

PushAround: Collaborative Path Clearing via Physics-Informed Hybrid Search

Zili Tang¹, Zongyuan Li², Zishao Qiao¹, and Meng Guo¹

Abstract—Large vehicles in cluttered environments are often blocked by movable obstacles, motivating mobile robot teams to clear traversable corridors. Navigation Among Movable Obstacles (NAMO) approaches often abstract obstacle displacements while neglecting robot dimensions, obstacle mass, contact geometry and forces, yielding physically infeasible plans. *PushAround* is a physics-informed hybrid search that couples geometric connectivity reasoning with short-horizon simulation to construct executable clearing plans. Repeated W-clearance connectivity graph (WCCG) queries expose frontier gaps requiring widening and rank them by estimated downstream costs. The search generates direct or recursive pushing candidates, selecting obstacles, directions, contacts and forces, and admits physics-evaluated candidates. Numerical experiments validate WCCG against standard connectivity checks and demonstrate scaling to denser clutter and teams of up to four robots, with robustness to simulated execution-side physical-parameter mismatch.

Index Terms—Multi-Robot Systems, Task and Motion Planning, Manipulation Planning.

I. INTRODUCTION

IN warehouses, disaster sites and urban spaces, movable obstacles such as pallets, boxes and equipment can block large vehicles, motivating mobile robot teams to clear traversable corridors. Conventional path planners assume static obstacles and fail once routes are blocked [1]. Navigation Among Movable Obstacles (NAMO) creates routes by pushing, pulling or rotating obstacles [2], yet many methods abstract away robot dimensions, obstacle size and mass, contact geometry, forces and dynamics, producing *geometrically valid but physically unrealizable plans*. As shown in Fig. 1, a large robot may be unable to reach obstacle boundaries while a small robot lacks pushing force. Cooperative pushing by several small robots offers an alternative, but simultaneous pushes can trigger chained motions of several obstacles, introducing physical coupling that complicates planning.

A. Related Work

1) *Navigation Among Movable Obstacles*: NAMO extends motion planning to cluttered environments by planning obstacle displacements through pushing, pulling, or rotating [2],



Fig. 1. Collaborative path clearing in simulation (Top) and hardware (Bottom). Two robots push movable obstacles to create a W-clear path for a large vehicle from start to goal.

[3]. Later work considers heuristic search, learning-based local planning, physics-based rollouts, and collaborative pushing [3]–[7]. However, many NAMO formulations simplify robot geometry, actuation limits, object mass, and contact dynamics, which can yield unrealizable plans.

2) *Collaborative Pushing*: Studies cover pushing mechanics and limit surfaces [8], [9], cooperative transport with robot swarms [10], multi-robot box pushing with reinforcement learning [11], collaborative pushing of polytopic objects in cluttered scenes [7], and learning-based locomanipulation [12]. These studies focus on a prescribed object rather than multi-object corridor clearing.

3) *Physics-Informed Planning*: Dynamics models, contact reasoning, and simulation feedback inform motion generation: learned dynamics for humanoid contact planning [13], multi-contact force control [14], demonstration-guided optimal control over contact modes [15], physics simulation interleaved with multi-agent pathfinding [16], [17], and learned physical priors [18], [19]. These methods typically address a prescribed target or a small set of objects. Corridor clearing instead requires choosing which of many movable obstacles to rearrange so that a larger vehicle can pass.

B. Our Method

As shown in Fig. 2, this work introduces **PushAround**, a physics-informed framework for collaborative multi-robot pushing that constructs traversable corridors for large vehicles in cluttered environments. Its central novelty is a hybrid search that jointly determines which obstacles to displace and how to push them, coupling the high-level clearing sequence with low-level pushing modes specifying contact points, directions

Manuscript received: June 24, 2026; Revised: September 16, 2026; Accepted: October 20, 2026.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China under grants U2241214 and T2121002. (Corresponding author: Meng Guo.)

The authors are with Peking University, Beijing 100871, China: ¹School of Advanced Manufacturing and Robotics; ²College of Engineering (e-mail: meng.guo@pku.edu.cn).

Digital Object Identifier (DOI): see top of this page.

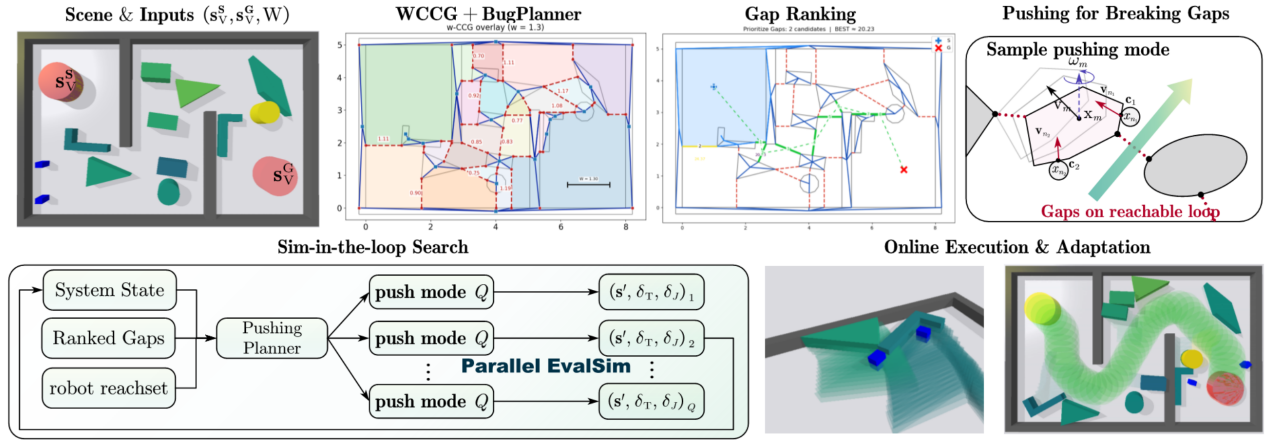


Fig. 2. Overview of the proposed framework. **Top:** Scene inputs, WCCG construction, gap ranking, and sampling of pushing modes for breaking frontier gaps; **Bottom-left:** Sim-in-the-loop hybrid search with parallel validation in simulation; **Bottom-right:** Online execution and adaptation during path clearing.

and forces. The framework integrates a W -clearance connectivity graph (WCCG) that certifies vehicle passage under width constraints, frontier-gap ranking by estimated effort, and pushing-mode generation under geometric and actuation constraints. Guided by the selected gap, the search proposes direct pushes or recursively clears blocking obstacles, and expands the plan through physics-evaluated transitions that account for robot dimensions, object mass, contact geometry and coupled pushing dynamics. Execution feedback updates connectivity and triggers replanning when needed.

