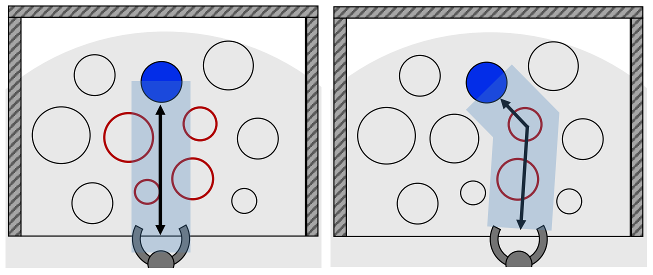
Planning for efficient rearrangement of obstacles for grasping objects in cluttered environments

Changjoo Nam, Jinhwi Lee, and ChangHwan Kim*

***Abstract*— We present a fast planning algorithm for grasping an object in clutter which computes a plan for relocating obstacles to generate a trajectory of an end effector to the target. We consider a configuration of objects where the density of the space is very high, so no collision-free path exists without relocating obstacles. Although there is a more appropriate objective value to be optimized for fast completion (or low energy consumption) of the grasping task, existing work does not set out to optimize such an objective value directly related to the task. Thus, the grasping task could be done inefficiently by spending much time on moving objects that are not necessarily moved.**

We propose an algorithm that aims to minimize the number of objects to be relocated instead of choosing the shortest traveling path of end effectors which is a common strategy in motion planning. The proposed algorithm is shown to run in time $O(N^3)$ where N is the number of obstacles surrounding the target. We show that the proposed algorithm runs very fast even with a large number of obstacles (less than 0.25 sec for 100 objects) and reduces the completion time of the grasping task significantly, compared with a baseline method which optimizes the travel distance of the end effector (39.0% of reduction).



(a) The distance-optimal path without considering the energy or time consumed for rearrangement

(b) An optimal path with the minimum number of obstacles to be relocated

Fig. 1: The target (blue circle) is surrounded by obstacles (empty circles) on a small constrained space like a shelf in a fridge. The space is bounded so objects only can move through the bottom side. The gray area shows the reachable workspace of the manipulator. (a) If the manipulator takes the shortest path to the target, it should remove the four obstacles on the path (red circles). (b) If the manipulator takes a longer path, the number of obstacles that should be rearranged reduces to two although the path is not distance-optimal.

I. Introduction

Grasping objects using robotic manipulators in clutter has long been considered as an important, practical, and difficult task until recently [1]. If obstacles surrounding a target object are populated densely, it is necessary to rearrange a subset of the obstacles to complete a mission taking out the target without damaging the target and obstacles as well as the manipulator. Although there have been numerous methods presented to manipulate a target object in clutter [2]–[8], the problem of optimizing the rearrangement process of the surrounding obstacles has not been received much attention. Thus, manipulation tasks in cluttered environments may be done inefficiently if the manipulator considers successful grasping only but does not take into account the entire task, including relocation of obstacles to avoid collisions between obstacles and the end-effector.

Figure 1 shows two planning examples in the same configuration of objects, showing how it is important to set the objective value of the manipulation task appropriately. The target (blue circle) is surrounded by obstacles (empty circles) on a small constrained space like a shelf in a fridge. The space is bounded, so objects only can move through the bottom side because we also assume that the top side is blocked or not accessible. The gray area shows the reachable workspace of the manipulator. If the manipulator takes the path with the minimum distance to the target (Figure 4a), which is a common objective value in path planning, it should remove the four obstacles on the path (red circles). On the other hand, if the manipulator takes a longer path as shown in Figure 4b, the number of obstacles that should be rearranged reduces to two although the path is not distance-optimal. Since grasping objects in general takes longer time and more energy than just transporting them, distance-optimal paths oftentimes are not practically meaningful in this setting.

In this work, we consider the problem of grasping objects in clutter while considering the overall task that is taking out the target object for further uses. We define the number of obstacles that are relocated by the manipulator as the objective value to be minimized. We consider 2D configurations of objects (so no "picking from the top" action considered) where the objects are densely located thus no collision-free path of the end-effector can be found without rearrangement of the objects. The environment is not necessarily static. In order to deal with dynamic environments, we assume that the manipulator can perceive dynamic changes of the object configuration.

