Object Rearrangement Planning for Target Retrieval in a Confined Space with Lateral View

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

***Abstract*— In this paper, we perform an object rearrangement task for target retrieval in an environment with a confined space and limited observation directions. The agent must create a collision-free path to bring out the target object by relocating the surrounding objects using the prehensile action, i.e., pick-and-place. Object rearrangement in a confined space is a non-monotone problem, and finding a valid plan within a reasonable time is challenging. We propose a novel algorithm that divides the target retrieval task, which requires a long sequence of actions, into sequential sub-problems and explores each solution through Monte Carlo tree search (MCTS). In the experiment, we verify that the proposed algorithm can find safe rearrangement plans with various objects efficiently compared to the existing planning methods. Furthermore, we show that the proposed method can be transferred to a real robot experiment without additional training.**

I. Introduction

Robotic manipulation in clutter is challenging, but it is often a necessary ability to use robots in the real world [1]–[8]. In this paper, we solve the target retrieval problem under the clutter in a lateral access environment such as a shelf, cabinet, and refrigerator as shown in Figure 1, where it is difficult to perform manipulation tasks for the following reasons. First, since the agent has limited directions to observe and access objects, occlusions and collisions between objects can occur. In addition, it is performed within a confined space, so an unoccupied space is limited. Therefore, we must relocate the surrounding object within the limited space to safely retrieve the occluded target object.

The object rearrangement task within a confined space is a non-monotone problem that can move the same object multiple times, which is NP-hard [9]. In addition, the agent has a much larger action space than simply finding a grasp order without considering placement [10], [11] because it considers not only what objects to move but also where to relocate them. There are previous studies that solve similar problems [11]–[14]. They propose task and motion planning (TAMP) methods to generate efficient and valid rearrangement plans. However, they have impractical assumptions: First, the agent knows both the size and position of all objects. Second, all objects can be grasped in all directions. Therefore, it is hard to apply these methods in more realistic circumstances considered in this paper.

To overcome the limitations of existing TAMP methods, we do not use ground truth information about objects. The agent estimates the shape and position of each object using lateral observations. For safe object rearrangement, we do not use non-prehensile actions that make objects prone to collapse, and we move objects by pick-and-place operations. The purpose of the rearrangement task is to retrieve the target with a smaller number of actions as possible while preventing falls due to collisions. We assume that the target object is partially observed at the initial state, and its location can be estimated. If the target object is completely occluded, we can relocate objects randomly until the target is detected, and then we can apply the proposed algorithm.

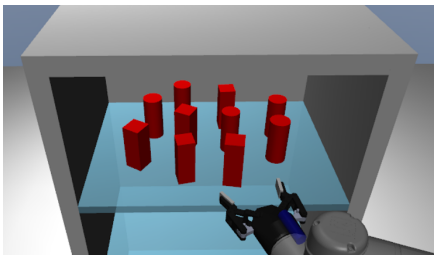


Fig. 1: Target retrieval task in a lateral access environment. The agent must rearrange the surrounding objects to bring out the target without collision.

For safely retrieving the target in a general situation, we combine a learning-based approach and TAMP. We predict object grasp results using a neural network that performs the following two roles. First, it determines whether an object can be grasped without collision in the given object arrangement. Second, it predicts grasp results in the approach direction considering the object shape.

We plan collision-free rearrangement sequences using MCTS. However, there are two problems in performing practical MCTS. First, MCTS in a robotic simulation is time-consuming because operating a robot requires extensive time. We significantly reduce the execution time of MCTS by replacing the robotic simulation with a point cloud-based simulation we proposed. Second, the search space for MCTS is impractically vast. To mitigate this problem, we divide the target retrieval task into sequential sub-problems and find each solution through the subgoal-conditioned MCTS (called sub-MCTS). In addition, we propose a task-specific reward function to determine how adequate the given state is for rearrangement planning. Using the above methods, we generate the entire rearrangement order in a practical time.

In the experiment, we consider various target retrieval scenarios using different objects that are difficult to solve using existing TAMP methods. Through these scenarios, we confirm that the proposed planning method can generate a safe and efficient rearrangement sequence by considering both the position and orientation of objects. Furthermore, we conduct an ablation study for reward function and experiment on cases where the target is completely occluded. In addition,

M. Kang, J. Kim, H. Kee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: {minjae.kang, junseok.kim, hogun.kee}@rllab.snu.ac.kr, songhwai@snu.ac.kr).

