ReloPush: Multi-object Rearrangement in Confined Spaces with a Nonholonomic Mobile Robot Pusher

Jeeho Ahn and Christoforos Mavrogiannis

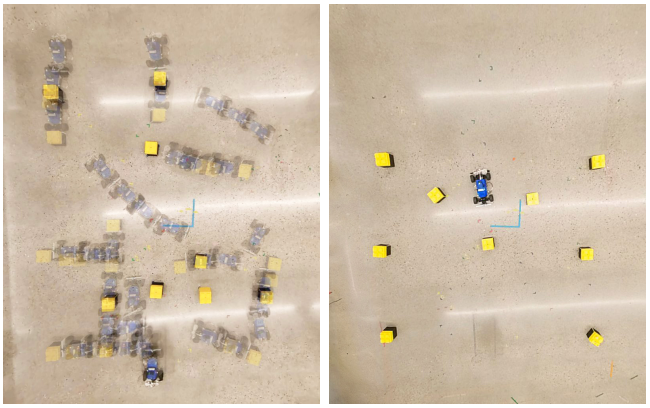
***Abstract*— We focus on push-based multi-object rearrangement planning using a nonholonomically constrained mobile robot. The simultaneous geometric, kinematic, and physics constraints make this problem especially challenging. Prior work on rearrangement planning often relaxes some of these constraints by assuming dexterous hardware, prehensile manipulation, or sparsely occupied workspaces. Our key insight is that by capturing these constraints into a unified representation, we could empower a constrained robot to tackle difficult problem instances by modifying the environment in its favor. To this end, we introduce a *Push-Traversability* graph, whose vertices represent poses that the robot can push objects from, and edges represent optimal, kinematically feasible, and stable transitions between them. Based on this graph, we develop *ReloPush*, a graph-based planning framework that takes as input a complex multi-object rearrangement task and breaks it down into a sequence of single-object pushing tasks. We evaluate ReloPush across a series of challenging scenarios, involving the rearrangement of densely cluttered workspaces with up to nine objects, using a 1/10-scale robot racecar. ReloPush exhibits orders of magnitude faster runtimes and significantly more robust execution in the real world, evidenced in lower execution times and fewer losses of object contact, compared to two baselines lacking our proposed graph structure.**

I. Introduction

Autonomous mobile robots have revolutionized fulfillment by offering a robust and scalable solution for completing massive rearrangement tasks. Fulfillment sites tend to be highly structured, requiring extensive workspace engineering, like the installation of rails along rectilinear grids, and specialized docking mechanisms for handling packages. While effective, this approach can be prohibitively costly and impractical for many critical domains like construction, waste management, and small/medium-sized warehouses. These environments involve rearrangement tasks for a wide range of object geometries, require precise navigation among static and movable objects contrast, and impose practical constraints like respecting tight geometric boundaries and robot kinematics.

A practical technique to expand the diversity of objects that mobile robots can rearrange is pushing. Pushing is a form of *nonprehensile manipulation*, a class of manipulation that does not require secure grasping but rather exploits the task mechanics to rearrange an object [22]. This is appealing because it enables the rearrangement of large, heavy, or unstructured objects even by simple mechanisms. However, pushing introduces constraints to robot motion: to ensure object stability, robots need to avoid abrupt turns and high accelerations.

Department of Robotics, University of Michigan, Ann Arbor, USA. Email: {jeeho, cmavro}@umich.edu



(a) Executing a rearrangement plan. (b) Resulting rearrangement.

Fig. 1: In this work, we describe ReloPush, a planning framework for tackling multi-object rearrangement tasks with a nonholonomic mobile robot pusher.

Real-world applications impose additional pragmatic constraints that need to be taken into account, i.e., geometric constraints imposed by the workspace boundary and by obstacles within, and robot kinematics (e.g., nonholonomic constraints). These constraints, in conjunction with push-stability constraints may render many practical rearrangement tasks infeasible. Prior work has tackled constrained rearrangement tasks by embracing the paradigm of *planning among movable objects* [33, 35], in which the robot strategically modifies its environment to simplify planning. Previous applications emphasize the use of dexterous manipulators that are often capable of unconstrained overhand grasping, ignore orientation specifications on goal object poses, and assume generous workspace boundaries. These assumptions severely limit the potential of robot deployments for achieving practical productivity in real-world applications.

In this work, we focus on the problem of rearranging a set of objects into a set of desired final poses within a confined workspace via pushing, using a nonholonomic mobile robot pusher. Our key insight is that we could capture geometric, physics, and kinematic constraints into a unified representation that could enable the robot to understand when and how to modify the environment to complete its downstream rearrangement task. To this end, we introduce a *push-traversability* graph whose edges represent kinematically feasible and stable object displacements. By planning on this graph, we achieve fast and effective rearrangement planning that can handle multiple objects in a confined space as shown in Fig. 1. Our code can be found at https://github.com/fluentrobotics/ReloPush and footage from our experiments at https://youtu.be/_EwHuF8XAjk.

