Manipulation Among Movable Objects for Pick-and-Place Tasks in 3D Workspaces

Dhruv Mauria Saxena[1] and Maxim Likhachev[1]

***Abstract*— In cluttered real-world workspaces, simple pick-and-place tasks for robot manipulators can be quite challenging to solve. Often there is no collision-free trajectory that allows the robot to grasp and extract the desired object from the scene. This requires motion planning algorithms to reason about rearranging some of the *movable* clutter in the scene so as to make the task feasible. Our work focuses on solving these pick-and-place tasks in 3D workspaces where objects may tilt, lean on each other, topple, and slide. We formulate the problem as a search over a discrete graph – vertices are configurations of all movable objects in the scene, and edges encode rearrangement actions taken to reach one configuration from another. The search solves a multi-agent pathfinding abstraction of the problem to generate candidate nonprehensile and prehensile rearrangement actions. In order to account for complex multi-body interactions in the scene, and to ensure that object-centric "interaction constraints" are satisfied during all rearrangement actions, the search queries a rigid-body physics simulator when evaluating actions. This helps us guarantee that the solution we find (i) does not make contact with *immovable* obstacles, and (ii) does not tilt movable objects beyond allowed limits nor makes them fall out of the workspace or move with high velocities. In addition, we introduce an easy-to-implement parallelisation scheme to deal with uncertainty in relevant object parameters (mass and coefficient of friction). We present an extensive evaluation of this approach in simulation on scenes of varying difficulty with a PR2 robot.**

I. Introduction

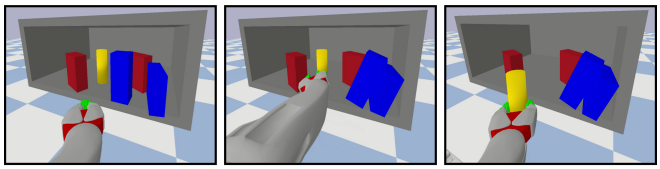
Manipulation Among Movable Objects (MAMO) [1] defines a general class of problems where a robot is given a manipulation task to complete in the presence of obstructing clutter in its workspace, some of which is known to be *movable*. In order to solve the manipulation task, the robot is allowed to rearrange the movable clutter while ensuring that it does not violate any object-centric constraints specified as part of the problem. These constraints typically encode desirable or acceptable qualities of robot-object interactions. For example, any *static* or *immovable* obstacles in the workspace will have interaction constraints associated with them that state neither the robot nor any movable object can make contact with them. Similarly, we can include constraints on how far the robot is allowed to tilt movable objects, whether they can be toppled over, and how fast the robot can move them around when pushing them.

This problem setup can be used to describe common uses of robot manipulators tasked with packing boxes in warehouses, assembling structures in manufacturing industries, and assisting humans inside households. As an example, Fig. 1 shows a MAMO problem a robot may be asked to solve. The task can be stated very simply – retrieve the can of beer (object-of-interest or OoI, outlined in yellow) from the refrigerator shelf. The complexity in solving this problem arises from the fact that the robot is only allowed to move the yogurt and almond beverage (movable objects outlined in blue). Furthermore, it cannot make these objects tilt beyond 25° along any axis, topple, or make contact with the (fragile) immovable obstacles outlined in red (eggs, cup of coffee, and glass bottles).

Fig. 1. An example MAMO problem to retrieve the beer can (object-of-interest or OoI, yellow outline). The yogurt and almond beverage are movable objects (blue outlines). All other objects in the scene are immovable obstacles (red outline).