This work was supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2022-0-00480, Development of Training and Inference Methods for Goal-Oriented Artificial Intelligent Agents). *(Corresponding author: Songhwai Oh.)*

we verify that the neural network trained using depth images obtained from simulation can be successfully transferred to the real environment by performing a real robot experiment.

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

- We propose a novel planning algorithm that solves the target retrieval task, considering the placement within a confined space through MCTS.
- By combining the learning-based approach with TAMP, we plan rearrangement sequences that consider both the location and orientation of various objects.
- We utilize the point cloud-based simulation and the task-specific reward function to perform MCTS.
- We transfer the trained network from simulation to a real robot without additional training.

II. Related Work

A. Object Rearrangement in a Lateral Access Environment

Recently, object rearrangement tasks in a lateral access environment have been studied. Cheong et al. [12] propose a TAMP algorithm that finds placement locations called valid slots within the polynomial-time to relocate objects for target retrieval. Lee et al. [13] plan object rearrangement orders with prehensile and non-prehensile actions through tree search using the A* search. However, the above target retrieval methods use only cylinders and assume that they know both the size and position of objects. We implement [13] as a baseline and perform object rearrangement using only prehensile actions for performance comparison.

B. Monte Carlo Tree Search in Manipulation Tasks

MCTS is often used to plan a long horizon action sequence in robot manipulations. Huang et al. [5], [15] separate the table-top clutter using push actions for object retrieval. They plan the optimal action sequence through MCTS combined with the grasp network and push prediction network. Song et al. [16] perform a sorting task on a table using push actions. They use MCTS to explore the optimal order using task-specific rewards while minimizing the number of actions.

Our previous work [10] solved target retrieval in a lateral environment. Similar to the proposed method, we planned object grasp orders through point cloud-based MCTS. However, since we assumed that there was enough space to hold relocated objects, we found the grasp order of objects without considering the placement. Unlike [10], the problem covered in this paper is performed within a confined space, so we also need to plan where to place the relocated objects.

III. Problem Statement

In this paper, we solve the target retrieval problem in a confined space by relocating the surrounding objects. For safe rearrangement, we only perform pick-and-place actions. We assume that there are N objects in the limited workspace $\mathcal{W}$. For practical rearrangement planning, we divide $\mathcal{W}$ into grids, and then relocate objects to the center of each grid. We make G_x and G_y grids for the x-axis and the y-axis, respectively, so a total of $G_x \times G_y$ grids are generated. When performing pick-and-place, the agent can only access an object at predetermined angles. We denote a set of possible approach angles as Θ. Finally, we want to create a collision-free path with an approaching direction $\theta \in \Theta$ to the target by rearranging objects with as few actions as possible while minimizing the number of fallen objects due to collisions.

IV. Rearrangement Plan for Target Retrieval

The proposed algorithm, **T**ree **S**earch with **A**pproaching **D**irection (TSAD), plans object rearrangement sequences for safe target object retrieval in a limited space. However, a non-monotone rearrangement problem has a vast search space, and it is usually impractical to explore this space by tree search. To solve this problem, TSAD divides the target retrieval problem into sub-problems, preventing excessive increases in the depth of tree search.

TSAD is performed in two stages as shown in Figure 2. First, TSAD finds the optimal direction to retrieve the target object. The optimal direction can be obtained using existing studies on grasp order in the lateral environment [10], [11]. And then, for the obtained angle, we determine objects that can block a clear access to the target object. These objects are considered as obstacles that must be relocated. Second, TSAD plans a rearrangement sequence for generating a collision-free path for the optimal direction through the sequential sub-MCTS. Initially, we need to find a long sequence to retrieve the target, but TSAD divides it into short plans and explores each path through sub-MCTS. The subgoal of each MCTS is to relocate at least one object currently blocking the target at the optimal direction. TSAD sequentially performs sub-MCTS to plan orders from the initial state or terminal state found in the previous tree search to solve the current sub-problem. These successive tree searches have loop invariant and eventually reach a state where the target can be grasped as analyzed in Section IV-C.

