

PushAround: Collaborative Path Clearing via Physics-Informed Hybrid Search

Abstract—The passage of large vehicles in cluttered environments is often blocked by movable obstacles, motivating the use of mobile robot teams to proactively clear traversable corridors. Existing approaches to Navigation Among Movable Obstacles (NAMO) mainly plan abstract sequences of obstacle displacements but neglect physical feasibility, overlooking robot dimensions, obstacle mass, contact geometry, and required forces, which often leads to strategies that fail in execution. This work introduces PushAround, a physics-informed framework that guarantees feasible path clearing by coupling geometric reasoning with dynamic validation. The core novelty is a hybrid search that jointly determines which obstacles to displace and how to push them, including contact points, directions, and forces. The framework builds a W-Clearance Connectivity Graph (WCCG) to test vehicle passage, ranks frontier gaps to prioritize obstacle-clearing actions, and incrementally expands feasible plans by validating pushing modes. This integration ensures that solutions are both geometrically valid and physically executable. Efficiency is achieved by combining compact geometric reasoning with prioritized evaluation, avoiding exhaustive simulation and improving scalability beyond prior NAMO methods. Extensive simulations and hardware experiments demonstrate robust success, physical validity, and superior efficiency compared to state-of-the-art approaches.

I. INTRODUCTION

In many real-world settings, the passage of large vehicles or transport units is obstructed by movable obstacles such as pallets, boxes, or equipment, motivating the deployment of mobile robot teams to actively clear traversable corridors and enable safe passage. This capability is particularly critical in cluttered and unstructured environments—warehouses, disaster sites, and dense urban spaces—where conventional path planning methods assume static obstacles and thus fail once direct routes are blocked [1]. Navigation Among Movable Obstacles (NAMO) has been studied as a means of creating new routes by pushing, pulling, or rotating obstacles [2], yet most existing methods abstract away physical feasibility. Factors such as robot dimensions, obstacle size and mass, contact geometry, applied forces, and the coupled dynamics of pushing are typically ignored, producing plans that are geometrically valid but physically unrealizable. The difficulty is amplified by the fact that a single large robot is often unable to clear a corridor unaided, smaller robots are limited by their reach and access to obstacle boundaries, and cooperative pushing introduces strong physical coupling where one action can trigger chained motions of multiple obstacles, making prediction and planning substantially more complex.

A. Related Work

Navigation Among Movable Obstacles (NAMO) has been studied as an extension of motion planning in cluttered

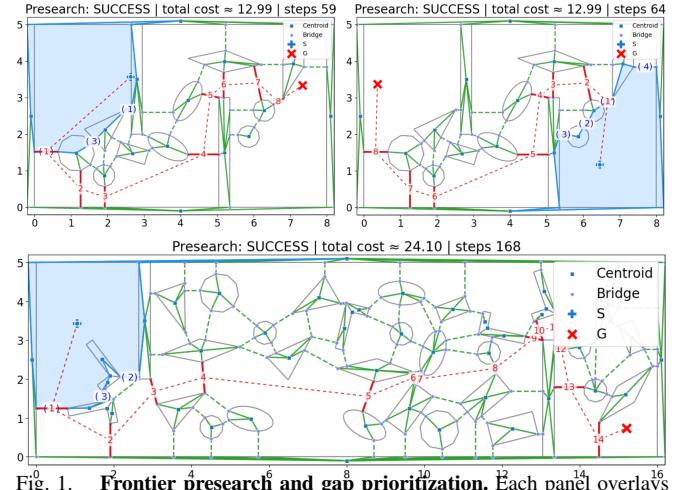


Fig. 1. Frontier presearch and gap prioritization. Each panel overlays the WCCG together with the currently selected face (light blue). Numbers in red mark the *gap-crossing order* returned by the presearch, and numbers in blue indicate the *local ranking* of first-hop gaps on the current face boundary. **Top left/right:** the same scene with start/goal swapped, results in symmetric gap orders with same cost (≈ 12.99). **Bottom:** a longer map with ~ 30 obstacles, showing a 14-gap sequence with cost ≈ 40.36 .

environments. Classical methods planned explicit sequences of obstacle displacements through pushing, pulling, or rotating [2], [3], while later work introduced heuristic search for scalability [3] and graph- and sampling-based formulations for larger workspaces [4]. Recent efforts extend NAMO to multi-agent settings where teams of robots clear passages collaboratively [5], [6]. However, most NAMO approaches still idealize robots as point agents with unlimited actuation and neglect robot size, obstacle mass, and realistic dynamics, which leads to plans that are geometrically valid but physically unrealizable.

Collaborative pushing has also been widely investigated in multi-robot manipulation. Foundational studies examined the mechanics of pushing and the limit surface model [7], [8], while more recent work addressed force synchronization [9], contact stability and slip margins [10], [11], and cooperative transport of large payloads in structured environments [12], [13]. Learning-based approaches have further explored emergent coordination in cluttered settings [14]. These studies demonstrate effective multi-robot cooperation but are generally limited to single-object tasks and assume strict collision avoidance with surrounding obstacles. As a result, inter-object interactions and the coupled dynamics of multiple movable obstacles remain largely unaddressed.

Physics-informed planning has emerged to integrate realistic dynamics into motion generation. Several methods use physics engines to validate candidate manipulations [15] or

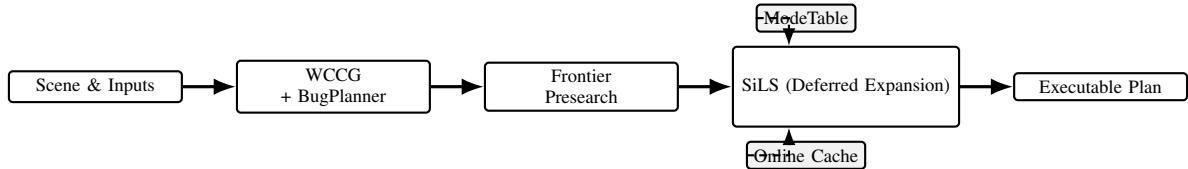


Fig. 2. TODO: Overall framework.

embed contact simulation directly in the planning loop [16]. Other approaches incorporate dynamic models into planning for non-prehensile manipulation [?], [?] or apply learning with physics priors and differentiable physics for contact-rich tasks [?], [?], [9], [10]. While these approaches highlight the benefits of embedding physical reasoning, they often decouple high-level planning from low-level feasibility checks or remain confined to single-object manipulation. Consequently, they do not scale to collaborative path clearing in cluttered environments with many movable obstacles.

