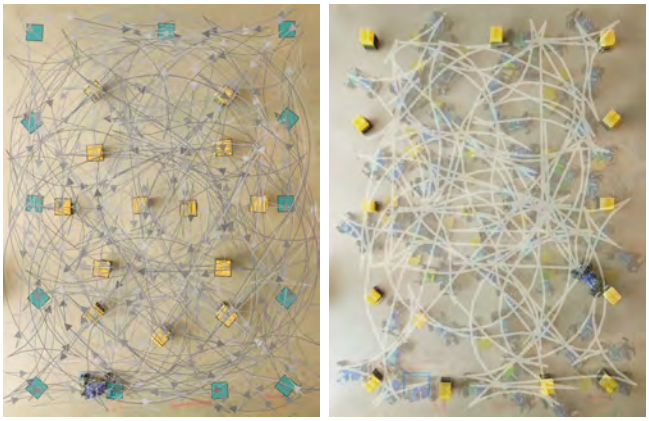
ReloPush-BOSS: Optimization-guided Nonmonotone Rearrangement Planning for a Car-like Robot Pusher

Jeeho Ahn and Christoforos Mavrogiannis

***Abstract*— We focus on multi-object rearrangement planning in densely cluttered environments using a car-like robot pusher. The combination of kinematic, geometric and physics constraints underlying this domain results in challenging *nonmonotone* problem instances which demand breaking each manipulation action into multiple parts to achieve a desired object rearrangement. Prior work tackles such instances by planning *prerelocations*, temporary object displacements that enable constraint satisfaction, but deciding where to *prerelocate* remains difficult due to local minima leading to infeasible or high-cost paths. Our key insight is that these minima can be avoided by steering a prerelocation optimization toward low-cost regions informed by Dubins path classification. These optimized prerelocations are integrated into an object traversability graph that encodes kinematic, geometric, and pushing constraints. Searching this graph in a depth-first fashion results in efficient, feasible rearrangement sequences. Across a series of densely cluttered scenarios with up to 13 objects, our framework, *ReloPush-BOSS*, exhibits consistently highest success rates and shortest pushing paths compared to state-of-the-art baselines. Hardware experiments on a 1/10 car-like pusher demonstrate the robustness of our approach. Code and footage from our experiments can be found at: https://fluentrobotics.com/relopushboss.**



(a) Initial (yellow) and goal (teal) object poses, connected through a *Push-Traversability* graph.

(b) Rearrangement path, extracted by planning on the graph, and robot motion traces from an experiment.

Fig. 1: We introduce *ReloPush-BOSS*, an optimization-based planning framework for nonprehensile multiobject rearrangement planning using a car-like robot pusher. By capturing kinematic, geometric, and physics constraints into a unified object traversability graph (a), our framework is capable of handling nonmonotone instances in densely cluttered workspaces with up to 13 objects (b).

I. Introduction

Rearrangement planning has revolutionized fulfillment by enabling rapid, large-scale transportation of goods by mobile robots in massive warehouses [7]. Transportation tasks in these domains are often facilitated by extensive workspace and systems engineering: robots often move along rectilinear grids in well-structured spaces; they manipulate a closed set of objects of regular shapes and sizes; they apply secure closure grasps through grippers of high design complexity and cost. This level of structure is not possible in other important domains like construction, waste management, and smaller/medium warehouses due to high cost and clutter density. In those environments, the ability to handle a wide range of objects is essential, and so is the need to address kinematic and geometric constraints. These requirements give rise to complex problem instances that motivate the investigation of new paradigms for rearrangement planning.

A practical strategy for tackling rearrangement tasks in unstructured domains is pushing [22, 24]: exploiting physics, rather than grasping, pushing allows for the reconfiguration of large, heavy, and irregular objects without requiring specialized grippers; a simple pushing surface is sufficient. However, this paradigm introduces the challenge of integrating physics into rearrangement planning. A practical way of doing so is via constraining interactions between the robot and pushed objects to be quasistatic. This approach further constraints robot motion: quasistatic pushing using a mobile robot reduces to the Dubins car problem [22]. When the robot is constrained, its maneuverability is also limited even when *not* pushing. In densely cluttered spaces, this combination of geometric, kinematic, and physics constraints renders many practical planning problems infeasible.

In this work, we introduce a planning framework for constrained multiobject rearrangement planning using a car-like robot pusher. Our framework simultaneously addresses geometric (obstacle avoidance, boundary constraints), kinematic (nonholonomic), and physics constraints (quasistatic pushing) to tackle challenging rearrangement tasks (see Fig. 1). Our approach builds upon our graph-based framework, ReloPush [1], which handles constraints by temporarily prerelocating objects along collision-free Dubins paths leading to object goal poses. In ReloPush, these prerelocations are planned along linear segments along the body frame axes of the pushed object, and thus often lead to high-cost solutions or failures. Here, we introduce *ReloPush-BOSS* (*ReloPush with Backtracking and Optimization with Seeded Start*), a planning framework that combines backtracking and prerelocation optimization informed by Dubins path classification to avoid excessively high-cost local minima. Across a series of simulated and real-world experiments involving the rearrangement of up to 13 objects by a 1/10th-

The authors are with the Department of Robotics, University of Michigan, Ann Arbor, USA. Email: {jeeho, cmavro}@umich.edu

scale robot racecar, we demonstrate the superior scalability and robustness of ReloPush-BOSS against ablated versions as well as a baseline from the literature.

Our contributions are as follows:

- *Optimization-based prerelocation search*: In contrast to our prior work [1], in which prerelocations are sampled from straight-line segments originating from an object's frame, we formulate prerelocation search as a continuous optimization producing lower-cost solutions.
- *Avoidance of high-cost local minima*: Exploiting insights from Dubins path classification [6, 9, 21, 30], we introduce seeded warm-starts in the optimization process to avoid getting trapped into high-cost local minima representing undesired paths.
- *Extensive empirical validation*: We validate our framework on challenging constrained planning problems, showing significant performance gains over baselines [1, 17]. We demonstrate the efficacy and robustness of our approach through real-world demonstrations on a 1/10 scale robot racecar [34].