II. Related Work

Planning among Movable Obstacles. Many real-world environments include movable obstacles that robots can manipulate to free up space to make path planning feasible. This problem, first formulated by Wilfong [40] is PSPACE-hard when the final positions of objects are specified (these instances are known as *labeled*) and NP-hard otherwise (instances known as *unlabeled*). However, the relevance of the problem has motivated extensive investigation. Chen and Hwang [5] devised a hierarchical planner that proved effective in environments with polygonal obstacles. Stilman and Kuffner [34] demonstrated real-time planning among movable obstacles in realistic, cluttered households. Later approaches exhibited important properties such as probabilistic completeness [38] and extensions to manipulation with high-DoF (degrees of freedom) arms [11, 15, 35].

We tailor the concept of planning among movable objects to a nonholonomic pusher, leveraging insights from Dubins path classification [20, 30]. In cases where the optimal rearrangement path violates the workspace boundary, we modify the object's starting pose to transition to a different optimal solution. When an obstacle is blocking an optimal object rearrangement, we remove the object to clear the way, by building on prior work on traversability graphs [24].

Rearrangement Planning. In rearrangement problems, the goal is to move a set of objects to (possibly predefined) goal poses. These are divided into two classes [26]: *monotone* instances, where each object only needs to be moved once, and *non-monotone* ones, requiring more than one movement per object. Much of the prior work focuses on monotone instances [3, 33, 35, 37], but real-world, densely cluttered spaces often give rise to non-monotone instances. While these have been shown to be NP-hard [14], recent algorithms have demonstrated practical performance in manipulation tasks [10, 14, 17, 27, 28]. While many works account also for geometric [1, 2, 24] and kinodynamic constraints [29], most approaches make simplifying assumptions such as tasks without object orientation constraints, holonomic robots [25], and high-DoF manipulators [17, 18, 27].

We target monotone and non-monotone, labeled problem instances for a pusher that accounts for geometric, kinematic, and stability constraints. Through a novel *push-traversability* graph whose edges incorporate all of these constraints, we manage to plan for challenging rearrangement instances in orders of magnitude lower runtimes than baselines.

Nonprehensile Rearrangement. Some works tackle rearrangement tasks using nonprehensile manipulation. Dogar and Srinivasa [6] describe a planner that iteratively removes clutter via pushing to retrieve objects of interest in cluttered tabletops. King et al. [15] develop a trajectory optimizer that pushes objects to simplify downstream manipulation tasks. Huang et al. [14] use iterated local search to handle problems like singulation and sorting in densely cluttered tabletops. Talia et al. [37] use multiagent pathfinding to distribute rearrangement tasks to a team of nonholonomic pushers. Some works focus on modeling the dynamics of pushing [4, 13] to inform motion planning [11] whereas others learn adaptive pushing control policies [19, 41] using data-driven techniques. For many domains, a practical assumption involves quasistatic pushing, for which analytical motion models exist [9, 12, 21, 23]. Quasistatic pushing can empower simple control laws or even open-loop systems to perform robustly on many real-world problems.

While much of prior on nonprehensile rearrangement planning assumes high-DoF manipulators [4, 11, 14, 15, 17, 18, 27, 35], we demonstrate the practicality of quasistatic pushing on a nonholonomic mobile robot pusher [32]. We move beyond prior work on nonholonomic nonprehensile rearrangement planning [16, 37, 37] by handling non-monotone problem instances in densely cluttered spaces (up to nine blocks of $0.15m$ side in a $4 \times 5.2m^2$ area).

III. Problem Statement

We consider a mobile robot *pusher* and a set of m polygonal *blocks* in a workspace $\mathcal{W} \subset SE(2)$. We denote the state of the pusher as $p \in \mathcal{W}$ and the states of the blocks as $o_j \in \mathcal{W}$, $j \in \mathcal{M} = \{1, \ldots, m\}$. The pusher follows rear-axle, simple-car kinematics $\dot{p} = f(p, u)$, where u represents a control action (speed and steering angle), and may push objects using a flat bumper attached at its front (see Fig. 7). The goal of the pusher is to rearrange the blocks from their starting configuration, $O^s = (o_1^s, \ldots, o_m^s)$, to a goal configuration, $O^g = (o_1^g, \ldots, o_m^g)$. We seek to develop a planning framework to enable the pusher to efficiently rearrange all objects into their goal poses. We assume that the pusher has accurate knowledge of its ego pose at all times, and of the starting configuration of all objects, O^s.

IV. ReloPush: Nonprehensile Multi-object Rearrangement

We describe ReloPush, a planning framework for multi-object rearrangement via pushing. ReloPush breaks down a complex rearrangement task into an efficient sequence of single-object push-based rearrangement subtasks.

A. System Overview

Given a workspace $\mathcal{W}$, an initial robot pose p_s, and a set of objects that need to be reconfigured from their starting poses O^s to their goal poses O^g, ReloPush finds a sequence of rearrangements in a greedy fashion. It first constructs a rearrangement graph (PT-graph) that accounts for robot kinematics, push stability, and workspace boundary constraints. Using graph search, ReloPush searches for the collision-free object rearrangement path of lowest cost. If such a path is found, the graph is updated to mark the rearranged object as an obstacle, and the planner is invoked again to find the next rearrangement of lowest cost. If the path found passes through a blocking object, ReloPush displaces that object out of the way first. If the path violates the workspace boundary, ReloPush displaces the object to be pushed until the path to its goal meets the boundary constraint. If the path is infeasible (i.e., fails to find a motion to approach the object to push), it replans with next rearrangement candidate that