The contributions are twofold: (I) it introduces a unified framework that couples collaborative pushing with physics-informed verification, producing executable plans for path clearing in cluttered environments; and (II) it demonstrates empirical gains in feasibility, efficiency and scalability over evaluated NAMO and collaborative-clearing baselines, with robustness to execution-side physics mismatch.

II. PROBLEM DESCRIPTION

A. Workspace and Robots

The workspace is a bounded planar region $\mathcal{W} \subset \mathbb{R}^2$ containing immovable structures $\mathcal{O} \subset \mathcal{W}$ and $M \geq 0$ movable rigid polygons $\Omega \triangleq \{\Omega_1, \dots, \Omega_M\}$. Each movable obstacle $\Omega_m \subset \mathcal{W}$ has mass M_m , inertia I_m , frictional parameters and state $\mathbf{s}_m(t) \triangleq (\mathbf{x}_m(t), \psi_m(t))$, with $\mathbf{x}_m(t) \in \mathbb{R}^2$ the centroid position and $\psi_m(t) \in \mathbb{R}$ the orientation; $\Omega_m(t)$ denotes the region it occupies at time t . A team of N robots, $\mathcal{R} \triangleq \{R_1, \dots, R_N\}$, operates as a cooperative unit. Each robot R_i is a rigid body with state $\mathbf{s}_{R_i}(t) \triangleq (\mathbf{x}_{R_i}(t), \psi_{R_i}(t))$ and footprint $R_i(t)$. The instantaneous free space is given by:

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

which holds for both the movable obstacles and robots.

B. External Vehicle and Clearance Goal

An external vehicle $\mathcal{V}$ whose footprint is enclosed by a disk of radius $W/2 > 0$ must navigate within the workspace from a start position $\mathbf{x}_V^S \in \mathcal{W}$ to a goal position $\mathbf{x}_V^G \in \mathcal{W}$. The robots rearrange obstacles to admit this disk. At time t , the admissible

center space is $\mathcal{F}_W(t) \triangleq \{p \in \widehat{\mathcal{W}}(t) \mid \mathbb{B}_{W/2}(p) \subset \widehat{\mathcal{W}}(t)\}$, where $\mathbb{B}_{W/2}(p)$ is the disk of radius $W/2$ centered at p . The W -clearance condition is:

$$\exists \mathcal{P}_V^W \subset \mathcal{F}_W(t) : \mathbf{x}_V^S \rightsquigarrow \mathbf{x}_V^G, \quad (2)$$

which certifies a collision-free centerline path for any vehicle footprint contained in the enclosing disk.

C. Collaborative Pushing Modes

To enlarge blocked gaps, the robots may actively reconfigure Ω by pushing movable obstacles. For obstacle $\Omega_m \in \Omega$, a pushing mode is denoted by $\xi_m \triangleq (\mathbf{c}_m, \mathbf{u}_m)$, where $\mathbf{c}_m = (c_1, \dots, c_N)$ is an ordered tuple of contact points on $\partial\Omega_m$, and the ordering specifies the robot-to-contact assignment. The vector $\mathbf{u}_m \in \mathbb{R}^{2N}$ stacks the corresponding planar contact forces, expressed in the body frame of Ω_m ; an inactive robot carries a null contact $c_i = \emptyset$ with zero force. For a contact tuple $\mathbf{c}_m$, the set of admissible forces $U_m(\mathbf{c}_m) \subset \mathbb{R}^{2N}$ collects contact forces satisfying the robot pushing limits, robot-object friction, and unilateral contact constraints. The body wrench induced by a mode is defined as:

$$\mathbf{q}_m(\xi_m) \triangleq G_m(\mathbf{c}_m)\mathbf{u}_m, \quad (3)$$

where $G_m(\mathbf{c}_m) \in \mathbb{R}^{3 \times 2N}$ is the contact-to-wrench mapping; and $\mathbf{q}_m(\xi_m) \in \mathbb{R}^3$. The admissible mode set is state dependent as denoted by $\Xi_m(\mathbf{s})$. It contains modes whose contacts are reachable from the current robot configurations, pre-contact configurations are feasible in the current clutter, and forces satisfy $\mathbf{u}_m \in U_m(\mathbf{c}_m)$. Hence, applying a feasible mode $\xi_m \in \Xi_m(\mathbf{s}(t))$ yields the physical transition:

$$\mathbf{s}(t^+) \triangleq \Phi(\mathbf{s}(t), m, \xi_m), \quad (4)$$

where $\mathbf{s}(t)$ stacks all robot and obstacle states. The force vector $\mathbf{u}_m$ serves as a feasibility variable to certify wrench generation under the modeled contact and actuation constraints rather than a directly commanded control input.

D. Problem Statement

The goal is to compute a *hybrid schedule* that reconfigures movable obstacles to admit a W -feasible path for the external vehicle. The schedule is defined as:

$$\pi \triangleq \{(\Omega_{m_k}, \mathbf{v}_k, \xi_k)\}_{k=1}^K, \quad (5)$$

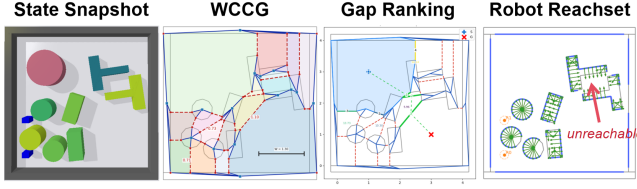


Fig. 3. Overview of the proposed physics-informed hybrid search (PIHS): WCCG construction, gap ranking, mode generation and physics validation.

where $\Omega_{m_k} \in \Omega$ is the manipulated obstacle, $\mathbf{v}_k$ its pushing direction, $\xi_k \in \Xi_{m_k}$ its pushing mode. The node cost is:

$$\mathcal{J}_{\text{chr}}(\pi) \triangleq T + \alpha \sum_{k=1}^K J(m_k, \xi_k, \mathbf{s}(t_k)), \quad (6)$$

where $T \triangleq \sum_{k=1}^K \Delta t_k$ denotes the robotic clearance duration excluding vehicle traversal, with $\Delta t_k > 0$ the duration that evaluation in (13) returns for the k -th push, and $J(\cdot)$ is the simulation-estimated manipulation cost weighted by $\alpha > 0$. Vehicle traversal enters only through the terminal W -clearance condition in (2). The constraints in (1)–(4) enforce valid pushing transitions. The realized clearance cost is defined in (6); PIHS samples finitely and does not guarantee optimality.