B. Our Method

This work introduces *PushAround*, a physics-informed framework for collaborative multi-robot pushing that actively constructs traversable corridors for large vehicles in cluttered environments. The central novelty is a hybrid search that jointly determines which obstacles to displace and how to push them, coupling the high-level sequence of obstacle-clearing actions with low-level pushing modes that specify contact points, directions, and forces. The framework integrates three components: a W-Clearance Connectivity Graph (WCCG) that certifies vehicle passage under width constraints, a gap-ranking strategy that prioritizes frontier gaps by estimated effort, and a search procedure that incrementally expands candidate plans while validating their physical realizability. This integration guarantees that the resulting plans are not only geometrically valid but also feasible under robot dimensions, object mass, contact geometry, and coupled pushing dynamics. Efficiency is achieved by combining compact geometric reasoning with prioritized evaluation, focusing computation on the most promising actions and avoiding the scalability issues of simulation-heavy NAMO approaches.

The contributions of this work are twofold: (I) it introduces the first unified framework that couples multi-robot collaborative pushing with physics-informed feasibility guarantees, producing executable plans for path clearing in dense cluttered environments; and (II) it demonstrates significant improvements in feasibility, efficiency, and scalability over existing NAMO and collaborative pushing methods, through extensive simulation and hardware validation.

II. PROBLEM DESCRIPTION

A. Workspace and Robots

The workspace is a bounded planar region $\mathcal{W} \subset \mathbb{R}^2$ that contains two types of obstacles. A set \mathcal{O} represents immovable structures, while a set of M movable rigid polygons is defined as $\Omega \triangleq \{\Omega_1, \dots, \Omega_M\} \subset \mathcal{W}$. Each movable obstacle

Ω_m is characterized by its mass M_m , inertia I_m , frictional parameters (either identified or estimated), and state $s_m(t) \triangleq (x_m(t), \psi_m(t))$, where $x_m(t) \in \mathbb{R}^2$ is the planar position of its centroid and $\psi_m(t) \in \mathbb{R}$ its orientation angle. The region occupied by Ω_m at time t is denoted $\Omega_m(t)$. Moreover, a small team of N robots, indexed as $\mathcal{R} \triangleq \{R_1, \dots, R_N\}$, operates as a cooperative unit. Each robot R_i is modeled as a rigid body with state $s_{R_i}(t) \triangleq (x_{R_i}(t), \psi_{R_i}(t))$, where $x_{R_i}(t) \in \mathbb{R}^2$ is its position and $\psi_{R_i}(t) \in \mathbb{R}$ its orientation. The footprint of R_i is denoted $R_i(t)$. The instantaneous free space is given by:

$$\widehat{\mathcal{W}}(t) \triangleq \mathcal{W} \setminus (\mathcal{O} \cup \{\Omega_m(t)\}_{m=1}^M \cup \{R_i(t)\}_{i=1}^N), \quad (1)$$

where $\widehat{\mathcal{W}}(t)$ excludes regions occupied by immovable obstacles, movable obstacles, and robots.

B. External Vehicle and Clearance Goal

An external vehicle V of radius $W/2 > 0$ must navigate within the workspace from a start configuration s_V^S to a goal configuration s_V^G . Since movable obstacles may obstruct the way, a direct passage is not always feasible. Feasibility at time t is captured by the W -clearance condition

$$\exists \mathcal{P}_V^W \subset \widehat{\mathcal{W}}(t) : s_V^S \rightsquigarrow s_V^G, \text{clr}(\mathcal{P}_V^W) \geq W, \quad (2)$$

where \mathcal{P}_V^W is a continuous curve connecting start and goal inside the free space, and $\text{clr}(\mathcal{P}_V^W)$ denotes its minimum clearance to surrounding obstacles.

C. Collaborative Pushing Modes

Thus, the robots may actively reconfigure Ω by pushing obstacles. The interaction with obstacle Ω_m is described by a pushing mode $\xi_m \triangleq (\mathcal{C}_m, \mathbf{u}_m)$, where $\mathcal{C}_m \in (\partial\Omega_m)^N$ is the set of contact points established by the robots, and $\mathbf{u}_m \in \mathbb{R}^{2N}$ encodes the body-frame forces or an equivalent wrench profile. The admissible set of modes is Ξ_m , determined by contact geometry and frictional limits. Furthermore, the system evolution under a pushing mode is captured by the transition operator below:

$$\mathbf{S}(t^+) \triangleq \Phi(\mathbf{S}(t), m, \xi_m), \quad (3)$$

where $\mathbf{S}(t)$ stacks all robot and obstacle states. The operator Φ models the joint dynamics of robots and obstacles under contact. In general, Φ is not available in closed form and is instead evaluated through physics simulation.

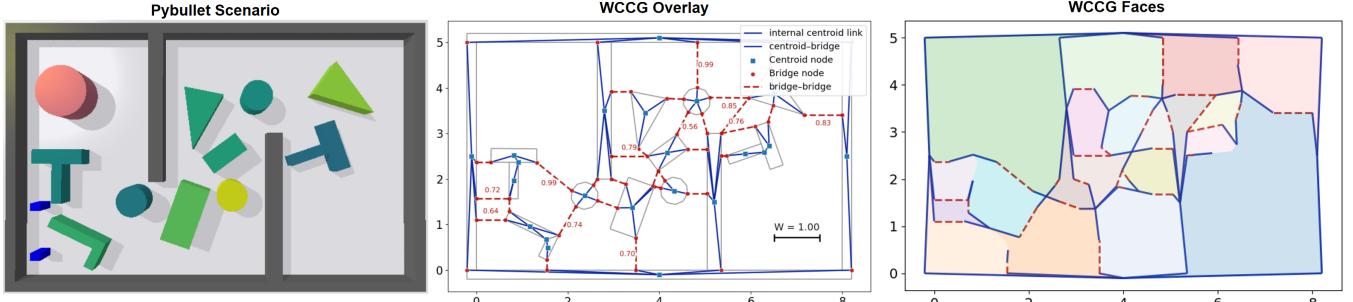


Fig. 3. Illustration of the W-Clearance Connectivity Graph (WCCG). **Left:** Cluttered PyBullet scenario with the immovable walls and movable objects; **Middle:** WCCG overlay with the centroid nodes (blue squares), bridge nodes (red circles), centroid-bridge edges (blue), and bridge-bridge edges (red dashed) annotated by the gap widths; **Right:** Induced faces of the WCCG, where the colors indicate distinct connected regions.