Since the goal for the robot is specified with respect to the OoI only, solving MAMO problems of this nature requires answers to three additional questions – *which* objects should the robot rearrange, *where* should they be relocated, and *how* can they be moved while satisfying all interaction constraints. Existing approaches simplify the problem in one of two major ways. If we restrict the robot to rearrange objects via prehensile (pick-and-place) actions only [1]–[7], the task is greatly simplified to finding a sequence of collision-free trajectories that make the OoI retrievable. This requires relatively simple and computationally cheap collision checking since once any object is grasped by the robot, it can be assumed to be rigidly attached to the kinematic chain for motion planning purposes. However, these methods do assume access to known grasp poses for all objects, and known and accessible stable placement locations for them. Even if this was feasible, not all objects are graspable. In fact, for our PR2 robot, both the movable objects in Fig. 1

[1]The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. e-mail: {dsaxena, mlikhach}@andrew.cmu.edu.



(a) (b) (c)

Fig. 2. The robot pushes two movable objects to the right side of the shelf in order to retrieve the OoI. Even though the movable objects tilt and lean on each other, they remain within acceptable limits specified in the problem.

Algorithm 1 Discrete Search

```
1: procedure SEARCH(v_start, V_goal, f)
2:     OPEN ← ∅
3:     Insert v_start into OPEN with priority f(v_start)
4:     while OPEN is not empty and time remains do
5:         v ← OPEN.TOP()
6:         if v ∈ V_goal then
7:             return RECONSTRUCTPATH(v)
8:         for v' ∈ GETSUCCESSORS(v) do
9:             if EVALUATEACTION(v, v') then
10:                Insert/update v' in OPEN with priority f(v')
11:    return ∅
```

are wider than what can fit in one end-effector. Thus the PR2 can only rearrange them via nonprehensile (pushing) actions.

The second simplification used to solve MAMO problems allows nonprehensile rearrangements, and consequently uses a physics-based simulator to forward simulate the effect of robot actions on the configurations of objects in the scene. However, existing approaches limit nonprehensile interactions to a planar surface [8]–[14]. This simplification ignores realistic interaction constraints on objects tilting and toppling.

In prior work [15]–[17], we have addressed the problem of nonprehensile rearrangements in 3D workspaces where objects may tilt, lean, topple, slide etc. The key contribution in these works was formulating an abstract multi-agent pathfinding (MAPF) relaxation to the MAMO problem where the movable objects are artificially actuated and are therefore able to rearrange themselves out of the way of the robot arm as it retrieves the OoI. This directly informs us of rearrangement actions that are suitable for the robot to try, thereby answering the questions of *which* objects the robot should rearrange and *where* they should be moved. A push action planner uses the MAPF solution to compute pushing trajectories to realise the suggested rearrangements in the real-world. Our planning algorithms are given access to a physics-based simulator (PyBullet [18]) to ensure all interaction constraints are satisfied while solving the task. The simulator is used to forward simulate the effect of any pushing actions and check for any constraint violations. Fig. 2 shows a solution found by our M4M ("Multi-Agent Pathfinding for Manipulation Among Movable Objects") algorithm [15] being executed in simulation.

This paper builds on the Enhanced-M4M or E−M4M algorithm presented in [16]. E−M4M solves MAMO problems by formulating a discrete graph search that searches over orderings of object rearrangements, different rearrangements of the scene, and different ways to rearrange each object. E−M4M only allowed the robot to use pushing actions to rearrange movable objects. In this paper we introduce the use of pick-and-place style prehensile rearrangements as well. We also relax the assumption made by M4M and E−M4M that required perfect knowledge of the physics properties (mass and coefficient of friction) for all objects.

II. Background: Discrete Graph Search

Algorithm 1 presents a pseudocode for a discrete graph search algorithm. We will use this to highlight the corresponding key pieces of the E−M4M algorithm. The SEARCH method is a basic graph search over a discrete graph $G = (V, E)$ where vertices $v \in V$ are the search *states* (V is thus the search *space*) and edges $e = (u, v) \in E$ represent actions/transitions that take us from state u to state v. SEARCH takes three input arguments – v_{start} is the start state or root of the search tree from where we would like to find a path, $V_{\text{goal}} \subset V$ is the set of goal states where a solution path may terminate, and f is a priority function that allows us to preferentially select more promising states (usually in terms of "closeness" to V_{goal}) to grow the graph from.