Remark 1. The objective couples obstacle selection, pushing-mode generation and switching in one hybrid planning problem. Its combinatorial-continuous search includes robot-team contacts and forces alongside the obstacle rearrangements considered in NAMO [2], [3]. ■

III. PROPOSED SOLUTION

As shown in Figs. 2 and 3, PushAround first uses the W -Clearance Connectivity Graph in Sec. III-A and gap ranking in Sec. III-B to identify and prioritize blocking gaps. It then generates and evaluates physically feasible collaborative pushing modes in Sec. III-C. These components are integrated into the simulation-in-the-loop hybrid search of Sec. III-D and the online execution procedure of Sec. III-E.

A. W -Clearance Connectivity Graph

The feasibility of routing the vehicle reduces to whether a W -width corridor connects $\mathbf{x}_V^S$ and $\mathbf{x}_V^G$. A W -clearance connectivity graph (WCCG) encodes obstacle adjacencies under clearance W for fast queries, as shown in Fig. 3. Grid- and sampling-based clearance checks typically depend on a chosen grid or sampling resolution and must update their queries when obstacles move [20]. Built from obstacle geometry, the WCCG needs no grid-resolution parameter in these repeated queries.

1) *Graph Construction:* The WCCG is built by decomposing all clearance-relevant obstacle and workspace boundaries into convex components or segments. 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 less 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 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 marked as potential bottlenecks. The resulting graph is given by:

$$\mathcal{G}_W \triangleq (\mathcal{V}, \mathcal{E}_c \cup \mathcal{E}_b), \quad (7)$$

where $\mathcal{V}$ contains centroid and bridge nodes; $\mathcal{E}_c$ contains centroid-bridge edges; and $\mathcal{E}_b$ contains bridge-bridge edges annotated by gap width.

2) *Connectivity Criterion:* Once $\mathcal{G}_W$ has been constructed, connectivity queries can be performed without explicitly computing a geometric path. Let $\text{Face}(\mathcal{G}_W)$ denote the set of connected planar faces induced by the embedded WCCG, i.e., the maximal regions separated by obstacle components and WCCG bottleneck edges. The criterion for (2) is given by

$$(\mathbf{x}_V^S, \mathbf{x}_V^G \in \mathcal{F}_W(t)) \wedge (\exists F \in \text{Face}(\mathcal{G}_W) : \mathbf{x}_V^S, \mathbf{x}_V^G \in F), \quad (8)$$

where the faces follow the standard configuration-space inflation interpretation: expanding obstacles by a disk of radius $W/2$ turns gaps narrower than W into overlapping barriers, and non-convex obstacles are decomposed into clearance-relevant components, after which WCCG records the separating bottlenecks. The consistency of WCCG connectivity with both continuous and rasterized configuration-space references is further evaluated numerically in Sec. IV-A3. A frontier-tracing procedure, similar to the BugPlanner [21], is then used to extract a skeleton Σ as an ordered sequence of centroid and bridge nodes. Starting from $\mathbf{x}_V^S$, it casts a ray toward $\mathbf{x}_V^G$ and follows the encountered frontier loop until either a valid exit is found or a blocking cycle is returned.

B. Ranking of Frontier Gaps

When (8) fails, the planner ranks blocking gaps on the reachable frontier of $\mathcal{G}_W$ by predicted downstream cost and passes them to the hybrid search as local clearance targets, linking the gap-ranking and pushing modules in Fig. 2.

1) *Frontier Extraction and Candidate Gaps:* A BugPlanner query on $\mathcal{G}_W$ above either confirms connectivity or returns a counter-clockwise frontier loop $\mathcal{L}$ that separates the two endpoints. The bridge-bridge edges on $\mathcal{L}$ define the first-hop candidate set $\Gamma_{\mathcal{L}} \triangleq \{g_1, \dots, g_K\}$, where each gap g_k is represented by its two closest boundary points, its width $w_{g_k} < W$, and the adjacent obstacles that can potentially be moved.

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 $\mathbf{o}_g$ the outside insertion point of gap g . The resulting one-hop cost is given by:

$$\mathcal{C}(g \mid \mathcal{L}, \mathbf{s}_{\mathcal{R}}) = \lambda_t \mathcal{C}_t(\mathbf{s}_{\mathcal{R}}, \mathbf{o}_g) + \lambda_p (\mathcal{C}_{\text{dp}}(g) + \mathcal{C}_{\text{rp}}(g)), \quad (9)$$

where $\lambda_t, \lambda_p > 0$ weight the costs and $\mathcal{C}_t, \mathcal{C}_{\text{dp}}, \mathcal{C}_{\text{rp}}$ are the transition, direct and recursive terms.

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., $h(g) = \eta \|\mathbf{o}_g - \mathbf{x}_V^G\|$, with the scaling constant $\eta > 0$. The *gap-lookahead* procedure performs an A*-style presearch with bounded width and depth, virtually crossing gaps and recomputing frontiers. The *cost-to-connect* is predicted by:

$$\widehat{\text{Cost}}(g) \triangleq \mathcal{C}(g \mid \mathcal{L}, \mathbf{s}_{\mathcal{R}}) + \sum_{g' \in \Pi^*(g)} \{\mathcal{C}(g' \mid \cdot) + h(g')\}, \quad (10)$$

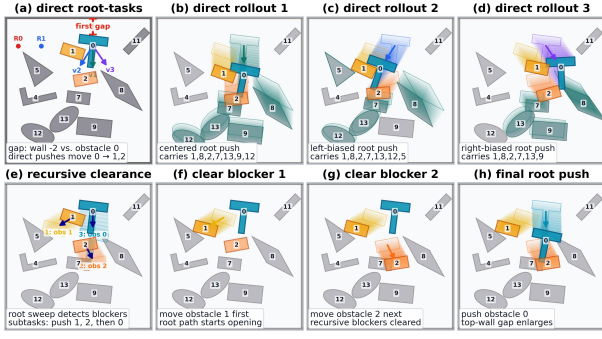


Fig. 4. **Direct pushing (Top)** moves a gap-side blocker along a gap-opening direction, whereas **recursive pushing (Bottom)** first clears movable blockers in the swept region and then returns to the root gap-opening task.

where $\Pi^*(g)$ is the sequence of subsequent gaps discovered after virtually crossing g . The module returns the ranked list $\text{Rank}(\Gamma_{\mathcal{L}}) \triangleq \text{argsort}_{g \in \Gamma_{\mathcal{L}}} \bar{\text{Cost}}(g)$, in ascending predicted cost, so promising gaps are expanded first. This geometric presearch estimates a promising bottleneck sequence without moving obstacles; the associated pushing actions are generated and evaluated in Sec. III-C.