D. Problem Statement

The goal is to compute a hybrid schedule for the robots that reconfigures the movable obstacles so that the vehicle admits a W -feasible path from start to goal. The overall schedule for the robotic fleet is defined as:

$$\pi \triangleq \{(m_k, \xi_k, \Delta t_k)\}_{k=1}^K, \quad (4)$$

where $\Omega_{m_k} \in \Omega$ specifies the movable obstacle to manipulate, $\xi_k \in \Xi_{m_k}$ is the pushing mode, and $\Delta t_k > 0$ the duration of execution. Thus, the optimization problem balances the execution time and the physical effort of the clearance process, subject to various constraints, i.e.,

$$\min_{\pi} \left\{ T + \alpha \sum_{k=1}^K J(m_k, \xi_k, \mathbf{S}(\tau_k)) \right\}, \quad (5)$$

where $T \triangleq \sum_{k=1}^K \Delta t_k$ is the total task duration, $\alpha > 0$ is a trade-off parameter, and $J(\cdot)$ is a simulation-based control effort or feasibility cost evaluated at state $\mathbf{S}(\tau_k)$. Note that the above optimization problem is constrained by (1), (2), and (3), which jointly ensure collision-free evolution within the workspace, valid dynamic transitions under pushing, and a terminal W -clearance path for the vehicle.

Remark 1. Problem (5) above uniquely couples obstacle selection, pushing modes, and timing into a single hybrid optimization. This yields a combinatorial search space far more complex than classical NAMO, where prior work often assumes simple object shapes or hand-crafted contact modes [13], [7], [11], [17]. ■

III. PROPOSED SOLUTION

The unified framework called PushAround is developed to enable collaborative multi-robot pushing with physical feasibility, as described in this section. The framework begins by constructing a W-Clearance Connectivity Graph, introduced in Sec. III-A, which captures the connectivity of the workspace under the clearance requirement and identifies frontier gaps that block passage. Building on this representation, Sec. III-B presents a gap-ranking strategy that assigns costs to candidate gaps and determines which obstacle-clearing actions are most promising. These ranked actions are then evaluated within a simulation-in-the-loop search, described in Sec. III-C, where a configuration-space tree

is expanded and candidate pushes are validated by parallel physical simulation. The complete execution flow together with generalization aspects is summarized in Sec. III-D.

A. W-Clearance Connectivity Graph

The feasibility of routing the vehicle depends on whether a corridor of width W exists between the start configuration \mathbf{s}_V^S and the goal configuration \mathbf{s}_V^G . To address this question, a *W-Clearance Connectivity Graph* (WCCG) is introduced. The WCCG encodes adjacency relations between obstacles under the clearance threshold W and supports efficient connectivity queries. Unlike grid maps or sampled roadmaps, it is constructed directly from obstacle geometry, avoiding discretization errors and ensuring consistent results. This is particularly important since clearance queries must be invoked repeatedly during planning.

1) *Graph Construction:* The WCCG is built by decomposing each movable obstacle $\Omega_m \in \Omega$ into convex components. From each component C , a centroid node v_c is created. When two components C_u and C_v have closest points $p_u \in C_u$ and $p_v \in C_v$ with distance smaller than W , bridge nodes are added at p_u and p_v . These bridge nodes are connected by a bridge-bridge edge, annotated with the corresponding gap width $w_{uv} \triangleq \|p_u - p_v\|$, and further linked back to their centroids with centroid-bridge edges. Narrow passages with $w_{uv} < W$ are explicitly marked as potential bottlenecks. The resulting graph is defined below:

$$\mathcal{G}_W \triangleq (\mathcal{V}, \mathcal{E}_c \cup \mathcal{E}_b), \quad (6)$$

where \mathcal{V} contains all centroid and bridge nodes; \mathcal{E}_c is the set of centroid-bridge edges; and \mathcal{E}_b the set of bridge-bridge edges annotated by widths w_{uv} .

2) *Connectivity Criterion:* Once \mathcal{G}_W has been constructed, connectivity queries can be performed without explicitly computing a geometric path. A frontier-tracing procedure, similar to the BugPlanner [18], starts from the vehicle start \mathbf{s}_V^S , casts a ray toward the goal \mathbf{s}_V^G , and explores the encountered loop of frontier edges. If a valid exit is discovered, the process continues until the goal is reached; otherwise a blocking cycle is returned. Successful execution produces a skeleton Σ , which is an ordered sequence of centroid and bridge nodes that certifies \mathbf{s}_V^S and \mathbf{s}_V^G lie in the same connected face. Let the W -clear free space be $\mathcal{F}_W \triangleq \mathbb{R}^2 \setminus (\mathcal{O} \oplus \mathbb{B}_{W/2})$, where \oplus denotes Minkowski addition and

$\mathbb{B}_{W/2}$ is a closed disk of radius $W/2$. A complete criterion for existence of a W -clear path \mathcal{P}_V^W is defined as:

$$\exists \mathcal{P}_V^W \subset \mathcal{F}_W : \mathbf{s}_V^S \rightsquigarrow \mathbf{s}_V^G \iff \left\{ \mathbf{s}_V^S, \mathbf{s}_V^G \text{ belong to the same face of } \mathcal{G}_W; \mathbb{B}_{W/2}(\mathbf{s}_V^S), \mathbb{B}_{W/2}(\mathbf{s}_V^G) \subset \mathcal{F}_W \right\}, \quad (7)$$

where $\mathbb{B}_{W/2}(\cdot)$ denotes a disk of radius $W/2$ centered at the argument. They align with (2) for the external vehicle.

3) *Skeleton to Path*: The skeleton Σ as an output of the previous step, is an ordered sequence of centroid and bridge nodes connected by frontier edges in \mathcal{G}_W . This skeleton serves as a compact certificate that \mathbf{s}_V^S and \mathbf{s}_V^G lie in the same connected face. Given as input the skeleton Σ , together with \mathcal{G}_W , the clearance W , and the endpoint disks $\mathbb{B}_{W/2}(\mathbf{s}_V^S)$ and $\mathbb{B}_{W/2}(\mathbf{s}_V^G)$, the output is an explicit W -clear path \mathcal{P}_V^W . This path is constructed by sliding each skeleton segment along the boundary of the inflated obstacles, offsetting slightly inward into \mathcal{F}_W , and attaching short connectors inside the endpoint disks. The resulting \mathcal{P}_V^W remains in \mathcal{F}_W , preserves the homotopy of Σ , and guarantees clearance of at least W .

Remark 2. The proposed WCCG differs from sampling- and grid-based planners in two key aspects: (I) It avoids discretization of \mathcal{W} and is therefore free from resolution-induced errors; (II) It relies purely on geometry, which makes queries highly efficient. These two features are crucial, since connectivity checks and clearance tests are invoked many times within the hybrid planner described in the sequel. ■

B. Ranking of Potential Blocking Gaps

When the condition in (7) fails, the vehicle cannot reach \mathbf{s}_V^G from \mathbf{s}_V^S through a W -clear path \mathcal{P}_V^W . In this case, the planner must prioritize *blocking gaps* on the reachable frontier of \mathcal{G}_W . The goal of this module is to provide an ordered list of such gaps, ranked by their predicted cost to eventually yield a feasible path, which serves as a critical guidance for the hybrid search in the sequel.