II. Related Work

There is a long history of work on rearrangement planning. We organize related work in terms of three key categories (see Table I): manipulation strategy; system dexterity; rearrangement hardness. We elaborate on each of them below.

Manipulation strategy. Much of the work on rearrangement planning assumes *prehensile* manipulation via grasping [10, 13, 17, 19, 35], including suction-based pipelines [13], pick-and-place sequences [10, 19], and grasp-based obstacle relocation for target access [35]. Recent prehensile methods address such sequential challenges via multi-objective planning with selective obstacle relocation to feasible points [41] or decomposition into easier pick-and-place subproblems [20]. In contrast, *nonprehensile* manipulation [12, 22, 24] leverages physics to handle diverse object sizes/shapes without specialized grippers, making it robust for unstructured domains like cluttered warehouses [5, 15, 32, 37, 40]. For instance, Huang et al. [15] employ policy rollouts and iterated local search to tackle large-scale planar rearrangement problems using a manipulator that pushes, and Song et al. [33] propose a nonprehensile, push-based rearrangement method for object sorting. Some work [8, 32] combines pushing with picking to enable grasping in dense clutter under occlusions, whereas recent work develops policies for robust pushing of diverse objects [5, 40].

Dexterity. For pick-and-place or sorting tasks, high–DoF manipulators are often employed thanks to their increased dexterity which enables complex rearrangements in clutter. In open-top scenes, arms exploit *overhand* access to simplify collision avoidance and sequencing [10, 13, 19, 32]. High-DoF manipulators are also effective in closed-top settings (shelves/racks/bins) by exploiting side-access grasps to complete challenging rearrangements [2, 25, 28]. However, for large planar workspaces, mobility is prioritized over dexterity. Holonomic bases allow unconstrained planar motion and simplify planning, but many common platforms are nonholonomically constrained (e.g., differential drive, or car-like robots), thus affording limited maneuverability in constrained spaces [9, 18, 27]. Ma et al. [23] incorporate kinematic constraints in a pickup–and–delivery problem under grid-world assumptions whereas Bertoncelli et al. [5] address more realistic settings in pushing, and Talia et al. [37] propose a framework for multirobot planar pushing using car-like robots. Closest to our settings, Ahn and Mavrogiannis [1] tackle multiobject rearrangement planning using a car-like robot, but their method is prone to high-cost local minima.

Hardness. Earlier work focused on *monotone* rearrangement tasks where each object is assumed to be moved at most once. The work of Stilman et al. [35] handles manipulations of obstacles assuming monotone transitions. Dogar and Srinivasa [8] present a planning framework for object retrieval formulated under *monotone* assumptions using a manipulator. Similarly, Ahn et al. [3] propose a coordination method of two manipulators in object retrieval, making problems monotone by introducing additional space for removing obstacles from a clutter. Talia et al. [37] address object rearrangement problems using nonholonomic robots, restricted to monotone cases. In contrast, many real-world domains like logistics, construction, and the home often require the rearrangement of multiple objects in dense clutter, thus giving rise to *nonmonotone* problem instances where multiple manipulation actions are required for rearranging any object. These are substantially more complex due to the combinatorial explosion in the search space and interdependent object movements [13, 17, 26]. This complexity escalates further when constrained by pushing actions and nonholonomic robot kinematics [33], and prior work tackles simplified problem instances like the rearrangement of a single object in an obstacle-free workspace [5, 40].

This work. We address labeled rearrangement tasks under nonprehensile, nonholonomic, and geometric constraints, which often result in nonmonotone instances. These instances are substantially more constrained compared to much of the literature, which tends to assume secure, prehensile manipulation (i.e., closure grasps), unconstrained robot motion (e.g., holonomic or overhand grasping), or position-only rearrangement planning. In prior work [1], we introduced *prerelocations* as temporary object displacements that unlock kinematically feasible object transitions along contact normals (pushing axes). This is constraining, as it limits the search space and often excludes more efficient rearrangement routes that could be unlocked by small displacements in other directions. Our new method, ReloPush-BOSS addresses this limitation by expanding the search for prerelocations beyond pushing directions, incorporating an optimization strategy that exploits insights from Dubins path classification [6, 9, 21, 30] to identify superior candidates while avoiding getting trapped in high-cost minima.

III. Problem Statement

We consider the problem of rearranging a set of m rigid, polygonal objects using a car-like robot pusher in a

TABLE I: Related works. † Relocation of obstacles for object retrieval. †† Relocation of objects for sorting. ††† Rearrangement of at most one object, and at most one reconfiguration of the object before transport. * Orientations are specified for object goal poses.

	Manipulation		Hardness		Dexterity		
	Prehensile	Nonprehensile	Nonmonotone	Orientation Requirement*	Overhand Grasp	Nonholonomic	DoF Assumed
Stilman et al. [35]	✓			✓			High-DoF arm
Dogar and Srinivasa [8]†	✓	✓		✓			High-DoF arm
Nam et al. [25]†	✓		✓				High-DoF arm
Ahn et al. [2]†	✓		✓				High-DoF arm
Ren and Qureshi [28]†	✓		✓				High-DoF arm
Lee et al. [19]	✓	✓	✓				High-DoF arm
King et al. [16]†††	✓	✓	✓	✓			High-DoF arm
Krontiris and Bekris [17]	✓		✓	✓			High-DoF arm
Shome et al. [31]	✓			✓	✓		High-DoF arm
Ahn et al. [4]††	✓		✓		✓		High-DoF arm
Han et al. [13]	✓		✓	✓	✓		High-DoF arm
Gao et al. [10]	✓		✓	✓	✓		High-DoF arm
Song and Boularias [32]	✓	✓	✓	✓	✓		High-DoF arm
Tang and Sukhatme [39]	✓	✓	✓	✓	✓		High-DoF arm
Huang et al. [15]†††		✓	✓	✓			High-DoF arm
Talia et al. [37]		✓		✓		✓	Low-DoF (car)
Our prior work, ReloPush [1]		✓	✓	✓		✓	Low-DoF (car)
This work (ReloPush-BOSS)		✓	✓	✓		✓	Low-DoF (car)

