Abstract—This paper presents a task and motion planning (TAMP) framework for a robotic manipulator in order to retrieve a target object from clutter. We consider a configuration of objects in a confined space with a high density so no collision-free path to the target exists. The robot must relocate some objects to retrieve the target without collisions. For fast completion of object rearrangement, the robot aims to optimize the number of pick-and-place actions which often determines the efficiency of a TAMP framework.

We propose a task planner incorporating motion planning to generate executable plans which aims to minimize the number of pick-and-place actions. In addition to fully known and static environments, our method can deal with uncertain and dynamic situations incurred by occluded views. Our method is shown to reduce the number of pick-and-place actions compared to baseline methods (e.g., at least 28.0% of reduction in a known static environment with 20 objects).

I. INTRODUCTION

Retrieving a target object from clutter using a robotic manipulator has long been considered as an important and practical task. Robots will perform such tasks in cluttered and confined spaces frequently in our home or workplace (e.g., shelves in fridges or cabinets) as illustrated in Fig. 1. If objects are populated densely and overhand grasps are not allowed, a subset of the objects should be relocated to secure a collision-free path for the manipulator to retrieve the target.

In manipulation planning, task planning focuses on generating high-level discrete actions while motion planning finds a sequence of robot configurations which result in continuous motions. Recently, tight coupling of task and motion planning (TAMP) has shown successful achievements [1]–[4] by generating symbolic task plans that are executable. However, TAMP frameworks could be inefficient for manipulation tasks in dense clutter. Planning robot motions in dense clutter could fail frequently but geometric motion planners cannot provide the cause of failure in the form of logical constraints. Thus, the task planners should iterate over symbolic plans if motion planning fails [5].

In this work, we propose a planning method taking the TAMP approach where the geometric task planner is specified for object rearrangement in dense clutter. Our task planner aims to generate a sequence of feasible pick-and-place actions to relocate objects until a target object becomes reachable. In order to establish a task plan that has valid continuous motions, the actions should be examined by a motion planner. However, determining the sequence is difficult as manipulation planning among movable obstacles

This work was supported by the Technology Innovation Program and Industrial Strategic Technology Development Program (10077538, Development of manipulation technologies in social contexts for human-care service robots). The authors are with Korea Institute of Science and Technology.

*Corresponding author: ckin@kist.re.kr.
mathematical formalization of the search and grasp planning problem. However, the algorithm shows exponential running time so may not be practically useful in environments with densely packed objects (planning takes longer than 25 sec for five objects). Another work [14] finds a sequence of actions of a mobile manipulator that minimizes the expected time to reveal all possible hidden target poses. This work defines admissible costs for its A* search, but planning takes long time owing to the high branching factor of the search (e.g., 40 sec with five objects).

In [15], the authors present methods dealing with object rearrangement. Although the suggested methods show high success rates in a confined space where overhand grasps are not allowed, they do not scale to the number of objects. [16] propose an asymptotically optimal algorithm for rearrangement and manipulation planning. However, the search in a graph structure takes exponential time (e.g., few seconds for an instance with only two objects). [7] and [17] consider object rearrangement on a tabletop. In both, scalable algorithms are proposed which can handle hundreds of objects. However, their problems allow overhand grasps so motion planning is relatively easy.

Our own work [18] proposes a fast algorithm for object relocation in clutter where overhand grasps are not allowed. However, it is a local planner so does not set out to achieve the global optimum. Also, it does not consider motion planning of the whole manipulator and partially known environments. In this work, we aim to develop a fast algorithm that optimizes the number of manipulation actions while considering robot kinematic constraints and partially known environments caused by occlusions.

III. PROBLEM FORMULATION

Target retrieval from clutter requires several different processes such as perception, task planning, motion planning, and grasping. We focus on the relocation task and motion planning. The problem of finding a path in a configuration of movable objects has an exponentially large search space in the number of objects. A simplified version of the problem with only one object is shown to be \( \mathcal{NP} \)-hard [19], [20] even in a perfectly known environment. We consider known environments as well as uncertainties incurred by occlusions.

Major assumptions: (i) No collision-free path exists for the end-effector without relocating some objects. The spaces to place relocated objects predetermined to be outside the workspace\(^1\), (ii) Overhand grasps are not allowed (e.g., the top is blocked by shelves), (iii) Objects are modeled by 3D cylinders (which could have different radii) so the objects can be grasped from any direction.