1) *Frontier Extraction and Candidate Gaps*: A BugPlanner query on \mathcal{G}_W , given $(\mathbf{s}_V^S, \mathbf{s}_V^G, W)$, either confirms connectivity or returns a counter-clockwise frontier loop \mathcal{L} that separates the two endpoints. The bridge–bridge edges visible on \mathcal{L} form the first-hop candidate set $\Gamma_{\mathcal{L}} \triangleq \{g_1, \dots, g_K\}$, where each g_k is a reachable candidate gap. These candidates are the inputs to the ranking module, while the output will be an ordered list of the same set sorted by predicted cost.

2) *Evaluation for Immediate Cost*: Each candidate $g \in \Gamma_{\mathcal{L}}$ is evaluated by combining the cost for the robots to reach the gap and the effort required to widen it. Let $\mathbf{s}_{\mathcal{R}}$ be the current robot positions, and \mathbf{o}_g as the outside insertion point of gap g . The resulting one-hop cost is given by:

$$C(g | \mathcal{L}, \mathbf{s}_{\mathcal{R}}) = \lambda_t C_t(\mathbf{s}_{\mathcal{R}}, \mathbf{o}_g) + \lambda_p C_p(g), \quad (8)$$

where $\lambda_t, \lambda_p > 0$ are weighting factors for the transition and pushing costs, respectively; function $C_t(\cdot)$ denotes the collision-free distance between two points; and the function $C_p(\cdot)$ measures the widening effort, which increases when the gap is narrower than W , or when the adjacent

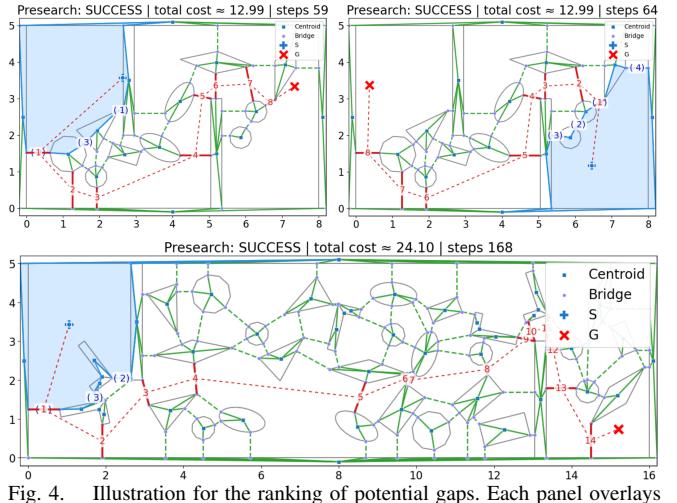


Fig. 4. Illustration for the ranking of potential gaps. Each panel overlays the WCCG together with the currently selected face (light blue). The algorithm (i) extracts the frontier loop from BugPlanner, (ii) enumerates candidate bridge–bridge gaps on the loop, (iii) assigns local first-hop ranks (blue numbers), and (iv) simulates a short presearch to predict the full gap-crossing sequence (red numbers). **Top:** identical environment with start and goal swapped; the resulting gap sequences are symmetric with identical predicted cost at 12.99; **Bottom:** a larger map with about 30 obstacles, where presearch returns a 14-gap sequence of predicted cost at 24.10.

obstacles are heavier as scaled by the mass of adjacent movable obstacles.

3) *Long-term Cost w.r.t. Goal*: Furthermore, to prioritize gaps closer to the goal, a heuristic is added to the evaluation, i.e., $h(g) = \eta \| \mathbf{o}(g) - \mathbf{s}_V^G \|$, where $\eta > 0$ is a scaling constant. Lastly, a short A^{*}-style presearch virtually crosses each candidate gap, recomputes the next frontier, and continues for a limited beam width and depth. The predicted cost-to-connect is given by:

$$\widehat{Cost}(g) \triangleq C(g | \mathcal{L}, \mathbf{s}_{\mathcal{R}}) + \sum_{g' \in \Pi^*(g)} \{C(g' | \cdot) + h(g')\}, \quad (9)$$

where $\Pi^*(g)$ is the sequence of subsequent gaps discovered after virtually crossing g ; the dots indicate updated inputs along that rollout; and the scalar score $\widehat{Cost}(g) > 0$ for each candidate gap. Consequently, the final output of this module is the ranked list of candidate gaps, i.e., $\text{Rank}(\Gamma_{\mathcal{L}}) \triangleq \text{argsort}_{g \in \Gamma_{\mathcal{L}}} \widehat{Cost}(g)$, which sorts $\Gamma_{\mathcal{L}}$ in ascending predicted cost. This ordered set is passed to the hybrid search module, such that only the promising gaps are expanded first.

Remark 3 (Practical Improvement). Efficiency of the above ranking procedure can be improved by caching frontier loops, storing the transitions $(\mathcal{L}, g) \mapsto \mathcal{L}'$, and accelerating the edge queries with axis-aligned bounding-box culling. With a small beam width and depth (typically around 8), the runtime of ranking remains negligible compared to the cost of simulation-based validation. ■

C. Physics-Informed Hybrid Search

The hybrid search couples high-level decisions about blocking gaps with low-level feasibility of multi-robot pushing. Unlike purely geometric planners, this procedure simultaneously determines a sequence of gaps to clear and phys-

Algorithm 1: Physics-Informed Hybrid Search

```

Input:  $s_0, \alpha, \text{EvalSim}(\cdot), W$ -criterion by (7)
Output:  $\pi^*$ 
    /* Initialization */ *
1  $\nu_0 \leftarrow (s_0, \emptyset)$ ,  $\chi(\nu_0) = 0$ ;
2 Init  $\mathcal{Q}$  by (10);
3 while not terminated do
    /* Selection */ *
4    $\nu \leftarrow \mathcal{Q}.\text{pop\_min}()$  by (10);
    /* Parallel Expansion */ *
5   foreach  $g \in \text{Rank}(\nu)$  do
6     /* Selection */ *
7       foreach  $(\mathbf{v}, \xi) \in \Xi_g$  do
8          $\tau = (g, \mathbf{v}, \xi)$ ;
9          $(s', \delta_T, \delta_J) \leftarrow \text{EvalSim}(\nu, \tau)$  by (11);
10        if success then
11           $\nu' \leftarrow (s', \pi \cup \{\tau\})$ ;
12           $\chi(\nu') \leftarrow \chi(\nu) + \delta_T + \alpha \delta_J$ ;
13           $\mathcal{Q}.\text{push}(\nu')$ ;
    /* Termination Check */ *
14  if  $s'$  admits  $\mathcal{P}_V^W$  by (7) then
15     $\pi^* \leftarrow \pi', \text{break}$ ;
15 return  $\pi^*$ ;

```

ically feasible pushing actions, including directions, contact modes, and forces. Parallel physics simulation is embedded so that many candidate push strategies can be evaluated simultaneously at each expansion, and the resulting successor states are returned to the search. This tight coupling of discrete graph reasoning with continuous pushing dynamics is unique for the considered problem.