$OPEN$ is a priority queue that contains states ordered according to the function f. The function GETSUCCESSORS : $V \rightarrow \mathcal{P}(V)$ returns the set of all states that may be reached from the input state. As such, the set of states returned by GETSUCCESSORS(v) can be considered successors/neighbours of v in G. In order to check whether the transition from v to one of its neighbours v' is valid, we call EVALUATEACTION(v, v'). If the transition is found to be valid, we may consider further growing the graph (towards V_{goal}) from v'. Once we reach a state in V_{goal}, we can backtrack from it to v_{start} along valid edges in the graph and return the solution path found.

III. E−M4M Graph Search

Each state in the E−M4M graph includes the configuration of all objects in the scene. The corresponding search space for the E−M4M search is then the space of all rearrangements that may be achieved. A valid edge $e = (v, v')$ between two states implies that we have found a robot arm trajectory that rearranges the objects from configuration v to v' while not violating any interaction constraints. E−M4M is unique in the way it implements the GETSUCCESSORS and EVALUATEACTION functions for MAMO problems.

For a state v input to GETSUCCESSORS, in order to determine which other states v' are its neighbours, E−M4M formulates and solves an abstract MAPF problem. Specifically, it first computes a robot arm trajectory that successfully retrieves the OoI in the *absence* of movable objects. The volume swept by the robot arm along this trajectory creates a "negative goal region" (NGR) [9] that, if devoid

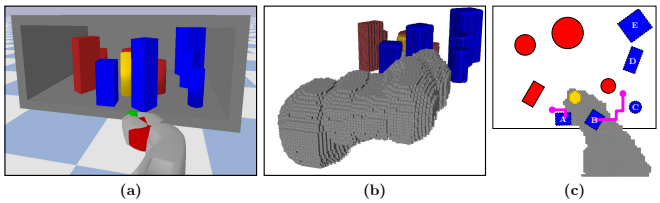


Fig. 3. (a) A simulated MAMO problem where the robot must extract the yellow OoI. (b) The initial negative goal region (NGR) computed for this scene is shown in gray, with collision models for all other objects shown in appropriate colours. (c) A 2D projection of the MAPF solution shows that objects A and B vacate the NGR along the pink paths.

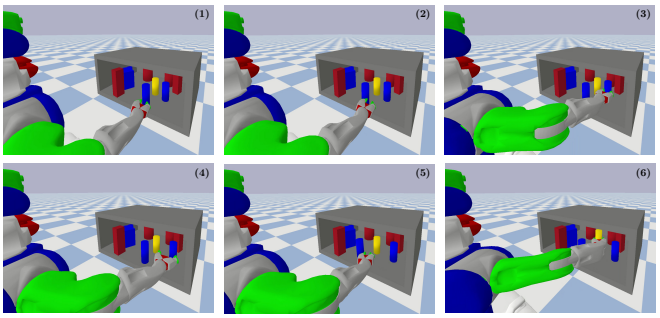


Fig. 4. (1) – (2) The movable cylinder in the front is pushed to the left. (3) – (4) The movable cylinder in the back is picked up and moved to the right. (5) The movable cylinder in the front is pushed further to the left. (6) The OoI can now be retrieved.

of movable objects, will allow successful retrieval of the OoI. For the abstract MAPF problem, movable objects are artificially actuated and tasked with moving themselves out of the initial NGR while avoiding collisions with each other and all immovable obstacles. We use Conflict-Based Search (CBS) [19] as our MAPF solver. Fig. 3 shows a simulated MAMO problem, the initial NGR for the problem, and a 2D visualisation of the MAPF solution for the problem. For *each* object (independent of others) that moves as part of the MAPF solution, E–M4M creates a successor state where the desired configuration for that object is the one achieved at the end of its path in the MAPF solution.