A. Smallest Rearrangement Set with Grasp Direction

Before performing a tree search, we first predict the optimal direction for retrieving the target using GP3 [10]. Because GP3 solves the monotone problem, which removes objects sequentially without considering the placement, it can efficiently find the optimal object grasp order and direction to reach the target. In this paper, we denote the optimal angle obtained through GP3 by θ^*.

To safely perform object rearrangement in a confined space, we utilize the prehensile decision network of GP3 to predict grasp results. The prehensile network ψ, consisting of two abstract modules of PointNet++ [17], predicts whether a collision between the target object and the other objects will occur when approaching the target with a direction θ. ψ receives the point cloud observation rotated by θ as input and performs binary classification for collision prediction.

Furthermore, we generate the smallest rearrangement set $\mathcal{R}_{\theta,s}$ using ψ. $\mathcal{R}_{\theta,s}$ is a set of objects that collide with the target with the approach angle θ in state s. If the agent tries to retrieve the target with a direction θ, all objects in $\mathcal{R}_{\theta,s}$ must be moved. After generating $\mathcal{R}_{\theta^*,s}$ for the optimal direction θ^*, we perform the sequential sub-MCTS based on $\mathcal{R}_{\theta^*,s}$.

B. Subgoal-Conditioned Tree Search for Target Retrieval

In this section, we plan rearrangement orders for retrieving the target object with the optimal approaching angle θ^* and the smallest rearrangement set $\mathcal{R}_{\theta^*,s}$ from the current state s. With the simple idea of sequentially reducing the size of the smallest rearrangement set $\mathcal{R}_{\theta^*,s}$, TSAD replaces the original tree search with a sequential small-scale tree search. By replacing an extensive tree search with several shallow tree searches, we significantly limit the exponentially growing search space as the tree search depth increases.

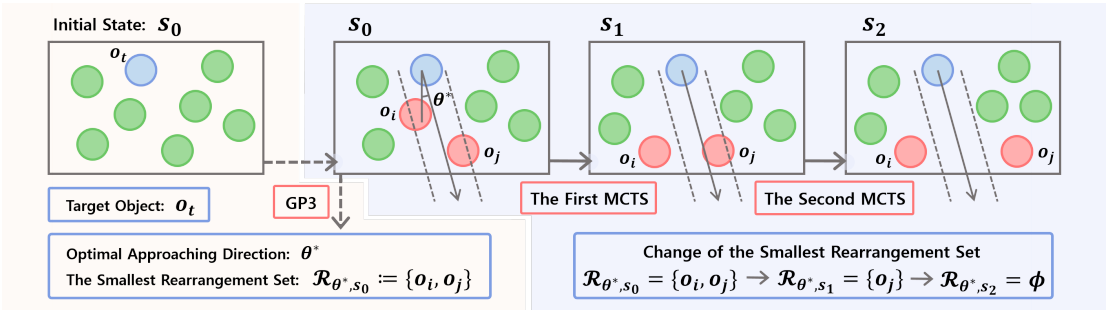


Fig. 2: Overview of TSAD. (*red*) First, TSAD obtains the optimal direction to the target object and the smallest rearrangement set accordingly. (*blue*) Second, TSAD gradually reduces the smallest rearrangement set through the sub-MCTS.

The sub-problem of target retrieval is to reduce the size of $\mathcal{R}_{\theta^*,s}$. Each sub-MCTS plans rearrangement orders with $\mathcal{R}_{\theta^*,s}$ until reaching a terminal state s' that satisfies the following two termination conditions:

1. $\exists o \in \mathcal{O}:\ o \in \mathcal{R}_{\theta^*,s}$ *and* $o \notin \mathcal{R}_{\theta^*,s'}$,
2. $\forall o \notin \mathcal{R}_{\theta^*,s}:\ o \notin \mathcal{R}_{\theta^*,s'}$,

where $\mathcal{O}$ is the set of objects. If sub-MCTS finds a sequence that reaches a terminal state s', TSAD takes s' as a new root node and performs the next sub-MCTS. The sequential tree search is performed until $\mathcal{R}_{\theta^*,\cdot}$ becomes an empty set. It means that there is no object colliding with the target object to the optimal angle θ^*, so we can grasp the target at θ^*.