A. Problem definition

Our goal is to complete the target retrieval task quickly so our objective value is the number of pick-and-place actions (the number of relocated objects equivalently). Suppose that an environment is with obstacles \( o_i \) for \( i = 1, \cdots, N - 1 \) \( \in \mathbb{Z}^+ \) and a target \( o_t \) (so total \( N \) objects). The centroid, radius, and height of object \( i \) is described by \( (x_i, y_i, h_i) \). Let \( \mathcal{O} = \{o_1, \cdots, o_{N-1}, o_t\} \). Let \( \mathcal{O}_R \subset \mathcal{O} \) be the sequence of objects to be relocated (including the target) where \( |\mathcal{O}_R| = k \leq N \). The home pose of the robot end-effector is described by \( p_R \). The thickness of the end-effector is \( r_e \). If it grasps an object whose radius is \( r_i \), the radius of the end-effector grasping the object is \( r_g = r_i + r_e \).

A mathematical definition of the problem is to find \( \mathcal{O}_R \) that minimizes \( k \). The solution sequence \( \mathcal{O}_R \) lists objects in the order in which they should be removed.

B. Dynamic and uncertain situations owing to occlusion

Objects could occlude each other in dense clutter. We need to define different situations occurring from occlusions so define relevant concepts. An object is occluded if it is partially visible to the robot. Occluded volume quantifies the space occluded by objects (Fig. 3b). An object is accessible if it can be grasped by the end-effector without relocating any objects. The set \( \mathcal{O}_A \subset \mathcal{O} \) includes all accessible objects. Fig. 3c shows four accessible objects (bold outlines). The leftmost object at the bottom is not accessible since there is no space for the end-effector to wedge its fingers into.

IV. TASK AND MOTION PLANNING FOR OBJECT RETRIEVAL FROM CLUTTER

Our task and motion planning framework consists of graph construction followed by task planning combined with motion planning. The graph represents traversability of objects considering the size of the robot hand. The task and motion planner finds a sequence of objects to be relocated using the graph while considering the robot kinematic constraints.

\( \)
A. Traversability graph

We construct a traversability graph (T-graph) representing movable paths of objects in clutter. An edge between a pair of objects means a collision-free path of the end-effector to move any object between the two poses represented by the nodes. The basic idea is illustrated in Figs. 3 and 6. Any object in \( \mathcal{O} \) can move between two poses if a path exists between the two corresponding nodes in the graph and the objects in the path are cleared. For example, \( \mathcal{V}_R = \{ v_R, v_2, v_3 \} \) in Fig. 5 is the shortest path from \( v_R \) to \( v_t \) so the end-effector can retrieve \( o_1 \) if \( o_2 \) is removed.

The T-graph construction is described in Alg. 1 which constructs an unweighted and undirected graph \( G(\mathcal{V}, \mathcal{E}) \) from \( \mathcal{O} \) where \( \mathcal{V} \) and \( \mathcal{E} \) are the sets of nodes and edges, respectively. Nodes represent the objects in \( \mathcal{O} \) so \( \mathcal{V} = \{ v_1, \cdots, v_t, v_R \} \) where \( v_t \) and \( v_R \) are the target and robot node, respectively (line 2). For every pair of nodes \( v_i \) and \( v_j \) where \( i \neq j \), an edge \( (i, j) \in \mathcal{E} \) is connected if any object grasped by the end-effector can access \( o_2 \) and \( o_3 \) so \( v_R \) has edges incident to \( v_2 \) and \( v_3 \).

Fig. 6: An example of T-graph construction. The largest object \( o_2 \) cannot move between \( (x_1, y_1) \) and \( (x_2, y_2) \), so nodes \( v_2 \) and \( v_3 \) are not connected. Similarly, \( v_4 \) and \( v_5 \) are not connected. The end-effector can access \( o_2 \) and \( o_3 \) so \( v_R \) has edges incident to \( v_2 \) and \( v_3 \).