workspace $\mathcal{W} \subset SE(2)$. Without loss of generality, we assume that all objects are K-polygons. The robot can push objects through a flat bumper attached at its front. We represent the state of the robot as $p \in \mathcal{W}$, and the state of each object $j \in \mathcal{M} = \{1, \ldots, m\}$ as $o_j \in \mathcal{W}$. The robot has rear-axle, simple-car kinematics of the form $\dot{p} = f(p, u)$, where u is a control input (speed and steering angle) from a space of controls $\mathcal{U}$. The objective of the robot is to rearrange the objects from their initial poses $O^s = (o_1^s, o_2^s, \ldots, o_m^s)$ to their goal poses $O^g = (o_1^g, o_2^g, \ldots, o_m^g)$. We seek to develop a planning framework to enable the robot to efficiently rearrange all objects to their goals. We assume the robot has accurate knowledge of: its ego pose; the starting configuration of all objects, O^s; the bumper-object friction coefficient.

IV. ReloPush-BOSS: Optimization-guided Nonmonotone Rearrangement Planning

We give an overview of our proposed architecture *ReloPush-BOSS*, and then elaborate on its core components.

A. Rearrangement Sequence Planning

An overview of ReloPush-BOSS is illustrated in Fig. 2. At the high level, ReloPush-BOSS involves a sequence planner, implemented as a depth-first search (DFS) over a sequence of rearranged objects. This search starts from depth $d0$, representing the initial configuration of the world, i.e., objects lying at O^s, robot lying at p^s. A push-traversability (PT)-graph is first constructed, describing how objects in $\mathcal{W}$ can move via stable pushing (see Fig. 3). For each object to be rearranged from o_j^s to o_j^g, a graph search is performed, resulting in a corresponding rearrangement path. These paths are then ordered wrt the distance required to push-transfer them to their goal pose, and stored in an ordered candidate queue (shorter paths are better). We check if the rearrangement of shortest path is reachable by the robot from its current pose. If reachable, the rearrangement is feasible. Thus, the rearrangement is reflected in an updated world configuration $d1$. If not reachable, the rearrangement is infeasible. In that case, the planner backtracks and tries the next best option in the queue. This whole process is repeated until all objects reach their goals or the candidate queue is

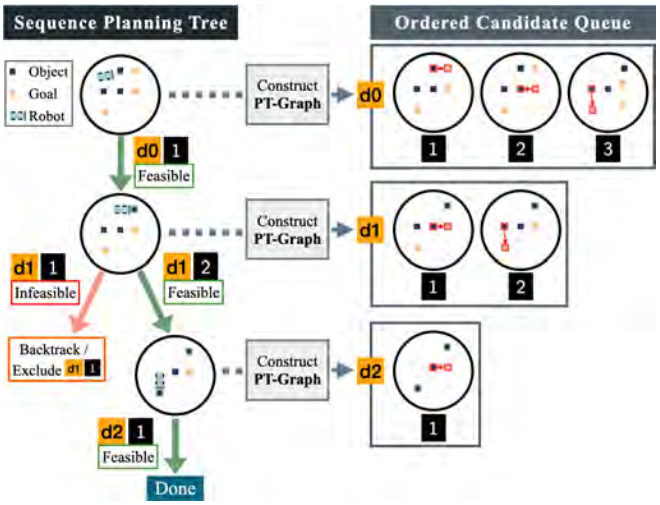
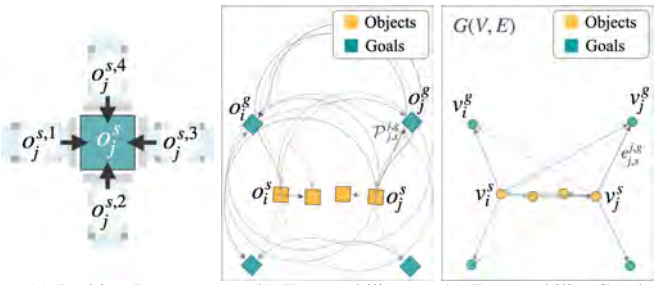


Fig. 2: ReloPush-BOSS embeds a PT-graph in a depth-first high-level planner. Starting from the initial configuration (d0), it builds a PT-graph at each depth, enumerates candidate rearrangements, and orders them by cost in a priority queue (ordered candidate queue). Candidates are tested in order; infeasible ones (e.g., inaccessible objects) are pruned and the planner backtracks until it finds a feasible sequence or exhausts all options.

exhausted, representing a failure. This cost-ordered depth-first search with backtracking represents an improvement over ReloPush [1], which marks failure when encountering an infeasible rearrangement. While the greedy selection does not guarantee a globally optimal sequence, we employ the strategy to avoid the combinatorial explosion associated with exhaustive sequence planning for global optimum.

B. Push-Traversability Graph

Construction. Object j, lying at a pose o_j admits a finite set of *pushing poses* $F(o_j) = \{p_j^1, \ldots, p_j^k\}$, defined by stable contact points on the object's boundary and the surface normal in which stable pushing is possible (i.e. K=4 for rectangular object. see Fig. 3a). As shown in Fig. 3, the PT-graph $G = (V, E)$ comprises a set of vertices $V = V^s \cup V^g$, where V^s represents the set of all pushing poses for all objects when lying at their initial poses O^s, and V^g represents the set of all pushing poses for all objects when lying at their goal poses O^g. For any pair of vertices $v_{\text{src}} \in$

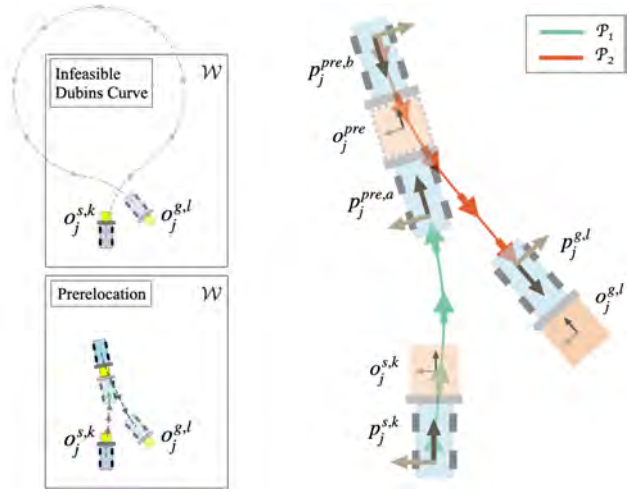


(a) Pushing Poses. (b) Traversability. (c) Traversability Graph.