C. Analyses of Subgoal-Conditioned Tree Search

In this section, we theoretically describe the sufficient condition for TSAD to find a solution. First, we prove that the solution of each sub-MCTS exists if there is a valid rearrangement sequence for the given direction.

Theorem 1. *If a valid rearrangement plan for target retrieval at direction θ exists, then there is a feasible solution for the sequential sub-MCTS of TSAD.*

Proof. Let the pick-and-place sequence $\tau_\theta = \{p_1, ..., p_M\}$ be a valid rearrangement plan at direction θ. Each action $p = \{o_i, x, y\}$ contains an object o_i to be relocated and placement coordinate (x, y). We define a set $\mathcal{S} = \{s_0, ..., s_M\}$, where s_0 is the initial state and state s_j is the state after performing action p_j in state s_{j-1}. τ_θ must include all actions for each object in the smallest rearrangement set $\mathcal{R}_{\theta,s_0}$. Using τ_θ, we first show the existence of solution for the first sub-MCTS of TSAD, which is started from s_0. We define a set of states in $\mathcal{S}$ that satisfy the termination conditions of the first MCTS as $\mathcal{S}_1$. Since the last state s_M satisfies termination conditions of all tree searches ($\because \mathcal{R}_{\theta,s_M} = \emptyset$), $\mathcal{S}_1$ is not an empty set, and let s_{t_1} be the first state of $\mathcal{S}_1$. The action sequence from s_0 to s_{t_1} exists in τ_θ, i.e., $\{p_1, ..., p_{t_1}\}$ is feasible, and s_{t_1} satisfies the termination conditions of the first MCTS, the first MCTS has at least one valid solution.

In a similar way, we can show that a solution exists in all subsequent subgoal-conditioned tree searches. Let s_i be the state in which the i-th MCTS starts. By the same reason for $\mathcal{S}_1$, $\mathcal{S}_i$, which is a set of states in $\mathcal{S}$ satisfying the termination conditions of the i-th tree search, is not empty. We now show that there is a valid action sequence from s_i to s_{t_i}, the first state of $\mathcal{S}_i$. First, since the pick-and-place action we use is reversible, there is a valid sequence from s_i to s_0. Furthermore, τ_θ includes a sequence from s_0 to s_{t_i}, i.e., $\{p_1, ..., p_{t_i}\}$. Therefore, there is a valid sequence from s_i to s_{t_i}, that is, a solution of the i-th MCTS exists. □

Since there is an action sequence satisfying the termination conditions of each sub-MCTS, every tree search has at least one solution. Therefore, TSAD can find the valid entire sequence through the sequential sub-MCTS. Next, we prove that if the sequential sub-MCTS is performed successfully, TSAD reaches the state in which the target can be grasped.

Theorem 2. *If each sub-MCTS of TSAD is performed successfully, the following loop invariant about the size of the smallest rearrangement set $\mathcal{N}$ is satisfied at the beginning of each tree search: Let $\mathcal{N}_0 = N$ and $\mathcal{N}_i = \mathcal{C}(\mathcal{R}_{\theta^*,s_i})$, $\forall i \in \{1, 2, ..., T\}$, where N is the number of objects in the environment, T is the number of tree searches of TSAD, $\mathcal{C}(A)$ is the cardinality of set A, and state s_i represents the initial state of the i-th tree search. Then, $\mathcal{N}_i < \mathcal{N}_{i-1}$ is true for all $i \in \{1, 2, ..., T\}$.*

Proof. **Initialization**: Since the smallest rearrangement set $\mathcal{R}$ does not contain the target, the size of all the smallest rearrangement sets is less than the total number of objects N. Therefore, $\mathcal{N}_1 < N = \mathcal{N}_0$ is established. **Maintenance**: Due to the second termination condition, a new object cannot be included in $\mathcal{R}$, so these sets cannot be increased. Moreover, by the first termination condition, at least one object is excluded from $\mathcal{R}$. Therefore, $\mathcal{N}_i \leq \mathcal{N}_{i-1} - 1 < \mathcal{N}_{i-1}$ is established. **Termination**: The sequential sub-MCTS is terminated if $\mathcal{R}$ becomes an empty set. That is, at the beginning of the last tree search, at least one object exists in $\mathcal{R}$. Therefore, $\mathcal{N}_{T+1} = 0 < \mathcal{N}_T$ is established, where $\mathcal{N}_{T+1}$ is the size of $\mathcal{R}$ after the last tree search is completed. □

In other words, as the sequential sub-MCTS progresses, the size of the smallest rearrangement set decreases, and eventually, the smallest rearrangement set becomes empty, i.e., TSAD can grasp the target. In summary, if a valid action sequence exists for angle θ, the preceding theorems guarantee that the sequential sub-MCTS performs successfully so that TSAD generates a rearrangement order for target retrieval.

