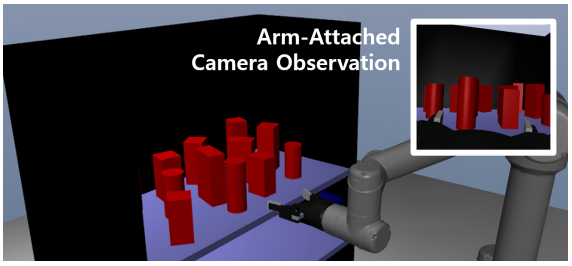


Fig. 1: The lateral access simulation environment with a UR5e robot. The agent tries to bring out the occluded target object using prehensile actions without collisions.

For retrieving a target object, we sequentially rearrange the surrounding objects to create collision-free paths to grasp the target. We plan the grasping order based on Monte Carlo tree search (MCTS), a widely used planning method in robotic manipulation [13]–[15]. Since it is difficult to predict the collision between objects and the grasping result according to the shape of the object, the rearrangement problem in the lateral environment requires robotic simulation. However, using the robotic simulation for tree search is time-consuming, making MCTS impractical.

To handle this problem, we propose a novel tree search method that determines the validity of the object rearrangement sequence in the lateral environment without robotic simulation. A valid sequence means that all objects in the sequence are prehensile when relocated. First, the proposed method synthesizes new observations using point cloud observations from various viewpoints. Next, we plan valid rearrangement orders by sequentially removing objects judged prehensile using the proposed two networks: the prehensile decision network and the priority decision network. The prehensile decision network verifies whether there is an approaching direction in which an object becomes prehensile, and the priority decision network excludes or recommends candidates for objects to be rearranged for efficient tree search. As a result, we find the optimal object rearrangement order without simulation, allowing us to save the computation cost required for tree search.

In the experiment, we confirm that the proposed method shows the best and most stable performance for various

M. Kang, H. Kee, J. Kim, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: minjae.kang@rllab.snu.ac.kr, hogun.kee@rllab.snu.ac.kr, junseok.kim@rllab.snu.ac.kr, songhwai@snu.ac.kr). This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, (SW Star Lab) Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning, 50%, and No. 2019-0-01371, Development of Brain-Inspired AI with Human-Like Intelligence, 50%). *(Corresponding author: Songhwai Oh.)*

sizes and types of objects compared to the existing TAMP-based methods. Also, the proposed method successfully plans the rearrangement order in various arrangement scenarios, including even unseen objects. Moreover, we verify that the proposed method trained in simulation can retrieve the target object using a real robot without additional training.

In summary, the contributions of this paper are as follows:

- It is the first study of a learning-based method for occluded target grasping using only prehensile actions in the lateral access environment.
- We propose a point cloud observation-based tree search, which does not require complex simulation, hence, reducing the computation time of MCTS.
- The proposed method plans the rearrangement orders with stable performance for various scenarios, which cannot be solved with the existing TAMP methods.
- We successfully transfer the trained model in simulation to a real robot without additional training.

II. Related Work

A. Task and Motion Planning in a Lateral Environment

There are studies to address occluded object retrieval tasks in a shelf while minimizing the number of relocating objects based on TAMP. Lee et al. [5] modify the local path planner, vector field histogram+ [16], to calculate the cost of the approach angle to the target considering surrounding objects. They find an angle with the lowest histogram value and remove an object closest to that angle. [6] extends [5] by using the cost function of the A* search algorithm with polar histograms to select the next object to be relocated.

Since the aforementioned TAMP-based methods assume that all objects are cylinders that can be grasp from all directions, only positions of the objects are important and postures are not considered. It is also assumed that the agent knows the location and size of all objects. To overcome theses limitations, in this paper, we consider various objects of different shapes and estimate the location, size, and posture from sensed observations.

B. Monte Carlo Tree Search for Object Rearrangement

Monte Carlo tree search (MCTS) has been used for object rearrangement problem. Song et al. [14] perform the table-top sorting task through MCTS using non-prehensile actions. The authors use a two-dimensional physics engine, Box2D, for simulation. Labbe et al. [13] handle the object rearrangement problem on a table with a minimum number of pick-and-places based on MCTS. Eljuri et al. [15] solve the rearrangement problem that considers the object positions and the postures using MCTS with pick-and-place actions.