Algorithm 1 GENGRAPH

```
Input: object geometry \( \mathcal{O} \), workspace \( \mathcal{W} \), robot position \( p_R \), robot size \( r_r \), safety margin \( r_s \)
Output: an unweighted undirected graph \( G(\mathcal{V}, \mathcal{E}) \)
1. \( N = |\mathcal{O}| - 1 \) // \( N \) counts obstacles only
2. \( \mathcal{V} = \{ v_1, \cdots, v_t, v_R \} \) // nodes of all objects and the end-effector
3. \( \mathcal{E} = \emptyset \)
4. \( r_g = r_{\text{max}} + r_r + r_s \) // \( r_{\text{max}} \) is the radius of the largest object
5. for each \( v_i \in \mathcal{V} \)
6. \( \mathcal{V} = \mathcal{V} \setminus v_i \) // no self-loop considered so \( i \neq j \)
7. for each \( v_j \in \mathcal{V}' \)
8. if \( \text{isCollision}(r_g, o_1, o_2, \mathcal{W}) \) // true if there is a collision for
9. \( \mathcal{E} \leftarrow \mathcal{E} \cup (i, j) \) // in the presence of immovable objects in \( \mathcal{W} \)
10. end for
11. end for
12. return \( G(\mathcal{V}, \mathcal{E}) \)
```

B. Relocation planning

Alg. 2 finds the minimum-hop path on a T-graph \( G \), which is \( \mathcal{V}_R \) (line 1) from the robot node \( v_R \) to the target \( v_t \). The path represents a sequence of objects to be relocated \( O_R \). Path finding on \( G \) is implemented using Breadth first search (BFS) [22]. If there are multiple paths that have the same number of hops, we compare their Euclidean distances to break the ties (lines 2–4). This algorithm does not consider the kinematic constraints of the manipulator. Thus, Alg. 3 combines motion planning with task planning where motion planning failures are handled in an online manner to find an alternative.

Initially, a task plan is found offline by Algs. 1 and 2 (lines 1–2). For each pick-and-place action for object \( o \in O_R \), motion planning is performed (lines 4–5) for both pick and place actions. If feasible motions are found, \( o \) is removed from the scene (line 7). After \( o \) is removed, the robot senses the scene to reflect the changes in traversability accordingly (lines 8–9). If no motion is found, the edge \( (v_R, v) \) (representing the path from the robot to \( o \)) is removed from \( G \) (line 11). If one or more new objects are found, they are included in the graph to update the task plan (lines 13–15). The whole procedure iterates until \( O_R \) becomes empty. If the robot does not have feasible motions for picking and placing any of the objects, the algorithm fails (lines 17–19).

Algorithm 2 RELOCPath

```
Input: graph \( G(\mathcal{V}, \mathcal{E}) \), target and robot poses \( o_t, p_R \)
Output: a sequence of objects to be relocated \( O_R \)
1. \( \mathcal{V}_R = \text{MINHOPPATH}(G, v_R, v_t) \) // find a min-hop path \( \mathcal{V}_R \) with
2. if multiple paths with \( k \) nodes
3. Choose the one with the minimum Euclidean distance
4. end if
5. \( O_R = \{ o_i \mid i \text{ is the index of a node in } \mathcal{V}_R \} \) // \( O_R \) is implemented using a queue
6. return \( O_R \)
```

Lastly, we propose an online planner that also can deal with Case III together with other cases. In Case III, the robot is tasked with target search first. In [13], an object is chosen to be removed such that the volume revealed after the removal is maximized. In our preliminary study [23], we tested three simple search strategies that are the one
Algorithm 3 BASEPLANNER

**Input:** object geometry \( \mathcal{O} \), target \( \alpha_t \), workspace \( \mathcal{W} \), robot kinematics \( X \), robot position \( p_R \), robot size \( r_v \), safety margin \( r_s \)

**Output:** Done

1. \( G(V, E) = \text{GENGRAPH}(\mathcal{O}, \mathcal{W}, p_R, r_v, r_s) \)
2. \( \mathcal{O}_R = \text{RELOCPath}(G, \alpha_t, p_R) \) \hspace{1cm} // compute the initial plan
3. while \( \alpha_t \) is not grasped
4. \( o = \text{DEQUEUE}(\mathcal{O}_R) \)
5. \( \text{result} = \text{MOTIONplanning}(o, \mathcal{O}, \mathcal{W}, p_R, X) \)
6. if \( \text{result} \)
7. Remove \( o \) from \( \mathcal{O} \)
8. Update \( \mathcal{O} \) with sensor inputs
9. \( G(V, E) = \text{GENGRAPH}(\mathcal{O}, \mathcal{W}, p_R, r_v, r_s) \)
10. else
11. \( G(V, E) = G(V, E \setminus (v_R, v)) \) \hspace{1cm} // remove the infeasible edge
12. if a new object(s) is found \hspace{1cm} // update the graph and replan
13. \( G(V, E) = \text{GENGRAPH}(\mathcal{O}, \mathcal{W}, p_R, r_v, r_s) \)
14. end if
15. \( \mathcal{O}_R = \text{RELOCPath}(G, \alpha_t, p_R) \)
16. if \( \mathcal{O}_R = \emptyset \) and \( \alpha_t \) is not grasped
17. return Fail \hspace{1cm} // terminate if no path to \( \alpha_t \) is available
18. end if
19. end while
20. return Done