D. Implementation Details for Monte Carlo Tree Search

This section describes the process details of each sub-MCTS in TSAD to find the order of object rearrangement. First, in sub-MCTS, the state s is a point cloud observation $\{o_1^{x_1,y_1,\theta_1}, ..., o_N^{x_N,y_N,\theta_N}\}$, which means object o_i is placed at the coordinate (x_i, y_i) with orientation θ_i as shown in Figure 3. Next, a pick-and-place operation p is represented as $\{n, \theta^{pick}, x, y, \theta^{place}\}$, which consists of the object index

$n \in \{1, ..., N\}$, the angle $\theta^{pick} \in \Theta$ to be picked, the two-dimensional coordinate $(x, y) \in \mathbb{R}^2$ of the position to be relocated, and the angle $\theta^{place} \in \Theta$ to be placed. Among the elements of p, $\{n, x, y\}$ are selected by MCTS, and $\{\theta^{pick}, \theta^{place}\}$ are determined through the prehensile decision network using a given state s and next state s', respectively. Accordingly, the number of actions in MCTS for the state in which N objects are observed is at most $N \times G_x \times G_y$. In summary, if we select an action $a = \{n, x, y\}$ through MCTS and perform a pick-and-place $p = \{n, \theta^{pick}, x, y, \theta^{place}\}$, the object o_n in the next state is expressed as $o_n^{x,y,\theta^{place}}$.

Sub-MCTS sets the current state as the root node and seeks for the optimal path to reach a state satisfying the termination conditions. Each MCTS consists of the following four steps.

Selection: To judge whether the given state s is promising for the object rearrangement problem, we define task-specific state reward $R(\cdot)$ and state value $V(\cdot)$ as follows:

$$\begin{aligned} R(s) &= r_t(s) + w_o r_o(s) + w_e r_e(s), \\ V(s) &= \max\left[R(s), \max_{s'} V(s') - \alpha\right], \end{aligned} \tag{1}$$

where s' indicates the child state of s, and α is a positive constant for step penalty. The state reward $R(s)$ is calculated as the weighted sum of the following three task-specific reward components with the given state s. First, $r_t(s)$ indicates whether s satisfies termination conditions of the tree search. If all conditions are satisfied, it has a value of 1, otherwise 0. Next, $r_o(s)$ is a negative reward to prevent the occurrence of new obstacles blocking the target object. If s violates the second termination condition, $r_o(s)$ becomes -1, otherwise 0. Lastly, $r_e(s)$ checks the number of empty spaces to make s better to rearrange. To reduce the computational cost of r_e, we do not use the prehensile decision network to determine whether it is empty, but count the number of empty grids ϵ_s in which no object exists within d_{min}, the predefined minimum distance. We define $r_e(s)$ to be proportional to the increased number of empty spaces compared to the initial state s_0 (i.e., $r_e(s) \propto \epsilon_s - \epsilon_{s_0}$). At the initial value of 0, the value increases by 0.1 each time ϵ_s increases, and the maximum value is 1. To consider the importance of each reward component, we set components weights w_o and w_e less than 1. We experiment a set of ablation studies to confirm the effect of each component in Section VI-C.

In the selection step, we select next action using the state-action value $Q(\cdot, \cdot)$ modified from upper confidence bounds (UCB) formula as follows:

$$Q(s, a) = V(s') + c\sqrt{\frac{2 \log N_v(s)}{N_v(s')}},$$

where s is the current state, a is the selected action, s' is the next state after performing a in s, $N_v(s)$ is the number of visits to s, and c is the exploration coefficient. We choose the action with the highest state-action value.

Expansion: When we reach a new node that is not explored by a tree search, we add the corresponding node to the tree. To estimate the value of the expanded node, we perform a simulation step to measure the state value.