There are differences between the above tasks and the task considered in this paper. First, aforementioned methods allow the robot to approach objects from top to bottom such that an object can be picked without collisions. In addition, since [14] and [13] perform tree search on a plane, they can perform MCTS within five seconds using 2D simulation. However, in the proposed method and [15], it is difficult to plan with a simple simulation because they need to consider 3D object shapes. In [15], it takes more than 10 minutes to plan the entire rearrangement order for four objects. On the other hand, the proposed method requires only 15 seconds on average to perform MCTS.

III. Problem Statement

We consider an object rearrangement problem to retrieve a target object without collisions in the lateral access environment. We want to solve the problem using only lateral depth view without the ground truth locations of objects and the top view of objects. By utilizing a depth camera attached to the robot arm, the agent determines the object types, sizes, locations, and whether it is possible to grasp. For safe object rearrangement, we consider only prehensile action, i.e., pick-and-place. The agent can move only one object in one action, and overhand grasping is not allowed. Lastly, we assume that there is an enough space to relocate objects.

Our goal is to retrieve the target object while minimizing the number of actions. Assume that there are N objects in the environment and $O = \{o_1, o_2, ..., o_N\}$ is the set of objects. We want to find feasible grasping sequences with the minimum number of pick-and-places to create a path that can allow a robot to grasp the target object without collisions.

IV. Planning for Occluded Object Grasping

This section introduces a method for finding the optimal object grasping sequence based on MCTS. For every iteration of MCTS, a tree search is performed in four steps: *selection*, *expansion*, *simulation*, and *backpropagation*. First, in the selection step, child nodes are sequentially chosen starting from a root node until it reaches a leaf node. Next, in the expansion step, it creates a child node from the reached leaf node. After that, in the simulation step, an episode is proceeded using a rollout policy until it reaches a terminal node. Lastly, in the backpropagation step, the number of node visits and episode results are updated. We repeatedly perform tree search consisting of these four steps to explore various rearrangement sequences for target retrieval.

MCTS usually obtains the next node by performing a selected action in a simulation representing the current node. However, grasping an object using robotic simulation is time-consuming. To address this issue, we propose **G**rasping **P**lanning with **P**oint cloud **P**rocessing (GP3), point cloud-based MCTS, which creates feasible next nodes without simulation. GP3 performs tree search using point cloud observations with object indexing information indicating which point belongs to which object or background. The point cloud observation and object indexing information are processed from the depth image described in Section V. Since it is a partially observable environment, new objects that have been hidden before can be detected when an object is moved. Therefore, we plan the grasping sequence through GP3, relocate only the first object and then re-plan the order by observing the environment again as shown in Figure 2.

A. Rigid Transformation of Point Cloud

We describe the rigid transformation of point cloud observations before introducing GP3. Each point cloud observation $\mathcal{S} \in \mathbb{R}^{M\times 3}$ consists of M three-dimensional points. In contrast to a 2D image, a point cloud can be rotated, translated, and merged because each point in the point cloud represents a three-dimensional Cartesian coordinate. Therefore, using the point cloud observation, it is possible to generate an observation from other viewpoints without obtaining new images from different viewpoints.

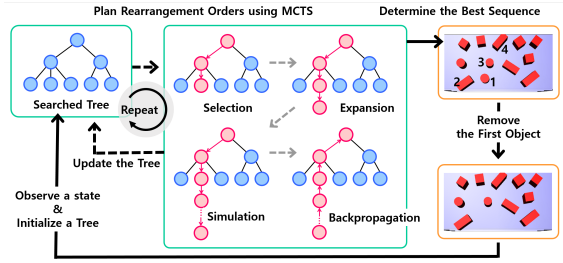


Fig. 2: The process overview of GP3. GP3 repeatedly performs tree search to explore the order of objects rearrangement (green boxes). Once the optimal grasping sequence is obtained, the agent relocates the first object, observes the environment, and initializes a tree (orange boxes).