Algorithm 4 RELOCPLANNER

**Input:** object geometry \( \mathcal{O} \), target \( \alpha_t \), workspace \( \mathcal{W} \), robot kinematics \( X \), robot position \( p_R \), robot size \( r_v \), safety margin \( r_s \)

**Output:** Done

1. while \( \alpha_t \) is not detected
2. \( G(V, E) = \text{GENGRAPH}(\mathcal{O}, \mathcal{W}, p_R, r_v, r_s) \)
3. \( (\mathcal{O}_A, V_A) = \text{GETACCESSIBLEOBJ}(G) \)
4. for each \( \alpha_i \in \mathcal{O}_A \)
5. Compute the metric \( m_i \) \hspace{1cm} // revealed volume
6. end for
7. if \( \text{result} \)
8. Remove \( o \) from \( \mathcal{O} \)
9. Update \( \mathcal{O} \) and \( \mathcal{O}_A \) with sensor inputs
10. else
11. \( \mathcal{O}_A = \mathcal{O}_A \setminus o \)
12. if \( \mathcal{O}_A = \emptyset \)
13. return Fail \hspace{1cm} // terminate if motion planning fails for all objects
14. end if
15. end if
16. end while
17. BASEPLANNER(\( \mathcal{O}, \alpha_t, \mathcal{W}, X, p_R, r_v, r_s \))
18. return Done

similiar to [13] (which we call **Volume** strategy) and two based on the Euclidean distance between the end-effector and objects. Experiments showed that **Volume** outperforms others so we choose the strategy in this present work. While the target is not detected, Alg. 4 removes object \( o \) that reveals the largest volume (lines 1–21). If motion planning for picking and placing \( o \) is successful, it is removed and the scene is updated. If no motion is found, the object becomes inaccessible so another object is chosen. If motion planning fails for all objects, Alg. 4 terminates. Once the target is found, the base planner dealing with Cases I and II is executed.

In Fig. 7 we show an example result of running Alg. 4 for Case I. Fig. 8 is an example execution for Case II. Two objects are initially undetected and revealed during execution. In Fig. 4 **Volume** strategy removes objects sequentially until the target is detected.

**C. Analysis of algorithms**

We provide the analysis of some algorithms.

**Lemma 4.1.** Alg. 4 has polynomial time complexity.

**Proof.** Collision checking (line 8) runs for every pair of objects (lines 5–11). There are total \( N(N-1)/2 \) pairs. The collision checker that we used runs in \( O(N(N + N)) = O(N^2) \) [18]. Thus, the time complexity is \( O((N(N-1)/2) \times (N(N + N))) = O(N^4) \).

The pseudocode assumes that the target does not become undetected once it is detected for simplicity. However, it can be modified easily for the case where the target is not recognized even after it is located.

**Theorem 4.2.** Alg. 2 has polynomial time complexity.

**Proof.** The graph \( G(V, E) \) has \( N \) nodes and at most \( N(N-1)/2 \) edges (fully connected). BFS runs in time \( O(|V| + |E|) = O(N(N-1)/2 + N + 1) = O(N^2) \) to find all min-hop paths between \( v_R \) and \( v_t \). Choose the one with the shortest Euclidean distance takes \( O(N) \). Thus, the time complexity is \( O(N^2 + N) = O(N^2) \).

**Theorem 4.3.** Alg. 2 is complete if \( G \) is connected.

**Proof.** First, we want to show that Alg. 1 is complete. Collision checking in line 8 of Alg. 1 is complete [18], which means that \( G \) is constructed after a finite number of iterations. By definition of connected graphs, a path exists from one of the accessible nodes to the target. BFS is complete [22] so finds the shortest path. Thus, Alg. 2 always returns a path which is the shortest.

