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. Extensive simulations and hardware experiments demonstrate robust success, physical validity, and efficiency.**

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. As shown in Fig. 1, 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. Snapshots of collaborative path clearing in both simulation (**Top**) and hardware (**Bottom**). Two robots actively push movable obstacles to create a W–clear path for a large vehicle to traverse from start to goal, via hybrid search over the sequence of obstacles to push, and the associated pushing directions and modes.

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 embed contact simulation directly in the planning loop [16]. Other approaches incorporate dynamic models into planning for non-prehensile manipulation [17], [18] or apply learning

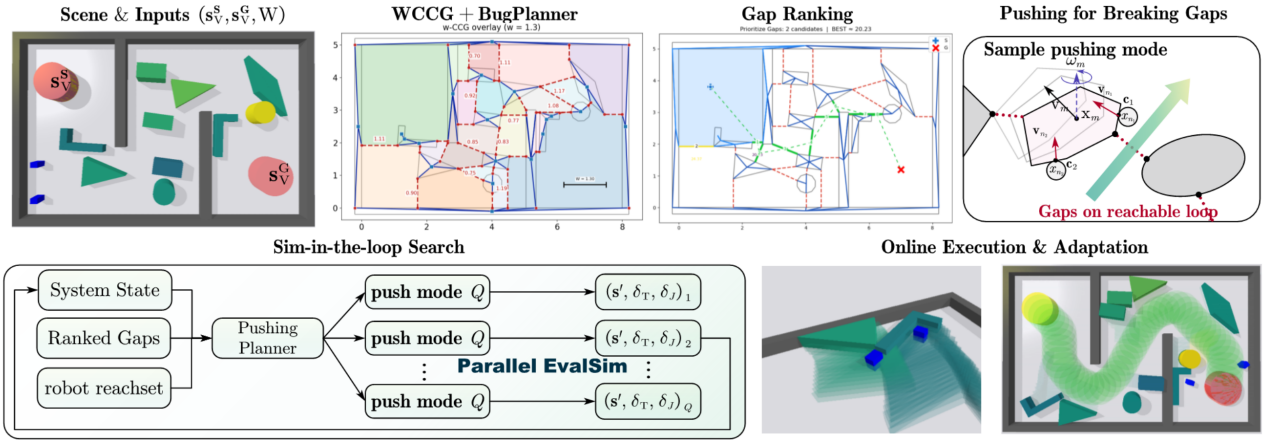

Fig. 2. Overview of the proposed PushAround framework. **Top:** Scene inputs, WCCG construction with BugPlanner, gap ranking, and sampling of pushing modes for breaking frontier gaps. **Bottom-left:** Sim-in-the-loop hyrid search where candidate push modes are validated in parallel. **Bottom-right:** Online execution and adaptation during path clearing.

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

As shown in Fig. 2, 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 $\mathbf{\Omega} \triangleq \{\Omega_1, \cdots, \Omega_M\} \subset \mathcal{W}$. Each movable obstacle Ω_m is characterized by its mass M_m, inertia I_m, frictional parameters (either identified or estimated), and state $\mathbf{s}_m(t) \triangleq (\mathbf{x}_m(t), \psi_m(t))$, where $\mathbf{x}_m(t) \in \mathbb{R}^2$ 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 $\mathcal{R} \triangleq \{R_1, \cdots, R_N\}$, operates as a cooperative unit. Each robot R_i is modeled as a rigid body with state $\mathbf{s}_{R_i}(t) \triangleq (\mathbf{x}_{R_i}(t), \psi_{R_i}(t))$, where $\mathbf{x}_{R_i}(t) \in \mathbb{R}^2$ 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:

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

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 $\mathbf{s}_\mathrm{V}^\mathrm{S}$ to a goal configuration $\mathbf{s}_\mathrm{V}^\mathrm{G}$. 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

$$\exists\, \mathcal{P}_\mathrm{V}^W \subset \widehat{\mathcal{W}}(t) : \ \mathbf{s}_\mathrm{V}^\mathrm{S} \rightsquigarrow \mathbf{s}_\mathrm{V}^\mathrm{G}, \ \mathtt{clr}(\mathcal{P}_\mathrm{V}^W) \geqslant W, \quad (2)$$

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

C. Collaborative Pushing Modes

Thus, the robots may actively reconfigure $\mathbf{\Omega}$ by pushing obstacles. The interaction with obstacle Ω_m is described by a pushing mode $\boldsymbol{\xi}_m \triangleq (\mathcal{C}_m, \mathbf{u}_m)$, 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