Fig. 3: Push-Traversability graph (PT-graph). Vertices denote pushing poses for objects/goals; directed edges encode feasible pushes weighted by total pushing length. (a) Pushing poses are defined as contact normals on object's sides. (b) Workspace view showing traversable connectivity between objects/goals. (c) Corresponding graph; for clarity, only one vertex per object is shown.

V^s, $v_{\text{dst}} \in V^s \cup V^g$, a *directed* edge $e = (v_{\text{src}}, v_{\text{dst}}) \in E$ is added iff there exists a collision-free Dubins path [9] from the pushing pose corresponding to v_{src} to the pushing pose corresponding to v_{dst}. To ensure stable pushing during push-transfer between two pushing poses, we set the turning radius of the robot ρ to the quasistatic steering limit $\rho_p \leq \rho_{max}$. In a densely cluttered space, a Dubins path under this limit is often not be feasible due to boundary violations or collisions with objects (see Fig. 4a). In such cases, we search for a prerelocation, i.e., an intermediate object pose $o_j^{pre} \in \mathcal{W}$, splitting the rearrangement of object j from o_j^s to o_j^g into a sequence of two collision-free Dubins paths: one from o_j^s to o_j^{pre} and another from o_j^{pre} to o_j^g (see Fig. 4b). If found, we embed the paths associated with the o_j^{pre} to an edge to connect the two vertices. While in our prior work [1], this search was performed via sampling along the directions of the object's pushing poses, in this paper we describe an optimization-based approach that searches over a continuous region for a prerelocation that minimizes the push-transfer length (see Sec. IV-C, Sec. IV-D). For a finite number of vertices, the PT-graph is constructed in polynomial time [1].

Search. For a pair of vertices $v_j^{s,k}$ to $v_j^{g,l}$, describing the rearrangement of object j from o_j^s to o_j^g via a path from pushing pose $p_j^{s,k}$ to pushing pose $p_j^{g,l}$, a push-transfer path can be found through standard graph search techniques. Note that the initial pushing pose, l, can be different than the final one, k, since our graph includes prerelocations, i.e., breaks of contact enabling transitions to different pushing poses. If the path contains a vertex other than $v_j^{s,k}$, $v_j^{g,l}$, that vertex represents an object $b \neq j$ at pose o_b, blocking the robot's path from $p_j^{s,k}$ to $p_j^{g,l}$. In that case, a straight-line push-transfer path is found to displace object b from o_b to make the rearrangement of object j feasible. That path is found by sampling points along the contact normals of o_b and finding the closest one that is not in collision with the rearrangement path of object j, and with any other objects. This action, referred to as *Obstacle Relocation*, acts as a primitive clearing heuristic; we strictly limit this to straight-line pushes to ensure the inner loop of the search remains efficient, rather than expanding the search space to find optimal poses for secondary objects. Relocation of the blocking object b will be executed before transferring object j, which will now pass unobstructed through the region that was previously occupied by object b.



(a) (b)

Fig. 4: Illustration of a prerelocation. (a) Top: A continuous push-transfer Dubins path from $o_j^{s,k}$ to $o_j^{g,l}$ is infeasible; bottom: transitioning to a new pushing pose when the object is transferred to a prerelocation enables a feasible push transfer. (b) Zoom in on a prerelocation. A robot path $\mathcal{P}_1$, from $p_j^{s,k}$ to $p_j^{pre,a}$ (shown in green) transfers the object from $o_j^{s,k}$ to o_j^{pre}. After the robot transits to a new pushing pose $p_j^{pre,b}$ (transit path not shown), $\mathcal{P}_2$ transfers the object to $o_j^{g,l}$.

C. Optimizing Prerelocations

ReloPush-BOSS optimizes a prerelocation wrt the total length of the push-transfer portion of the rearrangement given a start pose, a goal pose, and a minimum turning radius; transit paths to and from objects are not accounted for. For object j moved from o_j^s to o_j^g through a transition from pushing pose $p_j^{s,k}$ to pushing pose $p_j^{g,l}$, an optimized prerelocation o_j^{pre} is obtained by solving the following optimization (see Fig. 4b):

$$C(o_j^{pre}) = \arg\min \mathcal{L}(\mathcal{P}_1) + \mathcal{L}(\mathcal{P}_2) \tag{1}$$

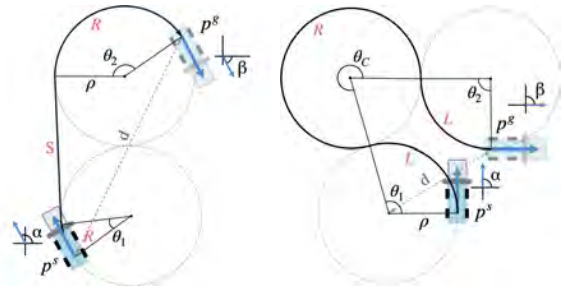
$$s.t.\ \mathcal{P}_1 = Dubins(p_j^{s,k}, p_j^{pre,a}) \tag{2}$$

$$\mathcal{P}_2 = Dubins(p_j^{pre,b}, p_j^{g,l}) \tag{3}$$

$$\mathcal{P}_1, \mathcal{P}_2 \in \mathcal{W} \tag{4}$$

$$\mathcal{P}_1, \mathcal{P}_2 \cap O = \emptyset, \tag{5}$$

where $C(\cdot)$, $\mathcal{L}(\cdot)$, $Dubins(\cdot,\cdot)$ are the cost of a *prerelocation*, the length of a path, and a function describing a Dubins path between the poses passed as its arguments, respectively. $p_j^{pre,a}$ is the pose of the robot when arriving at o_j^{pre}, $p_j^{pre,b}$ is the pose of the robot when initiating transfer to o_j^g. Eqs (4) and (5) enforce respectively the constraints that these segments are within the workspace boundary and not in collision with any objects. In this work, we focus on analyzing the finding of at most one prerelocation per object as optimization for sequential prerelocations would transform the problem into a different optimization problem. Our analysis confirms that a single prerelocation shows



(a) CSC Dubins path. (b) CCC Dubins path.