Since Algs. 3 and 4 include motion planning, they do not...
have polynomial time complexities in the number of objects. Their time complexities depend on the motion planner used. Note that sampling-based planners have time complexities that range from \( O(n \log n) \) to \( O(n^2) \) for both the process and query where \( n \) is the number of samples [24].

V. EXPERIMENTS

In this section, we show the experimental results of our method. We measure the task and motion planning time and the success rate of Alg. 4. We also run experiments in a simulated environment to measure the total running time including the execution of pick-and-place actions. We consider scenarios for all Cases I, II and III. We assume the line of sight of the fixed camera on the robot in Cases II and III. In Case II, 20% of the total objects are hidden initially and revealed if some front objects are removed. In Case III, only the front objects can be recognized.

We compare our method with other two fast planning methods. The first one is adopted from [9] which removes obstacles on the distance-optimal path of the end-effector (\texttt{Distance}). The second one is the method presented in [18] described in Sec. III (\texttt{VFH+}). We choose these two methods since ours aims to achieve fast planning for practical uses. While other methods have shown long running time even with small instances as reviewed in Sec. III, the chosen methods often run faster than ours. Since it is not known how they replan if motion planning fails, we implement replanning methods for them for fair comparisons instead of just terminating. We assume that it is beneficial if additional objects are removed when motion planning fails because having a larger configuration space increases the chance to succeed in planning motions. In \texttt{Distance}, more objects are added by increasing the width of the distance-optimal path of the end-effector (increasing the width by 2 cm for every failure from the initial width 6 cm). In \texttt{VFH+}, the angle range for finding obstacles to be removed increases by 10 deg for every failure from the initial value 90 deg.

Motion planning is implemented using a motion planning library [25] in MoveIt motion planning framework [26]. We use RRTConnect [27] which shows the best performance in our pilot studies. The same motion planner is used for all compared methods. We impose a time limit for task and motion planning. A successful planning takes less than one minute. Considering time limits in other work (e.g., 30 mins in [15]), our time limit is significantly short even with the large number of objects (up to 20 objects) and the incorporation of motion planning. The number of objects would not be increased beyond 20 since we use a manipulator with a fixed base so the range the robot can reach is limited. The system is with Intel Core i9 3.6GHz with 32G RAM and Python 2.7.

A. Algorithm tests

We test the algorithms with random instances to measure the computation time for task and motion planning. We randomly generate 20 instances for each size where \( N = 12, 16, 20 \). We use a model of Kinova JACO1, which is a 6-DOF manipulator in MoveIt. The values used for the robot size \( r_r \) and safety margin \( r_s \) are 5.0 cm and 0.5 cm, respectively. Objects are uniformly distributed at random in a workspace where the dimension is 0.9 m (length) \( \times \) 0.45 m (width) \( \times \) 0.45 m (height). The diameters and heights of objects are randomly sampled from \( \mathcal{U}(5, 6) \) and \( \mathcal{U}(6, 7) \) where the unit is centimeter.

We measure the number of relocated objects (i.e., pick-and-place actions) which is the main objective value. We also measure the success rate given the time limit (1 min). We first run 20 instances and compute the success rate. Then we run additional instances to collect 20 data points to have the equal sample size for computing the statistics. Comparisons are done for Cases I and II because the compared methods cannot deal with Case III as they must know the pose of the target. The result is shown in Figs. 10 and Tables III and IV.

Discussion: Motion planning time has large variances since it depends on the poses of the objects and the configuration space of the robot differing from each random test instance. Motion planning time does not have significant differences across the compared methods. The total planning time of ours is longer than others in average. However, ours still takes up to 8 sec per each action including motion planning in all cases, which is not prohibitively long in such complex
TABLE I: Task and motion planning results of the proposed and compared algorithms for Cases I, II, and III (20 repetitions). Except the number of relocated objects and the success rate, all numbers show the mean time (sec) and standard deviation (in parentheses). The number of relocated objects includes the target object.