1) *Tree and Initialization:* The search tree \mathcal{T} is composed of nodes $\nu \triangleq (s, \pi)$, where s is the current system state including the positions and orientations of all robots and movable obstacles as in (3), and π is the partial pushing strategy realized so far as in (4). Two global functions are maintained: $\text{Rank}(\nu)$ stores a ranked list of candidate gaps at this node together with their exploration status, derived from (9); and $\chi(\nu)$ returns the cumulative execution cost from the root to ν . The root is initialized as $\nu_0 \triangleq (s_0, \emptyset)$, where s_0^S corresponds to the initial system state s_0 . At initialization, $\chi(\nu_0) = 0$, and $\text{Rank}(\nu_0)$ is constructed by first computing the frontier loop \mathcal{L}_ν associated with s_0 , then extracting the visible gaps $\Gamma_{\mathcal{L}_\nu}$, and finally ranking them by their predicted cost-to-connect.

2) *Node Selection:* At each iteration, the node with minimum best-first priority is selected from the queue. The priority function balances the realized execution cost and the estimated effort of remaining gaps:

$$f(\nu) \triangleq \chi(\nu) + \min_{g \in \text{Rank}(\nu)} \widehat{\text{Cost}}(g), \quad (10)$$

where $\chi(\nu)$ is the cumulative execution cost so far, and the second term is the minimum predicted remaining cost among the unexplored gaps of ν . This scoring ensures that nodes are expanded in an order that jointly accounts for physical effort already incurred and the most promising future actions.

3) *Node Expansion with Parallel Simulation:* When a node ν is selected for expansion, a batch of *pushing strategies* from $\text{Rank}(\nu)$ is evaluated in parallel by simulation. Each pushing strategy is defined as $\tau \triangleq (g, \mathbf{v}, \xi)$, where $g \in \Omega$ is the chosen gap; $\mathbf{v} \triangleq (v_x, v_y, \omega) \in \mathbb{R}^3$ is a short-horizon body velocity for the manipulated obstacle; and $\xi \triangleq (\mathcal{C}_m, \mathbf{u}_m) \in (\partial\Omega_m)^N \times \mathbb{R}^{2N}$ is a contact mode specifying robot contact points and pushing forces as in (3). Moreover, each candidate τ is first checked by a geometric quick-pass. If it passes, it is evaluated by the simulator with the current state and pushing strategy:

$$(s', \delta_T, \delta_J) \triangleq \text{EvalSim}(\nu, (g, \mathbf{v}, \xi)), \quad (11)$$

where s' is the successor state; $\delta_T > 0$ is the time cost; and δ_J is the control effort as in (5). Thus, a successful evaluation produces a child $\nu' \triangleq (s', \pi')$, where π' is obtained by appending π with τ . The incremental realized cost of τ is $\widehat{\text{C}}(\nu, \tau) \triangleq \delta_T + \alpha \delta_J$, with $\alpha > 0$ the same trade-off parameter as in (5). The cumulative cost is then updated by $\chi(\nu') \triangleq \chi(\nu) + \widehat{\text{C}}(\nu, \tau)$. Since many pushing strategies are simulated concurrently, numerous successors can be expanded in parallel at each iteration.

4) *Termination:* Termination occurs when a node ν reaches a state s for which a W -clear path \mathcal{P}_V^W exists from s_V^S to s_V^G , as certified by (7). In this case, the schedule π stored in ν constitutes a complete solution to (5), encoding both the sequence of gaps to clear and the physically feasible pushing actions that realize them.

Remark 4 (Parallel Expansion and Mode Reuse). Efficiency of the hybrid search stems primarily from parallel evaluation of pushing strategies, which allows many candidate futures to be simulated at once for each expansion. Deferred expansion ensures that nodes with long candidate lists remain in the priority queue until all tasks are attempted. Moreover, validated (\mathbf{v}, ξ) pairs are cached locally within a subtree and stored in a persistent ModeTable for reuse across similar obstacles, significantly reducing repeated physics calls. These mechanisms yield orders-of-magnitude speedup relative to naive simulation-based search. ■

D. Overall Analyses

1) *Execution and Online Adaptation:* The output of the hybrid search is a pushing schedule $\pi = \{\tau_1, \dots, \tau_K\}$, where each τ_k specifies a target gap, a short-horizon velocity $\mathbf{v}_k \triangleq (v_x, v_y, \omega)$, and a contact mode ξ_k . Execution proceeds sequentially under a hybrid controller that alternates between transition and pushing phases.

During transition, robots plan collision-free paths in the instantaneous freespace $\widehat{\mathcal{W}}(t)$ to reach their designated contact points. If two paths are imminently head-on, the associated contacts are swapped to avoid deadlock. During pushing, the commanded velocity \mathbf{v}_k is integrated to form a short reference trajectory for the manipulated obstacle. This trajectory is mapped to per-robot contact states using ξ_k , and each robot R_n executes proportional velocity control:

$$\mathbf{v}_n = K_p(\widehat{\mathbf{p}}_n^c - \mathbf{p}_n^c), \quad \omega_n = K_r(\widehat{\psi}_n^c - \psi_n^c),$$

where $(\hat{\mathbf{p}}_n^c, \hat{\psi}_n^c)$ are the reference contact states, $(\mathbf{p}_n^c, \psi_n^c)$ are the measured states, and $K_p, K_r > 0$ are control gains. Small offsets are adapted online to compensate for yaw drift and maintain stable contact.

Execution is monitored continuously. Transition feasibility is checked by timeouts, while pushing is monitored by early-stop tests. If the commanded widening succeeds, execution advances to τ_{k+1} ; otherwise, the executor returns control to the planner with the current state. At a lower rate, the WCCG is rebuilt and the execution terminates immediately once a W -clear path \mathcal{P}_v^W exists between s_v^S and s_v^G .