Rotation and Translation of Point Clouds. We can obtain a rotation matrix and translation vector for viewpoint switching, using two pairs of position and quaternion of the viewpoint observed and the desired viewpoint. We define a rigid transformation vector $q \in \mathbb{R}^7$ consisting of a translation vector and a quaternion vector. The transformed observation $\mathcal{S}_q$ from observation $\mathcal{S}$ is defined using q as follows:

$$\mathcal{S}_q^T = R(q)\mathcal{S}^T - T(q), \tag{1}$$

where $R(\cdot) \in \mathbb{R}^{3\times3}$ is a rotation matrix and $T(\cdot) \in \mathbb{R}^3$ is a translation vector.

Merging of Point Clouds. Using the transformation of (1), we can merge multiple point cloud observations into one observation. The merged observation $\mathcal{S}_m$ for the designated viewpoint is obtained using point clouds observed at different K viewpoints as $\mathcal{S}_m = \bigcup_{i=1,\dots,K} \mathcal{S}^i_{q_i}$, where $\mathcal{S}^i$ is a point cloud observation at the i-th viewpoint and q_i is a transform vector between the i-th viewpoint and the designated viewpoint. Note that every state in GP3 is a merged observation. In this paper, $\mathcal{X}$ denotes a merged point cloud observation.

B. Prehensile Decision Network

In this section, we propose a prehensile decision network ψ that determines the existence of a collision-free path for object grasping from the point cloud observation. If there is a viewpoint in which a target object can be grasped without collisions, the target is prehensile. A robot gripper will approach straight from the viewpoint to the target coordinate. Therefore, the prehensile network predicts whether collisions with other objects occur and whether the target object can be grasped considering the posture when approaching straight.

The network ψ performs binary classification with a four-dimensional point set $\mathcal{P} \in \mathbb{R}^{M\times4}$ as an input (i.e., $\psi : \mathcal{P} \mapsto [0, 1]$). We define *an object-prehensile state* $\mathcal{P}^{\mathcal{X}}_{t,i}$ to check whether there is a collision between the target object o_t and another object o_i in observation $\mathcal{X}$. We concatenate an 1D feature to each 3D coordinate using object indexing information. The feature of each point has a value of 1 if it is the target object o_t, 0 if it is object o_i, and -1 if it is other objects or backgrounds as shown in Figure 3. For $\mathcal{P}^{\mathcal{X}}_{t,t}$, two objects are the same and there is no point with value 0. This observation is used to determine whether the target object o_t can be picked by approaching straightly. The prehensile decision network outputs 1 if the object is prehensile, i.e., if there is no collision between two objects or if the object can be picked at the given angle, and 0 otherwise.

In summary, the prehensile decision of the target object o_t through the network ψ from $\mathcal{X}$ and a transformation vector set $V_q = \{q_1, ..., q_K\}$ of K viewpoints is as follows:

$$\begin{aligned} &o_t \in O_p^{\mathcal{X}} \quad \text{if } \forall i \in \{1, ..., N\}\ \exists q_k \in V_q,\ \psi(\mathcal{P}^{\mathcal{X}_{q_k}}_{t,i}) > 0.5, \\ &o_t \notin O_p^{\mathcal{X}} \quad \text{otherwise,} \end{aligned} \tag{2}$$

where q_k is a transformation vector between the k-th viewpoint and the front view and $O_p^{\mathcal{X}}$ is a set of prehensile objects in observation $\mathcal{X}$. In summary, if we try to grasp an object o_i, we check the prehensile decision of o_i and find valid viewpoints and approaching directions to grasp o_i using (2).

The prehensile decision network ψ consists of three set abstraction modules of pointnet++ network [17] and three fully connected layers. To train ψ, we randomly set the simulation with object positions, the number of objects, the target object, and the approaching angle and collect data of collisions and grasping results. Since the proposed model has binary outputs, we optimize ψ using the cross-entropy loss.