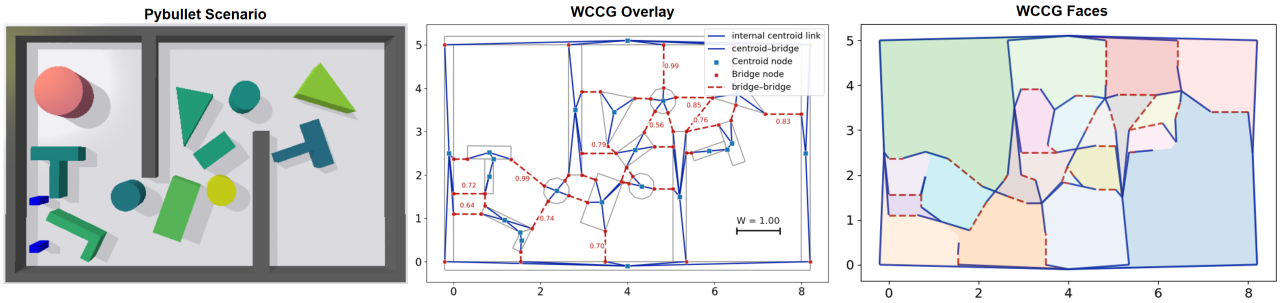

Fig. 3. Illustration of the W–Clearance Connectivity Graph (WCCG). **Left:** Cluttered PyBullet scenario with the immovable walls and movable objects; **Middle:** WCCG overlay with the centroid nodes (blue squares), bridge nodes (red circles), centroid–bridge edges (blue), and bridge–bridge edges (red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

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:

$$\mathbf{S}(t^+) \triangleq \Phi\big(\mathbf{S}(t),\, m,\, \boldsymbol{\xi}_m\big), \tag{3}$$

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.

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:

$$\pi \triangleq \big\{(m_k, \boldsymbol{\xi}_k, \Delta t_k)\big\}_{k=1}^{K}, \tag{4}$$

where $\Omega_{m_k} \in \boldsymbol{\Omega}$ specifies the movable obstacle to manipulate, $\boldsymbol{\xi}_k \in \Xi_{m_k}$ 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.,

$$\min_{\pi} \Big\{ T + \alpha \sum_{k=1}^{K} J(m_k,\, \boldsymbol{\xi}_k,\, \mathbf{S}(\tau_k)) \Big\}, \tag{5}$$

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], [19]. ■

III. Proposed Solution

The PushAround framework enables collaborative multi-robot pushing with physical feasibility. As shown in Fig. 2, it first constructs a W-Clearance Connectivity Graph as described in Sec. III-A, which captures workspace connectivity and identifies blocking gaps. A gap-ranking strategy in Sec. III-B then prioritizes obstacle-clearing actions, which are evaluated through a simulation-in-the-loop search in Sec. III-C. This search expands a configuration-space tree and validates pushes by parallel simulation. The overall flow and possible extensions are 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 $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ and the goal configuration $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$. To address this question, a *W–Clearance Connectivity Graph* (WCCG) is introduced. As shown in Fig. 3, 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 $\Omega_m \in \boldsymbol{\Omega}$ 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:

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

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 [20], starts from the vehicle start $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$, casts a ray toward the goal $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$, 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

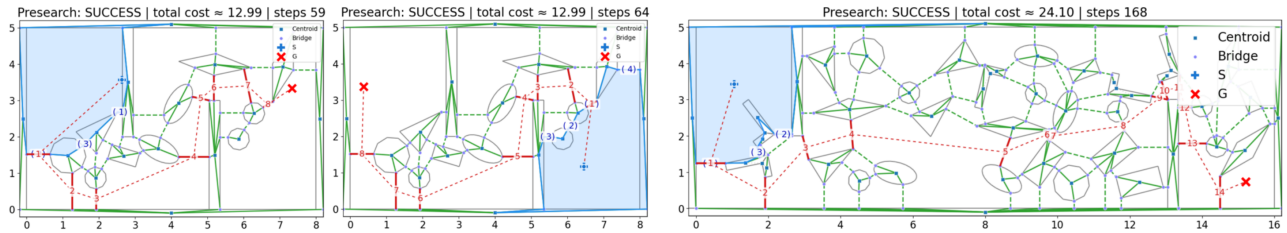

Fig. 4. Illustration for the ranking of potential gaps. Each panel shows the WCCG with the current face (light blue). The algorithm (i) extracts the frontier loop from BugPlanner, (ii) enumerates candidate bridge–bridge gaps on the loop, (iii) assigns local first-hop ranks (blue numbers), and (iv) simulates a short presearch to predict the full gap-crossing sequence (red numbers). **Top**: start and goal swapped in the same map give symmetric sequences with cost 12.99; **Right**: a larger map with about 30 obstacles produces a 14-gap sequence with cost 24.10.