Fig. 5: For a car-like robot, a Dubins path describes the optimal path of transitioning from a start pose p^s to a goal pose p^g. A Dubins path can be of two different type families, i.e., CSC or CCC, where S is a a straight line segment, and $C \in \{L, R\}$ denotes a left or right arc. When the distance d is too small relative to the respective heading difference, only CCC paths are admissible (the short-distance regime), yielding longer paths with a wider spatial sweep [30]. Circles indicate minimum–turning-radius arcs; α and β are the start and goal headings.

considerable effectiveness in finding feasible solutions that pose lower costs than those of the prior work.

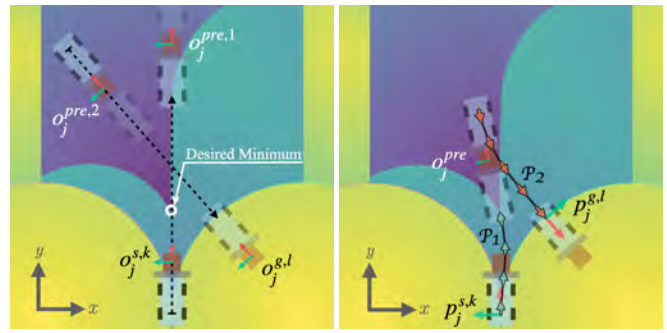
D. Avoiding High-Cost Local Minima

We show that a gradient-based optimization over the cost of eq. (1) is prone to local minima, and propose a technique for warm-starting an optimizer from a high-quality solution.

Dubins path classification. A Dubins path from a pose p^s to a pose p^g comprises exactly three segments and is either of type *CSC* or *CCC* [30], where *S* represents a straight line segment, and $C \in \{L, R\}$ represents an arc segment under turning radius ρ, where *L* is a left turn and *R* is a right turn. As shown in Fig. 5, when the required heading change between p^s and p^g is too large compared to their respective position distance, d, only *CCC* paths are admissible; these tend to be long as they feature three consecutive turns [30].

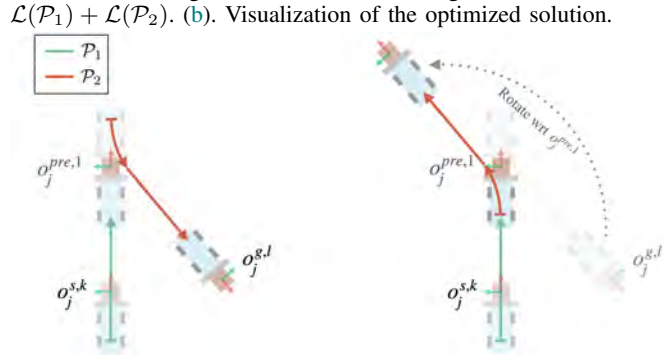
Non Convexity of n-point Dubins path. Goaoc et al. [11] study the problem of computing the shortest path of bounded curvature visiting a predefined sequence of n points in the plane. They show that the length of this path as a function of the local curve orientation at these points is locally convex, but globally non-convex, indicating the existence of multiple local minima. The prerelocation optimization problem of eq. (1) represents a more general version, where the prerelocation is optimized with respect to the resulting total curve length, $\mathcal{L}(\mathcal{P}_1)+\mathcal{L}(\mathcal{P}_2)$, over both its *position* and orientation. Thus, it is also non-convex.

Warm-starting the optimization. For the n-point Dubins path problem, considering fixed positions at a $d > 4\rho$ distance apart and varying orientations at the intermediate points, Goaoc et al. [11] show that an approximation of the shortest path (at most 1.91 times of the optimum) can be found by connecting consecutive waypoints with S and CS curves. Since prerelocation optimizes a full pose rather than an orientation, it induces a higher freedom for path optimization, offering a potential to reduce the 1.91 factor. Nevertheless, we cannot directly leverage this approximation because unlike the original n-point Dubins path problem, a prerelocation, by definition, results in a break of contact corresponding to a change in the pushing pose.



(a) Prerelocation Optimization. (b) Optimized solution.

Fig. 6: Cost landscape of $\mathcal{L}(\mathcal{P}_1) + \mathcal{L}(\mathcal{P}_2)$ plotted over $\mathbb{R}^2$. Darker colors represent lower costs. For visualization purposes, the plot assumes a fixed heading at each position but the proposed optimization minimizes over $SE(2)$. (a) $o_j^{pre,1}$ and $o_j^{pre,2}$ are two possible initializations following the approximation discussed in Sec. IV-D. The former is constructed through a sequence of CS and S curves, whereas the latter through a sequence of S and CS curves. Descending from either seed converges to a minimum of $\mathcal{L}(\mathcal{P}_1) + \mathcal{L}(\mathcal{P}_2)$. (b). Visualization of the optimized solution.



(a) Initial guess. (b) Conversion to 3PDP [6].

Fig. 7: Initial guess as a sequence of $\mathcal{P}_1$ (S) and $\mathcal{P}_2$ (CS) curves. By rotating $\mathcal{P}_2$ and $p^{g,l}$ wrt o_j^{pre} such that the orientation of $p_j^{pre,a}$ matches the orientation of the rotated $p_j^{pre,b}$, we convert our problem to the 3PDP [6, 29], enabling us to leverage a high-quality approximation of the optimum.