C. Priority Decision Network

In this section, we introduce a priority decision network ϕ that decides the grasp priority of objects. We define a four-dimensional point set, *object-conditioned state* $\mathcal{Q} \in \mathbb{R}^{M\times4}$ as an input of the priority decision network. $\mathcal{Q}^{\mathcal{X}}_{t,i}$ is a state in which the object o_i is chosen when the target object is o_t in observation $\mathcal{X}$. At this time, o_i is a different object from o_t. Therefore, $N-1$ object-conditioned states can be generated when N objects are detected as shown in Figure 3. Similar to the object-prehensile state $\mathcal{P}$, each point of $\mathcal{Q}$ consists of a 3D coordinate and additional 1D feature. The feature is 1 if the point belongs to the target o_t, 0 if it belongs to the conditioned object o_i, and -1 otherwise.

The priority decision network estimates an object-conditioned state value $\phi(\mathcal{Q}^{\mathcal{X}}_{t,i})$, which is the expected cumulative reward that can be obtained by selecting the object o_i when the target is o_t (i.e., $\phi : \mathcal{Q} \mapsto \mathbb{R}$). To calculate the value of the state $\mathcal{Q}^{\mathcal{X}}_{t,i}$, we define a reward function $r : \mathcal{Q} \mapsto \mathbb{R}$ and a done function $d : \mathcal{Q} \mapsto \{0, 1\}$ as follows:

$$r(\mathcal{Q}^{\mathcal{X}}_{t,i}) = \begin{cases} 1 & o_i \in O_p^{\mathcal{X}},\ o_t \in O_p^{\mathcal{X}_{\{i\}}} \\ -p & o_i \in O_p^{\mathcal{X}},\ o_t \notin O_p^{\mathcal{X}_{\{i\}}} \\ -p & o_i \notin O_p^{\mathcal{X}} \end{cases},$$

$$d(\mathcal{Q}^{\mathcal{X}}_{t,i}) = \begin{cases} 0 & o_i \in O_p^{\mathcal{X}} \\ 1 & o_i \notin O_p^{\mathcal{X}} \end{cases},$$

where p is a positive penalty value sufficiently smaller than one and $\mathcal{X}_{\{i\}}$ denotes the observation, which removes $\mathcal{O}_i$, the point cloud representation of object o_i, from observation $\mathcal{X}$, i.e., $\mathcal{X}_{\{i\}} = \mathcal{X} \setminus \mathcal{O}_i$.

Using defined functions r and d, the object-conditioned state value $\phi(\mathcal{Q})$ is repeatedly updated through the Bellman equation as follows:

$$\phi(\mathcal{Q}^{\mathcal{X}}_{t,i}) = r(\mathcal{Q}^{\mathcal{X}}_{t,i}) + \gamma\,(1 - d(\mathcal{Q}^{\mathcal{X}}_{t,i}))\max_{j\notin\{t,i\}}[\phi(\mathcal{Q}^{\mathcal{X}_{\{i\}}}_{t,j})]. \tag{3}$$

The priority decision network consists of three set abstraction modules and four fully connected layers. We use a random policy to select the next object and grasp the selected object for collecting train data. We save state-action transitions and object grasping results with collisions in buffer $\mathcal{B}$ and train the priority decision network to minimize one-step prediction error of (3) using transition data of $\mathcal{B}$.

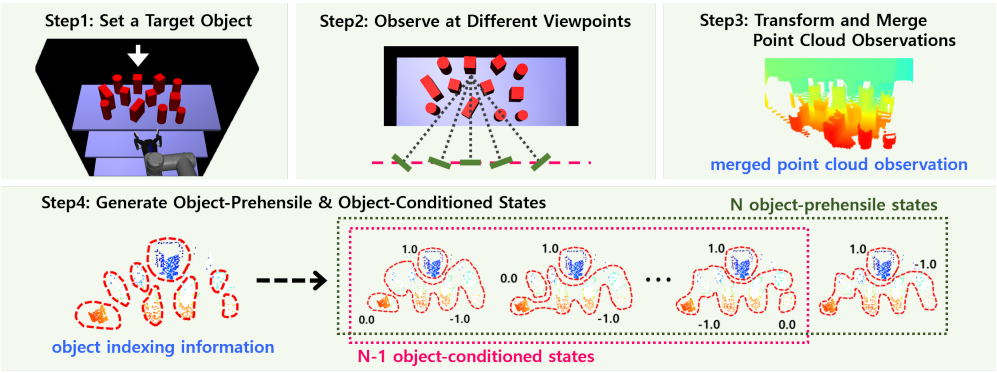


Fig. 3: Overview of generating point cloud states. Object-prehensile states are used as inputs to the prehensile decision network and object-conditioned states are used for the priority decision network.