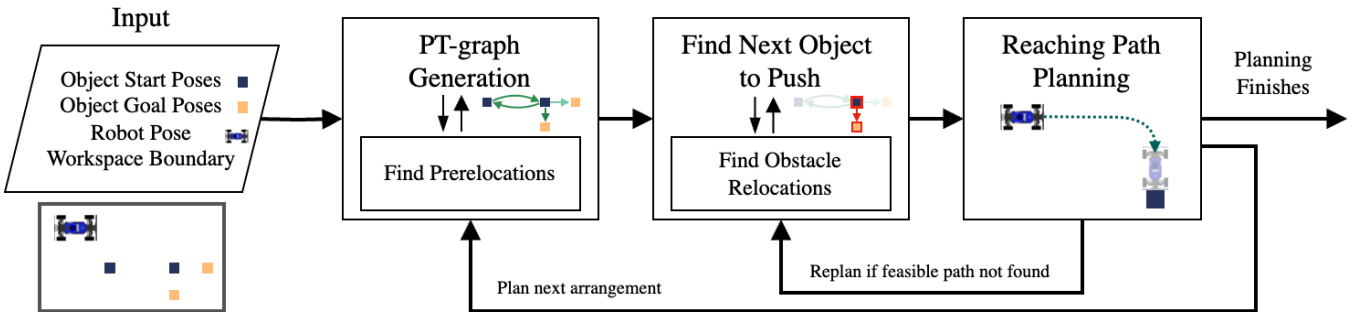
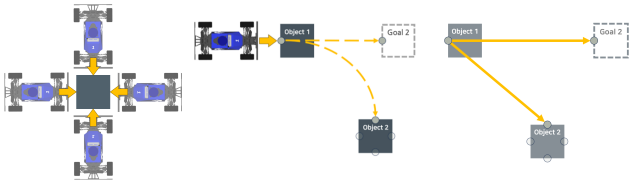
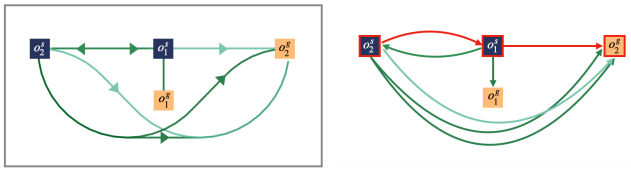


Fig. 2: The ReloPush architecture. Given the initial pose of the pusher and a rearrangement task in the form of start/goal object poses, ReloPush plans an efficient sequence of rearrangement subtasks to be executed by the robot via pushing.



(a) Pushing poses (b) Path Plan (c) PT-graph

Fig. 3: PT-graph generation. (a) First, every object is assigned K pushing poses (e.g., a cubic object has 4 pushing poses). (b) For any pair of pushing poses, we check if a collision-free path that respects the steering limit for quasistatic pushing can be drawn. (c) For each valid path, we construct a directed edge between its start/goal vertices.



(a) Object traversability. (b) PT-graph for the task in (a).

Fig. 4: (a) Two objects (navy squares) need to be rearranged to goal poses (yellow squares). (b) The PT-graph: nodes are pushing poses and edges are Dubins paths connecting them. By searching the graph, we can determine if any blocking objects need to be removed. For instance, the initial pose of object 1 is found to be blocking the shortest rearrangement of object 2 (red path).

has the next lowest cost. This process is repeated until a full rearrangement sequence for all objects is found. An overview of our architecture is shown in Fig. 2.

B. Push-Traversability Graph

A *traversability* graph (T-graph) is a representation of how movable objects can be reconfigured in a cluttered scene [24]. In its original form, vertices represent (starting and goal) positions of objects and edges represent collision-free transitions between them. By searching the graph, a collision-free rearrangement plan can be found.

Here, we build on the T-graph representation to introduce the *push-traversability* graph (PT-graph) $G(V, E)$, which not only captures the spatial relationships among movable objects but also integrates the kinematic constraints of the pusher and push-stability constraints of objects within the edges. Because in push-based manipulation of polygonal blocks, the block orientation is important, each vertex in our graph $v_i \in V$ represents a valid robot *pushing pose* p_i, i.e., a pose from which the pusher can start pushing a block (see Fig. 3).

For any pair of vertices (v_s, v_g) representing a pair of start and goal *pushing poses* (p_s, p_g), a directed edge e is formed from v_s to v_g if the optimal path from p_s to p_g is collision-free and within the workspace boundary. Optimality in the transitions is motivated by the heavily constrained problem domain which further demands efficient use of space. For a nonholonomically constrained mobile robot, the optimal path from p_s to p_g is a Dubins curve, and can be synthesized using L, R, and S primitives corresponding respectively to left, right, and straight motion [8]. To account for push stability, the L/R primitives are implemented using a minimum turning radius ρ that ensures stable pushing under the quasistatic assumption [9, 15, 21, 37]. If the Dubins curve is collision-free and within the boundary, a directed edge is constructed from v_s to v_g, and assigned a weight that is equal to the length of the curve. The edge direction is dictated by pushing stability constraints (e.g., contact cannot be maintained if the pusher moves backwards). Alg. 1 describes the graph construction and Fig. 4 shows an example.