The most relevant work to our problem is presented in [3]. The planner in that work uses non-prehensile actions to make a collision-free path to an object in clutter. It finds obstacles to be removed by running a forward simulation

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: ckim@kist.re.kr.

detecting obstacles that could collide with the manipulator if it follows a distance-optimal path to the target. However, the plan taking the distance-optimal path may take longer time than a plan relocating the minimum number of obstacles (but with a suboptimal distance). This shows that travel distance of end effectors would not be an appropriate objective value to be minimized.

The following are contribution of this work:

- We define an appropriate criterion for the grasping task in clutter which achieves energy- and time-efficient manipulation.
- We propose a polynomial-time algorithm that computes a plan specifying the obstacles to be relocated and the order of relocation, while aiming to minimize the number of obstacles to be moved.
- We experimentally compare our method with a baseline method, which only considers distance-optimal paths of the end-effector, using various metrics.

II. Related Work

In [3], a planning framework that utilizes non-prehensile actions, such as pushing and pulling, to grasp an object in clutter. The framework removes obstacles that the end effector collides with if it follows a straight trajectory to the target. Although this method generates the distance-optimal path of the end effector, some obstacles could have to be removed unnecessarily since the objective value is not the number of obstacles to be removed. Also, the planners presented in [2], [6] do not directly optimize energy or time used for performing grasping tasks but mainly concern about validity of their plans.

Recently, a randomized motion planner is proposed to grasp an object in clutter where no collision-free path exists [7]. The planner generate a path that the manipulator follows while pushing obstacles along the path. However, the path is generated from a random tree expanding in a workspace whose expansion strategy only concerns improving coverage of the tree rather than the number of obstacles to be pushed to reach the target object.

In another recent work [8], pushing actions are generated from a learned policy using deep reinforcement learning. However, the configuration considered in the work has collision-free paths and the goal is to push the target along one of the collision-free path. Thus, no rearrangement of obstacles is considered. In addition, the work considers relatively simple cases which have obstacles up to four.

Among these, no work has directly tackled the problem of minimizing the number of obstacles to be relocated for efficient task performance, which is the goal of this work. By considering a more appropriate and practical objective value, we aim to perform the grasping task more efficiently.

III. Problem Formulation

The mission taking out a target object from clutter consists of several different processes that are, for example, perception, planning, grasping, and navigation. We focus on the planning problem of the manipulator especially the rearrangement of obstacles, which is shown to be NP-hard [9], [10]. Other processes are out of the scope of this work so we assume that the manipulator knows the configuration of objects perfectly before planning begins. The environment could change dynamically where the manipulator is assumed to be able to perceive any dynamic changes (e.g., updated locations of objects).

We consider configurations of one target object and N obstacles in a 2D space. We assume that the density of the workspace is very high. Formally, the minimum distance between any two adjacent obstacles i and j, i.e., d_{ij}, is shorter than the maximum distance r between two points on the contour of a closed shape representing the target. More formally,

$$d_{ij} < r, \forall i, j \text{ where } i \neq j. \tag{1}$$

Under this assumption, there is no collision-free path between the target and any outside location of the clutter if no obstacle is removed. The volume (or the size) of the manipulator can be ignored by expanding the sizes of the obstacles (i.e., generating a C-space). In the following, we assume that any given map of objects is already in the C-space representation.

In this setting, we set out to optimize the energy to complete the mission. Note that we assume that energy consumption is proportional to the time spent for the mission so will use energy and time interchangeably. We also assume that grasping actions take longer time or greater energy compared to transporting grasped objects. Then, this problem can be posed as an optimization problem where the objective value is the number of objects to be relocated to avoid collisions while the end-effector navigates between its initial position and the target object and vice versa. Then a mathematical formulation of the problem is $\min k$ subject to $o_i \in W \subset \mathbb{R}^2$ for $i = 1, \cdots, k$ where k is the number of obstacles to be relocated, o_i is the i-th obstacle to be relocated, and W is the reachable workspace of the manipulator.[1] We assume that all obstacles are located in the reachable workspace of the manipulator without loss of generality.

IV. Approach

In this section, we describe the algorithm and provide an analysis of the algorithm.