Simulation: To determine the state value $V(\cdot)$ of a leaf node, rollout is performed until the tree search depth becomes the maximum depth d_{max} we define. In the simulation step, an action is selected by a random policy and the selected action is judged for its feasibility using the prehensile decision network.

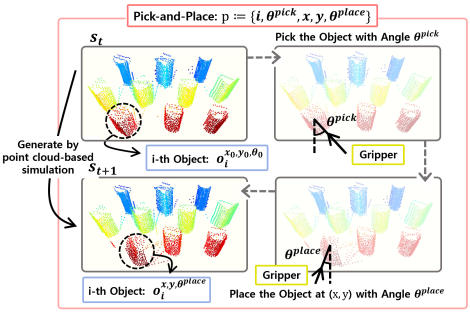


Fig. 3: Next state generation after performing the pick-and-place action using point cloud-based simulation.

Backpropagation: When one search process ends through a selection or simulation step, we update the state value $V(\cdot)$ of all passed states. The state values are adjusted using (1) while reversing the searched trajectory from the leaf node to the root. After the backpropagation step, we return to the root node and perform tree search iteratively.

We perform MCTS by repeating the above process by a fixed number of times. After the tree search, we find an optimal rearrangement plan by sequentially selecting the child nodes with the highest state value, starting with the root node, which represents the current state.

Since we perform the robotic task in a partially observable environment, there may be occluded objects. Therefore, we do only planned rearrangement sequence from the first sub-MCTS at each robot implementation. To achieve efficient planning, we perform only the first sub-MCTS instead of finding the entire plan. After executing planned actions, TSAD repeats MCTS until the target object can be grasped.

V. Point Cloud-Based Simulation

In general, to perform MCTS, we need to obtain the future state when an action is performed in a given state. However, implementing actions using a robot is time-consuming, making tree searches impractical. Therefore, we propose a novel point cloud-based simulation that replaces the complex robotics simulation. The point cloud-based simulation infers derivable states from point cloud observation obtained from lateral views. Although GP3 similarly performs tree search by processing point cloud observations, their simulation is much simpler because it simply removes objects sequentially.

Using the point cloud-based simulation, we generate the observation where objects are relocated and rotated after performing pick-and-place as shown in Figure 3. First, through point cloud processing of GP3 and SD Mask R-CNN [18], a method for segmenting objects from a depth image, we express a state s in which N objects are observed as a point cloud observation $\{o_1^{x_1,y_1,\theta_1}, ..., o_N^{x_N,y_N,\theta_N}\}$. If we perform pick-and-place $\{i, \theta^{pick}, x, y, \theta^{place}\}$, we can generate next scene with the converted i-th object $o_i^{x,y,\theta^{place}}$.

VI. Experiment

A. Simulation Setup

In this experiment, we model the lateral access environment using the MuJoCo [19] simulation framework, robosuite [20], for performing the target retrieval task as shown

in Figure 1. We solve the object rearrangement problem within a 32×50 shelf using a UR5e robot. The agent obtains 128×128 depth images using an arm-attached camera. Depth images are converted into point clouds through Open3D library [21]. The agent observes and approaches objects at seven angles separated by 10° from -30° to 30°.

Similar to the experiments in GP3, we experiment on three object arrangement scenarios with 10 objects as shown in Figure 4. Scenario 1 consists of cylinders and square pillar objects that can be grasped at any approach direction. Scenario 2 adds rectangular cylinders to Scenario 1 that can be picked at limited directions. Therefore, we have to plan a rearrangement sequence considering the object shape. Scenario 3 adds a large non-prehensile object to Scenario 1. To solve this scenario, we must predict that the large object cannot be moved, and plan the rearrangement order that does not include this object.