D. Monte Carlo Tree Search for Object Rearrangement

In this section, we plan object grasping orders to retrieve the target object using GP3, an MCTS method with the prehensile decision network and priority decision network. A node of GP3 is a point cloud observation $\mathcal{X}$, and an action is an object selection among objects in $\mathcal{X}$. GP3 initializes a tree by taking a current observation as a root node and generates child nodes by sequentially erasing objects selected through actions until it reaches a terminal node. The terminal node of GP3 is divided into *a success node* and *a failure node*. The success node refers to a state in which the target object is prehensile. On the other hand, the failure node is an infeasible state in which a removed object is non-prehensile due to collisions or grasping failure. Eventually, the purpose of GP3 is to find success nodes with the minimum tree depth and obtains action sequences to reach those nodes.

GP3 uses the prehensile decision network to distinguish invalid plans. The valid plan means that all objects in the rearrangement order are prehensile when moved. In the selection and simulation steps, the prehensile decision network checks the following facts: 1) Whether the chosen object to be relocated is non-prehensile (i.e., whether the child node is a failure node). 2) Whether the target object is prehensile (i.e., whether the current node is a success node). All prehensile decisions in GP3 are determined through (2).

In addition, GP3 utilizes the priority decision network to save computation costs by significantly reducing the number of plans to be explored with tree search. In GP3, the priority decision network performs the following roles: 1) In the selection step, the priority decision network prunes non-prehensile objects using object-conditioned state values. 2) In the simulation step, GP3 selects the next object using object-conditioned state values as a rollout policy.

Selection: At each node $\mathcal{X}$, we select an action a_i of choosing an object o_i to be rearranged. Then, GP3 generates a new point cloud observation $\mathcal{X}_{\{i\}}$, which removes points of object o_i from $\mathcal{X}$, as a child node. For performing tree search, we store the number of node visits $N(\mathcal{X})$ and define a node value function $F : \mathcal{X} \mapsto [0, 1]$ and a node-action value function $G : (\mathcal{X}, a) \mapsto [0, 1]$ as follows:

$$F(\mathcal{X}) = \max_i [G(\mathcal{X}, a_i)],$$
$$G(\mathcal{X}, a_i) = \begin{cases} 0 & \text{if } o_i \notin O_p^{\mathcal{X}} \\ 1 & \text{else if } o_i \in O_p^{\mathcal{X}},\ o_t \in O_p^{\mathcal{X}_{\{i\}}} \\ \max[-0.1 + F(\mathcal{X}_{\{i\}}),\ 0] & \text{otherwise} \end{cases}, \tag{4}$$

where n_{depth} is the tree depth of node $\mathcal{X}$. Note that the terminal node has only two node values: 1 if the target is prehensile and 0 if the current node is infeasible. The parent node value is less than the maximum node value of child nodes by 0.1. For example, if the node value of $\mathcal{X}$ is 0.8, it means that $\mathcal{X}$ can reach a success node in two actions.

Until GP3 reaches a leaf node, it selects an action using (5), which modifies upper confidence bounds (UCT) [18] value $U(\mathcal{X}, \cdot)$ to balance exploration and exploitation.

$$U(\mathcal{X}, a_i) = G(\mathcal{X}, a_i) + c\sqrt{\frac{2 \log N(\mathcal{X})}{N(\mathcal{X}_{\{i\}})}}, \tag{5}$$

where c is an exploration term. Finally, the agent chooses the next object of the maximum UCT value among all objects.