C. Pushing Mode Generation and Evaluation

For a selected frontier gap g , this subsection constructs candidate pushing strategies $\Xi_g \triangleq \{(\Omega_m, \mathbf{v}, \xi)\}$, where $\mathbf{v} = (v_x, v_y, \omega) \in \mathbb{R}^3$ is a short-horizon pushing direction for Ω_m , and $\xi = (c_m, \mathbf{u})$ is its mode in (4). Wrench feasibility screens contact configurations that can realize the required motion within actuation limits. The forces certify feasibility rather than serving as hardware setpoints; candidate motion is then checked by rollout or quick-pass approximation.

1) *Direct and Recursive Pushing Tasks:* For *direct push*, let Ω_m denote one of the two movable obstacles adjacent to g . For each such obstacle, a small set of candidate body velocities $\{\mathbf{v}_m^k\}_{k=1}^{K_m}$ is sampled with a bias toward enlarging the gap g , while a few angular components are included to account for local geometry. If the swept region of a direct push intersects movable blockers, *recursive push* tasks are required instead: for each out-layer blocker Ω_m , the nominal direction moves it away from both the swept region and the reference gap sequence $\Pi^*(g)$. The bounded dependency chain in Fig. 4 prioritizes the outermost blocker Ω_m . Clearing this blocker can free space for its parent push, allowing the search to return toward the original gap-opening task.

2) *Pushing Mode Generation:* At a search state $\mathbf{s}$, a pushing mode is generated for each candidate direction $\mathbf{v}_m^k$ and obstacle Ω_m . In this subsection, contact tuples, object velocities, forces, and wrenches are expressed in the body frame of Ω_m . For a contact tuple $\mathbf{c}$, its multi-directional force feasibility is:

$$J_{\text{MF}}^m(\mathbf{c}, \mathbf{v}_m^k) \triangleq \sum_{\mathbf{v}_d \in \mathcal{D}_m^k} w_d \min_{\mathbf{u} \in U_m(\mathbf{c})} \|\mathbf{q}_m(\mathbf{c}, \mathbf{u}) + \mathbf{q}_m^\mu(\mathbf{v}_d)\|, \quad (11)$$

where $\|\cdot\|$ is a weighted ℓ_1 wrench norm, the finite set $\mathcal{D}_m^k$ contains perturbations of $\mathbf{v}_m^k$ with weights $w_d > 0$, and $\mathbf{q}_m^\mu(\mathbf{v}_d)$ is the quasi-static friction wrench; the inner minimization is a small LP over the admissible forces $U_m(\mathbf{c})$.

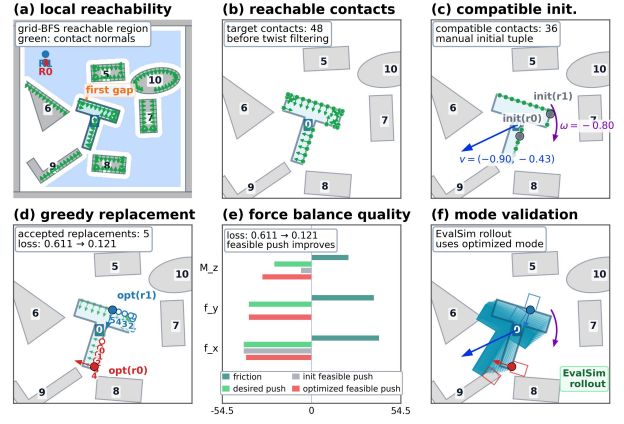


Fig. 5. Mode generation for a gap-opening task: (a) local reachability, (b) the reachable contact candidates after twist filtering, (c) a compatible initial contact tuple, (d) greedy replacement of individual contacts, (e) the force balance of the optimized push, and (f) rollout validation of the mode.

A state- and direction-dependent contact set $\mathcal{A}_m^k(\mathbf{s}) \subset \partial\Omega_m$ is drawn from the robot-reachable boundary of Ω_m by velocity–force compatibility filtering (Fig. 5), retaining only contacts with collision-free pre-contact poses. Starting from cached layouts or random initializations, sequential greedy replacement refines the tuple. At iteration ℓ , $\mathbf{c}_{r,c}^\ell$ denotes the tuple with contact r replaced by candidate c . The replacement is:

$$\mathbf{c}^* = \underset{c \in \mathcal{A}_m^k(\mathbf{s})}{\text{argmin}} J_{\text{MF}}^m(\mathbf{c}_{r,c}^\ell, \mathbf{v}_m^k), \quad (12)$$

where the tuple is updated with $\mathbf{c}^*$. Contact slots are swept until J_{MF}^m stops decreasing or the evaluation budget is exhausted. The best tuple over initializations gives $\mathbf{c}_m^{k,*}$, from which $\mathbf{u}_m^k$ follows from the inner minimization in (11), yielding $\xi_m^k = (\mathbf{c}_m^{k,*}, \mathbf{u}_m^k)$; each mode is stored in the *mode-cache* for subsequent warm starts. If the sequential replacement cannot assign a contact to every robot, a bounded backtracking search over robot–contact combinations attempts the full-team assignment. If no feasible full-team mode is found within this budget, for instance on small faces or under the pairwise pusher-clearance constraint, one robot is temporarily left inactive and the assignment is repeated with the remaining team. Reduced-team modes carry a ranking penalty, so a feasible full-team mode is preferred, and the inactive robot returns for the next task. The best few admissible pairs are retained for gap g .

3) *Pushing Strategy Evaluation:* For a current state $\mathbf{s}$ and candidate strategy $(\Omega_m, \mathbf{v}, \xi) \in \Xi_g$, a short-horizon rollout evaluates the transition in (4) under the selected mode. As shown in Fig. 5, a strategy is accepted if the robots reach the pre-contact poses, maintain stable contact and induce obstacle motion above threshold. The simulation returns:

$$(\mathbf{s}', \delta_T, \delta_J) \triangleq \text{EvalSim}(\mathbf{s}, (\Omega_m, \mathbf{v}, \xi)), \quad (13)$$

where $\mathbf{s}'$ contains the resulting robot and object configurations, $\delta_T > 0$ is duration and δ_J control effort. Joint rollout captures contact-induced interactions among all bodies. Optionally, quick-pass approximates a transition using a cached rollout or a mechanically feasible reference trajectory, subject to compatible relative configurations and physical parameters, reachability, and swept-clearance checks. The trajectory end-point and associated duration and effort estimates supply $\mathbf{s}'$,

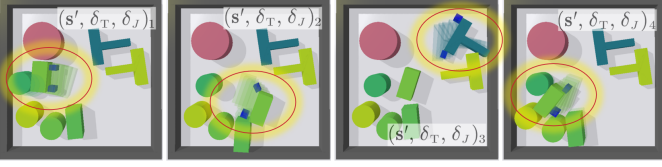
Evaluation of Pushing Modes via Parallel Simulation:

Fig. 6. Sampled push tasks are evaluated via parallel simulation, producing updated states for search expansion.

