Efficient Target Singulation with Multi-fingered Gripper using Propositional Logic

Hyojeong Kim[1,2], JeongYong Jo[1,3], Myo-Taeg Lim[2] and ChangHwan Kim[1,*]

***Abstract*— When multiple tablewares are closely packed on a table, rearranging obstacles to make space is necessary to grasp the target, often called target singulation. Due to the nature of handling fragile tablewares (i.e. plates, bowls), we make a few assumptions for the target singulation problem. First, tableware is grasped with a multi-fingered gripper; second, rearrangement is based on prehensile motions like pick-and-place. Under these assumptions, we aim to generate a relocation plan that guarantees global optimality. Furthermore, if any relocation plan cannot singulate the target, we aim to determine it quickly. Therefore, we propose a search method that utilizes the relationship between the object and its nearby obstacles expressed in propositional logic. We define the problem as determining logical entailment (i.e., whether the target can be singulated) and expand the search tree from the target while generating an optimal relocation plan. We demonstrate the performance of our algorithm by increasing the number of objects and validate the plan in a simulation environment.**

I. INTRODUCTION

Manipulation planning in clutter is still an actively researched area. In this paper, we assume a table clearing scenario where a robot clears tableware after people have finished eating. In such scenario, it is easy to imagine a situation where the space between dishes is too narrow for the robot to grasp a target dish. Then, some obstacles should be relocated to make enough space for grasping the target object, which is often described as a *rearrangement problem* for target singulation. Unfortunately, sequentially deciding which object needs to be relocated and where to is known to be NP-hard [1].

Singulation has been focused on utilizing non-prehensile motions such as push. However, non-prehensile motions are not always desirable depending on environmental constraints. For instance, pushing dishes on a table can be considered unsafe since most tableware is fragile, and it is important to ensure that dishes do not leave the table area. However, on a densely cluttered table, the table boundary might be too tight, leaving too little room to push. Moreover, it is difficult to model push dynamics due to the physical characteristics of tablewares. Therefore, precise prehensile motions such as pick-and-place are required for singulating tablewares.

For object rearrangement with prehensile motion, it is necessary to consider spatial requirements to ensure a collision-free grasp. The space required for the gripper varies depending on grasping method, gripper shape, and size. Fig. 1 shows our table-clearing robot with a multi-fingered gripper, designed for handling tablewares with overhead grasping. To grasp an object with such a gripper, we need to consider whether there is enough space to insert the gripper into the clutter without any of the fingers colliding. Let's say the obstacles within the radius of the gripper for a target object are referred to as *local* obstacles. According to the rotational angle of the gripper for grasping the target, the set of obstacles to be relocated can be determined. However, it is not sufficient to only consider the local obstacles, since those obstacles might not be graspable. Therefore, for target singulation, a *global* relocation plan, which takes into account all obstacles on the table, must be generated.

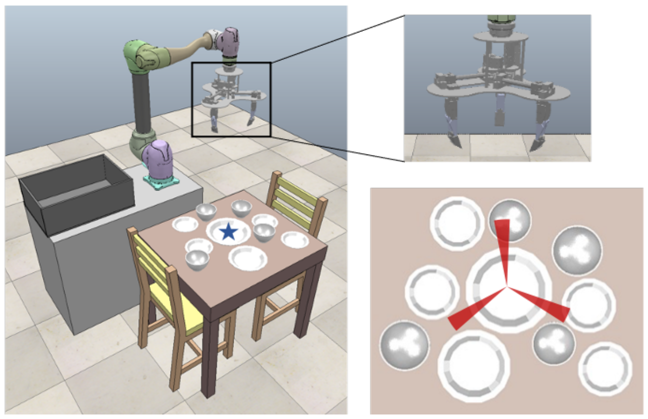


Fig. 1: Left: Table-clearing robot with 3-finger gripper. The table is densely filled with dishes. The target dish (blue star) is not graspable without rearranging the surrounding dishes. Right: An example of a target's local obstacle set, which collides with the gripper (red region). If the ones in the set are also not graspable, then additional obstacles must be relocated in order to make them graspable.

The obstacles to be relocated depend on the rotational angle of the gripper at which the target object is grasped. Therefore, finding a globally optimal relocation plan is difficult. Moreover, it is not always possible to generate a relocation plan using only prehensile motion. For example, consider cases where objects are tightly packed, making it impossible to relocate any object, no relocation plan would enable target singulation, and we refer to these situations as deadlocks. Especially, the more cluttered the table is, the higher the likelihood of occurring deadlock.