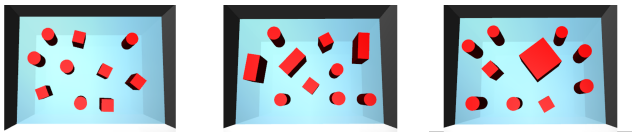
B. Performance Comparison with Baselines

To compare the performance of TSAD, we implement ROTS [13] and GP3 [10] as baselines. ROTS is a TAMP method that uses modified vector field histogram+ [22] to find collision-free approach directions and plans rearrangement actions through tree search. Originally, ROTS utilizes both prehensile and non-prehensile actions, but we only use pick-and-place in this experiment. Since ROTS assumes that all objects are cylinders, it is difficult to apply to Scenario 2 and 3. Therefore, we implement ROTS*, which determines whether a sequence generated by ROTS is feasible using the prehensile decision network of TSAD. On the other hand, GP3 is a learning-based method that performs point cloud-based MCTS, similar to TSAD. However, GP3 only determines the object grasp order and does not plan where to place them. In the case of GP3, we find the grasp order with GP3 and then place the object in the location with the highest number of remaining empty spaces after placement.

The performance comparison results are shown in Table I. We have performed 30 episodes for each scenario and measured the success rate and the average number of pick-and-place actions and fallen objects. We list the averages for successful episodes and for all episodes, respectively. The plan is failed if the target falls, an object falls out of the workspace, or the algorithm fails to plan the rearrangement order. First, GP3 has low success rates for all scenarios. This means that there is not enough empty space in the initial state, and the agent should create more empty space for the successful target retrieval. Next, ROTS shows low success rates and high object fall counts for Scenario 2 and 3, as expected. This is because ROTS plans the rearrangement sequence without considering the shape of objects. To solve this, ROTS* uses the prehensile decision network to predict grasp results. However, since rearrangement planning and the collision prediction process are performed separately, ROTS* is difficult to generate an efficient rearrangement order. On the other hand, TSAD shows high success rates with smaller fall counts for all scenarios. This means that TSAD can successfully plan rearrangement sequences by considering the location to place and the approaching direction.

C. Ablation Study for Reward Function

We perform an ablation study to confirm the importance of each reward component of the state reward function. The



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

Fig. 4: Object arrangement scenarios with different objects.

(a) Target retrieval results of Scenario 1

Algorithm	TSAD	GP3	ROTS	ROTS*
Success Rate	0.97	0.33	0.87	0.87
Avg. of Actions	3.72 / 3.67	2.90 / 3.07	3.96 / 4.07	4.12 / 4.33
Avg. of Falls	0.17 / 0.23	0.10 / 0.13	0.31 / 0.30	0.19 / 0.33

(b) Target retrieval results of Scenario 2

Algorithm	TSAD	GP3	ROTS	ROTS*
Success Rate	0.93	0.17	0.63	0.70
Avg. of Actions	3.96 / 4.23	3.20 / 3.47	4.42 / 4.70	4.95 / 5.10
Avg. of Falls	0.21 / 0.27	0.20 / 0.27	0.74 / 0.90	0.24 / 0.23

(c) Target retrieval results of Scenario 3

Algorithm	TSAD	GP3	ROTS	ROTS*
Success Rate	0.93	0.20	0.67	0.70
Avg. of Actions	4.71 / 5.00	3.17 / 3.83	5.40 / 5.73	5.81 / 6.07
Avg. of Falls	0.29 / 0.33	0.17 / 0.23	0.80 / 1.10	0.33 / 0.37

TABLE I: Performance comparison with baselines. For the average number of actions and the average number of falls, the first number is the average from successful episodes and the second number is the average from all episodes. This rule applied to all tables in this paper.

Algorithm	TSAD	w/o r_o	w/o r_e
Success Rate	0.97	0.93	0.90
Avg. of Actions	3.72 / 3.67	4.04 / 4.07	4.30 / 4.33
Avg. of Falls	0.17 / 0.23	0.21 / 0.27	0.22 / 0.23

TABLE II: Ablation study results for reward function.

reward r_s is an essential component that checks whether the termination conditions of the sub-MCTS are satisfied, whereas the two reward elements, r_o and r_e, are intuitive rewards to aid the tree search. Therefore, we experiment on the effects of r_o and r_e. We set w_o and w_e to 0.5 and 0.3.

We have performed 30 episodes for Scenario 1, and the results are in Table II. First, TSAD using all components shows the best success rate and the lowest number of prehensile actions. Next, in the absence of r_o, the success rate decreased by 0.04, and on average, 0.32 more actions are performed compared to TSAD. It shows that r_o helps MCTS find the optimal sequence by penalizing unnecessary actions. Last, when performed without r_e, the success rate decreased by 0.07, and on average, 0.58 more actions are performed compared to TSAD. r_e makes the tree search perform efficiently by inducing empty spaces to increase.