δ_T , and δ_J . This shortcut trades prediction accuracy for reduced online simulation effort.

D. Physics-Informed Hybrid Search

The physics-informed hybrid search (PIHS) selects blocking gaps and evaluates pushing strategies in parallel, see Fig. 6 and Alg. 1, using short-horizon simulation or the optional quick-pass approximation of Sec. III-C. The evaluated clearance cost of a partial schedule in (6) supplies the accumulated-cost term in the node priority of (14). Finite sampling gives no global optimality or completeness guarantee.

1) *Tree and Initialization*: Tree nodes $\nu = (s, \pi)$ contain the states in (4) and partial schedule in (5). $\text{Rank}(\nu)$ orders frontier gaps using (10), and $\chi(\nu)$ accumulates execution cost from the root. Alg. 1 initializes $\nu_0 = (s_0, \emptyset)$ and its queue.

2) *Node Selection*: At each iteration, line 4 of Alg. 1 selects the node with minimum priority:

$$f(\nu) \triangleq \chi(\nu) + \min_{g \in \text{Rank}(\nu)} \widehat{\text{Cost}}(g), \quad (14)$$

where the second term estimates the minimum remaining cost over unexplored gaps. Gap ranking orders local clearance targets within a node, whereas this priority compares partial schedules with different clearing histories. This heuristic need not be admissible, thus without A* optimality guarantee.

3) *Node Expansion with Parallel Simulation*: Alg. 1 expands the selected node $\nu = (s, \pi)$ in lines 7–17. For each sampled gap $g \in \text{Rank}(\nu)$, its predicted gap sequence $\Pi^*(g)$ in (10) is first derived. Then, fresh direction samples and mode optimization by (11) provide a candidate strategy set $\Xi_g \triangleq \{(\Omega_m, \mathbf{v}, \xi)\}$, including direct pushes of gap-side obstacles and recursive pushes of outer blockers. Thus, expansion is prioritized first by the gap ordering in $\text{Rank}(\nu)$, and then by the candidate ordering within Ξ_g . Each candidate is evaluated by (13), using quick-pass approximation or a rollout as described in Sec. III-C. A successful evaluation yields (s', δ_T, δ_J) , from which a child node $\nu' \triangleq (s', \pi')$ is created with $\pi' \triangleq \pi \cup \{\tau\}$, appending $\tau \triangleq (\Omega_m, \mathbf{v}, \xi)$ to π . The cumulative cost is updated by: $\chi(\nu') \triangleq \chi(\nu) + \widehat{C}(\nu, \tau)$, where $\widehat{C}(\nu, \tau) \triangleq \delta_T + \alpha \delta_J$ is the incremental cost. With *reinsertion* in line 17, the parent node is requeued for later expansion with new target and direction samples.

4) *Termination*: Alg. 1 terminates when a node admits a W -clear path by (8). The returned schedule satisfies nominal clearance and evaluated pushing transitions but need not minimize $\mathcal{J}_{\text{clr}}$ in (6). Vehicle traversal follows clearing.

Remark 2. Gap-lookahead reduces unproductive branches, MODE-CACHE warm-starts optimization, and optional quick-pass approximates transitions to reduce rollout effort. ■

Algorithm 1: Physics-Informed Hybrid Search

Input: $s_0, \alpha, \text{EvalSim}(\cdot), W$ -criterion by (8)
Output: Clearing schedule π^* , or failure

```

1  $\nu_0 \leftarrow (s_0, \emptyset), \chi(\nu_0) = 0;$ 
2 Init  $\mathcal{Q}$  by (14);
3 while  $\mathcal{Q} \neq \emptyset$  and budget remains do
4    $\nu = (s, \pi) \leftarrow \mathcal{Q}.\text{pop\_min}()$  by (14);
5   if  $s$  admits  $\mathcal{P}_V^W$  by (8) then
6     return  $\pi$ ;
7   foreach sampled  $g \in \text{Rank}(\nu)$  do
8     Compute  $\Pi^*(g)$ , generate  $\Xi_g$ ;
9     foreach  $(\Omega_m, \mathbf{v}, \xi) \in \Xi_g$  do
10       $\tau = (\Omega_m, \mathbf{v}, \xi)$  // Direct or recur.
11       $(s', \delta_T, \delta_J) \leftarrow \text{EvalSim}(s, \tau)$  by (13);
12      if success then
13         $\pi' \leftarrow \pi \cup \{\tau\};$ 
14         $\nu' \leftarrow (s', \pi');$ 
15         $\chi(\nu') \leftarrow \chi(\nu) + \delta_T + \alpha \delta_J;$ 
16        Add  $\nu'$  to  $\mathcal{Q}$ ;
17   Reinsert  $\nu$  into  $\mathcal{Q}$  for further sampling;
18 return failure;
```

E. Overall Analyses

1) *Execution and Online Adaptation*: The schedule $\pi = \{\tau_1, \dots, \tau_K\}$ is executed in a receding-horizon manner, with $\tau_k = (\Omega_{m_k}, \mathbf{v}_k, \xi_k)$. Robots reach the assigned contacts and follow $\mathbf{v}_k$ from the current state in (13). On hardware, the optimized forces certify wrench feasibility within actuation limits; they are not control setpoints. At each control interval, EvalSim predicts object motion tracked with motion-capture feedback. Between pushes, robot reassignment preserves contact order along the obstacle boundary [7] to reduce path crossings. After each push, the state and WCCG are updated until a W -clear path connects $\mathbf{x}_V^S$ and $\mathbf{x}_V^G$. Failed contacts, stalled pushing, or invalidated connectivity trigger replanning by Alg. 1 from the latest observed state.

2) *Computational Complexity*: Given M movable obstacles, the computational cost of one node expansion in Alg. 1 is dominated by candidate generation and physics validation. WCCG updating and gap ranking on the node's frontier loop $\mathcal{L}_\nu$ incur $\mathcal{O}(M \log M + |\mathcal{V}| + |\mathcal{E}_c| + |\mathcal{E}_b|)$ and $\mathcal{O}(|\Gamma_{\mathcal{L}_\nu}| \log |\Gamma_{\mathcal{L}_\nu}|)$ costs. The dominant cost is physics validation, which grows with the number of evaluated candidates; parallel workers reduce the rollout wall time. Robot-team size affects mode generation: each greedy sweep evaluates at most $N|\mathcal{A}|$ single-contact substitutions, where $|\mathcal{A}|$ bounds the reachable contact candidates considered per robot, rather than $|\mathcal{A}|^N$ joint tuples, and refinement is capped by a bounded evaluation budget; failures are handled by bounded backtracking with budget B and a reduced-team fallback over at most N one-robot-excluded subsets. Each candidate is checked by an LP over at most $2N$ force variables.