produces a skeleton Σ, which is an ordered sequence of centroid and bridge nodes that certifies $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ and $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$ lie in the same connected face. Let the W–clear free space be $\mathcal{F}_W \triangleq \mathbb{R}^2 \backslash (\mathcal{O} \oplus \mathbb{B}_{W/2})$, 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 $\mathcal{P}_{\mathrm{V}}^W$ is defined as:

$$\exists\, \mathcal{P}_{\mathrm{V}}^W \subset \mathcal{F}_W :\ \mathbf{s}_{\mathrm{V}}^{\mathrm{S}} \rightsquigarrow \mathbf{s}_{\mathrm{V}}^{\mathrm{G}} \Longleftrightarrow \Big\{ \mathbf{s}_{\mathrm{V}}^{\mathrm{S}}, \mathbf{s}_{\mathrm{V}}^{\mathrm{G}} \text{ belong to the same face of } \mathcal{G}_W;\ \mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}),\ \mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}) \subset \mathcal{F}_W \Big\}, \tag{7}$$

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 $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ and $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$ lie in the same connected face. Given as input the skeleton Σ, together with $\mathcal{G}_W$, the clearance W, and the endpoint disks $\mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{S}})$ and $\mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{G}})$, the output is an explicit W–clear path $\mathcal{P}_{\mathrm{V}}^W$. 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 $\mathcal{P}_{\mathrm{V}}^W$ 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 $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$ from $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ through a W–clear path $\mathcal{P}_{\mathrm{V}}^W$. In this case, the planner must prioritize *blocking gaps* on the reachable frontier of $\mathcal{G}_W$. As shown in Fig. 4, 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 $(\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}, \mathbf{s}_{\mathrm{V}}^{\mathrm{G}}, W)$, 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 $\Gamma_{\mathcal{L}} \triangleq \{g_1, \cdots, g_K\}$, 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:

$$\mathsf{C}\big(g \mid \mathcal{L}, \mathbf{s}_{\mathcal{R}}\big) = \lambda_{\mathrm{t}}\, \mathsf{C}_{\mathrm{t}}(\mathbf{s}_{\mathcal{R}}, \mathbf{o}_g) + \lambda_{\mathrm{p}}\, \mathsf{C}_{\mathrm{p}}(g), \tag{8}$$

where $\lambda_{\mathrm{t}}, \lambda_{\mathrm{p}} > 0$ are weighting factors for the transition and pushing costs, respectively; function $\mathsf{C}_{\mathrm{t}}(\cdot)$ denotes the collision-free distance between two points; and the function $\mathsf{C}_{\mathrm{p}}(\cdot)$ measures the widening effort, which increases when the gap is narrower than W, or when the adjacent 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., $h(g) = \eta \left\| \mathbf{o}(g) - \mathbf{s}_{\mathrm{V}}^{\mathrm{G}} \right\|$, 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:

$$\widehat{\mathsf{Cost}}(g) \triangleq \mathsf{C}\big(g \mid \mathcal{L}, \mathbf{s}_{\mathcal{R}}\big) + \sum_{g' \in \Pi^\star(g)} \big\{ \mathsf{C}(g' \mid \cdot) + h(g') \big\}, \tag{9}$$

where $\Pi^\star(g)$ is the sequence of subsequent gaps discovered after virtually crossing g; the dots indicate updated inputs along that rollout; and the scalar score $\widehat{\mathsf{Cost}}(g) > 0$ for each candidate gap. Consequently, the final output of this module is the ranked list of candidate gaps, i.e., $\mathsf{Rank}(\Gamma_{\mathcal{L}}) \triangleq \mathbf{argsort}_{g \in \Gamma_{\mathcal{L}}} \widehat{\mathsf{Cost}}(g)$, 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

As shown in Fig. 5, the hybrid search couples high-level decisions about blocking gaps with low-level feasibility of