Remark 5 (Endpoint Booster). The W -CCG criterion of (7) may be conservative near the endpoints. If connectivity holds but the disks $\mathbb{B}_{W/2}(s_v^S)$ or $\mathbb{B}_{W/2}(s_v^G)$ intersect obstacles, the planner generates a small set of auxiliary pushes directed away from the endpoints. These pushes clear the disks and allow execution to proceed. ■

2) *Computational Complexity*: The cost of each node expansion is dominated by simulation of pushing strategies. Construction of the W -clearance graph \mathcal{G}_W and the associated connectivity tests scale nearly linearly in the number of movable obstacles $|\Omega|$, while gap ranking by presearch (Sec. III-B) is lightweight because it is limited to a small subset of frontier gaps. At expansion, only a fraction of the strategies in $\text{Rank}(\nu)$ survive the geometric quick-pass and early-stop checks, which shortens the horizon of the simulator calls defined by EvalSim. Parallel execution of these simulations across multiple workers reduces the effective cost nearly linearly until communication overhead is reached. Deferred expansion further improves efficiency by distributing the evaluation of $\text{Rank}(\nu)$ across iterations, avoiding redundant restarts and keeping the priority queue focused on promising frontiers.

3) *Generalization*: The framework admits several natural extensions: (I) *Heterogeneous teams*. The controller structure admits per-robot gain tuning, and mode priors can be augmented with robot-specific constraints. This enables heterogeneous robots to cooperate within the same framework; (II) *Concurrent pushing*. While the executor applies one push task at a time, the framework can be extended to allow concurrent pushing of multiple obstacles. Such an extension requires multi-object mode generation and transition planning to account for inter-robot conflicts; (III) *Dynamic environments*. Periodic W -connectivity checks and on-demand re-planning enable adaptation to moderate disturbances such as object drift or unmodeled contacts. The framework can therefore generalize to settings where the environment changes gradually during execution.

IV. NUMERICAL EXPERIMENTS

We evaluate the proposed simulation-in-the-loop NAMO planner (SiLS) in cluttered environments with both movable and immovable obstacles. All components (W -CCG presearch, frontier extraction, and ModeTable prior) are integrated as described in Sec. III. The implementation is in Python 3 and simulations are run in PyBullet [19]

on a laptop with an Intel Core i7-1280P CPU. Videos and logs are provided in the supplementary material.

A. Numerical Simulations

1) *Setup*: The physics step is $\Delta t = 1/120$ s and the control period is $1/40$ s. Robots are disk/box pushers with risk radii consistent with the W -clearance definition. Movables have masses uniformly sampled in $[5, 15]$ kg; immovables are modeled with mass 0. A trial succeeds when a W -clear path from start to goal exists and the target reaches its goal disc.

Workspace and scenarios. We use a *nominal scenario* with an 8×5 m bounded workspace, two internal bars (bottlenecks), and a mixed pool of movable shapes (curved and polygonal, including rings, ellipses, X/T/L/diamond, arrow-like, rectangles, and cylinders). Two robots start in the lower left; the target goal lies on the right side. Movables are randomly placed with a minimum separation. We test 10–15 objects over multiple seeds.

Geometry and caching. Collision checks use polygon/curve-edge models with convex decomposition when needed. Ray–edge queries for frontiers are vectorized with AABB culling. Frontier loops are cached by signatures and transitions $(\mathcal{L}, g) \mapsto \mathcal{L}'$ are memoized.

2) *Algorithm Configuration*: Unless stated otherwise: per-task simulation horizon = 80 steps, up to 64 candidate push tasks per expansion, and priority $f = g + \widetilde{\text{Cost}}_{\text{to-go}}$ with heuristic factor 10. Gap sampling uses a softmax with temperature 0.05. ModeTable is enabled (auto-baked if missing). A *quick-pass* geometry screen may skip physics if a reference rollout already clears the gap; otherwise a short-horizon simulation with early stop is used. To avoid premature termination, a *deferred-reinsertion* rule keeps high-value but temporarily unexpanded nodes in the queue and revisits them later.

3) *Baselines*: We compare against three representative families, all using the same W -clearance criterion and contact models:

DFS-WCCG: Simulation-in-the-loop depth-first search that shares our physics predictor and W -CCG for goal checking. Each node is a snapshot; actions are four fixed axis-aligned pushes $! \leftarrow, \rightarrow, \uparrow, \downarrow!$ applied to any movable object (branching $\leq 4n$ for n objects).

SL-Push: Straight-line (or waypoints) route; blockers are cleared by (i) off-line minimal normal displacements or (ii) sim-in-the-loop normal pushes from near to far.

Rec-NAMO: Recursive routing/pushing on a cost-weighted visibility graph: Dijkstra for routing, push-decomposition for local clearing; failures prune edges and replan.

4) *Metrics*: We report success rate, wall-clock time, number of node expansions, number of simulated pushes, cumulative push time, average presearch cost-to-go, quick-pass ratio, early-stop ratio, and worker-pool utilization. Results are averaged over 10 seeds unless noted.

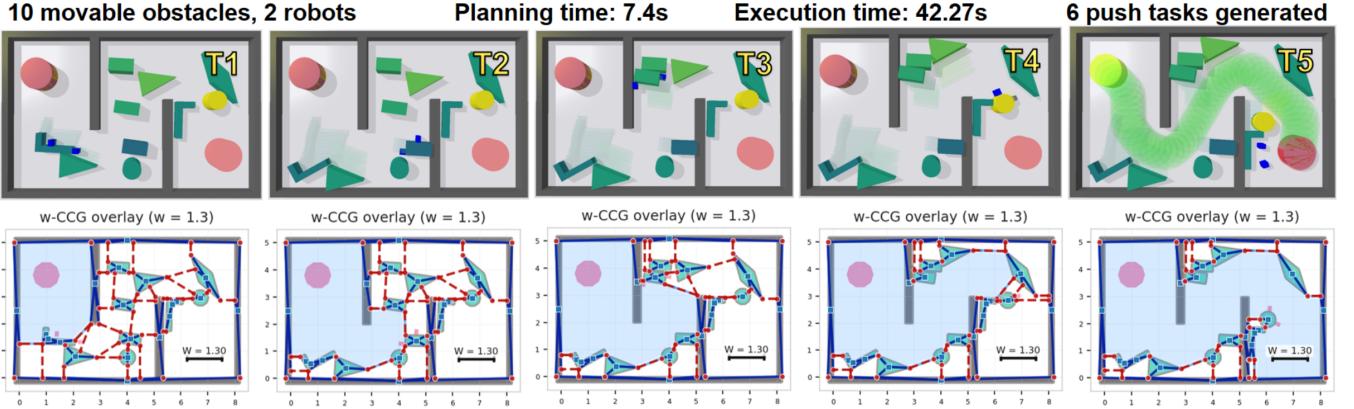


Fig. 5. Execution–planning alignment in the main scenario. **Top:** five PyBullet snapshots of the scene. **Bottom:** corresponding WCCG with the start face highlighted as a translucent blue polygon. The task moves a cylindrical object from the start $S = (1, 4)$ to the goal $G = (7, 2)$. In the final snapshot, the start face contains G , consistent with the green execution trajectory in the top row, confirming successful completion from S to G .