| Case  | Measure                      | N = 12 | N = 16 | N = 20 |
|-------|------------------------------|--------|--------|--------|
|       | Proposed | Distance | VFH+ | Proposed | Distance | VFH+ | Proposed | Distance | VFH+ |
| I     | #relocated objects | 2.55 (0.60) | 2.45 (0.76) | 3.11 (1.54) | 2.89 (0.66) | 3.68 (2.08) | 4.42 (4.13) | 3.6 (0.88) | 5.21 (2.92) | 5.00 (1.94) |
|       | Success rate (%) | 93 | 100 | 85 | 91 | 89 | 91 | 89 | 91 |
|       | Total time | 13.99 (18.22) | 7.61 (3.06) | 7.62 (3.04) | 23.01 (14.01) | 14.05 (9.68) | 13.21 (8.31) | 23.04 (10.25) | 22.13 (14.24) | 16.85 (8.26) |
|       | Time per action | 3.29 (2.08) | 2.29 (1.19) | 4.28 (0.72) | 8.06 (2.31) | 3.94 (2.46) | 2.94 (1.24) | 2.63 (2.22) | 4.14 (1.54) | 3.35 (1.18) |
| II    | #relocated obstacle | 2.4 (0.50) | 2.5 (0.95) | 3.12 (1.00) | 2.95 (1.08) | 3.32 (1.34) | 3.42 (2.01) | 4.55 (1.39) | 4.2 (1.11) | 5.00 (2.54) |
|       | Success rate (%) | 85 | 100 | 85 | 89 | 70 | 90 | 89 | 89 |
|       | Total time | 9.43 (3.90) | 8.01 (1.71) | 10.92 (1.29) | 20.43 (12.26) | 12.29 (5.09) | 11.78 (3.52) | 35.88 (18.17) | 8.76 (14.39) | 21.38 (14.89) |
|       | Time per action | 1.88 (1.04) | 2.98 (1.51) | 3.58 (2.08) | 6.72 (2.76) | 3.05 (1.64) | 3.58 (2.06) | 7.67 (2.00) | 8.41 (2.65) | 4.18 (1.05) |

TABLE II: Task and motion planning results of the proposed method for Case III (20 repetitions). Compared methods cannot solve the case so only our results are shown.

| Measure | N = 12 | N = 16 | N = 20 |
|---------|--------|--------|--------|
| Success rate (%) | 3.25 (1.16) | 4.37 (1.80) | 4.30 (1.47) |
| Time per action | 21.25 (13.97) | 26.09 (15.00) | 32.09 (15.04) |
| Time per action | 6.39 (3.79) | 5.93 (2.49) | 6.94 (2.23) |

environments. The planning time in Case III is much shorter because the method chooses the object to relocate instantly based on the revealed volume but does not plan globally.

The number of relocated objects is reduced 28.0% compared to VFH+ and 30.9% compared to Distance in Case I. If only obstacles are counted, the reductions are 35.0% and 38.2%, respectively. In Case II, the amount of improvement decreases since hidden objects incur inefficient relocation. The others are less affected by hidden objects as they do not plan globally across the entire workspace. The numbers of relocated objects from the compared methods look better than they what actually are because the instances failed owing to the timeout are not included in computing the mean. Such instances have large numbers of relocated objects while our method rarely has such instances.

Our method has the highest success rate for all instance sizes and the cases. Distance does not incorporate the relationship between objects in task planning thus the order of removing objects is determined without considering what objects should be removed to make next objects reachable. VFH+ shows better success rates than Distance since it considers the relationship. However, it is worse than ours because it is a local planner so may stuck in local minima where no object can be reachable. The success rate in Case III decreases since target search takes long in some instances.

B. Experiments in simulated environments

We test the proposed method in a simulated environment using a high fidelity robotic simulator V-REP [28] with Vortex physics engine. JACO1 is also used in the environment (Fig. 12). The environment and object specification are the same with the tests in Sec. VOC-A We compare the three methods in Case I and generate 10 random instances where N = 20. We omit other experiments owing to the space limit but the results are similar. The robot places removed obstacles in the box below the table. We measure the number of relocated objects, the total running time which includes planning and execution, and the success rate (Table III).

Discussion: As shown in Sec. V-A, the number of relocated objects reduces when our method is used. The reductions of 1.08 and 0.41 in the number of objects reduce 35.0 sec and 20.4 sec in total execution time with Distance and VFH+, respectively. The reductions in time are 32.2% and 21.7%. The success rate is higher in our method which is similar to the algorithm test in Sec. V-A The result shows that our method can finish the target retrieval task efficiently while maintaining high success rates.

VI. Conclusion

In this work, we study the problem of retrieving objects from clutter. Our objective is to minimize the number of objects to be relocated (pick-and-place actions) to generate a collision-free path for a robotic manipulator so as to reduce the total running time to retrieve the target object. We take the TAMP approach and develop an efficient replanning scheme if motion planning fails. In addition to known environments, we consider partially known environments incurred by occlusions. The results from extensive experiments show that our method reduces the objective value compared to baseline methods. The experiment using a dynamic simulator shows that our approach could works as expected for real problems.