IV. EXPERIMENTS

PIHS is evaluated in simulation and hardware for connectivity, clearing performance, scalability, and robustness to execution-side physical mismatch. The algorithms use Python3 and PyBullet [22] on an Intel i9-14900KF with an RTX 4080. The accompanying video illustrates the simulation and hardware experiments.

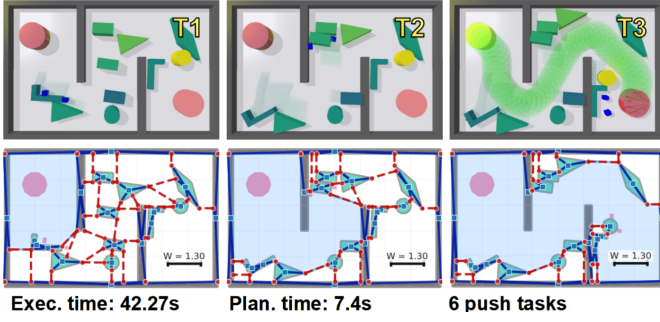
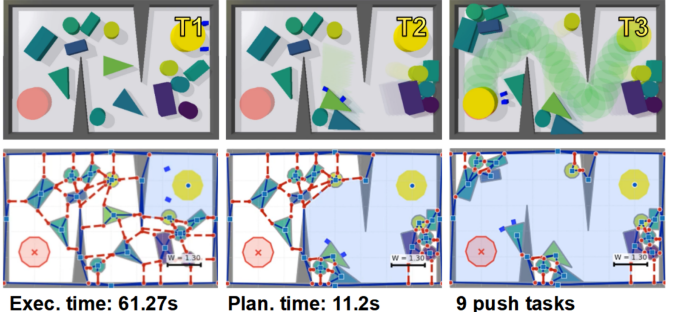
Scenario 1: 10 movable obstacles, 2 robots**Scenario 2: 14 movable obstacles, 2 robots**

Fig. 7. Execution and planning alignment in Scenario 1 (Left) and Scenario 2 (Right). **Top:** simulated snapshots during path clearing. **Bottom:** corresponding WCCG overlays with the start face in blue. The executed pushes progressively expand the reachable W -clearance region until the goal is connected.

TABLE I
WCCG CONNECTIVITY VALIDATION AND MEAN RUNTIME.

Type	Validation		Runtime (ms)		
	CC = FR	FP/FN [†]	CC	FR	WCCG
Convex	100/100	0/0	1.23	282.9	0.86
Non-convex	98/100	0/0	1.51	204.1	1.30
Mixed	98/100	0/0	1.33	242.4	1.05
Total	296/300	0/0	1.36	243.1	1.07

[†]WCCG errors relative to the continuous configuration-space reference (CC). All runtimes use the same 300-scene harness with the compiled WCCG backend.

TABLE II
PERFORMANCE COMPARISON AND ABLATIONS ON SCENARIO 1
OVER 40 RANDOMIZED TRIALS.^A

Method / Variant	Succ.(%)	PT (s)	ET (s)	#Sims	#Pushes
DFS-WCCG	25.0	>100.0	51.2	>500	8.0
SL-Push (offline)	62.5	0.06	44.1	0.0	7.3
SL-Push (sim)	75.0	18.2	64.6	20.0	10.2
Rec-NAMO	37.5	13.3	42.4	0.0	7.8
PushAround (ours)	97.5	10.3	28.6	121.5	6.0
w/o gap-lookahead	82.5	16.9	35.2	178.3	7.2
w/o mode-cache	92.5	17.1	33.4	164.6	6.5
w/o quick-pass	90.0	23.6	41.9	208.2	6.1
w/o reinsertion	85.0	12.8	31.1	151.7	7.8

^A Times are averaged over successful trials; failures count in the success rate only. For DFS-WCCG, PT and $\#Sims$ are not comparable and are given as lower bounds ($>$).

A. Numerical Simulations

1) *Setup and Configuration:* The simulation step is $\Delta t = 1/240$ s and the control period is $1/40$ s. Robots are disk or box pushers, and obstacle masses range from 5 to 15 kg in Scenarios 1–2 and from 20 to 30 kg in the generalization studies. The nominal 8×5 m workspace contains bottleneck-forming bar obstacles, with two robots and 10–15 movables under minimum-separation constraints. Unless stated otherwise, comparisons and ablations use Scenario 1 with ten movables and 40 trials; Scenario 2 has 14 movables and provides an additional cluttered case in Fig. 7. All methods share robot footprints, obstacle configurations, collision and reachability checks, and the execution simulator. They use eight workers, an 800 s planning cap, and an execution cap of 20,000 control steps, equivalent to 500 s. A trial succeeds when the vehicle reaches its goal through a W -clear path. Planning time PT and execution time ET average over successful trials unless stated otherwise; failures count only in success rates, while scalability timing curves cap failures at the budgets.

2) *Results:* Across the 40 Scenario 1 trials, PushAround achieves a 97.5% success rate, with mean planning and execution times of 10.3 s and 28.6 s, respectively, and 6.0



Fig. 8. Failure cases by baselines Rec-NAMO, SL-Push and DFS-WCCG.

pushes per successful trial, as shown in Table II. Fig. 7 shows representative runs in both scenarios. In Scenario 1 with 10 movable obstacles and two robots, PushAround generates a 6-push plan in 7.4 s and completes execution in 42.3 s. In Scenario 2 with 14 obstacles, it generates a 9-push plan in 11.2 s, with 61.3 s of execution. Planning effort is dominated by simulation-based validation: a successful trial uses 121.5 simulations on average for 6.0 executed pushes. The ablations indicate that mode-cache, quick-pass, gap-lookahead, and reinsertion mainly affect this validation load rather than plan length. Execution takes longer because each selected push is physically carried out and observed, whereas inter-push repositioning is handled by geometric reachability checks.

3) *WCCG Connectivity Validation:* Table I compares WCCG against two references on 300 random convex, non-convex, and mixed scenes. The continuous reference (CC) resolves collisions geometrically after inflating boundaries by $W/2$, with no grid discretization [23]; the fine-raster reference (FR) discretizes the workspace on a 0.01 m occupancy grid. CC and FR disagree in 4 of 300 cases, all near narrow contacts and rasterization-related, whereas WCCG matches CC on all 300 scenes without false positives or negatives. A query takes 1.07 ms, comprising 0.67 ms for querying and 0.40 ms for construction, versus 1.36 ms for CC and 243 ms for FR. It also supplies the gap graph for PIHS. Per-push WCCG rebuilding is negligible next to physics rollouts.