C. Prerelocation: Change of Starting Pushing Pose

Often, an edge between two vertices cannot be formed because the Dubins curve connecting them violates the workspace boundary. This is especially common due to the limited turning radius ρ imposed by the push stability constraint (see Fig. 5). Our insight is that a small change in the starting pushing pose might allow for an optimal, collision-free rearrangement that lies entirely within the workspace boundary. To this end, we leverage prior work on the classification of Dubins curves [20, 30]. Intuitively, because of the robot's kinematic constraints, if the start and goal poses are "too close", the Dubins curve connecting them will tend to require a sequence of wide turns that violate the workspace boundary. In particular, if the Euclidean distance d between the start and goal poses, normalized by the turning radius ρ, is smaller than a threshold d_{th}, the Dubins curve connecting them is a *Short* path that will likely include an excessive turning arc. If $d > d_{th}$, the corresponding Dubins curve is a *Long* path that will likely not require excessive wide turning. This threshold can be found to be $d_{th} = |\sin\alpha| + |\sin\beta| + \sqrt{4 - (\cos\alpha + \cos\beta)^2}$, where α and β represent the start and goal orientations with respect to the line connecting them and transformed to be horizontal [20, 30]. Fig. 5 shows two paths starting from

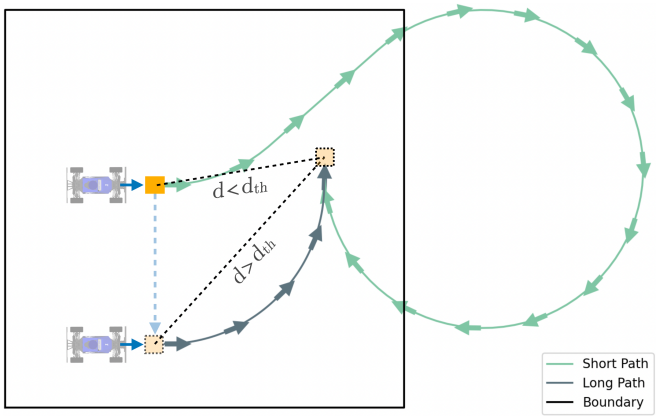


Fig. 5: Two Dubins curves with the same goal pose (top right) and maximum turning radius. When the start pose is too close to the goal ($d \leq d_{th}$), the resulting Dubins curve (green color) is a *Short Path* involving large turns violating the workspace boundary. Using Dubins path classification [20, 30], we can determine a *prerelocation* of the object's starting pose to allow reaching the goal via a *Long Path* ($d > d_{th}$) which will involve smaller turning arcs (gray color).

different poses and leading to the same goal pose via a *Long* and a *Short* path. By moving the object's starting pose slightly towards the bottom, a *Long* Dubins curve leading to the goal pose can be found to be within bounds.

For instances where an object rearrangement path is found to be out-of-bounds and meet the condition of a *Short* path (with excessive turning, i.e., $d \leq d_{th}$), we attempt to find another start pose p'_s that can be connected to p_g via a collision-free *Long* Dubins curve and that is reachable by the robot from p_s. We refer to this change of starting pose from p_s to p'_s as a *Prerelocation*. To find p'_s, we evenly sample configurations along the pushing directions of the object (see Fig. 3a), and choose a configuration that meets the conditions above, while requiring minimal displacement $||p_s - p'_s||$ from its initial configuration. This process is abstracted as the function FINDPRERELO in Alg. 1.

D. Removing Blocking Objects

Extracting a rearrangement path plan can be done by searching the PT-graph using any graph search algorithm. The extracted path may include a vertex that is different from the start and goal vertex. If that is the case, then that vertex corresponds to an object that is physically blocking the rearrangement path. This object needs to be displaced before the plan can be executed. To do so, we follow a similar technique to how we plan *Prerelocations*: we find the closest relocation along the object's pushing directions (see Fig. 3a) that unblocks the path execution. This method of finding what object to remove is shown to be complete [24].

E. Reaching to Push an Object

To execute a rearrangement plan, the pusher needs to navigate between objects. To do that, we invoke a motion planner to check if there is a collision-free path connecting the robot's pose with a pushing pose. For every object rearrangement, there is at least one motion plan invocation to find a motion to approach an object. If a rearrangement

Algorithm 1 GENGRAPH

Input: Object start poses O^s, Object goal poses O^g, Workspace $\mathcal{W}$, Min. Turning Radius ρ
Output: Push-Traversability Graph G

$\mathcal{V} =$ GENVERTICES(O^s, O^g)
..... // construct a vertex for each pushing pose for each object start/goal
$\mathcal{E} = \varnothing$ ..... // start with no edge
for each v_s in $\mathcal{V}$
  if v_s is vertex of O^g
    continue ..... // does not consider push from goals
  else
    for each v_g in $\mathcal{V}, O_{v_g} \neq O_{v_s}$ ..... // vertex of another object
      $\mathcal{P}_{Dubins}$ = VALIDDUBINS$(v_s, v_g, \rho, \mathcal{W})$
..... // find a Dubins curve from v_s to v_g and check if it is collision-free
..... // and within $\mathcal{W}$
      if $\mathcal{P}_{Dubins}$ ..... // if the Dubins curve is valid
        $\mathcal{E} \leftarrow \mathcal{E} \cup (v_s, v_g, \mathcal{P}_{Dubins})$ // $||\mathcal{P}_{Dubins}||$ to be used as the weight
      else
        if $\mathcal{P}_{Dubins}$ is out of bounds
          $p_{pre} =$ FINDPRERELO$(v_s, v_g, \mathcal{W})$
..... // find different start pose near v_s that induces a valid Dubins curve
          if p_{pre} ..... // a prerelocation made the push valid
            $\mathcal{P}'_{Dubins}$ = VALIDDUBINS$(v_{pre}, v_g, \rho, \mathcal{W})$
            $\mathcal{E} \leftarrow \mathcal{E} \cup (v_s, v_g, v_{pre})$ .... // $||\mathcal{P}'_{Dubins}||$ is new path weight
          end if ..... // pre-relocation
        end if ..... // out of bound
      end if ..... // validity of dubins curve
    end for ..... // v_g
  end if ..... // v not in O^g
end for ..... // v_s
return $G(V, E)$