To summarize, we aim to solve the target singulation problem by relocating obstacles with prehensile motion using a multi-fingered gripper. There are two main issues to be

[1]Korea Institute of Science and Technology, Seoul, Korea: Emails {hyojeongk82, jjy0607, ckim}@kist.re.kr. [2]Department of Electrical Engineering, Korea University, Seoul, Korea. mlim@korea.ac.kr [3]Department of Interdisciplinary Robot Engineering Systems, Hanyang University, Ansan, Korea. *Corresponding author.

addressed. First, we need to determine if there exists a relocation plan that can singulate the target using prehensile motions. Second, we need to find an optimal plan that can singulate the target with the least number of relocations.

Our key idea to solve the problem is as follows: (1) Convert the target singulation problem to a logical entailment problem, which is assessing the logical validity of singulating the target based on environmental conditions. (2) Use a polar histogram to identify all local obstacle sets for each object and express it in a logical form. (3) Propose a polynomial-time graph search algorithm that employs backward chaining to determine whether a relocation plan exists, while finding a globally optimal relocation plan for target singulation.

II. RELATED WORK

A. Rearrangement on tabletop

Tabletop rearrangement for singulation has been focused on utilizing non-prehensile motion. Cogsun et al. [2] suggested a search-based planner generating a sequence of push actions for making space for the target object to be placed within the table area. However, these model-based approaches require accurate push dynamics. Huang et al. [3] solved various large-scale tabletop rearrangement problems such as singulation, repositioning, and sorting etc. The author suggested an online local search method that iteratively replans after executing each push action. Each local search finds the best push action that minimizes the distance to the goal by evaluating policy rollouts in simulation. Several papers [4], [5] trained a push and grasp policy jointly for singulation using model-free deep reinforcement learning. The grasp policy serves as a discriminator to determine whether the target is graspable, while the push policy greedily selects which object to push and in which direction to make the target graspable. However, it is difficult to find a global optimal plan. Furthermore, the space required for the gripper is often overlooked.

B. Rearrangement on shelves

Rearrangement on shelves involves taking the target object out of a cluttered and confined space (e.g., shelf, fridge). Since they assume cases where the target is obscured by other obstacles, it is important to consider the space needed by the gripper to retrieve the target object without collision. Various search-based methods have been proposed to retrieve the target by minimizing the number of relocating obstacles in the shelve. Each method focuses on reducing the search time by guiding the search in different ways. Lee et al. [6] proposed MVFH+ for verifying the accessibility of the target. By drawing a polar histogram around the target, it recursively determines which obstacles should be removed. Lee et al. [7] also proposed a heuristic search approach to minimize the number of rearrangement actions, considering both prehensile and non-prehensile motion. Nam et al. [8] construct a traversability graph, which represents collision-free path existences between two object poses, and use it to guide the search algorithm. Saxena et al. [9], [10] considered only non-prehensile motion and formulated the problem as a Multi-Agent Pathfinding (MAPF). By decomposing the process into two phases—first finding an abstract MAPF plan and then motion planning—they were able to find the rearrangement plan efficiently.

III. PROBLEM DESCRIPTION

We assume a densely cluttered environment where there may not exist a collision-free grasp for the target object. To grasp the target without collision, it is necessary to relocate surrounding obstacles to create enough space for grasping the target. We define a tabletop rearrangement problem for target singulation under the condition of relocating within a table area with prehensile motion using k-finger gripper.

The major assumptions of this problem are as follows: (i) All objects have a circular shape. The configuration of all objects (i.e., pose and size) is known. (ii) There is always free space available within the table area to relocate the object, and stacking on top of another object is not allowed. (iii) The gripper has a symmetrical structure composed of k fingers, with uniform angles between fingers $\theta_k = 360/k$. (iv) Only overhead grasping is allowed.

An environment is configured with N objects including target: $O = \{o_1, o_2, ..., o_{N-1}, t\}$. The goal is to determine whether there exists a global relocation sequence O_s that enables the grasping of the target object t, and if O_s exists, to minimize the number of relocations. More formally, we'd like to determine $KB \models t$ while finding $\min |O_s|$. We represent 'An object is graspable' as 'Corresponding literal is True'. Therefore, determining whether t is entailed by KB is equivalent to determining whether the target can be singulated enough to grasp.

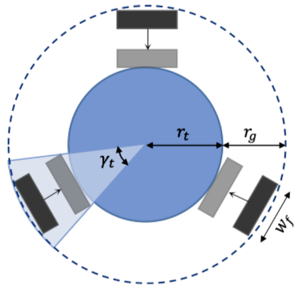
We employ graph search to solve this problem. The search tree is constructed by expanding the local obstacle set of the parent object as children. Each node N consists of three components: (1) O_c, objects to be relocated for current node N, which is in the form of *conjunctive clause*. (2) O_s^R, list of objects to be relocated until node N, which is a reverse order of relocation sequence. (3) $cost$, the cost of relocation sequence from the root node to node N, which we defined as the length of O_s.