4) *Ablation Studies:* Four ablations remove individual mechanisms: **w/o gap-lookahead** omits downstream costs in gap ranking; **w/o mode-cache** disables cached warm starts; **w/o quick-pass** evaluates every candidate by rollout; and **w/o reinsertion** expands each node only once. Table II shows success drops exceeding ten percentage points without gap-lookahead or reinsertion. Removing quick-pass lowers success

TABLE III
SCALABILITY AND ROBUSTNESS.

Case	Trials	Succ.(%)	PT (s)	ET (s)
<i>Obstacle density</i> (PushAround, $N = 2$, nominal)				
$M = 15$	10	100	2.5 ± 0.9	37.1 ± 21.0
$M = 30$	10	100	11.7 ± 3.7	100.9 ± 34.7
$M = 45$	10	100	53.4 ± 24.5	272.7 ± 97.4
<i>Robot-team size</i> ($M = 30$, nominal; 10 trials per cell)				
PushAround, $N = 3$	10	100	30.0 ± 12.7	97.3 ± 42.1
PushAround, $N = 4$	10	100	29.2 ± 8.9	110.1 ± 33.1
SL-Push (sim), $N = 3$	10	80	30.0 ± 46.3	193.3 ± 111.3
SL-Push (sim), $N = 4$	10	80	23.4 ± 18.2	171.7 ± 61.2
DFS-WCCG, $N = 3$	10	100	142.6 ± 183.1	178.1 ± 72.1
DFS-WCCG, $N = 4$	10	90	101.9 ± 103.9	175.9 ± 113.6
<i>Physics mismatch</i> ($M = 30$, $N = 2$)				
Nominal	10	100	16.8 ± 3.4	104.7 ± 37.4
Mass $\times 0.6$ –1.4	40	100	16.8 ± 5.4	105.9 ± 44.6
Lateral fric. $\times 0.6$ –1.4	40	97.5	16.5 ± 5.7	114.0 ± 48.5
Spin fric. $\times 0.6$ –1.4	40	97.5	15.2 ± 3.6	100.1 ± 35.7
Force limit $\times 0.6$ –1.4	40	95.0	18.9 ± 10.9	115.9 ± 58.5
All perturbed	160	97.5	16.9 ± 7.0	108.9 ± 47.4

Each perturbed row scales one parameter; the others stay nominal. Times and budget are as in Table II.

to 90.0%; removing mode-cache increases simulations.

5) *Comparisons*: Four baseline variants run on Scenario 1 over 40 randomized trials under the same W -clearance criterion, scene distribution, and execution simulator. **DFS-WCCG** searches fixed object–direction pushes depth first and checks transitions by physics rollout. **SL-Push** clears blockers from a straight corridor by pushing them sideways; its offline variant plans geometric displacements, while its simulation-validated variant checks candidate pushes by rollout. **Rec-NAMO** recursively clears blocking objects using geometric planning without rollout validation during search. These variants represent search-based, local-pushing, and recursive NAMO strategies [3]–[5], [16]. Table II shows that PushAround reaches the highest success rate of 97.5%, at the lowest combined planning-and-execution time among successful trials by 38.9 s. DFS-WCCG exhibits rapid branching growth, offline SL-Push produces displacements that fail in execution, and Rec-NAMO often fails to connect the W -clear corridor in clutter as shown in Fig. 8.

6) *Scalability and Robustness Analysis*: Table III and Figs. 9–10 report generalization in an 8×8 m workspace. Density runs query a precomputed mode table; its preparation is excluded from PT . Mismatch and team-size runs optimize modes greedily. With $M = 30$ and $N = 2$, nominal PT is 11.7, 16.8, and 23.9 s, respectively. The team-size study uses a different realization of the same seeds.

(I) **Obstacle density**: with two robots and ten trials per density, PushAround maintains 100% success for $M = 15$ –45, planning in 2.5–53.4 s. At $M = 45$, none of the evaluated baselines succeeds, while the no-recursion ablation solves 7/10 trials. Across the clutter sweep, branching stays near six while expansions rise from 3.5 to 24.5. Median search effort is ten expansions and 48 visited snapshots, with about 60 simulations at 165 ms each versus 22 ms per mode.

(II) **Robot-team size**: at $M = 30$, PushAround solves all ten trials for each of $N = 2, 3$, and 4. The shared mode generator first tries the full team and may temporarily leave one robot inactive. Across $N = 2$ –4, search effort remains near ten expansions and 60–63 simulations; aggregate mode-generation time rises from 15.8 to 46.4 s.

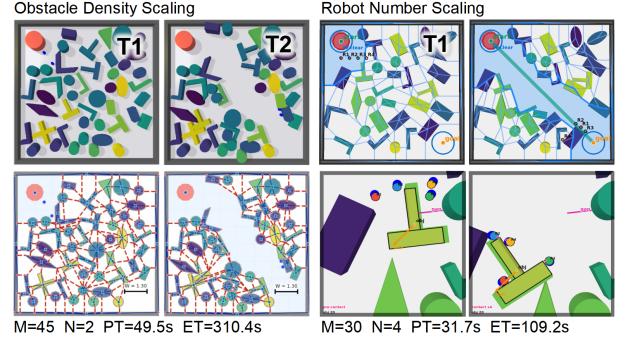


Fig. 9. Scalability of the clearing process. **Left**: density scaling ($M=45$, two robots): snapshots during clearing (**top**) and WCCG overlays with the start face in blue (**bottom**). **Right**: team scaling ($M=30$, four robots): initial and final states (**top**) and one push before contact and during execution (**bottom**).

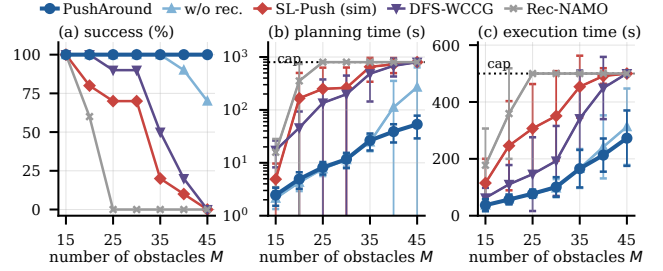


Fig. 10. Obstacle-density scaling (8×8 m, 10 seeds per density). Times are budget-capped (planning 800 s, execution 20,000 steps = 500 s; dotted).

(III) **Physics mismatch**: the planner retains nominal physics while execution varies one parameter family at a time. Nominal and perturbed conditions reuse ten base seeds for PushAround and five per baseline. Under $\pm 40\%$ perturbations to four parameter families, PushAround, SL-Push (sim), and DFS-WCCG achieve 97.5%, 32.5%, and 57.5% success, respectively. Their PT averages 18.4, 31.5, and 229.1 s, averaging successful-trial means over conditions with successes. Nominal success is 10/10 for PushAround and 3/5 per baseline. Comparisons reflect end-to-end performance, not normalized degradation; joint errors are untested.