A. Algorithm

The algorithm consists of (i) graph structure construction and (ii) optimal path computation. Algorithm 1 describes the entire procedure. The algorithm receives the information of the objects (i.e., centroids of the target and obstacles, their radii). Particularly, the information includes the locations of the outermost obstacles in the reachable workspace. The

[1]A rigorous formulation of the problem may be a multi-objective optimization problem to consider both the number of objects to be relocated and the travel distance. But we assume that differences in travel distance among different solutions are not significant as the manipulator works in a relatively small space.

algorithm returns an ordered set of obstacles from the target to an outermost obstacle. The set tells the manipulator what and when to remove to reach the target.

Specifically, Algorithm 1 generates a Voronoi diagram $\mathbb{C}$ based on the centroids of all objects X (line 2). An unweighted graph $G(V, E)$ is generated using the Voronoi diagram $\mathbb{C}$ (line 3) where V and E represent the sets of vertices and edges of the graph, respectively. For each outermost object $t_i \in T$, the algorithm finds an ordered set $P_i = \{s, \cdots, t_i\}$ (line 5) where using Breadth-First Search (BFS) [11] where s is the target object. The graph is unweighted so the shortest path means the minimum-hop path from s to t_i. The algorithm maintains the shortest path P^* chosen among all shortest paths from s to each t_i (lines 6–8). For the paths with the same number of hops (i.e., ties), the one with the shorter Euclidean distance is chosen (lines 9–10). Using P^*, the manipulator can removes obstacles from t_i to reach s (the reverse order of P^*).

Algorithm 2 shows how the unweighted graph G is constructed from the Voronoi diagram $\mathbb{C}$. For each Voronoi cell $C_i \in \mathbb{C}$, vertex i is generated (line 3). Then FINDADJCELLS finds all adjacent cells[2] (line 4). For each adjacent cell C_j, vertex j is generated (line 6) and then i and j are connected with edge (i, j) (line 7). This generation of vertices and edges repeats for all $N + 1$ cells (including N obstacles and the target).

Figure 2 shows an instance of the problem and the result of running Algorithm 1. The left figure is the C-space of a configuration of objects in a 1 m × 1 m plane where the blue circle represents the target object (the objects are dilated by 4 cm considering the size of the end-effector). The center figure shows the Voronoi diagram generated from the C-space where the gray circles show the outermost objects in the workspace of the manipulator (suppose that the left and right sides are not accessible by the manipulator owing to the limited reachability or obstacles like walls, similar to the environment in Figure 1). Then the graph in the right figure is generated where the red edges show the optimal path between the target and one of the outermost objects. If the manipulator removes the obstacles on the path, the target can be taken out from the clutter. If the manipulator takes the distance-optimal path like [3] (the shortest line between the target and the bottom side), total five obstacles should be removed to avoid collisions. If the path shown in the right figure is taken, only two obstacles need to be relocated.

Now we provide a complexity analysis of the algorithm. Computing a Voronoi digram runs in time $O(|V| \log |V|) = O((N + 1) \log(N + 1)) = O(N \log N)$ using an efficient algorithm such as the method proposed in [12]. The graph generation shown in Algorithm 2 takes time $O(N^2)$ because FINDADJCELLS runs in $O(N)$ to check all cells except C_i, and this function is repeated $N + 1$ times (i.e., $O(N(N + 1)) = O(N^2)$). The graph $G(V, E)$ generated by Algorithm 2 has $N + 1$ vertices and at most $(N + 1)^2$ edges (if fully connected). Since BFS runs in time $O(|E| + |V|) = O(N(N + 1)/2 + N + 1) = O(N^2)$, calling this repeatedly for all outermost objects in T takes $O(N^3)$ as $|T|$ is at most N. Therefore, Algorithm 1 runs in polynomial time, which is $O(N \log N + N^2 + N^3) = O(N^3)$.

Since this algorithm runs efficiently in polynomial time, it can be used to recompute a new plan quickly if the configuration is updated according to more accurate or recent sensing information reflecting dynamic changes in the environment. The feasibility of using the algorithm for online replanning will be shown through extensive evaluations in the following section.

[2]If two cells share an edge in the Vorinoi diagram, they are adjacent. Please see the center figure of Figure 2 as an example Voronoi diagram.