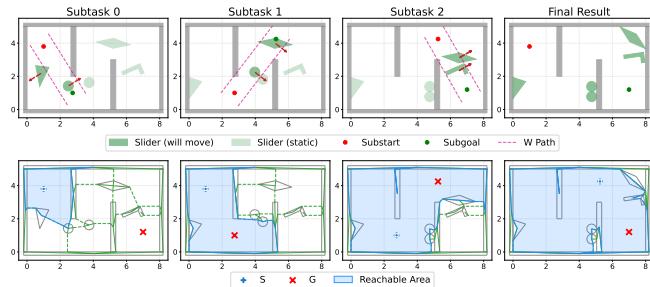


Fig. 6. SL-Push. Compute(offline) or simulate(sim-in-the-loop) the pushing directions for movable obstacles along the W-width straight path from sub-start to sub-goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

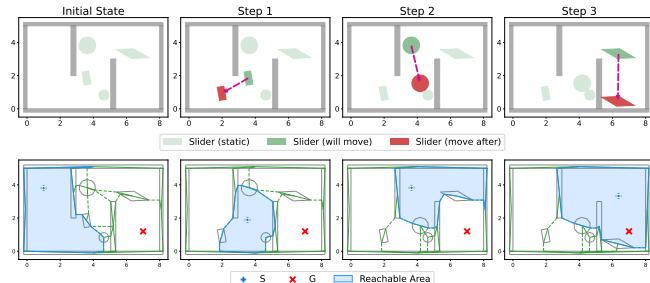


Fig. 7. Rec-NAMO. Rec-NAMO builds path segments sequentially but fails to construct the full W-clear path from start to goal. Top: Pushing steps. Bottom: Reachable region changes during pushing

5) Comparison of Main Results: Table I summarizes the performance of four baseline methods and our proposed SiLS method on the nominal scenario. DFS-WCCG exhibits high computational overhead due to the exponential growth of the search space with the number of movable obstacles, resulting in very long planning times and frequent timeouts, which is reflected in its lowest success rate. SL-Push (offline) achieves the fastest planning times, typically under 0.1 s, but the lack of physics-informed simulation leads to physically infeasible solutions. The simulation-in-the-loop type of SL-Push improves solution feasibility; however, it still requires

TABLE I
PERFORMANCE ON THE NOMINAL SCENARIO WITH
PUSH-COUNT (MEAN \pm STD).

Method	Succ. (%)	#PT (s)	#ET (s)	#Sims	#Pushes
DFS-WCCG	25.0	41.8	N/A	–	–
SL-Push (off-line)	62.5	0.03	145.1	0.0	7.0
SL-Push (sim)	75.0	25.1	246.8	16.0	8.0
Rec-NAMO	37.5	10.5	179.3	0.0	7.0
SiLS (ours)	92.5	10.3	28.6	121.5	6.0

Metrics. Succ. = success rate; #PT = planning time; #ET = execution time; #Sims = simulations invoked; #Pushes = length of executed push sequence.

many simulation calls, resulting in increased planning times. Furthermore, the straight-line pushing strategy may generate unnecessarily long action sequences, as the direct path is not always the minimal pushing path. Figure 6 illustrates the step-by-step execution of SL-Push with two manually defined subgoals, showing how the environment gradually becomes connected such that the start and goal are in the same connected face.

Rec-NAMO, while faster than simulation-in-the-loop approaches and effective in classical NAMO tasks, struggles to generate fully connected W-clear paths in our setting. Although its recursive search produces locally connected regions along the pushing sequence, the final map often fails to contain a continuous W-clear path, as demonstrated in Figure 7. This limitation is reflected in its relatively low success rate in Table I.

Our SiLS method attains higher success rates with fewer simulation calls and shorter planning and execution times, thanks to (i) frontier-based gap ranking, (ii) ModeTable-guided push directions, and (iii) quick-pass/early-stop. In addition, deferred reinsertion prevents priority-queue starvation by revisiting previously generated high-value nodes when a batch of actions fails, further improving efficiency and solution quality.

6) Ablations: We remove one component at a time: (i) W-CCG presearch (random gap order), (ii) ModeTable prior

6 movable obstacles, 2 robots

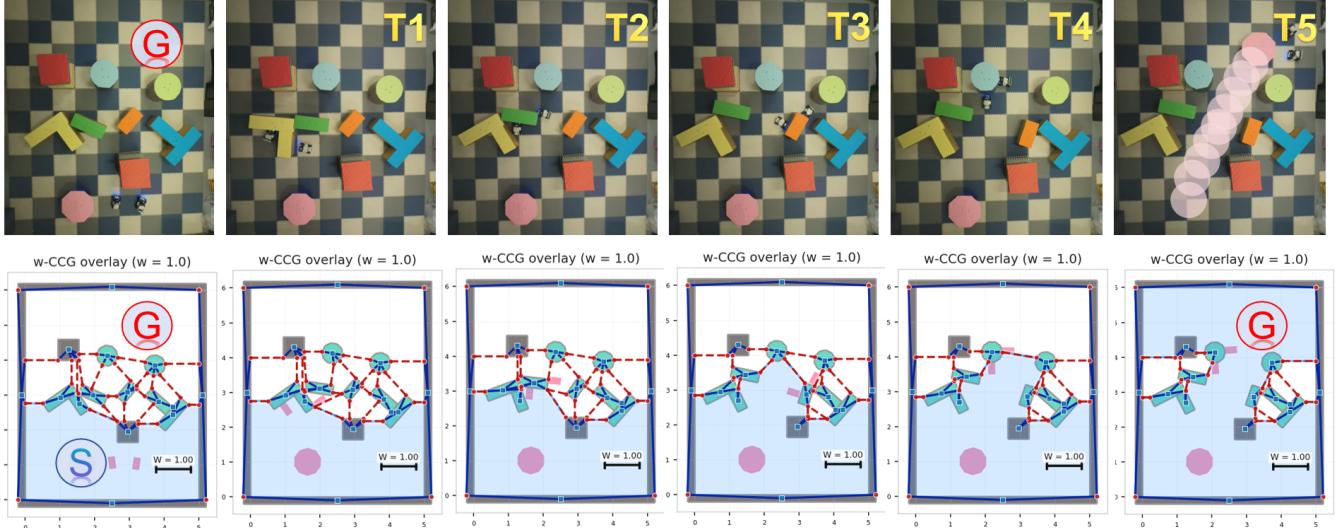


Fig. 8. Real-world pushing experiment with execution–planning alignment. **Top:** six snapshots from the physical scene as two robots push a target object across a 5×6 m workspace with both movable pieces and fixed boundary segments. **Bottom:** corresponding WCCG overlays, where the start face is highlighted as a translucent blue polygon. Planning is posed with clearance $\bar{W} = 1.0$, start $S = (1.65, 1.01)$, and goal $G = (3.4, 4.8)$. Across snapshots, the graph structure and candidate gaps evolve consistently with the observed object motion, illustrating that the planned corridors remain compatible with the real execution.