involves a *prerelocation* or the removal of a *blocking object*, an additional motion plan is invoked. Any motion planner could be used but we found convenient to use Hybrid A* [7].

F. Analysis of the Algorithm

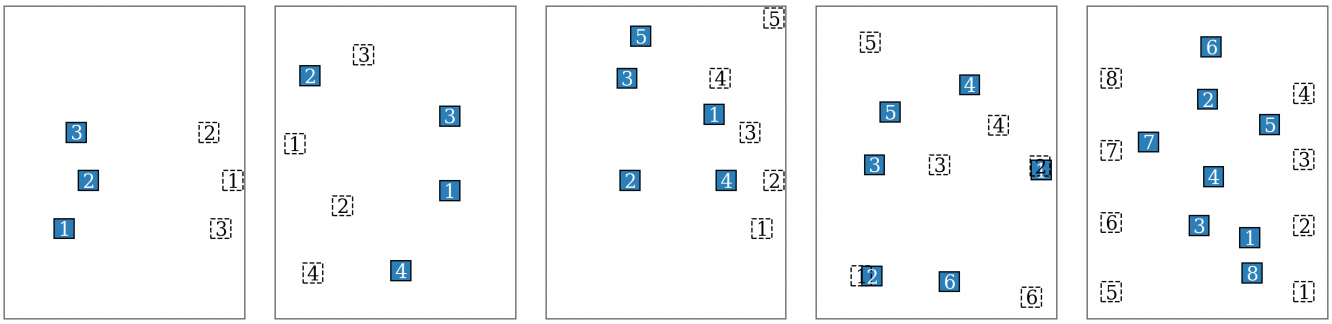
Theorem 4.1: Assuming a bounded number of pushing poses per object, K_{max}, the graph construction runs in polynomial time.

Proof: The number of vertices per object is bounded by K_{max}, thus, a fully connected graph in our case has $K_{max} \cdot m$ vertices. For each edge, a Dubins path is found in $O(1)$ and its collision checking is done in $O(1)$ assuming a bounded number of configurations checked due to our confined workspace. Searching a graph with Dijkstra's algorithm runs in $O(|V|^2)$ in a directed complete graph (the number of edges dominates). Thus, the runtime for a the rearrangement of m objects reduces to $O(m^3)$. To plan motion to approach an object, we invoke Hybrid A*, whose runtime also reduces to a polynomial expression on the number of objects assuming fixed workspace discretization, resolution of driving directions, and replanning attempts. ■

Theorem 4.2: Alg. 1 is complete if FINDPRERELO is complete.

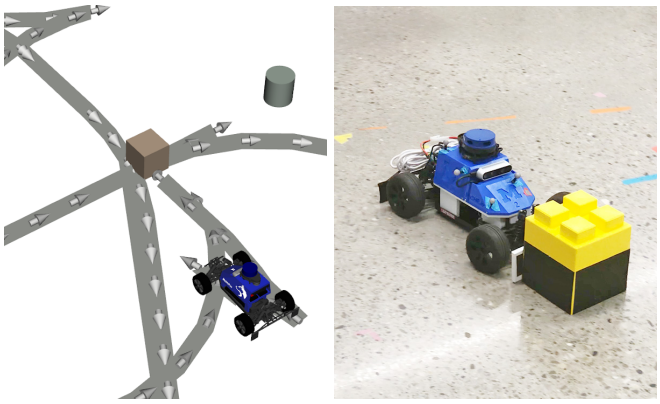
Proof: The collision checking for each Dubins path is complete since it always finds a collision if one exists by checking all grid cells that the path occupies. Thus, Alg. 1 is complete since it constructs the graph G after a finite number of iterations, assuming that FINDPRERELO is complete. ■

Theorem 4.3: RELOPUSH is complete if Alg. 1 is complete and at least one object pose is reachable by the robot.



(a) $m = 3$ (b) $m = 4$ (c) $m = 5$ (d) $m = 6$ (e) $m = 8$

Fig. 6: Evaluation scenarios. Solid squares represent starting object poses and dashed squares represent goal poses.



(a) Simulated trial. (b) Real-robot trial.

Fig. 7: Stills from simulated (a) and real robot (b) experiments.

Proof: By construction, an edge $e \in E$ between a pair (v_s, v_g) represents a feasible path from p_s to p_g. Thus, any path on G is a collision-free path within the workspace $\mathcal{W}$. Therefore, for any path on G, if the robot can plan a path to its starting vertex, a complete rearrangement is feasible. ■

V. Evaluation

We investigate the efficacy, scalability, and robustness of ReloPush in simulations and hardware experiments (Fig. 7).