IV. METHOD

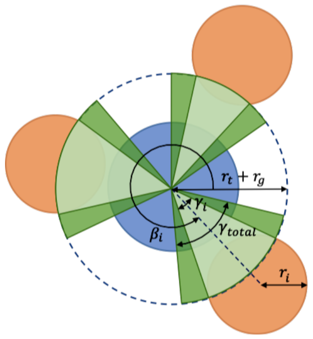
We propose a method for computing an efficient relocation plan for target singulation, which consists of the following three steps. (1) Identify the local obstacle set for each object using TG-MVFH+ and construct a knowledge base from it. (2) Find the optimal relocation sequence for target singulation while determining whether the target can be singulated via relocation by utilizing a backward chaining style graph search. (3) Determine the relocating position for each object in the sequence obtained from (2).

A. Identify local obstacle set with TG-MVFH+

In this section, we describe a modified version of MVFH+ for k-fingered Top Grasping, which we call TG-MVFH+. From the modified polar histogram, we identify the local obstacle set for a target object and represent it in propositional

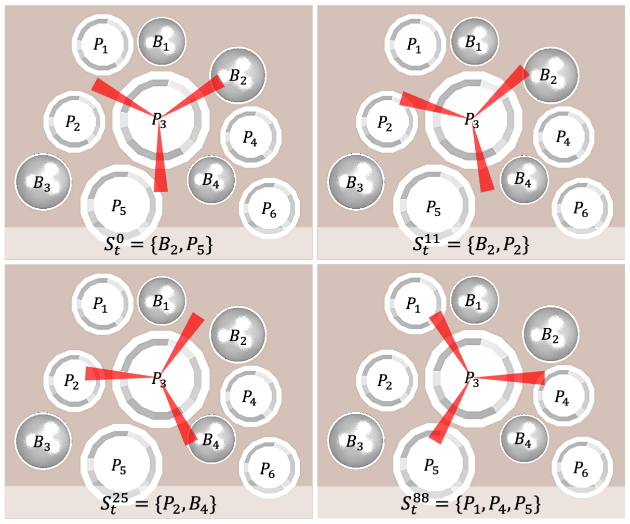


(a) The angle occupied by a finger denoted as γ_t, when grasping the target with a radius of r_t using 3-finger gripper.



(b) The angle occupied by object i denoted as γ_i and the angle γ_{total}, which is expanded by the gripper.

Fig. 2: (a) shows the gripper closing after approaching the target with an additional radius r_g. In (b), All obstacles(orange) within the radius r_g from the target(blue) are uniformly enlarged by $\gamma_t/2$ on both sides.



P_1
B_1
B_2
P_2
P_3
P_4
B_3
B_4
P_5
P_6
$S_t^0 = \{B_2, P_5\}$
$S_t^{11} = \{B_2, P_2\}$
$S_t^{25} = \{P_2, B_4\}$
$S_t^{88} = \{P_1, P_4, P_5\}$

$(B_2 \cap P_5) \cup (B_2 \cap P_2) \cup (P_2 \cap B_4) \cup (P_1 \cap P_4 \cap P_5) \dots \Rightarrow P_3$

Fig. 3: An example of the local obstacle set for target t (Above) and its representation as a propositional logic (Below). The four figures above show the local obstacle set S_t^z with different z values. S_t^z refers to the set of objects that must be relocated to grasp the target in the rotational angle z. The target becomes graspable if at least one of S_t^z is relocated. Therefore, it can be expressed as an implication sentence with a premise in DNF and the target as below.

logic. Then, we obtain a Knowledge Base by determining the local obstacle sets for all objects.

MVFH+ was proposed to determine the approach direction of the manipulator for retrieving the target object located in a cluttered shelf [6]. To generate a collision-free motion plan for grasping a target, a polar histogram is computed by the density of surrounding obstacles with consideration of physical constraints. In [6], the author assumed lateral grasping for taking the target object out from the cluttered shelf and generated a polar histogram accordingly. On the other hand, we assume overhead grasping with a k-finger gripper for the tabletop rearrangement problem. Therefore, we modify the process of generating a polar histogram accordingly and introduce an additional process for identifying the local obstacle set from the polar histogram.

We denote the required radius for the gripper to grasp an object as r_g and the width of the finger as w_f as shown in Fig. 2a. Then, we can determine the angle γ_t of the area occupied by a single finger when grasping a target t with a radius of r_t. For a target t, we only compute the polar histogram of local obstacles, which refers to objects within the r_g area around t. Fig. 2b shows each local obstacle o_i with radius r_i, located at a distance d_i from the target. The angle γ_i of the area occupied by each of local obstacles o_i can also be defined. To prevent collision between the gripper and o_i when grasping t, γ_i needs to be enlarged by γ_t, which is the amount of angle that the gripper requires.