Algorithm 1 MINREARRANGE

Input: the set of the centroids of all objects X, a target object s and the set T including the outermost obstacles in the reachable workspace of the end-effector
Output: a set of obstacles to be relocated P^* where $|P^*| = k$

```
1  $d^* = \infty$, $P^* = \emptyset$
2  $\mathbb{C} = $ VORONOI($X$)
3  $G(V, E) = $ GENGRAPH($\mathbb{C}$) ....// see Alg. 2 for the graph generation
4  for each $t_i \in T$ ..................// repeat for each outermost object
5    $P_i = $ SHORTESTPATH($G, s, t_i$) ........// run Breadth-First Search
6    if $|P_i| < d^*$ .........// if the path length is shortest, keep that path
7      $d^* = |P_i|$
8      $P^* = P_i$
9    else if $|P_i| = d^*$ and $l(P_i) < l(P^*)$ // if path lengths are the same,
   ..// keep the one w/ the min. Euclidean distance computed using $l(path)$
10     $P^* = P_i$
11   end if
12 end for
13 return $P^*$
```

Algorithm 2 GENGRAPH

Input: A Voronoi diagram $\mathbb{C} = \{C_1, \cdots, C_{N+1}\}$ where C_i is the i-th cell
Output: an unweighted graph $G = (V, E)$

```
1  $V = E = \emptyset$
2  for each $C_i \in \mathbb{C}$
3    $V \leftarrow V \cup i$
4    $\mathbb{C}_{\text{adj}} = $ FINDADJCELLS($\mathbb{C}, i$) .......// find all adjacent cells of $C_i$
5    for each $C_j \in \mathbb{C}_{\text{adj}}$
6      $V \leftarrow V \cup j$
7      $E \leftarrow E \cup (i, j)$ .........// connect edges between adjacent cells
8    end for
9  end for
10 return $G(V, E)$
```

V. EVALUATION

In this section, we evaluate the performance of the algorithm. First, we measure the running time of the algorithm with random instances. We also run experiments in a simulated environment to see how the proposed method improves performance of the grasping task practically. The result is compared with a baseline method which uses the distance-optimal path of the end effector presented in [3].

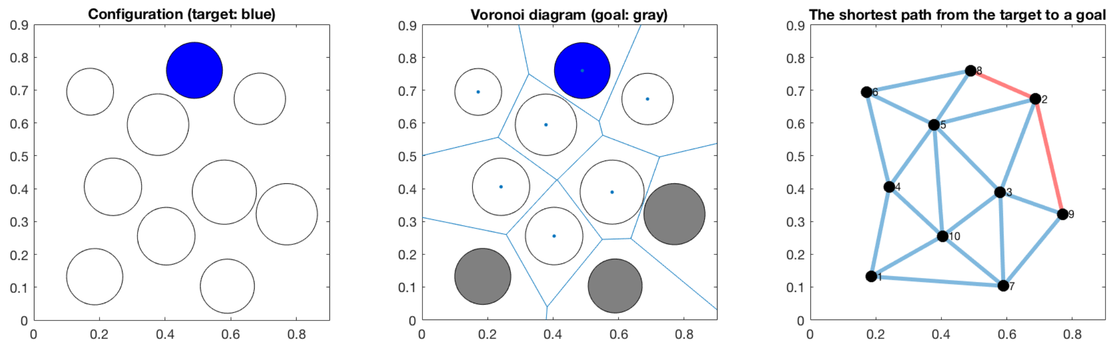


Fig. 2: An instance of the problem and the result of running Algorithm 1. (Left) A C-space of objects in a 1 m × 1 m plane where the blue circle represents the target object. (Center) The Voronoi diagram generated from the C-space where the gray circles show the outermost objects in the workspace of the manipulator (Right) A graph generated where the red edges show the optimal path between the target and one of the outermost objects. If the distance-optimal path (the shortest line between the target and the bottom side) is taken, four obstacles should be removed whereas the path shown in the right figure only requires relocation of two obstacles.

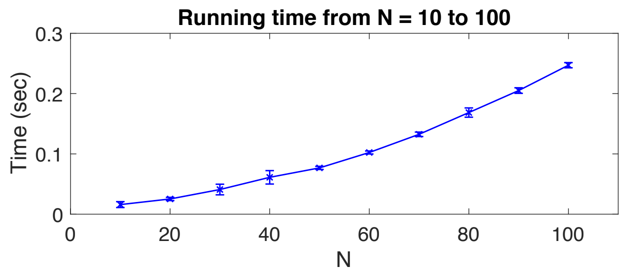


Fig. 3: Running time of Algorithm 1 from $N = 10, \cdots, 100$ with an interval of 10 (20 repetitions for each N).