A. Implementation

Experimental Setup. We implement our framework on MuSHR [32], an open-source 1/10th-scale mobile robot race-car, augmented with a 3D-printed flat bumper for pushing (see Fig. 7b), similar to the one used by Talia et al. [37], deployed in a workspace of area $4 \times 5.2m^2$. Simulations do not involve physics computations – insights about the pushing performance can be extracted from our real-world experiments. Across simulations and hardware experiments, we assume access to accurate robot localization (in real experiments, we make use of an overhead motion-capture system). We use objects of cubic shape with a side of $0.15m$ and a mass of 0.44 kg. The friction coefficient on the bumper-object surface was measured to be ~ 0.73.

Software. We implement our framework using the Open Motion Planning Library for Dubins path planning [36], and the code of Wen et al. [39] for Hybrid A* planning. Across simulated and real-world experiments, we use a Receding Horizon Controller (RHC) based on the implementation of the MuSHR [32] ecosystem. We run graph construction and search using Boost Graph Library [31]. For searching the PT-Graph, we use Dijkstra's algorithm but any graph search algorithm can be used. All planning experiments are conducted on a desktop equipped with an Intel Core i7-13700 CPU and 32G RAM. We share our software implementation online at **https://github.com/fluentrobotics/ReloPush**.

Metrics. We evaluate our system with respect to the following metrics:

- S: Success rate – a trial is successful if a planner successfully finds a feasible rearrangement sequence.
- T_p: The time it takes to compute a complete rearrangement plan.
- L_t: The total length of the path that the robot travelled, including the reaching and pushing segments.
- N_{loss}: The total number of objects the robot lost contact with.
- T_e: The total time takes to execute a complete rearrangement plan.

We also extract insights on the planning behavior of all algorithms using the following indices:

- N_{pre}: The total number of objects prerelocated to a feasible starting pushing pose (see Fig. 5).
- N_{obs}: The total number of removed blocking objects (see Fig. 4).
- L_p: The total length of path segments involving pushing.

Baselines. We compare the performance of ReloPush against two baselines:

- NO-PRERELO (NPR): a variant of RELOPUSH that also uses the PT-graph to handle non-monotone cases but does not plan *prerelocations*. Instead, it invokes Hybrid A* if a collision-free Dubins path is found, to add edges.
- MP: a variant of our system that does not make use of the PT-graph at all but rather invokes Hybrid A* to plan a sequence of rearrangement tasks in a greedy fashion, and thus can only handle monotone cases.

Experimental Procedure. We consider a series of rearrangement scenarios of varying complexity (see Fig. 6) instantiated in simulation and the real world. To extract statistics on planning performance, we instantiated 100 trials of each scenario by locally randomizing the start and goal positions of objects within a range of $\pm 0.05m$ around the

TABLE I: Planning performance. Each cell lists the mean and the standard deviation over 100 trials per scenario.

Scenario	$m=3$			$m=4$			$m=5$			$m=6$			$m=8$		
Algorithm	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP
S (%)	**100**	55	52	**100**	**100**	73	**80**	64	56	**89**	25	3	**86**	12	6
T_p (ms)	**40** (4.3)	1864 (141.1)	465 (67.4)	**86** (3.5)	5376 (131.5)	956 (36.0)	**146** (6.8)	9034 (581.5)	1175 (117.6)	**318** (20.8)	14489 (682.3)	1487 (57.4)	**529** (23.5)	25654 (2205.6)	2073 (149.5)
L_t (m)	**32.3** (3.1)	42.1 (5.2)	38.7 (7.3)	**40.3** (0.8)	45.5 (2.5)	45.1 (2.5)	**48.3** (3.5)	60.8 (6.5)	61.0 (9.1)	77.6 (5.5)	74.5 (15.2)	**62.9 (1.0)**	**90.3** (6.0)	105.8 (4.2)	101.6 (3.5)

TABLE II: Planning behavior. Each cell lists the mean and the standard deviation over 100 simulated trials per scenario.

Scenario	$m=3$			$m=4$			$m=5$			$m=6$			$m=8$		
Algorithm	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP
N_{pre}	0.9 (0.29)	-	-	1.0 (0.00)	-	-	1.6 (0.49)	-	-	3.0 (0.45)	-	-	4.4 (0.59)	-	-
N_{obs}	0.0 (0.00)	0.0 (0.00)	-	0.0 (0.00)	0.0 (0.00)	-	0.5 (0.52)	1.0 (0.25)	-	1.0 (0.00)	0.5 (0.50)	-	0.6 (0.49)	0.0 (0.00)	-
L_p (m)	**8.4** (0.1)	19.1 (4.4)	17.8 (4.3)	**8.5** (0.10)	18.1 (0.8)	17.3 (0.9)	**11.2** (0.5)	29.6 (10.1)	37.6 (7.2)	**13.9** (0.9)	37.2 (7.8)	35.6 (0.4)	**23.7** (0.76)	59.3 (3.1)	55.9 (4.0)

TABLE III: Results from real-world trials. Each cell lists the mean and the standard deviation over 5 trials per scenario.