We convert our *prerelocation* optimization to a 3-point Dubins path problem (3PDP [6]) to leverage this result in the construction of an optimization seed. First, we find the object pose $o_j^{pre,1}$ enabling a straight-line path $\mathcal{P}_2$ (S) from $p_j^{pre,b}$ to $p_j^{g,l}$ and a (SC) curve $\mathcal{P}_1$ from $p_j^{s,k}$ to $p_j^{pre,a}$. By symmetry, we could also define a similar initial guess comprising a S segment for $\mathcal{P}_1$, and a SC segment for $\mathcal{P}_1$. We show both solutions in Fig. 6a. Next, we rotate $\mathcal{P}_2$ and $p_j^{g,l}$ wrt o_j^{pre} such that the orientation of $p_j^{pre,a}$ matches the orientation of the rotated $p_j^{pre,b}$ (Fig. 7). We show that at this configuration represents a high-quality upper bound of an optimal solution.

Theorem 4.1: Let the straight segments of $\mathcal{P}_1$ and $\mathcal{P}_2$ of $o_j^{pre,1}$ be S_1 and S_2, respectively. Then $C(o_j^{pre,1})$ is larger than an optimal cost by at most $\mathcal{L}(S_1) + \mathcal{L}(S_2)$.

Proof: To meet the orientation goal of rearranging $o_j^{s,k}$ to $o_j^{g,l}$, the robot is required to rotate the object by $\Delta\theta = |\theta_j^{s,k} - \theta_j^{g,l}|$ where $\theta_j^{s,k}$ and $\theta_j^{g,l}$ are the orientation of $o_j^{s,k}$ and $o_j^{g,l}$, respectively. Because the robot is constrained by the minimum turning radius ρ, the robot needs to turn along its turning arc of at least $\rho \cdot \Delta\theta$ length. Thus, any valid solution of o_j^{pre} must incur a travel distance of at least $\rho \cdot \Delta\theta$, which is the same length as the only C segment of $o_j^{pre,1}$.

Therefore, the total length $o_j^{pre,1} = \mathcal{L}(\mathcal{P}_1) + \mathcal{L}(\mathcal{P}_2)$ is at most $\mathcal{L}(S_1) + \mathcal{L}(S_2)$ larger than the optimal cost. ■

In our confined settings, the straight segments are generally short, and concurrently, each Dubins path generally falls into $d < 4\rho$. As discussed in prior work, the length of a Dubins path is discontinuous when $d < 4\rho$ at the boundary between CSC and CCC [30, 42], with a CCC solution being strictly longer than any CSC. As the initialization $o_j^{pre,1}$ is a solution with a straight path in both $\mathcal{P}_1$ and $\mathcal{P}_2$, it is in the CSC region for both paths. While the region is generally not convex, the Dubins path is analytic with respect to d and start/goal orientations as long as the path type remains consistent [30]. Thus, a gradient-based optimizer will converge to a first-order stationary point, a local minimum that generally poses shorter total length than that of $o_j^{pre,1}$. By the symmetry of the construction, *Theorem* 4.1 also holds for the other initialization, $o_j^{pre,2}$.

Conditional Completeness. The overall method is *conditionally complete*, conditioned on the existence of a solution within the search space defined by these primitives: at most one prerelocation, straight-line obstacle clearing, and standard Dubins connections. Since these constraints ensure a finite number of candidate sequences, our DFS complete within the search space.

V. Evaluation

We present an empirical evaluation of *ReloPush-BOSS*, assessing its scalability, robustness, and real-world transfer.

A. Experiment design

Methods. Our evaluation involves the following methods:

- *plRS-Push*. Adaptation of the piecewise-linear Rearrangement Search (plRS) [17], originally for pick-and-place, restricted to *pushing* with a car-like robot. The high-level planner exhaustively enumerates object-order permutations (in randomized order) and returns a plan upon success or after all permutations are exhausted.
- *ReloPush*. The vanilla version of ReloPush from our prior work Ahn and Mavrogiannis [1].
- *ReloPush-B*. *ReloPush* augmented with a *B*acktracking high-level planner (depth-first search) that exhaustively explores the solution tree (Sec. IV-A). During graph construction, prerelocations are not optimized, and are found by sampling points along pushing poses [1].
- *ReloPush-BO*. Replaces the search method for *prerelocation* in *ReloPush-B* with gradient-based *O*ptimization but without informed warm-starting (Sec. IV-C).
- *ReloPush-BOSS*. The proposed method, corresponding to *ReloPush-BO*, augmented with a *S*eeded-*S*tart: each relocation optimization is warm started using the proposed warm-starting (Sec. IV-C).

Metrics. We evaluate performance in terms of the following metrics: *Success rate* – fraction of instances with a feasible plan found within 20 min (1,200 s); *Planning Time (s)* – time taken to find a feasible rearrangement plan; *Total Pushing Length (m)* – sum of all pushing segments for every object; *Total Path Length (m)* – overall distance traveled by the robot, including both transfer and transit segments.

Scenarios. All methods were tested across the six scenarios of increasing hardness depicted in Fig. 8, which involve the rearrangement of $m = 8$ to $m = 13$ objects in constrained workspace. To collect statistical insights, for each scenario, we generate 100 planning instances by locally perturbing the position and the orientation of each object or a goal within a range of ± 0.05 m and ± 0.1 rad, respectively.