Algorithm 1: Physics-Informed Hybrid Search

Input: $\mathbf{s}_0$, α, $\mathsf{EvalSim}(\cdot)$, W–criterion by (7)
Output: $\pi^\star$
/* Initialization */
$\nu_0 \leftarrow (\mathbf{s}_0, \varnothing)$, $\chi(\nu_0) = 0$;
Init $\mathcal{Q}$ by (10);
while *not terminated* **do**
  /* Selection */
  $\nu \leftarrow \mathcal{Q}.\mathtt{pop\_min}()$ by (10);
  /* Parallel Expansion */
  foreach $g \in \mathsf{Rank}(\nu)$ **do**
    foreach $(\mathbf{v}, \boldsymbol{\xi}) \in \Xi_g$ **do**
      $\tau = (g, \mathbf{v}, \boldsymbol{\xi})$;
      $(\mathbf{s}', \delta_T, \delta_J) \leftarrow \mathsf{EvalSim}(\nu, \tau)$ by (11);
      if *success* **then**
        $\nu' \leftarrow (\mathbf{s}', \pi \cup \{\tau\})$;
        $\chi(\nu') \leftarrow \chi(\nu) + \delta_{\mathrm{T}} + \alpha\, \delta_{\mathrm{J}}$;
        $\mathcal{Q}.\mathtt{push}(\nu')$;
  /* Termination Check */
  if $\mathbf{s}'$ *admits* $\mathcal{P}_V^W$ *by* (7) **then**
    $\pi^\star \leftarrow \pi'$, **break**;
return $\pi^\star$;

multi-robot pushing. Unlike purely geometric planners, this procedure simultaneously determines a sequence of gaps to clear and physically-feasible pushing actions, including directions, contact modes, and forces. Parallel physics simulation is embedded so that candidate push strategies can be evaluated simultaneously at each expansion, and the resulting successor states are returned to the search. The algorithmic details are summarized in Alg. 1 and explained below.

1) Tree and Initialization: The search tree $\mathcal{T}$ is composed of nodes $\nu \triangleq (\mathbf{s}, \pi)$, where $\mathbf{s}$ 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: $\mathsf{Rank}(\nu)$ 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 $\nu_0 \triangleq (\mathbf{s}_0, \varnothing)$, where $\mathbf{S}^{\mathrm{S}}$ corresponds to the initial system state $\mathbf{s}_0$. At initialization, $\chi(\nu_0) = 0$, and $\mathsf{Rank}(\nu_0)$ is constructed by first computing the frontier loop $\mathcal{L}_\nu$ associated with $\mathbf{s}_0$, 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:

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

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

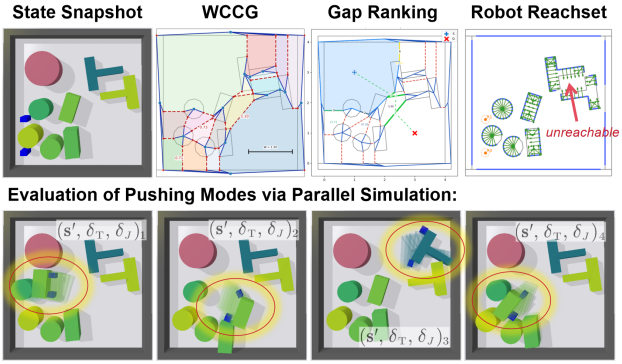

Fig. 5. **Top:** Core components of the proposed physics-informed hybrid search; **Bottom:** Sampled push tasks are evaluated via parallel simulation, producing updated states for search expansion.

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 $\mathsf{Rank}(\nu)$ is evaluated in parallel by simulation. Each pushing strategy is defined as $\tau \triangleq (g, \mathbf{v}, \boldsymbol{\xi})$, where $g \in \boldsymbol{\Omega}$ 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 $\boldsymbol{\xi} \triangleq (\mathcal{C}_m, \mathbf{u}_m) \in (\partial\Omega_m)^N \times \mathbb{R}^{2N}$ 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:

$$(\mathbf{s}',\, \delta_{\mathrm{T}},\, \delta_{\mathrm{J}}) \triangleq \mathsf{EvalSim}\big(\nu,\, (g, \mathbf{v}, \boldsymbol{\xi})\big), \tag{11}$$

where $\mathbf{s}'$ is the successor state; $\delta_{\mathrm{T}} > 0$ is the time cost; and δ_{J} is the control effort as in (5). Thus, a successful evaluation produces a child $\nu' \triangleq (\mathbf{s}', \pi')$, where π' is obtained by appending π with τ. The incremental realized cost of τ is $\widehat{\mathsf{C}}(\nu, \tau) \triangleq \delta_{\mathrm{T}} + \alpha\, \delta_{\mathrm{J}}$, with $\alpha > 0$ the same trade-off parameter as in (5). The cumulative cost is then updated by $\chi(\nu') \triangleq \chi(\nu) + \widehat{\mathsf{C}}(\nu, \tau)$. 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 $\mathbf{s}$ for which a W–clear path $\mathcal{P}_{\mathrm{V}}^W$ exists from $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ to $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$, 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 $(\mathbf{v}, \boldsymbol{\xi})$ 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. ■