D. Case Study for Complete Target Occlusion

To consider the general situation, we perform target retrieval for object arrangements where the target is completely occluded in Scenario 1 and 2. Since we cannot obtain the optimal approaching angle in these cases, we randomly rearrange the objects until the target is partially observed. After the target is detected, we can apply TSAD. We conduct 15 episodes for each scenario and the results are shown in

TSAD	Scenario 1	Scenario 2
Success Rate	0.93	0.93
Avg. of Random Actions	3.79 / 3.93	4.07 / 4.00
Avg. of Actions	4.21 / 4.13	4.43 / 4.47
Avg. of Falls	0.14 / 0.27	0.21 / 0.27

TABLE III: Target occlusion experiment results.

Fig. 5: Real environment with practical objects using a UR5 robot and an Intel Realsense D435 camera.

Algorithm	TSAD	GP3	ROTS*
Success Rate	0.87	0.13	0.60
Avg. of Actions	3.69 / 3.47	3.00 / 1.93	4.22 / 3.93
Avg. of Falls	0.31 / 0.40	0.00 / 0.13	0.33 / 0.33

TABLE IV: Real robot experiment results.

Table III. The results show that TSAD successfully solves the problem where the target is completely blocked.

E. Real Robot Experiment

We solve the target retrieval problem in a real environment. As shown in Figure 5, we use a UR5 robot with a Robotiq 2-Finger 85 gripper and obtain depth images with an Intel Realsense D435 camera. The task is performed within a $40cm \times 28cm$ shelf with practical objects. We place five objects for each episode and perform five episodes for each scenario, totaling 15 episodes. The real experiment details can be confirmed through the supplementary video. Note that, in this experiment, we use the prehensile decision network trained in simulation without additional training.

The results of real experiment are shown in Table IV. First, GP3 fails to retrieve the target in most cases in a limited space where there is not enough space for placement. Also, ROTS* shows low efficiency and often fails to plan. However, TSAD performs rearrangement tasks safely and efficiently in the real environment. This result shows that TSAD can successfully perform the target retrieval task using practical objects as well as transfer networks trained from simulation to the real environment.

VII. Conclusion

We have proposed TSAD which combines a learning based approach with TAMP and demonstrated that TSAD efficiently solves the target retrieval problem under occlusion within confined spaces. TSAD plans the rearrangement sequence that includes both the placement positions and the approaching angles using lateral views via MCTS using the point cloud-based simulation. Since TSAD considers both the location and the shape of objects, it can be applied to various object arrangement problems. Furthermore, TSAD can be successfully applied to real-world problems without additional training through real robot experiments.

References

[1] A. Boularias, J. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[2] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6232–6238.

[3] 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.

[4] 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.

[5] 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.

[6] X. Lou, Y. Yang, and C. Choi, "Collision-aware target-driven object grasping in constrained environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6364–6370.

[7] B. Wen, W. Lian, K. Bekris, and S. Schaal, "Catgrasp: Learning category-level task-relevant grasping in clutter from simulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6401–6408.

[8] P. K. Murali, A. Dutta, M. Gentner, E. Burdet, R. Dahiya, and M. Kaboli, "Active visuo-tactile interactive robotic perception for accurate object pose estimation in dense clutter," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4686–4693, 2022.

[9] G. Wilfong, "Motion planning in the presence of movable obstacles," in *Proceedings of the fourth annual symposium on Computational geometry*, 1988, pp. 279–288.

[10] M. Kang, H. Kee, J. Kim, and S. Oh, "Grasp planning for occluded objects in a confined space with lateral view using monte carlo tree search," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[11] 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.

[12] 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.

[13] 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.

[14] J. Ahn, C. Kim, and C. Nam, "Coordination of two robotic manipulators for object retrieval in clutter," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1039–1045.

[15] B. Huang, T. Guo, A. Boularias, and J. Yu, "Interleaving monte carlo tree search and self-supervised learning for object retrieval in clutter," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 625–632.

[16] 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.

[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] 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.

[19] 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.

[20] 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.

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

[22] 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.