The polar histogram refers to the obstacle density for each angular sector around the target. For each angular sector z, ranging from 0 to 360 deg, the polar histogram of o_i for target o_j is computed as follows.

$$H_i^j(z) = \begin{cases} (c_i)^2(a - bd_i^2), & \text{if } z \in [\beta_i - \frac{\gamma_{total}}{2\alpha}, \beta_i + \frac{\gamma_{total}}{2\alpha}] \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where c_i is the coefficient that indicates the certainty of sensor measurements, β_i represents the angle from the target to o_i. This paper assumes environments with definite measurements, so c_i is set to 1. α denotes the angular resolution, which describes the degree to divide the sectors of the polar histogram. In this paper, α is set to 1, meaning that the magnitude of vector is calculated for every degree. The constants a, b are defined in Eq. 8.

Furthermore, for a collision-free grasp of the target object, it is crucial that all of the k fingers simultaneously avoid colliding with other obstacles. The angles between k fingers are uniform, with $\theta_k = 360/k$, by our assumption. To consider all k directions at the same time, we aggregate k histograms with θ_k intervals. Therefore, the polar histogram of o_i for target o_j, considering k fingers, is as follows.

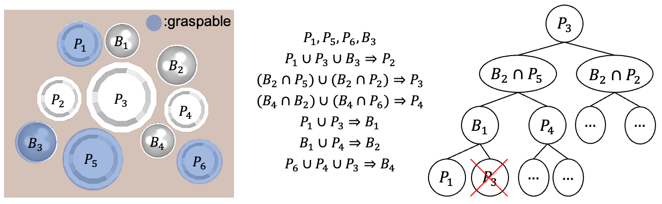
$$G_i^j(z) = \sum_{m=0}^{k-1} H_i^j(z + m \times \theta_k), \text{ for } z \in [0, \theta_k] \tag{2}$$

We can identify which obstacles are blocking a specific angular sector z from the aggregated histogram of local obstacles O''. Therefore, when the target object is o_j, the local obstacle set for an angular sector z can be defined as follows. $I_{O''}$ indicates the index of local obstacles for target.

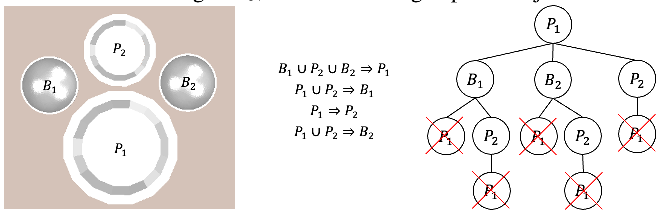
$$S_j^z = \{o_i | G_i^j(z) > 0, \forall i \in I_{O''}\} \tag{3}$$

We define a set of all possible local obstacle sets, S_j, which comprises S_j^z for all angular sectors. S_j does not contain duplicates sets S_j^z with different z.

$$S_j = \{S_j^z | z \in [0, \theta_k]\} \tag{4}$$


:graspable
P_1 B_1 B_2 P_2 P_3 P_4 B_3 P_5 B_4 P_6
P_1, P_5, P_6, B_3
$P_1 \cup P_3 \cup B_3 \Rightarrow P_2$
$(B_2 \cap P_5) \cup (B_2 \cap P_2) \Rightarrow P_3$
$(B_4 \cap B_2) \cup (B_4 \cap P_6) \Rightarrow P_4$
$P_1 \cup P_3 \Rightarrow B_1$
$B_1 \cup P_4 \Rightarrow B_2$
$P_6 \cup P_4 \cup P_3 \Rightarrow B_4$
P_3
$B_2 \cap P_5$ $B_2 \cap P_2$
B_1 P_4 ... ...
P_1 P_3 ... ...

(a) An example where a relocation plan exists; In the search tree started from the target P_3, it reaches the graspable object P_1.


B_1 P_2 B_2 P_1
$B_1 \cup P_2 \cup B_2 \Rightarrow P_1$
$P_1 \cup P_2 \Rightarrow B_1$
$P_1 \Rightarrow P_2$
$P_1 \cup P_2 \Rightarrow B_2$
P_1
B_1 B_2 P_2
P_1 P_2 P_1 P_2 P_1
P_1 P_1

(b) An example where a relocation plan does not exist; In the search tree started from the target P_1, all nodes are removed due to duplication in the path, leaving no more nodes to explore.

Fig. 4: Environment, Knowledge Base, Search Tree (from left to right) for both (a) and (b). During the expansion of the search tree, if duplicate objects exist in the path, the corresponding node, indicated by an x mark, is removed.