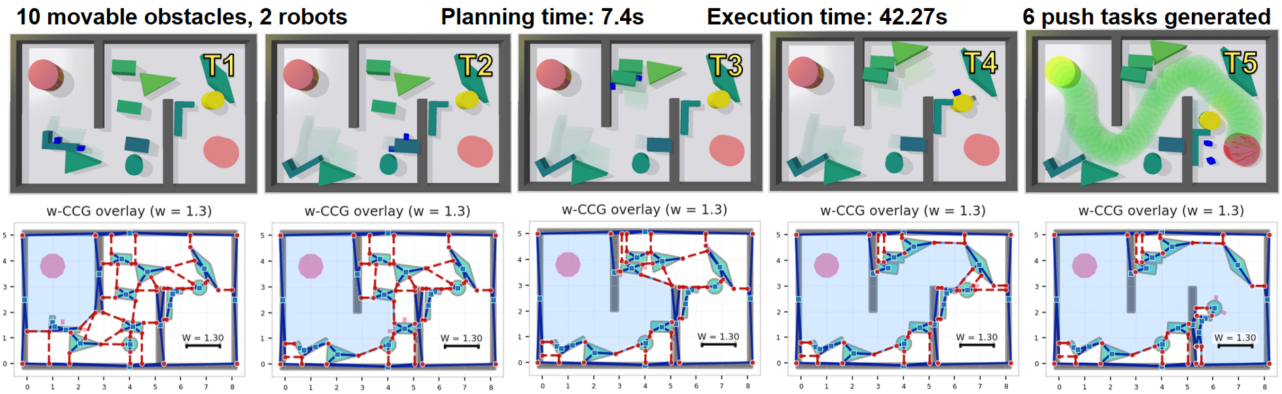

Fig. 6. Execution–planning alignment in the main scenario. **Top:** 5 PyBullet snapshots as 2 robots push a cylindrical object. **Bottom:** corresponding WCCG overlays with the start face in blue. The task moves the object from $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}} = (1, 4)$ to $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}} = (7, 2)$. In the final state, the start face contains G, matching the executed trajectory and confirming success.

D. Overall Analyses

1) Execution and Online Adaptation: The hybrid search outputs a pushing schedule $\pi = \{\tau_1, \cdots, \tau_K\}$, 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 $\boldsymbol{\xi}_k$. Execution proceeds sequentially under a hybrid controller alternating between transition and pushing.

During transition, robots plan collision-free paths in the instantaneous freespace $\widehat{\mathcal{W}}(t)$ to reach designated contact points. If two paths are imminently head-on, the corresponding contacts are swapped to avoid deadlock. During pushing, the velocity $\mathbf{v}_k$ is integrated to generate a short reference trajectory, which is mapped to per-robot contact states using $\boldsymbol{\xi}_k$. Each robot R_n applies proportional velocity control: $\mathbf{v}_n = K_p(\hat{\mathbf{p}}_n^{\mathrm{c}} - \mathbf{p}_n^{\mathrm{c}})$, $\omega_n = K_r(\hat{\psi}_n^{\mathrm{c}} - \psi_n^{\mathrm{c}})$, where $(\hat{\mathbf{p}}_n^{\mathrm{c}}, \hat{\psi}_n^{\mathrm{c}})$ are reference states, $(\mathbf{p}_n^{\mathrm{c}}, \psi_n^{\mathrm{c}})$ are measured states, and $K_p, K_r > 0$ are control gains. Small offsets are adapted online to counter yaw drift and maintain stable contact.

Execution is monitored by timeouts during transition and early-stop tests during pushing. If a push succeeds, execution advances to τ_{k+1}; otherwise, control is returned to the planner. At a lower rate, the WCCG is rebuilt and execution terminates once a W–clear path $\mathcal{P}_{\mathrm{V}}^{W}$ connects $\mathbf{s}_{\mathrm{V}}^{\mathrm{S}}$ and $\mathbf{s}_{\mathrm{V}}^{\mathrm{G}}$.