Scenario	$m=3$			$m=4$			$m=5$			$m=6$			$m=8$		
Algorithm	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP	RELOPUSH	NPR	MP
N_{loss}	**0.0** (0.0)	0.6 (0.49)	1.0 (0.0)	**0.0** (0.0)	0.6 (0.49)	1.0 (0.63)	**0.0** (0.0)	0.4 (0.49)	0.8 (0.75)	**0.2** (0.40)	1.4 (0.49)	1.2 (0.40)	**0.2** (0.4)	2.0 (0.63)	1.6 (0.49)
T_e (s)	**96.6** (0.07)	114.4 (0.49)	104.0 (4.23)	**121.7** (0.32)	137.4 (1.05)	133.7 (0.37)	**148.1** (0.66)	154.0 (1.46)	235.8 (1.71)	256.2 (3.08)	198.9 (2.74)	**182.97** (1.20)	**274.4** (0.33)	314.9 (2.28)	285.9 (1.40)

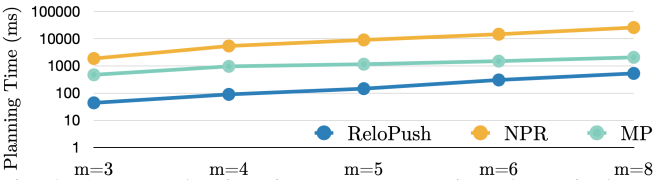


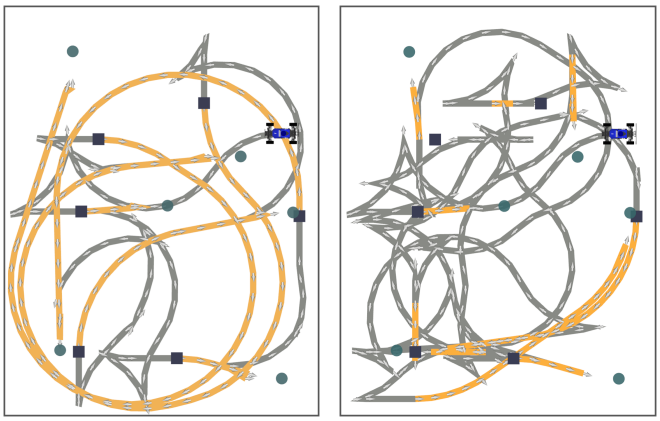
Fig. 8: Average planning time across scenarios, shown in logarithmic scale. ReloPush scales well with the number of objects compared to baselines.

nominal instances of Fig. 6. To evaluate the robustness of our complete architecture, we executed the same scenarios in a physical workspace on a real MuSHR [32] robot. To ensure fairness in real-robot experiments, we chose instances where all algorithms were successful in planning. We ran each scenario 5 times per algorithm.

B. Results

Planning Performance. ReloPush dominates baselines in success rate and planning time (see Table I, Fig. 8), with the gap becoming more pronounced as the clutter (number of objects) increases. This happens because increased clutter is more likely to lead to non-monotone instances. For example, most $m = 6$ instances are non-monotone because two objects overlap with goals of other objects. Since MP can only handle monotone rearrangements, it fails more frequently in these harder instances. It is also worth observing that kinematic constraints make some instances harder to solve *regardless* of the number of objects. For example, some of the $m = 3$ instances are challenging because o_2^s is situated so close to its goal o_2^g that the optimal path connecting them goes out of the boundary. In contrast, *ReloPush* handled this scenario effectively via *prerelocation*. Table II provides intuition on the planning decisions that ReloPush made enabled increased performance. We see that as clutter increases, ReloPush makes increasingly more workspace modifications (prerelocations and blocking-object removals).

Real Robot Experiments. ReloPush successfully completed all trials, dominating baselines in terms of execution time (see Table III). ReloPush never lost contact with any objects, in contrast to baselines. One reason for that is that ReloPush maintains low pushing path length (for better or similar total path length) as shown in Table II. The shorter the pushing distance, the lower the risk of losing the object due to model errors and uncertainties. By accounting for this during planning through prerelocations (see Fig. 9), ReloPush reduces the burden on the path tracking controller which will inevitably accrue errors during execution (the same path tracking controller was used for all algorithms). A video with footage from our experiments can be found at https://youtu.be/_EwHuF8XAjk.



(a) MP (b) RELOPUSH

Fig. 9: Paths generated by MP (a) and RELOPUSH (b) for the $m = 6$ scenario. Squares and circles represent respectively start and goal object poses. Continuous lines represent planned paths with pushing segments shown in yellow. RELOPUSH plans substantially shorter pushing segments to minimize the risk of losing contact with an object during execution.

VI. LIMITATIONS

While ReloPush is capable of handling densely cluttered problem instances, it assumes that at least one object is initially accessible by the pusher. Future work involves enabling the pusher to "break" cluttered configurations through impact to make space for planning. To achieve object stability during pushing, we used sandpaper to increase friction at the pusher-object contact, and locked the pusher's steering below the quasistatic limit, which further complicated planning. Ongoing work involves learning a model of push dynamics to relax planning and close the loop for push stability during execution. ReloPush could be further improved by optimizing the use of space when planning *prerelocations* and when removing blocking objects. Extensions to this work will study scenarios involving the rearrangement of unstructured objects like debris, and construction materials.