Algorithm 1: Local Obstacles-based Backward Search (LOBS)

```
   Input: Knowledge Base KB, target $t$
   Output: relocation sequence $O_s$
1  $Root.O_c \leftarrow t$ // objects to be relocated
2  $Root.O_s^R \leftarrow t$ // reversed relocation sequence
3  $Root.g \leftarrow 0$
4  OPEN $\leftarrow \{Root\}$
5  while $OPEN \neq \emptyset$ do
6      $N \leftarrow \arg\min_{N' \in \text{OPEN}} N'.cost$
7      OPEN $\leftarrow$ OPEN $\setminus \{N\}$
8      $D \leftarrow \emptyset$ // list of local obstacle set
9      for $o$ in $N.O_c$ do
10         $p \leftarrow$getPremise(KB, $o$) // get premise of $o$ from KB
11         if $p$ is not True then
12             $D \leftarrow D \cup p$
13     if $|D| == 0$ then
14         return reverse($N.O_s^R$)
15     dnf $\leftarrow$ conjugateDNFs(D) // convert D into a single DNF by
         conjugation
16     for clause in dnf do
17         $N' \leftarrow$ new node
18         $N'.O_c \leftarrow$ clause
19         $N'.O_s^R \leftarrow$ findDup($N.O_s^R$, clause) // find duplicate object in
             the path from Root to $N'$
20         if len($N'.O_s^R$) $\neq$ 0 then
21             $N'.g \leftarrow len(N'.O_s^R)$
22             OPEN $\leftarrow$ OPEN $\cup N'$
23 return $\emptyset$
```

From S_j, we define the condition for the target to be grasped in propositional logic. To grasp the target o_j with the rotational angle of the gripper z, all objects in S_j^z should be relocated. We represent it as the conjunction of all objects o in S_j^z must be *True*. Additionally, the target o_j is graspable, if only one of the obstacle set S_j^z in S_j is relocated, which can be expressed as the disjunction of S_j^z should be *True*. Thus, to be relocated obstacles for grasping o_j are represented as a Disjunctive Normal Form (DNF)[1] as follows.

$$p_j = \begin{cases} \bigvee_{S_j^z \in S_j} \bigwedge_{o \in S_j^z} o, & \text{if } \emptyset \notin S_j \\ True, & \text{otherwise} \end{cases} \tag{5}$$

For the target object o_j, a sentence can be written as an implication[2] form whose premise p_j is a DNF and whose conclusion is a single positive literal o_j. Therefore, we can generate a Knowledge Base by identifying local obstacles set and representing them as an implication sentence for each object in O as follows.

$$KB = \{p_j \Rightarrow o_j | j = 1, ..., \text{N}\} \tag{6}$$

B. Local obstacles-based backward search

We propose a search algorithm for determining whether the target can be singulated and, if so, finding the global optimal relocation plan for target singulation. We employ Backward Chaining, starting from the target, and repeating the chain of reasoning to determine whether the target is True. Furthermore, we employ the Uniform Cost Search algorithm to select the node that requires minimum relocation, thereby obtaining the global optimal relocation plan.

Backward Chaining [11] generally assumes that KB consists of Horn Clauses[3], which is an implication with premise as a clause and conclusion as a single literal. Therefore, And-Or graph is constructed with a single object literals as nodes for determining whether the query is True. However, we assume KB consists of implications with a premise as a disjunction of one or more AND clauses, referred to as DNF. Thus, we define each node with AND clause and apply OR-graph search. During the chaining process, the premises of the node are conjugated and generate child nodes that have OR relations with each other. Since all premises are in DNF, it is possible to convert into a single DNF conjugation from the multiple DNFs and construct a search tree by defining each clause in a DNF as a child node. Node with AND clause also conveys the meaning of local obstacle set colliding with k fingers at some angular sector z.

Alg. 1 describes our method. It starts from the root node with target object t, and enqueue to OPEN set (lines 1-4). OPEN is a data structure that keeps track of candidate nodes for expanding the search tree. Since we use the Uniform Cost Search (UCS) method, we select a node that has the minimum value of *cost* and dequeue from OPEN (lines 6-7). D is a list for local obstacle set of $N.O_c$. For each object o of Node N, we find a sentence with the o as the conclusion from KB append its premises to D (line 9). Since the premise

[1]Disjunctive Normal Form (DNF) refers to the disjunction(OR) of conjunctive(AND) clauses.

[2]Implication refers to a sentence in the form of $X \Rightarrow Y$, which means that if X is True, then Y is also True. X is called *premise*, and Y is called *conclusion*.

[3]Horn clause is a disjunction of literals in which at most one is positive, forming an implication where the premise is the clause and the conclusion is a single literal.

of each object o is in DNF, D is a list of DNFs (lines 10-12). If all objects in $N.O_c$ are graspable, then there are no objects to be relocated, and the size of D is 0. Therefore, empty set D indicates that N is a goal node. If it reaches to goal, the relocation sequence is returned (lines 13-14).

If the goal is not reached, then child nodes are generated from D. Since the objects of $N.O_c$ are in a conjugation relationship with each other, each DNF in D also has a conjugation relationship with each other. The conjunction of multiple DNFs can be converted into a single DNF by applying the distribution law. Then, each AND clause in a single DNF generates a child node. To avoid generating unnecessary duplicates, logically equivalent clauses in a DNF are removed in this process (line 15).