TABLE I: Running time of Algorithm 1 (20 repetitions). The standard deviations are omitted as they are all smaller than 0.012.

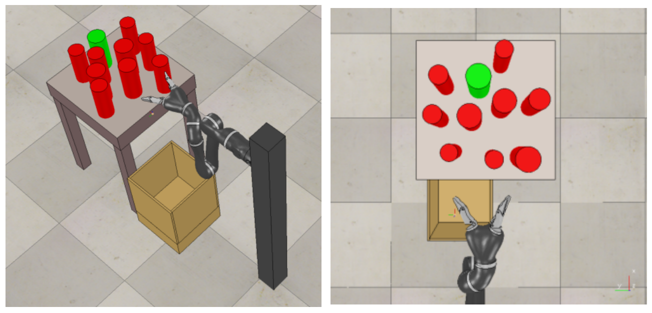
N	10	20	30	40	50
Time (sec)	0.016	0.025	0.0409	0.0611	0.0768
N	60	70	80	90	100
Time (sec)	0.1023	0.1325	0.1686	0.2050	0.2471

A. Performance of the algorithm

We test random instances to measure practical running time of Algorithm 1. Note that the system is with Intel Core i5 2.7GHz with 8G RAM and MATLAB R2017b. We randomly generate 20 instances for each N where N increases from 10 to 100 with an interval of 10. The result is shown in Figure 3 and Table I. The algorithm can compute a path within 0.25 sec for the instances with 100 obstacles, showing that the algorithm is suitable for real-time applications. Also, online replanning in dynamic environments is also possible up to several tens of obstacles (may vary depending on the system).

B. Experiments

1) Setting: We test the proposed algorithm in a simulated environment using a high fidelity robotic simulator V-REP [13] where dynamics can be modeled by one of several physics engines. An example configuration of the experiments is shown in Figure 4. Ten objects are randomly populated on a 0.5 m by 0.5 m tabletop, whose diameters are randomly sampled from [3.5, 5] whose unit is centimeter. A C-space is generated by dilating the objects by 2 cm to ignore the size of the end effector (notice that Figure 4 is not in the C-space representation). The green object is the target and the box below the table is prepared to place removed obstacles. We assume that the manipulator only can pull out objects through the bottom half of the table (the direction toward the manipulator's position) owing to the limited reachability of the end effector. For any random configuration, the position of the end effector is fixed in the middle at the bottom side of the table. The simulation is done in a system with Intel Core i7 4.20GHz and 8GB RAM.



(a) An example configuration (b) Top view

Fig. 4: Screenshots of the simulated environment.

From our proposed algorithm, a simple path of the end effector which consists of line segments connecting the obstacles in P^* is calculated. Note that this path does not consider kinematics of the manipulator, so the manipulator may collide with obstacles if it cannot follow the original path. Also, other possible (nonlinear) paths that modify the original linear path to detour obstacles are not considered.

2) Results: Table II shows the results from 10 random configurations. For the same configuration, we run both our method and a baseline method used in [3] choosing a distance-optimal path between the initial position of the end effector and the target object. We measure the number of relocated objects computed by Algorithm 1. We also record the number of collisions between the manipulator and the objects (those collisions that change object positions are counted only). We measure the total running time of the

Method	Baseline				Proposed			
Configuration #	(A) No. of relocated objects	(B) No. of collisions	(A+B)	Total time (sec)	No. of relocated objects	No. of collisions	(A+B)	Total time (sec)
1	3	1	4	103.4	2	1	3	91.3
2	4	0	4	131.6	2	0	2	91.4
3	3	0	3	103.4	1	1	2	51.0
4	5	0	5	159.7	2	0	2	91.4
5	4	0	4	131.5	2	1	3	91.4
6	5	0	5	159.7	2	0	2	91.4
7	5	0	5	159.5	2	0	2	91.3
8	6	0	6	187.8	2	0	2	91.4
9	5	1	6	159.5	2	1	3	88.0
10	4	1	5	131.5	2	2	4	91.3

(a) The results from 10 random configurations. The same configuration is used for the baseline and the proposed method for comparison.