B. Implementation

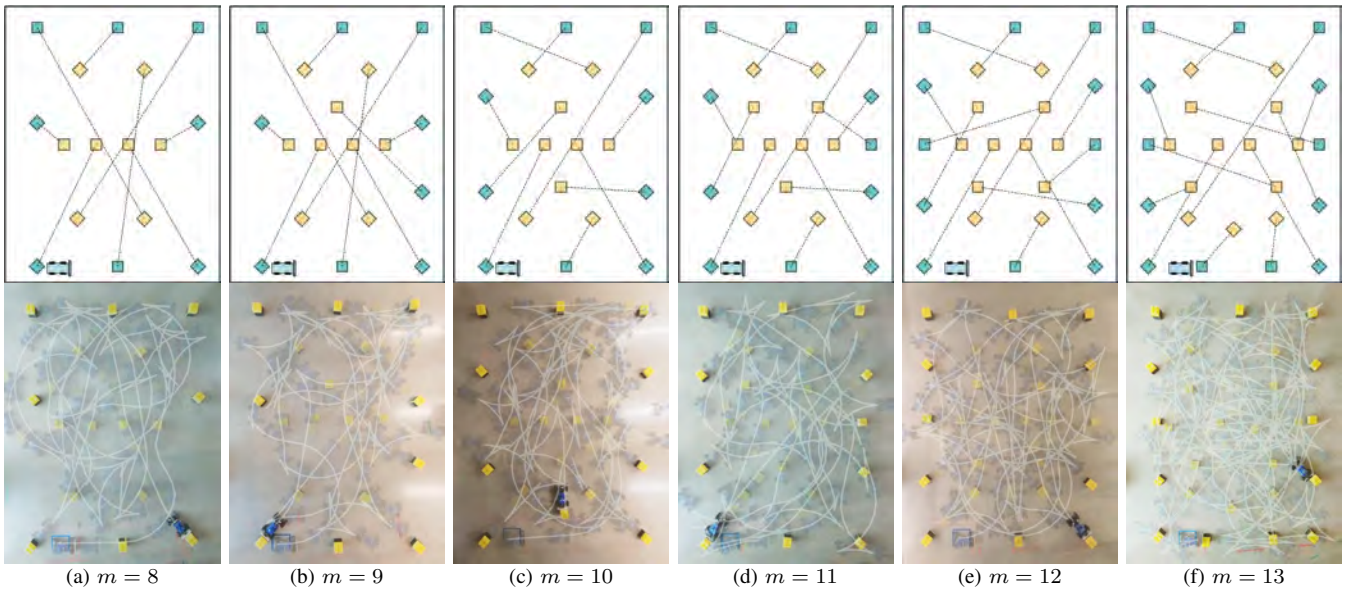
For our experiments, we use an open-source 1/10 racecar from the MuSHR ecosystem [34, 38], augmented with a flat pushing bumper, following prior work [1, 37]. All scenarios are instantiated in a $4 \times 5.2\,\mathrm{m}^2$ rectangular workspace (resolution 0.1 m for map discretizations and collision checking). Objects are rigid square blocks with a 0.15 m side, 0.44 kg weight, and measured bumper–object friction of ~ 0.73. Steering is constrained to a minimum turning radius of $\rho_p = 1.43$ m to preserve quasistatic motion during pushing, and $\rho_{np} = 1.09$ m during transit. Notably, ρ_p constitutes $\sim 35\%$ of the workspace width such that a full turning circle largely does not fit. Following limit-surface calculations [12, 14, 22], the minimum turning radius for stable pushing is $\rho_{p,min} = 0.815$ m, thus our setting represents a conservative bound. We exploit the symmetry of rearranged objects (cubic shape) to allow for multiple possible goal orientations (i.e., each object can be rearranged to one of four possible configurations). ReloPush-BOSS is implemented in C++. Dubins primitives and transit paths are computed using OMPL [36] and the Hybrid A^* module of Wen et al. [43]. Collision checking assumes exact geometry for the robot, objects, and workspace boundaries. We use Dijkstra's algorithm for searching on the PT-graph. We modified *plRS* implementation by Song and Boularias [32] into a *plRS-Push* variant, suitable for pushing. Our code can be found at: https://fluentrobotics.com/relopushboss.

C. Planning Performance

Fig. 9 depicts the performance of ReloPush-BOSS against baselines. We see that ReloPush-BOSS attains the highest success rate across the board. Out of successful trials, ReloPush-BOSS also achieves the *lowest* transfer length, and total length (see Fig. 10), whereas its planning time is consistently among the lowest, close to ReloPush.

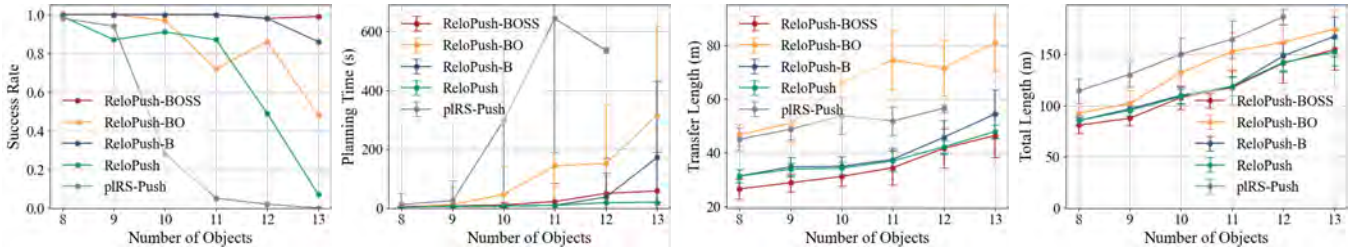
Hardness of benchmark. Our benchmark is unique in that it involves the rearrangement of dense clutter of even up to 13 objects inside a constrained workspace. This results in a substantially denser workspace than prior work [1, 16, 37, 40]. This hardness can be traced in the path lengths required to execute the underlying scenarios, which reached larger than 150 m for the more challenging ones (see Fig. 9d).

Optimization-guided robustness. As the optimization module of ReloPush-BOSS searches over a wider range of $SE(2)$ for prerelocations, it results in a graph that makes more efficient use of the constrained space, resulting in higher success rates even as m grows. A byproduct of that is that ReloPush-BOSS tends to find a solution earlier than other methods, leading to shorter planning times. This is nontrivial as a generic optimization approach generally takes longer than an analytic approach (see ReloPush-B).



(a) $m = 8$ (b) $m = 9$ (c) $m = 10$ (d) $m = 11$ (e) $m = 12$ (f) $m = 13$

Fig. 8: Top: evaluation scenarios involving $8 \sim 13$ objects; for each scenario, dotted lines connect initial (blue) and goal (yellow) object poses. Bottom: frame stacks from real-world execution for each scenario, overlaid with the planned path.



(a) (b) (c) (d)

Fig. 9: Comparative planning performance across all metrics and scenarios.