Each conjunctive clause of the DNF becomes $N'.O_c$, which is the objects for the child node N' (lines 17-18). The relocation sequence for child node $N'.O_s^R$ is defined by adding $N'.O_c$ to the relocation sequence of the parent node $N.O_s^R$. The function **findDup** checks if a duplication object exists in the relocation sequence of the child node and returns $N'.O_s^R$. $N'.O_s^R$ depends on whether the duplicate object is graspable. If duplicate object o_d is not graspable, then the relocation sequence does not exist; therefore, **findDup** returns an empty set. However, if o_d is graspable, then **findDup** returns by removing o_d from $N.O_s^R$ and adding to the $N'.O_c$ (line 19). If there is no duplicate object, then N' is added to the OPEN list. The *cost* value of N' is defined by the length of its O_s^R (lines 20-22). If there are no more nodes to explore in the OPEN list before reaching the goal, it concludes that the relocation plan does not exist and returns the empty set (line 23).

C. Reposition Algorithm

We propose a method for determining the relocation positions for obstacles along the relocation sequence obtained through Backward Chaining. The relocation positions should ensure that the grasped object does not overlap with other objects, and consideration should be given to the clearance radius r_g when opening the gripper during grasping and placing. Additionally, the relocation should be performed in a way that does not affect subsequent sequences. To achieve this, we propose the following steps for selecting the relocation positions.

First, excluding the currently grasped obstacle in Fig. 5b, we create a binary image by expanding the clearance radius r_g that the gripper can be placed and opened around all objects (Fig. 5c). Second, we convolve the binary image from Fig. 5c with a mask image of the size of the object to be relocated (Fig. 5d). Third, using a threshold value that considers the k-finger gripper and r_g, we mark the relocation position area on the convolution image (Fig. 5e). The threshold is as follows:

$$threshold = \begin{cases} 255\left(1 - \frac{2\theta_g}{360} + \frac{sin(\theta_g)}{\pi}\right), & \text{if } \theta_g < \theta_k \\ 255\left(1 - \frac{2\theta_k}{360} + \frac{sin(\theta_k)}{\pi}\right), & \text{otherwise} \end{cases} \tag{7}$$

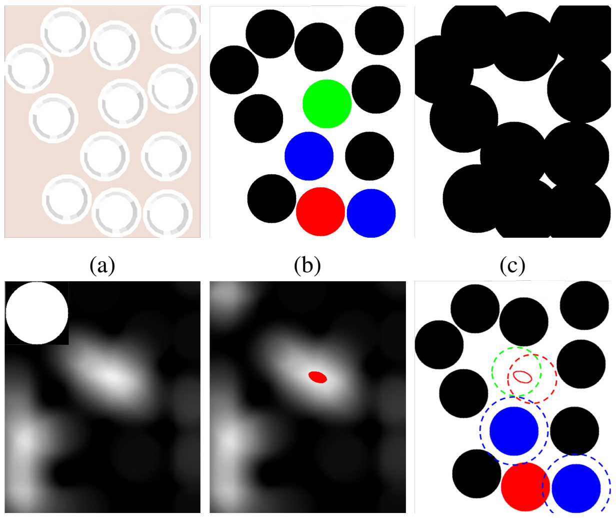


Fig. 5: (a) Current environment. (b) Current image. Red is the current target object. Blue is the remaining obstacle in the sequence. Green is the current obstacle in the sequence. (c) Binary image extended by r_g in the current image. (d) Mask image and result of the convolution with the binary image. (e) The image emphasizing areas that exceed the threshold. (f) The final selected image. The red dashed circle is the original obstacle position. The blue dashed circle is the area expanded by r_g around the obstacles is the sequence. The green dashed circle is the selected relocation position.

Where 255 is the value obtained when the mask image region completely overlaps in convolution. The values in parentheses following it represent the ratio of the area excluding the portion covered by the circle for the given angle in the entire area of the circle. Furthermore, θ_g represents the angle from the center of the circle when it overlaps by r_g, and θ_k is the angle between the gripper fingers.

Finally, among the marked areas, we select a position that is farthest from the target object, ensuring that it does not interfere with subsequent relocations (Fig. 5f).

V. EXPERIMENT

In this section, we show the experimental results of our algorithms. First, we demonstrate the performance of our search algorithm (Section III.B) for obtaining the global optimal relocation sequence. Second, we show the success rate of the repositioning algorithm (Section III.C) given the relocation sequence. Finally, we execute the entire process (Section III) of generating the relocation plan and validate it in a simulation environment.

A. Experiment Setup

We evaluated our algorithm in a simulated environment using the robotic simulator CoppeliaSim [12] where dynamics are modeled by various physics engines. A 6 DOF manipulator Doosan Robot M1013 with a custom three-fingered gripper is used for simulation. The gripper configuration is as follows: $k = 3, \theta_k = 120\text{deg}, w_f = 0.02\text{m}, r_g = 0.05\text{m}$.