B. Hardware Experiments

1) *System Description*: Experiments use a 5×6 m workspace of interlocking 0.6×0.6 m foam mats (Fig. 11), 2 UGVs and 6 cardboard boxes; robots and movables carry motion-capture markers streamed to PyBullet in real time. The UGVs are velocity-controlled, and the optimized contact forces are used only for feasibility evaluation, not for motor control. Motion-capture poses are compared with EvalSim predictions: position or orientation errors above 0.3 m or $\pi/4$ trigger replanning, as do the failures in Sec. III-E1.

2) *Results*: Two representative real-world runs are shown in Fig. 11. In hardware Scenario 1, PIHS considers five candidate gaps and plans a four-task schedule in 6.5 s using 12 node expansions and 49 visited snapshots at depth 6; the robots then execute it, first rotating the yellow L-shaped object to open a gap near an immovable obstacle and then separating the green and orange objects until a W -clear path is established at $t = 52$ s. In hardware Scenario 2, PushAround succeeds in 8/10 trials with an average of 5.2 pushes and 11.4 s of planning. In the illustrated run, the first direct push on the

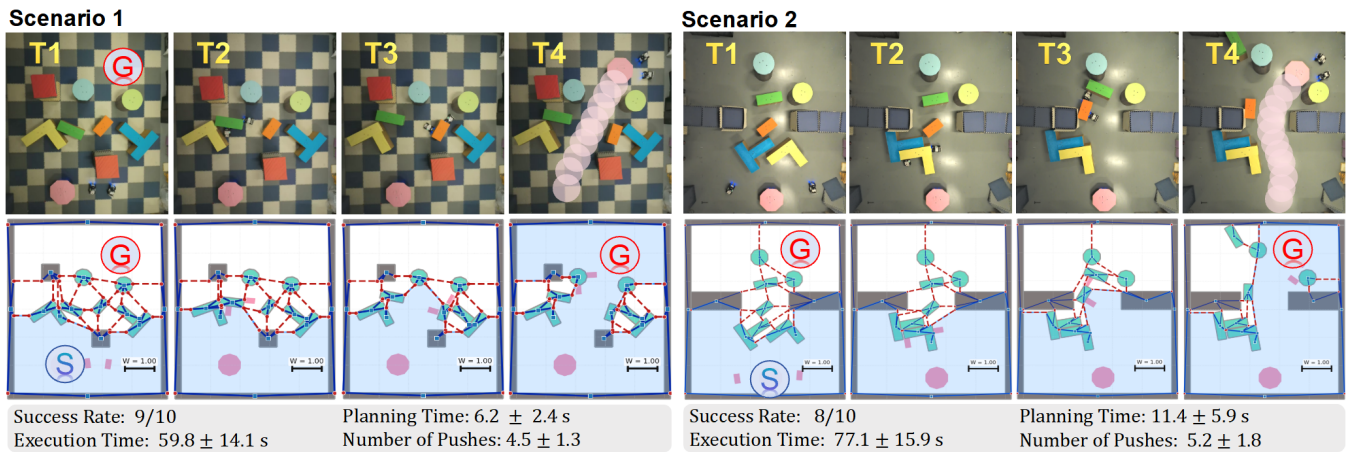


Fig. 11. Real-world pushing experiments with execution-planning alignment (over 20 runs, 2 scenarios). **Top**: two robots pushing obstacles in a 5×6 m workspace. **Bottom**: WCCG overlays with the start face in blue.

blue L-shaped object also moves the adjacent T-shaped block and opens the first gap. Coupled rigid-body rollout in EvalSim predicts this neighboring-object motion without a separately encoded action. The resulting snapshot updates the WCCG before execution continues. Execution takes 77.1 ± 15.9 s (mean $\pm$ standard deviation); online replanning handles deviations due to force limits, drift and wheel slip.

V. CONCLUSION

This work presents PushAround, a multi-robot framework that jointly plans obstacle selection, contacts and forces through a hybrid search. Geometric gap reasoning identifies where the corridor must widen, while pushing evaluation assesses the corresponding rearrangements under physical constraints. Simulation studies examine obstacle density, team size, and execution-side physics mismatch; hardware experiments demonstrate path clearing with state-feedback replanning. Future work targets obstacle-state estimation without external sensing or known masses.

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 *IEEE International Conference on Robotics and Automation (ICRA)*, 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 3062–3068.
- [5] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 533–539.
- [6] J. J. Weeda, S. Bakker, G. Chen, and J. Alonso-Mora, "Pushing through clutter with movability awareness of blocking obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 512–518.
- [7] Z. Tang, Y. Zhang, and M. Guo, "PushingBots: Collaborative pushing via neural accelerated combinatorial hybrid optimization," *IEEE Transactions on Robotics*, vol. 42, pp. 579–599, 2026.
- [8] 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.
- [9] K. M. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 1992, pp. 416–421.
- [10] 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.
- [11] Y. Wang and C. W. De Silva, "Multi-robot box-pushing: Single-agent Q-learning vs. team Q-learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 3694–3699.
- [12] Y. Feng, C. Hong, Y. Niu, S. Liu, Y. Yang, and D. Zhao, "Learning multi-agent loco-manipulation for long-horizon quadrupedal pushing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 14 441–14 448.
- [13] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5280–5286.
- [14] 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.
- [15] T. Xue, H. Girgin, T. S. Lembono, and S. Calinon, "Demonstration-guided optimal control for long-term non-prehensile planar manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 4999–5005.
- [16] D. M. Saxena and M. Likhachev, "Planning for complex non-prehensile manipulation among movable objects by interleaving multi-agent pathfinding and physics-based simulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 8141–8147.
- [17] K. Ren, G. Wang, A. S. Morgan, L. E. Kavraki, and K. Hang, "Object-centric kinodynamic planning for nonprehensile robot rearrangement manipulation," *IEEE Transactions on Robotics*, vol. 41, pp. 5761–5780, 2025.
- [18] R. Ni and A. H. Qureshi, "Progressive learning for physics-informed neural motion planning," in *Robotics: Science and Systems*, 2023.
- [19] Y. Liu, R. Ni, and A. H. Qureshi, "Physics-informed neural mapping and motion planning in unknown environments," *IEEE Transactions on Robotics*, vol. 41, pp. 2200–2212, 2025.
- [20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [21] 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.
- [22] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," <http://pybullet.org>, 2016–2019.
- [23] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, 1983.