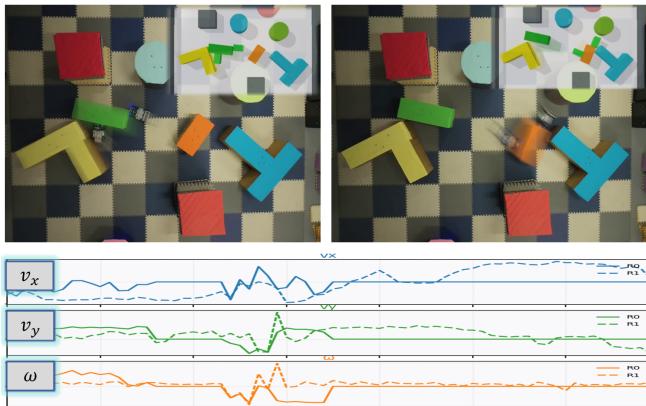


Fig. 9. Two-frame highlight of chain pushing in the real-world experiment. **Top:** two consecutive video snapshots illustrating a contact-induced cascade: Robot R1 pushes a movable object, which in turn imparts motion to a neighboring object (chain push). **Bottom:** control panel showing commanded velocities for both robots (v_x , v_y , ω). The temporal alignment between the frames and the plots reveals the contact transition: a brief rise in translational speed and differential angular commands near the interaction, followed by moderated controls as the secondary object begins to move. This pairing demonstrates that the planned interaction is executed as intended by both robots.

(fallback “away + jitter” only), (iii) quick-pass/early-stop (always full physics), (iv) deferred reinsertion (terminate on empty queue). Table II shows consistent drops in success and increased time/#Sims when any component is disabled.

7) *Efficiency and Scalability:* **Frontier caching** makes BugPlanner near-linear in visible edges. **Parallel prediction** uses a persistent worker pool with per-round broadcast of snapshots and reachable-contact sets to reduce IPC. **Quick-pass** and **early-stop** skip or truncate physics when geom-

TABLE II

ABALATIONS ON THE NOMINAL SCENARIO (AVERAGED OVER 40 RUNS).

Variant	Succ. (%)	Time (s)	#Sims	Quick-pass (%)
SiLS (full)	92.5	28.6	122	57.5
w/o presearch	85.0	35.2	178	25.0
w/o ModeTable	80.0	33.4	164	40.0
w/o quick-pass/ES	90.0	41.9	208	0.0
w/o reinsertion	82.5	31.1	151	47.5

etry suffices. **Deferred reinsertion** maintains steady depth growth even when many pushes are rejected.

8) *Qualitative Results:* Figure ?? shows a typical run: frontiers and ranked gaps, a top-ranked gap sequence, simulated pushes that widen bottlenecks, and the final W -clearance path. Per-step overlays and GIFs are produced by a lightweight snapshot logger.

9) *Reproducibility:* Random seeds are fixed, JSONL snapshots are logged, and the driver and scenario generator are released. ModeTable entries are auto-generated if absent to ensure repeatability across machines.

V. CONCLUSION

This work introduces a multi-robot approach for path clearing in unstructured environments, utilizing a hybrid search algorithm to plan and execute the sequence of obstacles, contact points, and forces efficiently. The framework demonstrates real-time adaptability to dynamic scenarios. Future work will address uncertainties such as estimating obstacle positions without external monitoring, handling uncertain obstacle masses, and managing partial robot visibility to improve robustness and performance in real-world, dynamic environments.

REFERENCES

- [1] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, 2023.
- [2] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.
- [3] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE, 2007, pp. 3327–3332.
- [4] L. Yao, V. Modugno, A. M. Delfaki, Y. Liu, D. Stoyanov, and D. Kanoulas, "Local path planning among pushable objects based on reinforcement learning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 3062–3068.
- [5] Z. Tang, Y. Feng, and M. Guo, "Collaborative planar pushing of polytopic objects with multiple robots in complex scenes," *arXiv preprint arXiv:2405.07908*, 2024.
- [6] Z. Ren, B. Suvonov, G. Chen, B. He, Y. Liao, C. Fermuller, and J. Zhang, "Search-based path planning in interactive environments among movable obstacles," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 533–539.
- [7] S. Goyal, A. Ruina, and J. Papadopoulos, "Limit surface and moment function descriptions of planar sliding," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1989, pp. 794–795.
- [8] K. M. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *IROS*, vol. 1, 1992, pp. 416–421.
- [9] R. Ni and A. H. Qureshi, "Progressive learning for physics-informed neural motion planning," *arXiv preprint arXiv:2306.00616*, 2023.
- [10] Y. Liu, R. Ni, and A. H. Qureshi, "Physics-informed neural mapping and motion planning in unknown environments," *IEEE Transactions on Robotics*, 2025.
- [11] J. Chen, M. Gauci, W. Li, A. Kolling, and R. Groß, "Occlusion-based cooperative transport with a swarm of miniature mobile robots," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 307–321, 2015.
- [12] R. Ni and A. H. Qureshi, "Physics-informed neural motion planning on constraint manifolds," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 12 179–12 185.
- [13] Y. Wang and C. W. De Silva, "Multi-robot box-pushing: Single-agent q-learning vs. team q-learning," in *2006 IEEE/RSJ international conference on intelligent robots and systems*, 2006, pp. 3694–3699.
- [14] Y. Feng, C. Hong, Y. Niu, S. Liu, Y. Yang, and D. Zhao, "Learning multi-agent loco-manipulation for long-horizon quadrupedal pushing," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 14 441–14 448.
- [15] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5280–5286.
- [16] Q. Rouxel, S. Ivaldi, and J.-B. Mouret, "Multi-contact whole-body force control for position-controlled robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5639–5646, 2024.
- [17] Z. Tang, J. Chen, and M. Guo, "Combinatorial-hybrid optimization for multi-agent systems under collaborative tasks," in *IEEE Conference on Decision and Control (CDC)*, 2023.
- [18] K. N. McGuire, G. C. H. E. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *Robotics and Autonomous Systems*, vol. 121, p. 103261, 2019.
- [19] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," <http://pybullet.org>, 2016–2019.