TABLE I: The performance of our search algorithm LOBS for Case I (N=16, 18, 20) and Case II (N=25, 37, 50). N represents the number of objects, and the value in parentheses next to N indicates the occupancy rate, which is the area occupied by objects relative to the table area. The baseline method is shortened as Base.

Measure	N=16 (34%)		N=18 (38%)		N=20 (42%)		N=25 (≈45%)		N=37 (≈45%)		N=50 (≈45%)	
	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours	Base	Ours
deadlock rate (%)	0	20	0	25	0	55	0	35	0	55	0	35
success rate (%)	95	100	80	100	70	100	55	100	35	100	0	100
search time (sec)	3.58	0.14	12.88	0.17	19.08	0.21	6.13	0.26	25.96	0.53	60.00	0.93
# relocation	2.00	2.00	2.00	2.07	2.07	2.44	2.09	2.23	2.00	2.33	0.00	2.54

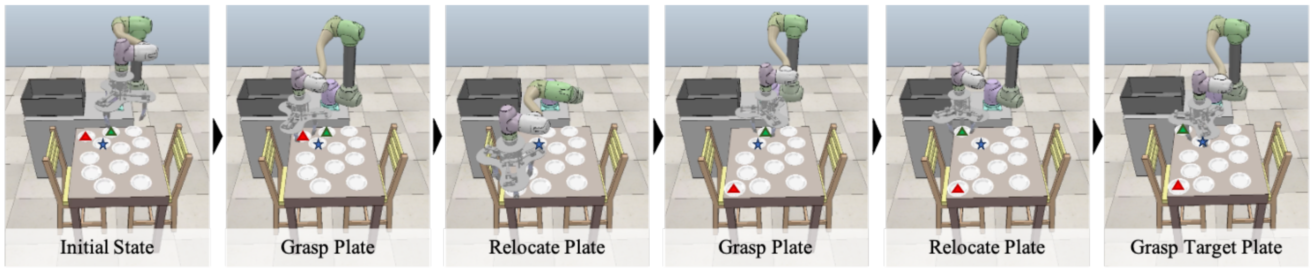


Fig. 6: An example sequence of executing a relocation plan in a simulation environment. In the first figure described as *Initial State*, a star represents the target, and triangles represent objects to be relocated. Relocating objects in the order of the red triangle and then the green triangle creates enough space to grasp the target, as shown in the final figure.

We assume a scenario where multiple objects are densely placed on the table. We set the size of all objects i to the same radius $r_i = 0.075$m and a rectangular table of different sizes. We evaluated by increasing the number of objects. All experiments were on a 2.9 GHz Intel Core i7-10700 desktop with 64 GB RAM with one minute runtime limit.

The baseline method for our search algorithm LOBS is based on A* search [13]. Different heuristics have been proposed to find a global optimal rearrangement plan in [14], [7]. Unlike LOBS, the baseline is a forward search approach that starts from a graspable object and defines each node with a single object. Starting from the root node, it considers the selected object has been relocated and generates the next graspable object as a child node, expanding the search tree until the target is reached. We define heuristic as $G^t_{min} = \min_{z\in[0,\theta_k]} G^t(z)$, which is the minimum number of local obstacle for grasping the target t. $G^t(z)$ is defined in Eq. 11. The baseline method does not use KB but instead assumes that the selected object has been relocated and updates the graspable objects as the search tree expands.

B. Experiment Result

We conducted experiments on two cases to demonstrate the performance of the proposed search algorithm compared with the baseline method. In Case I, we increased the number of objects on a fixed-sized table. In Case II, we increased the number of objects while also increasing the table size in proportion to maintain the occupancy rate, which is the ratio of the area occupied by objects.

Table I presents the averaged results across 20 episodes for each column. For each episode, all objects are randomly placed, and the target was selected as the object with the largest G^j_{min}. We consider an episode a success if a relocation sequence is found or a deadlock is identified within a 1-minute time limit. Therefore, *success rate* refers to the proportion of successful episodes to the total number of episodes. Likewise, *deadlock rate* refers to the proportion of episodes where deadlock is identified to the total number of episodes. *# relocation* refers to the average length of the relocation sequence across non-deadlock episodes. *search time* represents the average search time across all episodes.

The experiment result for Case I is shown in Table I. As the number of objects N increases, the our method(Ours) identifies more episodes as deadlocks. On the other hand, the baseline method(Base) is likely to fail to identify deadlock within the time limit. Therefore, as N increases, the search time of **Base** significantly increases, leading to a decrease in the success rate. In contrast, **Ours** maintains a 100% success rate and keeps the search time below 1 second, regardless of the deadlock rate. Since both are optimal planners, the number of relocations remains the same when N is relatively small. However, as N increases, **Base** fails to find a relocation plan in 1 minute, even if it exists. Thus, when N=20, the average number of relocations is smaller in **Base** than **Ours**.