In order to verify whether (i) there exists an action (robot arm trajectory) corresponding to this transition, and (ii) whether that action is valid with respect to interaction constraints, E–M4M queries a nonprehensile push planner and physics-based simulator within the EVALUATEACTION function. The EVALUATEACTION function can be broken down into three main steps. First, E–M4M computes an appropriate end-effector pose in $SE(3)$ on the basis of the path found in the MAPF solution and plans a trajectory to this pose in the configuration space of the robot (while avoiding collisions with *all* objects). If this trajectory is found, it uses inverse kinematics (IK) (along the MAPF path) to compute the pushing action. If the pushing action is found (IK does not hit joint limits or any immovable obstacles), the pushing action is forward simulated in the physics-based simulator for constraint verification. This helps not only determine the validity of an edge $e = (v, v')$ in the E–M4M search graph, but also the exact state v' achieved by the action corresponding to this edge (if valid).

IV. MODIFICATIONS TO E–M4M

A. Adding Prehensile Rearrangement Actions

Given the structure of the E–M4M graph search, it is straightforward to add different rearrangement actions that we would like the robot to be capable of executing. In this paper, we introduce the use of pick-and-place rearrangement actions within the E–M4M algorithm. The GETSUCCESSORS routine is mostly unchanged; only the instantiation of outgoing edges from a state v is modified. For objects that are "graspable", i.e. those that will fit inside the end-effector for our robot, we now create two successor states v'_1 and v'_2 corresponding to rearranging an object by a push action and pick-and-place action respectively. The edge $e = (v, v'_1)$ is evaluated the same as before, described in Section III. Edges corresponding to pick-and-place rearrangement actions such as $e = (v, v'_2)$ proceed similarly – we first compute a pick-and-place trajectory, and if one is found, validate it in the simulator.

In order to compute a pick-and-place trajectory, we do assume access to known grasp poses for each object. The rearrangement action itself can then be broken down into four parts – a trajectory to reach the grasp pose, the grasp maneuver, a trajectory to reach the placement pose, and the placement action. For pick-and-place rearrangement actions, we impose the restriction that the entire trajectory must be free of collision with all objects (movable or immovable). The placement pose is obtained from the MAPF solution path for the object we want to rearrange. Grasping and placement actions are hardcoded movements of the end-effector relative to the object from the grasp pose and placement pose repectively. If such a collision-free trajectory is found, we go on to simulate the grasping and placement actions in the simulator to ensure that no interaction constraints are violated. We do not need to simulate any other parts of the trajectory since by construction they are collision-free and hence contact-free and therefore cannot violate any interaction constraints. Fig. 4 shows a sequence of images where we are able to rearrange the scene with a combination of nonprehensile and prehensile actions in order to solve the MAMO problem.

B. Robustness to Object Parameter Uncertainty

Prior to planning, E–M4M assumes perfect knowledge about which objects are in the scene, where they are, and what their physical properties (masses, and coefficients of friction) are. There has been a lot of advancement in localising objects in a scene with great accuracy [20], [21]. However, accurately perceiving the mass of an object or computing its coefficient of friction remains a challenge. We would like to be robust to some bounded uncertainty in these physical properties of objects.

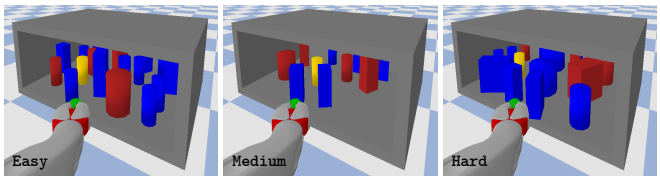


Fig. 5. MAMO problems of varying difficulty. The difficulty level is determined by the number of movable objects that overlap with the initial NGR we compute (sec:em4m).

Fortunately, our use of a physics-based simulator for action evaluation makes this easy to implement. Taking inspiration from existing work [22], [23], we utilise parallelised simulations of the same trajectory in scenes where each object is instantiated with different values of their mass and coefficient of friction. To be precise, given some budget on the number of parallelised simulators we are allowed to launch, we instantiate multiple copies of each object within each simulator. For each object copy, we sample its mass and coefficient of friction from within the known bounded uncertainty we assumed for that parameter. We ensure that contacts and collisions within the simulator are computed appropriately – object copies do not collide with each other, but they do collide with one copy each of all other objects. An action is valid or not based on some user-defined threshold $\delta \in (0, 1]$ for success. Given N parallel simulators and M copies of each object, we say that an action is valid if $\max(1, \lfloor \delta N M \rfloor)$ samples are valid in simulation.