Value of initial guess. We see that ReloPush-BO scales poorly with the scenario hardness, exhibiting lower success rates than ReloPush-BOSS and ReloPush-B, and planning inefficient rearrangement paths with excessively high times. In contrast, we see that ReloPush-BOSS, which is just ReloPush-BO augmented with our proposed module for warm-starting the optimizer, leads to scalable and robust performance. As discussed in Sec. IV-C, this gap in performance is explained by the nonlinearity of the domain which often traps ReloPush-BO into excessively high-cost local minima.

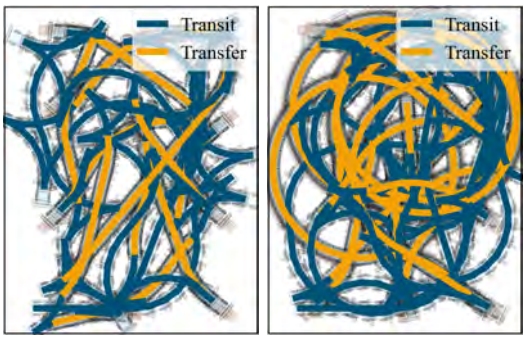
Effectiveness of Depth-First Search. From the results of ReloPush-B, we observe higher success rates and overall competitive performance against baselines just by integrating a depth-first search high level planner into ReloPush.

D. Real-Robot Demonstration

We deployed the framework on a MuSHR platform [34] to execute randomized benchmark instances. Using overhead motion capture for robot localization and MPC for path tracking, we successfully completed all trials, including a 13-object task, under open-loop object manipulation. These results (Fig. 8) demonstrate robustness against real-world friction and tracking errors. Footage from our experiments can be found at: https://fluentrobotics.com/relopushboss.

VI. Limitations

The conservative bound on the turning radius makes planning in constrained spaces especially challenging. On-



(a) ReloPush-BOSS (b) pIRS-Push

Fig. 10: Paths resulting from tackling the $m = 9$ instance using our method (a), and pIRS-Push (b). Yellow and blue lines represent push-transfer and transit path portions, respectively.

going work looks at relaxing it through the incorporation of data-driven models of pushing that account for variations in dynamics. Despite strong performance across complex scenarios, the proposed method is limited by its dependence on at least one unarranged object being accessible to the robot and by the consideration of at most one prerelocation per object. In severely cluttered workspaces, the robot may find no clear path to any object or optimization solution within the workspace. Future work will leverage impulsive manipulation to "break" the clutter and open up free paths. Finally, the implementation of obstacle clearing as a straight-line displacement is limiting and can be improved by leveraging the machinery of prerelocation optimization.

REFERENCES

[1] J. Ahn and C. Mavrogiannis. ReloPush: Multi-object rearrangement in confined spaces with a nonholonomic mobile robot pusher. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 13196–13203, 2025.

[2] 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.

[3] J. Ahn, C. Kim, and C. Nam. Coordination of two robotic manipulators for object retrieval in clutter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1039–1045, 2022.

[4] 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.

[5] F. Bertoncelli, F. Ruggiero, and L. Sabattini. Linear time-varying mpc for nonprehensile object manipulation with a nonholonomic mobile robot. In *Proceedings of the IEEE international conference on robotics and automation (ICRA)*, pages 11032–11038, 2020.

[6] Z. Chen and T. Shima. Shortest Dubins paths through three points. *Automatica*, 105:368–375, 2019.

[7] R. D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639, 2012.

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

[9] 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.

[10] K. Gao, J. Yu, T. S. Punjabi, and J. Yu. Effectively rearranging heterogeneous objects on cluttered tabletops. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2057–2064, 2023.

[11] X. Goaoc, H.-S. Kim, and S. Lazard. Bounded-curvature shortest paths through a sequence of points using convex optimization. *SIAM Journal on Computing*, 42(2):662–684, 2013.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[19] J. Lee, C. Nam, J. Park, and C. Kim. Tree search-based task and motion planning with prehensile and non-prehensile manipulation for obstacle rearrangement in clutter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 8516–8522, 2021.

[20] S. Levit, J. Ortiz-Haro, and M. Toussaint. Solving sequential manipulation puzzles by finding easier subproblems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 14924–14930, 2024.

[21] 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.

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

[23] H. Ma, W. Hönig, T. S. Kumar, N. Ayanian, and S. Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7651–7658, 2019.

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

[25] 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.

[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] J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[28] 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.

[29] A. Sadeghi and S. Smith. On efficient computation of shortest Dubins paths through three consecutive points. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 6010–6015, 2016.

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

[31] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin. Fast, high-quality dual-arm rearrangement in synchronous, monotone tabletop setups. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 778–795, 2018.

[32] C. Song and A. Boularias. Object rearrangement with nested non-prehensile manipulation actions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6578–6585, 2019.

[33] H. Song, J. A. Haustein, W. Yuan, K. Hang, M. Y. Wang, D. Kragic, and J. A. Stork. Multi-object rearrangement with monte carlo tree search: A case study on planar nonprehensile sorting. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 9433–9440, 2020.

[34] 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].

[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] S. Talia, M. Schmittle, A. Lambert, A. Spitzer, C. Mavrogiannis, and S. Srinivasa. Hound: An open-source, low-cost research platform for high-speed off-road underactuated nonholonomic driving. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[39] B. Tang and G. S. Sukhatme. Selective object rearrangement in clutter. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1001–1010, 2023.

[40] Y. Tang, H. Zhu, S. Potters, M. Wisse, and W. Pan. Unwieldy object delivery with nonholonomic mobile base: A stable pushing approach. *IEEE Robotics and Automation Letters*, 8(11):7727–7734, 2023.

[41] C. B. Tuncer, M. Toussaint, and O. S. Oguz. Mo-segman: Rearrangement planning framework for multi objective sequential and guided manipulation in constrained environments. *arXiv preprint arXiv:2511.01476*, 2025.

[42] P. Vána and J. Faigl. Optimal solution of the generalized Dubins interval problem. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[43] 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.