In Case II, we increased N while maintaining the occupancy rate of almost 45%. **Base** shows a significant decrease in success rate as N increases. This is due to the increase in search time, similar to Case I. On the other hand, **Ours** shows no meaningful increase in both the success rate and search time, even with the increase in N and # relocation.

Table II shows the performance of our reposition algorithm. It achieved a 100% success rate and a runtime of about 3 seconds in all episodes. Table III shows the performance of our methods, including searching for the relocation sequence and determining the relocation position, which is the total process of generating a relocation plan and executing it in simulation. A relocation plan consists of a sequence of objects to be relocated and the positions to relocate them

TABLE II: The success rate and runtime of the proposed repositioning algorithm. Only 10 non-deadlock episodes of Case I are considered for evaluation.

	N=16	N=18	N=20
success rate (%)	100	100	100
reposition time (sec)	2.67	2.93	2.76

TABLE III: The total success rate and individual runtime for relocation planning and execution in a simulation environment. Relocation planning includes the process of searching for relocation sequence and repositioning. Only 10 non-deadlock episodes of Case I are considered for evaluation.

	N=16	N=18	N=20
total success rate (%)	100	100	100
planning time (sec)	2.82	3.09	2.96
execution time (sec)	77.99	77.60	76.49

to. For each object to be relocated, pick-and-plan actions are generated for the simulated robot with a motion planner RRTConnect [15]. We consider an execution is success if all actions are executed without any collisions with objects on the table until the target is picked up. Our method achieved a 100% success rate including execution and a planning time of about 3 seconds in all episodes.

VI. CONCLUSIONS

We proposed an algorithm that generates a relocation sequence through backward search based on a local obstacle set. Our algorithm was able to maintain a 100% success rate and determine whether the deadlock occurred or generate a global relocation plan within 3 seconds, regardless of the number of objects N. However, as N increases, the deadlock rate also increases significantly. Therefore, it is necessary not only to determine whether it is a deadlock but also to generate a plan for breaking the deadlock by utilizing non-prehensile actions. Furthermore, we assume that there is always enough space to reposition the object within the table area. However, even if there is free space to relocate, a collision between the gripper and obstacles may occur while placing the object in the new position.

APPENDIX

In a grid map with a square size of w_s, two constants $a, b \in \mathbb{R}^+$ are defined by the following relationship.

$$a - b\left(\frac{w_s - 1}{2}\right)^2 = 1 \tag{8}$$

In Fig. 2a, the finger width of gripper is w_f and the radius of the target is r_t. Then, the area occupied by each finger γ_t is defined as follows.

$$\gamma_t = 2tan^{-1}(\frac{w_f}{2r_t}) \tag{9}$$

As shown in Fig. 2b, the area occupied by an obstacle o_i, which is at a distance d_i from t and exists within the r_g boundary, can be defined as follows.

$$\gamma_i = 2cos^{-1}\left(\frac{(r_t + r_g)^2 + d_i^2 - r_i^2}{2d_i(r_t + r_g)}\right) \tag{10}$$

For all object i, $G^j(z)$ is defined as the sum of the histogram for each object i as follows.

$$G^j(z) = \sum_i G_i^j(z) \tag{11}$$

ACKNOWLEDGMENT

This work was supported by the Technology Innovation Program and Industrial Strategic Technology Development Program (20018256, Development of service robot technologies for cleaning a table).

REFERENCES

[1] M. Stilman and J. Kuffner, "Planning among movable obstacles with artificial constraints," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1295–1307, 2008.

[2] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 4627–4632.

[3] E. Huang, Z. Jia, and M. T. Mason, "Large-scale multi-object rearrangement," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 211–218.

[4] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4238–4245.

[5] K. Xu, H. Yu, Q. Lai, Y. Wang, and R. Xiong, "Efficient learning of goal-oriented push-grasping synergy in clutter," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6337–6344, 2021.

[6] 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.

[7] 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.

[8] C. Nam, J. Lee, S. H. Cheong, B. Y. Cho, and C. Kim, "Fast and resilient manipulation planning for target retrieval in clutter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3777–3783.

[9] D. Saxena and M. Likhachev, "Planning for manipulation among movable objects: Deciding which objects go where, in what order, and how," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 33, no. 1, 2023, pp. 668–676.

[10] D. M. Saxena and M. Likhachev, "Planning for complex non-prehensile manipulation among movable objects by interleaving multi-agent pathfinding and physics-based simulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8141–8147.

[11] C. Hewitt, "Planner: A language for proving theorems in robots," in *Proc. of International Joint Conference on Artificial Intelligence*, vol. 1, 1971, pp. 295–301.

[12] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 1321–1326.

[13] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[14] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, vol. 33, pp. 217–236, 2012.

[15] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.