References

[1] J. Ahn, J. Lee, S. H. Cheong, C. Kim, and C. Nam. An integrated approach for determining objects to be relocated and their goal positions inside clutter for object retrieval. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6408–6414, 2021.

[2] J. Ahn, S. Lee, and C. Nam. Coordination of multiple mobile manipulators for ordered sorting of cluttered objects. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Sytems (IROS)*, pages 712–718, 2023.

[3] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Pérez. Manipulation with multiple action types. In *International Symposium on Experimental Robotics*, pages 531–545, 2013.

[4] M. Bauza and A. Rodriguez. A probabilistic data-driven model for planar pushing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3008–3015, 2017.

[5] P. Chen and Y. Hwang. Practical path planning among movable obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 444–449 vol.1, 1991.

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

[7] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Practical search techniques in path planning for autonomous driving. In *Proceedings of the International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR)*, 2008.

[8] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[9] S. Goyal, A. Ruina, and J. Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991.

[10] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu. Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. *The International Journal of Robotics Research*, 37(13-14):1775–1795, 2018.

[11] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour. Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3075–3082, 2015.

[12] R. D. Howe and M. R. Cutkosky. Practical force-motion models for sliding manipulation. *The International Journal of Robotics Research*, 15(6):557–572, 1996.

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

[14] E. Huang, Z. Jia, and M. T. Mason. Large-scale multi-object rearrangement. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 211–218, 2019.

[15] J. King, M. Klingensmith, C. Dellin, M. Dogar, P. Velagapudi, N. Pollard, and S. Srinivasa. Pregrasp manipulation as trajectory optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[16] J. E. King, M. Cognetti, and S. S. Srinivasa. Rearrangement planning using object-centric and robot-centric action spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3940–3947, 2016.

[17] A. Krontiris and K. E. Bekris. Dealing with difficult instances of object rearrangement. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[18] A. Krontiris, R. Shome, A. Dobson, A. Kimmel, and K. Bekris. Rearranging similar objects with a manipulator using pebble graphs. In *Proceedings of the IEEE-RAS Conference on Humanoid Robots*, pages 1081–1087, 2014.

[19] J. K. Li, W. S. Lee, and D. Hsu. Push-net: Deep planar pushing for objects with unknown physical properties. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[20] J. Lim, F. Achermann, R. Bähnemann, N. Lawrance, and R. Siegwart. Circling back: Dubins set classification revisited. In *Workshop on Energy Efficient Aerial Robotic Systems, International Conference on Robotics and Automation (ICRA)*, 2023.

[21] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[22] K. M. Lynch and M. T. Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. *The International Journal of Robotics Research*, 18(1):64–92, 1999.

[23] M. T. Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[24] C. Nam, S. H. Cheong, J. Lee, D. H. Kim, and C. Kim. Fast and resilient manipulation planning for object retrieval in cluttered and confined environments. *IEEE Transactions on Robotics*, 37(5):1539–1552, 2021.

[25] J. Ota. Rearrangement planning of multiple movable objects by using real-time search methodology. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 947–953 vol.1, 2002.

[26] Z. Pan, A. Zeng, Y. Li, J. Yu, and K. Hauser. Algorithms and systems for manipulating multiple objects. *IEEE Transactions on Robotics*, 39(1):2–20, 2022.

[27] H. Ren and A. H. Qureshi. Multi-stage monte carlo tree search for non-monotone object rearrangement planning in narrow confined environments. In *Proceedings*

of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 12078–12085, 2024.

[28] K. Ren, L. E. Kavraki, and K. Hang. Rearrangement-based manipulation via kinodynamic planning and dynamic planning horizons. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Sytems (IROS)*, pages 1145–1152, 2022.

[29] K. Ren, P. Chanrungmaneekul, L. E. Kavraki, and K. Hang. Kinodynamic rapidly-exploring random forest for rearrangement-based nonprehensile manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 8127–8133, 2023.

[30] A. M. Shkel and V. Lumelsky. Classification of the dubins set. *Robotics and Autonomous Systems*, 34(4): 179–202, 2001.

[31] J. Siek, L.-Q. Lee, and A. Lumsdaine. Boost graph library. http://www.boost.org/libs/graph/, June 2000.

[32] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, R. Scalise, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi. MuSHR: A low-cost, open-source robotic racecar for education and research, 2019. *arXiv:1908.08031* [cs.RO].

[33] M. Stilman and J. Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, 2008.

[34] M. Stilman and J. J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 02 (04):479–503, 2005.

[35] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour. Manipulation planning among movable obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3327–3332, 2007.

[36] I. A. Sucan, M. Moll, and L. E. Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[37] S. Talia, A. Thareja, C. Mavrogiannis, M. Schmittle, and S. S. Srinivasa. PuSHR: A multirobot system for nonprehensile rearrangement. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Sytems (IROS)*, pages 5380–5387, 2023.

[38] J. Van Den Berg, M. Stilman, J. Kuffner, M. Lin, and D. Manocha. Path planning among movable obstacles: a probabilistically complete approach. In *Algorithmic Foundation of Robotics VIII: Selected Contributions of the Eight International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 599–614. Springer, 2010.

[39] L. Wen, Y. Liu, and H. Li. Cl-mapf: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints. *Robotics and Autonomous Systems*, 150:103997, 2022.

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

[41] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang. Rearrangement with nonprehensile manipulation using deep reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 270–277, 2018.