Remark 5 (Endpoint Booster)**.** The W–CCG criterion may be conservative near endpoints. If connectivity holds but the disks $\mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{S}})$ or $\mathbb{B}_{W/2}(\mathbf{s}_{\mathrm{V}}^{\mathrm{G}})$ intersect obstacles, a small set of auxiliary pushes is applied to clear the disks and enable execution. ■

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 $|\boldsymbol{\Omega}|$, while gap ranking by presearch in Sec. III-B is lightweight as it is limited to a small subset of frontier gaps. At expansion, only a fraction of the strategies in $\mathsf{Rank}(\nu)$ survive the geometric quick-pass and early-stop checks, which shortens the horizon of the simulator calls defined by $\mathsf{EvalSim}(\cdot)$. 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 $\mathsf{Rank}(\nu)$ across iterations, avoiding redundant restarts and keeping the priority queue focused on promising frontiers.

3) Generalization: The framework admits several extensions: (I) *Heterogeneous teams.* Per-robot gain tuning and robot-specific mode priors enable cooperation among diverse robots; (II) *Concurrent pushing.* Multi-object mode generation and conflict-aware transition planning allow multiple obstacles to be pushed simultaneously; (III) *Dynamic environments.* Periodic W–connectivity checks and on-demand re-planning handle disturbances such as object drift or unmodeled contacts during execution.

IV. Experiments

We evaluate the proposed physics-informed hybrid search(PIHS) in cluttered environments containing both movable and immovable obstacles. All components—W-CCG presearch, frontier extraction, and the *ModeTable* prior—are integrated as described in Sec. III. The implementation is in `Python 3`; simulations run in `PyBullet` [21] on a laptop with an Intel Core i7–1280P CPU. Videos and logs are provided in the supplementary material.

A. Numerical Simulation

1) Setup and Configuration: As shown in Fig. 6, the simulation has a step of $\Delta t = 1/240\,\mathrm{s}$ and a control period of $1/40\,\mathrm{s}$. Robots are modeled as disk or box pushers with radii consistent with the W-clearance definition. Movable obstacles have masses uniformly sampled from $[5, 15]\,\mathrm{kg}$, while immovables are modeled with zero mass. A trial is successful when a W-clear path exists from start to goal and the target reaches its goal disc. The nominal scenario consists of an $8\times5\,\mathrm{m}$ bounded workspace with two bar obstacles forming bottlenecks and a mixed pool of curved and polygonal shapes including rings, ellipses, X/T/L/diamond, arrow-like, rectangles, and cylinders. Two robots start in the lower-left and the goal lies on the right, with 10–15 movables placed randomly under a minimum separation. The per-task simulation horizon is 80 steps, and up to 64 candidate push tasks are generated per expansion. The node priority follows (10) with heuristic factor 10, and gap sampling follows