GP3 limits infeasible actions by discriminating non-prehensile objects using the priority decision network to perform selection steps efficiently. In the case of non-prehensile objects, the corresponding object-conditioned state value becomes zero through (3). We set prehensile value threshold δ and judge an object o_i with a state value below δ as a non-prehensile object. (i.e., $o_i \notin O_p^{\mathcal{X}} \iff V(\mathcal{Q}_{t,i}^{\mathcal{X}}) < \delta$) By pruning object choices with values of less than δ, the number of orders to be searched by GP3 can be significantly reduced.

Expansion: If GP3 reaches a leaf node, it inserts a child node into the tree to expand the searched area. After the expansion step, the simulation step is performed to predict the value of the inserted child node.

Simulation: GP3 generates child nodes sequentially until it reaches a terminal node. For efficient simulation steps, we propose a new rollout policy using the priority decision network. First, we create a set of all object-conditioned states (i.e., $\{\mathcal{Q}_1^{\mathcal{X}}, ..., \mathcal{Q}_{t-1}^{\mathcal{X}}, \mathcal{Q}_{t+1}^{\mathcal{X}}, ..., \mathcal{Q}_N^{\mathcal{X}}\}$) from the current observation $\mathcal{X}$ and then obtain each node-action value using the priority decision network. GP3 selects the object with the highest value and removes it to create a child node. If GP3 reaches a success node, we set the node value to one and go backward with the generated rollout trajectory, reducing the node value by 0.1 compared to each child node. In other cases, if GP3 fails to grasp the target in the simulation step, the inserted child node value is set to zero.

Backpropagation: After the end of above steps, we update node-action values of all nodes and actions visited in the selection step according to (4). After this step, GP3 starts a new tree search with revised node values.

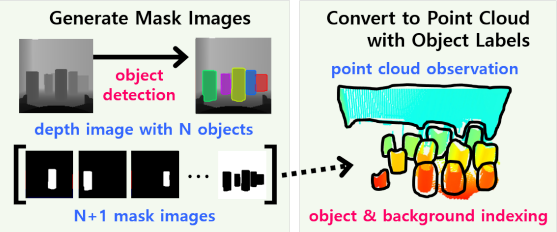


Fig. 4: Process of generating point cloud observation and object indexing information from a depth image.

Once GP3 is executed, we pick the first object in the generated optimal grasping sequence and put it down in an empty space. Since we assume that there is enough space for placement, we place it in a space that would not affect the target retrieval. After relocating the object, we obtain a new observation, initialize the tree, and perform GP3.

V. Point Cloud Observation Processing

We need to generate point could observation and object indexing information from raw depth images to perform GP3 as shown in Figure 4. Object indexing information indicates each point of point cloud observation belongs to which object or background. First, we detect objects from a lateral depth image using SD Mask R-CNN [19], which performs object instance segmentation from depth images. After extracting object masks, we combine the depth image and each mask image to generate point cloud of each object separately. In addition, we merge all mask images with the depth image to create a point cloud for backgrounds. We use the Open3D library [20] to convert depth images to point clouds. Consequently, we generate $N+1$ point clouds from N objects detected depth observation. We merge all separated point clouds into one point cloud observation and store information about which points are generated from which object, called object indexing information.

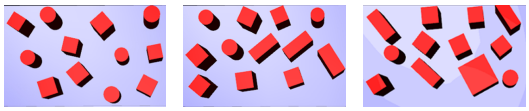
VI. Experiment

In this section, we solve the target retrieval problem in a shelf using only lateral views. We construct a simulation using the robosuite framework [21] based on the MuJoCo [22] physics engine and use UR5e with a Robotiq 2-Finger 85 gripper as shown in Figure 1. The agent observes a 128×128 depth image from an arm-attached camera, which has 75.0 field of view. We assume that the target object is partially detected in the initial state, so the agent cannot grasp the target directly, but its location can be estimated.

A. Experiment Setup

We randomly place 12 unfixed-sized objects, including cylinders, square pillars, and rectangular cylinders, on a 60 cm $\times$ 35 cm shelf in simulation. We compare the retrieval success rate, the number of pick-and-places, and the number of collisions during object rearrangement to verify the performance and safety of the planning methods. If there are more than three collisions, the corresponding episode is considered as a failure. We set seven angles divided by 10° apart from -30° to +30° as multiple viewpoints, and all observations and actions are performed only at these angles.