Method	(A) No. of relocated objects	(B) No. of collisions	(A+B)	Total time (sec)
Baseline	4.4 (0.97)	0.3 (0.48)	4.7 (0.95)	142.7 (271.5)
Proposed	1.9 (0.32)	0.6 (0.70)	2.5 (0.71)	87.0 (126.8)

(b) The average and standard deviation (in parentheses) of 10 random trials.

TABLE II: The result from experiments in a simulated environment with $N = 9$ (10 trials).

grasping task to see how reducing the number of relocated obstacles decreases the total execution time as well as energy consumption. Note that no failure occurs in taking out the target from the cluttered environment.

Since the objects are very densely located, a distance-optimal path must pass the area occupied by many obstacles. On the other hand, the path generated from our proposed method reduces the number of obstacles to be relocated as it directly aims to minimize rearrangement of objects. The second and fifth columns in Table IIa show that our method can reduce the number of relocation in all configurations.

The proposed method incurs more collisions in average because the generated path does not consider kinematics of the manipulator. The baseline method also does not consider kinematics but the simple straight path does not necessarily require to take into account kinematics. In hindsight, the collided objects also should have to be relocated for collision-free manipulation. Thus, it is reasonable to compare the sum of the numbers of relocated objects and collided objects (i.e., the columns showing (A+B) in Table IIa). Overall, the proposed method has a smaller number of obstacles that should be relocated to achieve collision-free manipulation of the target (reduced from 4.7 to 2.5 as shown in Table IIb). This reduction brings a significant improvement to the completion time of the grasping task. In average, 55.7 sec is saved for the completion of the task by reducing the number of relocated obstacles. In other words, 39.0% of the completion time of the baseline method is reduced by computing a non distance-optimal path using our proposed algorithm which takes less than 0.02 sec for the given number of objects.

C. Discussion

Although the proposed algorithm finds the obstacles to be removed with their removal order quickly, avoiding collisions is not guaranteed since the algorithm only considers straight path segments connecting obstacles to be removed. To deal with possible collisions, we will further take into account collision avoidance using existing kinodynamic planners. For those cases where no collision-free path can be found even after considering collision avoidance, we are developing a collision checker running a forward simulation similar to the method described in [3]. The collision checker will enable finding obstacles around the path which can collide with the manipulator. If such obstacles are found, the manipulator can try with other same hop paths[3] not to have additional relocations of obstacles. Or the manipulator may decide to relocate the obstacles from the path. A unified approach will follow in our future work to include planning, collision checking, collision avoidance, control, and online replanning.

VI. Conclusion

In this work, we propose a fast planning algorithm for grasping an object in clutter which finds the minimum number of objects to be removed for the manipulation mission taking out the object. We provide evaluations of the algorithm and realistic experiments in a simulated environment. We show that the proposed algorithm (i) runs very quickly even with a large number of obstacles (less than 0.25 sec for 100 objects) and (ii) saves the entire running time of a manipulation mission significantly, compared with a baseline method which optimizes the travel distance of the end effector (39.0% of reduction).

In the future, we will consider the cases where the configuration of the objects are not perfectly perceived in advance, so the manipulator should deal with dynamically updated configurations in an online manner. We will also consider the problem of deciding where to relocate the obstacles and how to plan for the relocation. Non-prehensile actions like pushing and pulling can be considered as well for those obstacles that do not require grasping. Also, we will consider affordance of grasp by considering asymmetric objects. Lastly, we will develop a unified framework combining the separated processes of perception, planning, and execution.

[3]Notice that we choose the path with the minimum Euclidean distance in Algorithm 1 if there are ties.

References

[1] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.

[2] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 639–646.

[3] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, vol. 33, no. 3, pp. 217–236, 2012.

[4] G. Havur, G. Ozbilgin, E. Erdem, and V. Patoglu, "Hybrid reasoning for geometric rearrangement of multiple movable objects on cluttered surfaces," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.

[5] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Backward-forward search for manipulation planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6366–6373.

[6] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3075–3082.

[7] 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. 2, pp. 712–719, 2018.

[8] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang, "Rearrangement with nonprehensile manipulation using deep reinforcement learning," *arXiv preprint arXiv:1803.05752*, 2018.

[9] G. Wilfong, "Motion planning in the presence of movable obstacles," *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 1, pp. 131–150, 1991.

[10] 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.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[12] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1-4, p. 153, 1987.

[13] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 1321–1326.