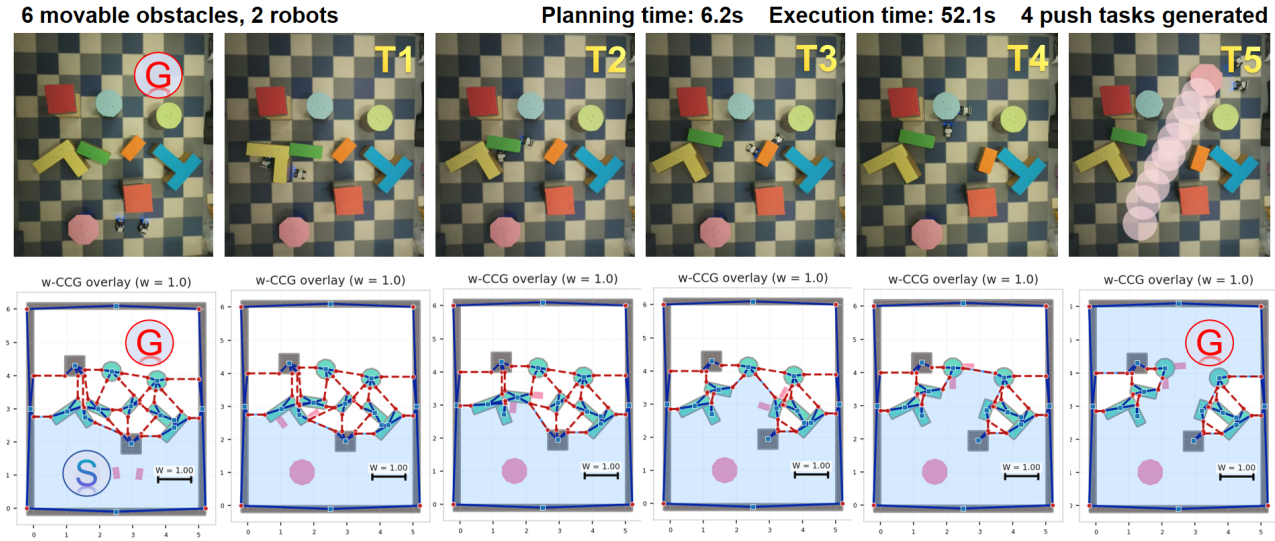

Fig. 7. Real-world pushing experiment with execution–planning alignment. **Top:** snapshots of two robots pushing obstacles in a $5\times6\,\mathrm{m}$ workspace with movable objects and fixed boundaries. **Bottom:** WCCG overlays with the start face in blue. Planning uses clearance $W=1.0$, start $\mathbf{s}_V^S=(1.65, 1.01)$, and goal $\mathbf{s}_V^G=(3.4, 4.8)$. The evolving graph remains consistent with observed motions, confirming that planned corridors match execution.

TABLE I

PERFORMANCE COMPARISON AND ABLATIONS.

Method / Variant	Succ.(%)	PT (s)	ET(s)	#Sims	#Pushes
DFS-WCCG	25.0	>100.0	51.2	>500	8.0
SL-Push (off-line)	62.5	0.06	44.1	0.0	7.0
SL-Push (sim)	75.0	18.2	64.6	20.0	10.0
Rec-NAMO	37.5	13.3	42.4	0.0	7.0
PushAround (ours)	**92.5**	**10.3**	**28.6**	**121.5**	**6.0**
w/o presearch	85.0	16.9	35.2	178	7.0
w/o ModeTable	80.0	15.1	33.4	164	6.0
w/o quick-pass	90.0	23.6	41.9	208	6.0
w/o reinsertion	82.5	12.8	31.1	151	7.0

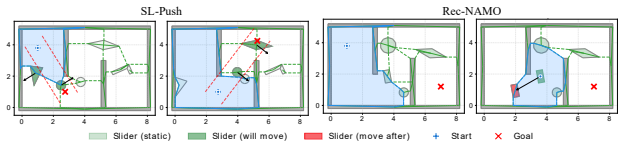

Fig. 8. **Left:** SL-Push, which determines pushing directions for movable obstacles along a W-width straight corridor between sub-start and sub-goal. **Right:** Rec-NAMO, which incrementally constructs path segments but often fails to establish a complete W-clear path from start to goal.

a softmax distribution with temperature 0.05. ModeTable is enabled and automatically updated when missing. A quick-pass geometric screen skips physics if a reference rollout already clears a gap; otherwise a short-horizon simulation with early stop is applied. To prevent premature termination, a reinsertion rule preserves high-value nodes that cannot be expanded immediately and revisits them later.

2) Comparisons and Ablation Studies: In total 8 planners are evaluated under the same W-clearance criterion and contact models. External baselines are: **DFS-WCCG**, a simulation-in-the-loop depth-first search guided by the W-CCG with four fixed axis-aligned pushes per movable, which scales poorly due to exponential branching; **SL-Push (off-fline)**, which clears blockers by minimal displacements along a straight or waypointed route without physics, planning quickly but yielding unrealizable executions; **SL-Push (sim)**, which validates the same route with physics, improving feasibility but requiring many simulations and long action sequences; and **Rec-NAMO**, which combines Dijkstra routing with local push decomposition and edge pruning, producing faster plans than simulation-heavy methods but often failing to establish a connected W-clear corridor in dense clutter. Internal ablations remove key modules of **PushAround**: **w/o presearch**, which disables frontier extraction and gap ranking; **w/o ModeTable**, which discards structured contact-mode priors; **w/o quick-pass**, which forces full-horizon rollouts; and **w/o reinsertion**, which prevents revisiting promising but previously failed nodes.