In the future, we will consider different shapes of objects so objects may have limited reachable directions. We will also consider non-prehensile actions like pushing and dragging since some objects may need to be moved slightly to avoid collisions.
REFERENCES

[1] N. T. Dantam, S. Chaudhuri, and L. E. Kavraki, “The task-motion kit: An open source, general-purpose task and motion-planning framework,” IEEE Robotics & Automation Magazine, vol. 25, no. 3, pp. 61–70, 2018.

[2] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, “Fifrob: Leveraging symbolic planning for efficient task and motion planning,” The International Journal of Robotics Research, vol. 37, no. 1, pp. 104–136, 2018.

[3] S. Srivastava, E. Fang, L. Riano, R. Chitimis, S. Russell, and P. Abbeel, “Combined task and motion planning through an extensible planner-independent interface layer,” in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), 2014, pp. 639–646.

[4] M. Toussaint, “Logic-geometric programming: An optimization-based approach to combined task and motion planning,” in Proc. of Int. Joint Conf. on Artificial Intelligence (IJCAI), 2015.

[5] F. Lagriffoul and B. Andres, “Combining task and motion planning: A culprit detection problem,” The Int. J. of Robotics Research, vol. 35, no. 8, pp. 890–927, 2016.

[6] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, “Manipulation planning among movable obstacles,” in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), 2007, pp. 3327–3332.

[7] S. D. Han, N. M. Stilffer, A. Krontiris, K. E. Bekris, and J. Yu, “Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps,” The Int. J. of Robotics Research, vol. 37, no. 13-14, pp. 1775–1795, 2018.

[8] S. D. Han, N. M. Stilffer, K. E. Bekris, and J. Yu, “Efficient, high-quality stack rearrangement.” IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1608–1615, 2018.

[9] M. Dogar and S. Srinivasa, “A planning framework for non-prehensile manipulation under clutter and uncertainty,” Auton. Robots, vol. 33, no. 3, pp. 217–236, 2012.

[10] W. Yuan, J. Stork, D. Kragic, M. Wang, and K. Hang, “Rearrangement with nonprehensile manipulation using deep reinforcement learning,” arXiv preprint arXiv:1803.05752, 2018.

[11] L. Haustein, J. King, S. Srinivasa, and T. Asfour, “Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states,” in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), 2015, pp. 3075–3082.

[12] M. Moll, L. Kavraki, J. Rosell et al., “Randomized physics-based motion planning for grasping in cluttered and uncertain environments,” IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 712–719, 2018.

[13] M. Dogar, M. Koval, A. Talavajhula, and S. Srinivasa, “Object search by manipulation,” Auton. Robots, vol. 36, pp. 153–167, 2014.

[14] Y.-C. Lin, S.-T. Wei, S.-A. Yang, and L.-C. Fu, “Planning on searching occluded target object with a mobile robot manipulator,” in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), 2015.

[15] A. Krontiris and K. E. Bekris, “Dealing with difficult instances of object rearrangement,” in Robotics: Science and Systems, 2015.

[16] W. Vega-Brown and N. Roy, “Rt-connect: An efficient approach to single-query path planning,” in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), vol. 2, 2000.

[17] E. Rohmer, S. P. Singh, and M. Freese, “V-rep: A versatile and scalable robot simulation framework,” in Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2013, pp. 1321–1326.

[18] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” The Int. J. of Robotics Research, vol. 30, no. 7, pp. 846–894, 2011.

[19] I. A. Sucan, M. Moll, and L. E. Kavraki, “The open motion planning library,” IEEE Robotics & Automation Magazine, vol. 19, no. 4, pp. 72–82, 2012.

[20] I. A. Sucan and S. Chitta, “MoveIt,” [http://moveit.ros.org] [Online].

[21] J. J. Kuffner Jr and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), vol. 2, 2000.

[22] 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,” 2020, accepted for presentation at IEEE Int. Conf. on Robotics and Automation (ICRA).

[23] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to algorithms. MIT press, 2009.

[24] C. Nam, J. Lee, Y. Cho, J. Lee, D. H. Kim, and C. Kim, “Planning for target retrieval using a robotic manipulator in cluttered and occluded environments,” arXiv preprint arXiv:1907.03956, 2019.