V. Experimental Analysis

We compare the performance of the original version of E−M4M [16] with the one introduced in this paper. For the version introduced in this paper, we use $N = 2$ simulator instances in parallel each containing $M = 4$ copies of an object. With $\delta = 0.8$, we say that an action is valid if it succeeds in $\max(1, \lfloor 0.8 \times 2 \times 4 \rfloor) = 6$ simulations (out of $2 \times 4 = 8$). E−M4M is effectively run with $N = 1$, $M = 1$, and $\delta = 1$. For planning purposes, including all simulations, E−M4M uses sampled values for all parameters. However, to check whether a found solution will succeed in the real-world, we simulate it with the true object parameters.

Both algorithms are run on the same 36 randomly generated MAMO scenes containing one OoI, four immovable obstacles, and varying numbers of movable objects with a 180s planning timeout. The difficulty of a scene is determined by the number of movable objects that overlap with the initial NGR we compute (details in Section III) – one for `Easy`, two for `Medium`, and three or more for `Hard` problems. Fig. 5 shows an example scene of each difficulty level. We include 12 scenes from each difficulty level.

Table I shows the result of this quantitative comparison between the original version of E−M4M and the one presented in this paper. We present the number of problems solved by each algorithm, and for the solved problems only we provide numbers for – the average number of rearrangement actions in the solution, minimum/median/maximum times spent overall, solving MAPF problems, and simulating actions. We can see that with the addition of prehensile rearrangement actions, and making the algorithm robust to parameter uncertainties, we are able to solve more problems with a smaller sequence of rearrangement actions on average. Clearly, for the version of E−M4M presented in this paper, we would expect the algorithm to spend more time computing trajectories for rearrangement actions to be simulated since in addition to the pushing trajectories computed by E−M4M, we are now also computing pick-and-place trajectories. This is reflected in the amount of the total planning time that is not spent solving MAPF problems and not spent simulating actions. The benefit of including prehensile rearrangements in particular is reflected in less time spent solving MAPF problems and simulating actions. This is because if we find a valid pick-and-place rearrangement action, given the strict conditions required of it (Section IV-A), we exactly achieve the desired configuration of a movable object as suggested by our MAPF solver. This results in much greater progress towards the goal of solving the MAMO problem in comparison to a push action which is subject to the complex multi-body contact dynamics that are not modeled by the MAPF solver and can lead to a successor state much different than the one the MAPF solver wanted to achieve.

TABLE I

MAMO Planning with E−M4M- problems solved, *avg* ± *std* number of rearrangement actions, and *min/median/max* planning, MAPF, and simulation times

Metrics	Difficulty	Planning Algorithms	
		E−M4M [16]	This Paper
Problems Solved	`Easy`	11	12
	`Medium`	10	12
	`Hard`	9	8
# Rearrangement Actions	`Easy`	2.73 ± 1.0	2.5 ± 0.9
	`Medium`	3.9 ± 1.37	3.42 ± 1.31
	`Hard`	4.22 ± 1.40	3.38 ± 1.3
Total Time (s)	`Easy`	6.6 / 11.3 / 57.7	7.2 / 17.0 / 162.5
	`Medium`	10.7 / 37.2 / 149.3	7.5 / 55.2 / 128.0
	`Hard`	35.1 / 80.0 / 108.6	37.0 / 55.6 / 95.5
MAPF Time (s)	`Easy`	0.006 / 0.04 / 0.9	0.005 / 0.1 / 1.9
	`Medium`	0.02 / 0.3 / 12.3	0.01 / 0.1 / 3.6
	`Hard`	0.07 / 0.3 / 3.5	0.05 / 0.3 / 0.7
Simulation Time (s)	`Easy`	3.3 / 7.1 / 46.5	2.0 / 9.6 / 77.2
	`Medium`	5.7 / 17.6 / 84.6	4.0 / 17.3 / 46.1
	`Hard`	26.7 / 47.0 / 75.9	18.5 / 26.0 / 48.4