The experiments are conducted on three object arrangement scenarios as shown in Figure 5. In the first scenario, only cylinders and square pillars that can be picked in all directions are arranged on the shelf. In the second scenario, rectangular cylinders with one side wider than the gripper are added. These rectangular cylinders can only be grasped at limited angles, and approaching them with invalid directions causes collisions. In the final scenario, there is an large object that cannot be grasped. These large objects are unseen objects in the training step of GP3. We verify whether the planning methods can judge objects that cannot be relocated and find new paths to avoid them through this scenario.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 5: Object arrangement examples of each scenario.

(a) Rearrangement results of Scenario 1

Algorithm	Ours	[5]	ROTS	[5]-ψ	ROTS-ψ
Success Rate	1.00	1.00	0.95	1.00	1.00
# of Pick-and-Places	3.70	3.55	3.84	3.70	3.65
# of Collisions	0.30	0.35	0.26	0.30	0.40

(b) Rearrangement results of Scenario 2

Algorithm	Ours	[5]	ROTS	[5]-ψ	ROTS-ψ
Success Rate	0.95	0.70	0.60	0.80	0.80
# of Pick-and-Places	3.95	4.07	3.75	4.88	4.81
# of Collisions	0.53	0.64	0.83	0.56	0.50

(c) Rearrangement results of Scenario 3

Algorithm	Ours	[5]	ROTS	[5]-ψ	ROTS-ψ
Success Rate	1.00	0.80	0.65	0.80	0.75
# of Pick-and-Places	5.20	4.13	3.92	5.75	5.87
# of Collisions	0.50	1.06	1.23	0.69	0.60

TABLE I: Performance comparison with TAMP methods.

B. Performance Comparison with TAMP methods

In this experiment, we compare the proposed method GP3 with the existing TAMP methods, [5] and ROTS [6], for the target retrieval problem in the lateral environment. [5] utilizes a modified vector field histogram+ to find the optimal approach angle to a target object. ROTS performs a tree search using the A* search algorithm for grasp planning.

Both baselines consider only cylinders and assume that they know the locations and sizes of all objects. However, we perform rearrangement tasks on three arrangement scenarios, and the locations and sizes of objects are estimated using lateral views. Since baselines cannot determine the prehensile decision, we add algorithms, [5]-ψ and ROTS-ψ, that combine each algorithm with a prehensile decision network ψ of GP3. In these methods, the object to be rearranged is chosen using TAMP methods, but the angle to grasp the object is selected through the prehensile decision network.

We perform 20 target retrieval episodes per scenario in the shelf environment. Note that we average the number of pick-and-place attempts and collisions only for successful episodes. Even if the average number of performed actions is small, if the success rate is low, this means that the method plans reckless orders. According to the experiment results as shown in Table I, GP3 shows the highest success rate in all scenarios while maintaining the number of collisions low. In the last scenario, GP3 determines that the unseen large object is non-prehensile and plans the rearrangement sequence that does not include it. Meanwhile, [5] and ROTS show stable performance in Scenario 1. However, there

Fig. 6: (left) Real shelf environment with UR5. (right) Post-processing of depth images from the D435 depth camera.

Algorithm	Ours	[5]-ψ	ROTS-ψ
Success Rate	1.00	0.60	0.80
# of Pick-and-Places	3.80	4.33	4.00
# of Collisions	0.30	0.33	0.25

TABLE II: Experiment results in the real environment

are many collisions in Scenario 2 and 3 because they do not consider grasping direction for rearrangement planning. Although [5]-ψ and ROTS-ψ significantly reduce the number of collisions, they often fail to find the optimal sequence.

C. Real Robot Environment

In this section, we perform the target retrieval by transferring GP3 trained in simulation to a real robot as shown in Figure 6. We set a 40 cm $\times$ 28 cm two-story shelf and use a UR5 robot with an Intel Realsense D435 camera. Since we implement GP3 to perform using only depth images, we can reduce the observation gap between simulation and real environments. Depth images from the real are improved using the depth image post-processing of Intel RealSense SDK. We set five angles divided by 10° apart from -20° to +20° as viewpoints for object observation and access. We construct 10 episodes of Scenario 2 with eight objects.