Table I shows that our method achieves the highest success rate of 92.5% with the lowest combined planning and execution time. DFS-WCCG reaches only 25% success due to exponential branching. As detailed in Fig. 8, SL-Push offline is extremely fast but produces physically invalid solutions, while SL-Push sim improves feasibility at heavy computational cost. Rec-NAMO is faster but often fails to produce complete solutions. The ablations reveal that presearch and ModeTable are critical, with success drops of 7.5% and 12.5% accompanied by increased simulations. Quick-pass reduces runtime overhead by 86 additional simulations and $13.3\,\mathrm{s}$ in planning with little effect on success, while reinsertion sustains robustness by recovering from local dead ends, as success drops by 10.0% when removed. Overall, the comparisons confirm that frontier-based ranking, structured contact priors, and selective simulation are essential for scalability and feasibility, and that deferred reinsertion further strengthens reliability.

3) Scalability Analyses: Scalability is examined in an $8\times5\,\mathrm{m}$ workspace containing 30 movable objects. Although the combinatorial complexity increases, the WCCG presearch and the ModeTable guidance maintain a small branching factor. As shown in Fig. 9, the run completes with 10 node expansions out of 115 visited nodes, 312 short simulation calls, and 7 executed push tasks, which yields

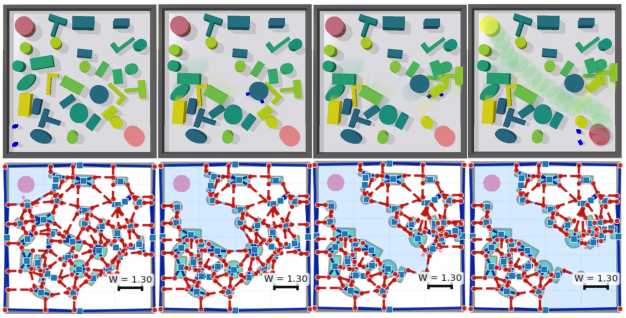

Fig. 9. Scalability in dense clutter with **30** movable objects. **Top:** snapshots during the path clearing of 24.5 s. **Bottom:** WCCG overlays with the active face in blue and ranked frontier gaps.

a connected W-clear corridor. The planning time is 21.2 s and execution 24.5 s, after which all robots reach the goal. These results confirm that the method sustains efficiency and solution quality as scene density increases significantly.

B. Hardware Experiments

1) System description: Experiments are conducted in a $5\,\mathrm{m} \times 6\,\mathrm{m}$ workspace assembled from interlocking $0.6\,\mathrm{m} \times 0.6\,\mathrm{m}$ foam mats, as illustrated in Fig. 7. Each trial employs 2 UGVs and 6 movable obstacles represented by cardboard boxes covered with colored paper for visual distinction. Both robots and movables are equipped with three to four motion-capture markers. A motion-capture system streams global poses to `PyBullet` in real time, enabling policy execution and visualization. Movables are randomly placed at the beginning of each trial to generate diverse clutter.

2) Results: A representative real-world run is shown in Fig. 1 and 7. The WCCG presearch completes within 10 s, during which 5 **gaps** are sequentially cleared through 64 node expansions and a maximum search depth of three. In the first task at 20 s, a yellow T-shaped object is rotated counterclockwise to expose a reachable contact on an adjacent green rectangle. In the second task at 55 s, the green rectangle is pushed to its target. The third and fourth tasks manipulate an orange rectangle and a blue cylinder at 71 s and 77 s, respectively, to open the gaps. By the 5-th task, all gaps in the WCCG are cleared, enabling a collision-free traverse from start to goal. Execution time varies due to occasional drifting or slippage of the Mecanum wheels near 45°, which lengthens certain transitions and introduces small deviations from simulation. Online replanning compensates for these effects. The sequence of pushing actions and contact modes confirms that the planned interactions are physically realizable on hardware.

V. Conclusion

This work presents a multi-robot framework PushAround for path clearing in unstructured environments using a hybrid search algorithm that jointly plans obstacles, contact points, and forces with efficiency. The method demonstrates adaptability to dynamic scenarios. Future directions include estimating obstacle states without external sensing, addressing uncertain masses, and managing limited robot visibility to enhance robustness in real-world 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," in *Proc. Robot., Sci. Syst.*, 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] T. Xue, H. Girgin, T. S. Lembono, and S. Calinon, "Guided optimal control for long-term non-prehensile planar manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 4999–5005.

[18] D. M. Saxena and M. Likhachev, "Planning for complex non-prehensile manipulation among movable objects by interleaving multi-agent pathfinding and physics-based simulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8141–8147.

[19] 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.

[20] 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.

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