VI. Conclusion

This paper extends our prior work on solving MAMO problems with a graph search algorithm E−M4M [16]. We introduce the use of prehensile rearrangement actions within E−M4M, and make it robust to known and bounded uncertainties in physical properties of objects by parallelising the simulation of actions to check their validity. Quantitatively we show the improvements these additions provide over the original E−M4M algorithm.

REFERENCES

[1] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *2007 IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2007.

[2] A. Krontiris, R. Shome, A. Dobson, A. Kimmel, and K. E. Bekris, "Rearranging similar objects with a manipulator using pebble graphs," in *14th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2014, Madrid, Spain, November 18-20, 2014*. IEEE, 2014, pp. 1081–1087.

[3] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, L. E. Kavraki, D. Hsu, and J. Buchli, Eds., 2015.

[4] J. Lee, Y. Cho, C. Nam, J. Park, and C. Kim, "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments," in *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. IEEE, 2019, pp. 183–189.

[5] C. Nam, J. Lee, S. 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 2020, Paris, France, May 31 - August 31, 2020*. IEEE, 2020, pp. 3777–3783.

[6] R. Shome and K. E. Bekris, "Synchronized multi-arm rearrangement guided by mode graphs with capacity constraints," in *Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2021*, S. M. LaValle, M. Lin, T. Ojala, D. A. Shell, and J. Yu, Eds., 2021.

[7] R. Wang, K. Gao, J. Yu, and K. Bekris, "Lazy rearrangement planning in confined spaces," in *International Conference on Automated Planning and Scheduling*, 2022.

[8] J. P. van den Berg, M. Stilman, J. Kuffner, M. C. Lin, and D. Manocha, "Path planning among movable obstacles: A probabilistically complete approach," in *Selected Contributions of the Eighth International Workshop on the Algorithmic Foundations of Robotics, WAFR 2008*, H. Choset, M. Morales, and T. D. Murphey, Eds., 2008.

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

[10] J. E. King, "Robust rearrangement planning using nonprehensile interaction," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, December 2016.

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

[12] B. Huang, S. D. Han, J. Yu, and A. Boularias, "Visual foresight trees for object retrieval from clutter with nonprehensile rearrangement," *IEEE Robotics Autom. Lett.*, vol. 7, no. 1, pp. 231–238, 2022.

[13] S. Park, Y. Chai, S. Park, J. Park, K. Lee, and S. Choi, "Semi-autonomous teleoperation via learning non-prehensile manipulation skills," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[14] E. Vieira, D. Nakhimovich, K. Gao, R. Wang, J. Yu, and K. E. Bekris, "Persistent homology for effective non-prehensile manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[15] 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)*, 2023, pp. 8141–8147.

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

[17] ——, "Imagine all objects are robots: A multi-agent pathfinding perspective on manipulation among movable objects," in *AAAI Workshop on Multi-Agent Path Finding*, 2023.

[18] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[19] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.

[20] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, H. Kress-Gazit, S. S. Srinivasa, T. Howard, and N. Atanasov, Eds., 2018.

[21] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Conference on Robot Learning (CoRL)*, 2018. [Online]. Available: https://arxiv.org/abs/1809.10790

[22] M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa, "Robust trajectory selection for rearrangement planning as a multi-armed bandit problem," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*. IEEE, 2015, pp. 2678–2685.

[23] W. C. Agboh and M. R. Dogar, "Robust physics-based manipulation by interleaving open and closed-loop execution," *CoRR*, vol. abs/2105.08325, 2021. [Online]. Available: https://arxiv.org/abs/2105.08325