The experiment results using a real robot are shown in Table II. We compare GP3 with [5]-ψ and ROTS-ψ and confirm that GP3 performs the best in the rearrangement task using real objects. It means that the prehensile decision network and priority decision network are successfully applied to the real robot without fine-tuning. The supplementary video shows the details of the real experiment.

VII. Conclusions

In this paper, we have proposed GP3 to solve the object rearrangement problems for target retrieval in the lateral access environment. To grasp an object in the lateral environment where observation and access direction are limited, the agent must consider both collision and the approach angle. Point cloud-based GP3 can generate feasible rearrangement orders without simulation and find the optimal sequence efficiently. GP3 successfully planned the rearrangement order not only for various objects but also for unseen objects in the experiments. Also, GP3 successfully transfer the trained model in simulation to a real robot without fine-tuning.

References

[1] B. Huang, S. D. Han, J. Yu, and A. Boularias, "Visual foresight trees for object retrieval from clutter with nonprehensile rearrangement," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 231–238, 2021.

[2] A. Kurenkov, J. Taglic, R. Kulkarni, M. Dominguez-Kuhne, A. Garg, R. Martín-Martín, and S. Savarese, "Visuomotor mechanical search: Learning to retrieve target objects in clutter," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 8408–8414.

[3] B. Huang, S. D. Han, A. Boularias, and J. Yu, "Dipn: Deep interaction prediction network with application to clutter removal," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4694–4701.

[4] T. Novkovic, R. Pautrat, F. Furrer, M. Breyer, R. Siegwart, and J. Nieto, "Object finding in cluttered scenes using interactive perception," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8338–8344.

[5] J. Lee, Y. Cho, C. Nam, J. Park, and C. Kim, "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 183–189.

[6] J. Lee, C. Nam, J. Park, and C. Kim, "Tree search-based task and motion planning with prehensile and non-prehensile manipulation for obstacle rearrangement in clutter," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8516–8522.

[7] S. H. Cheong, B. Y. Cho, J. Lee, C. Kim, and C. Nam, "Where to relocate?: Object rearrangement inside cluttered and confined environments for robotic manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7791–7797.

[8] Q. Ahmed, M. Arsalan, P. Chris, Y. Michael, and F. Dieter, "Nerp: Neural rearrangement planning for unknown objects," in *Proceedings of Robotics: Science and Systems*, 2021.

[9] O.-M. Pedersen, E. Misimi, and F. Chaumette, "Grasping unknown objects by coupling deep reinforcement learning, generative adversarial networks, and visual servoing," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 5655–5662.

[10] W. Bejjani, W. C. Agboh, M. R. Dogar, and M. Leonetti, "Occlusion-aware search for object retrieval in clutter," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4678–4685.

[11] H. Huang, M. Dominguez-Kuhne, V. Satish, M. Danielczuk, K. Sanders, J. Ichnowski, A. Lee, A. Angelova, V. Vanhoucke, and K. Goldberg, "Mechanical search on shelves using lateral access x-ray," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2045–2052.

[12] M. Hasan, M. Warburton, W. C. Agboh, M. R. Dogar, M. Leonetti, H. Wang, F. Mushtaq, M. Mon-Williams, and A. G. Cohn, "Human-like planning for reaching in cluttered environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7784–7790.

[13] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3715–3722, 2020.

[14] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork, "Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9433–9440.

[15] P. M. U. Eljuri, G. A. G. Ricardez, N. Koganti, J. Takamatsu, and T. Ogasawara, "Combining a monte carlo tree search and a feasibility database to plan and execute rearranging tasks," *IEEE Access*, vol. 9, pp. 21 721–21 734, 2021.

[16] I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," in *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, vol. 2. IEEE, 1998, pp. 1572–1577.

[17] 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.

[18] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[19] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg, "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

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

[21] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[22] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Proc. of the International Conference on Intelligent Robots and Systems*, Oct 2012.