Pass the Parcel: A Relay-Based Nonprehensile Approach to Manipulate Cargo with Multi-Agent Systems

Hui Zhi, *Student Member, IEEE*, Filippo Bertoncelli, Angelo Ferrando, Chenguang Yang, *Fellow, IEEE*, Lorenzo Sabattini, *Senior Member, IEEE* and David Navarro-Alarcon, *Senior Member, IEEE*

***Abstract*—This paper presents a relay-based strategy for multi-robot object transportation in constrained environments, where an object is passed sequentially between agents. To enable efficient and stable physical handovers, we employ nonprehensile pushing, a method that introduces challenges due to friction constraints. We propose a novel framework that jointly optimizes the locations of relay points and the connecting push trajectories. Our trajectory optimization explicitly incorporates pushing kinematics and handover pose constraints to ensure a smooth switching of pushing surfaces at relay points. Additionally, handover timing is coordinated to balance the workload among robots and achieve continuous, chain-like transportation. The effectiveness of the proposed paradigm is validated through extensive real-world experiments, demonstrating its practical viability.**

***Index Terms*—Path planning; Nonprehensile manipulation; Mobile robots; Relay tasks.**

I. Introduction

MULTI-ROBOT systems have demonstrated significant potential across a diverse range of fields, including search and rescue [1], environmental monitoring [2], and warehouse logistics [3]. Foundational research areas like multi-robot formation control [4] and task allocation [5] have been investigated extensively and are now well-established in the literature.

Building upon these principles, a burgeoning area of research has emerged that extends multi-robot systems to complex manipulation tasks, with cooperative object transportation being a prime example [6]. Research in this field predominantly addresses two canonical problems: (i) the cooperative manipulation of a single, large object whose size or mass precludes handling by an individual robot [7], and (ii) the coordination of multiple, independent transportation tasks within constrained environments, a challenge often framed as a multi-agent path finding (MAPF) problem [8].

Distinct from these paradigms, this paper explores a third approach: relay-based transportation, where a single object is passed sequentially through a chain of robots, as shown in Fig. 1. This model, which breaks a complex, long-distance task into a series of simpler sub-tasks, has been proven effective in domains like truckload shipping [9] and wireless communications [10]. However, its application in physical robotics is constrained by the difficulty of the handover itself. For simple agents like differential drive robots, reliable object transfer is a critical bottleneck.

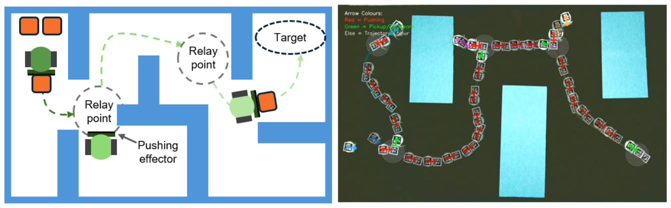

Fig. 1. Conceptual overview of the proposed multi-agent relay system. A parcel is passed sequentially between robots at designated RPs. The insets show snapshots from a real-world experiment.

To enable this physical handover, we turn to nonprehensile pushing manipulation [11]. Unlike caging, which often requires fixed formations [12], or grasping, which necessitates specialized hardware like robotic arms [13] or cable-suspended grippers [14], pushing is a flexible strategy that does not require complex mechanisms to physically attach to the object. We leverage the fact that a typical parcel offers multiple pushing faces [15], allowing a receiving robot to engage the object on a free surface as the delivering robot disengages. This facilitates a smooth, collision-free handover, making the powerful relay model viable for teams of simple robots. While stable pushing introduces challenges in trajectory planning due to contact dynamics [16] [17], it provides the foundational mechanism for our relay framework.

The classic formulation for relay-based systems is the Relay Network Design Problem (RNDP), which typically seeks to optimize the location of relay points (RPs), the assignment of nodes, and the transfer routes, often via mixed-integer programming [18]. While conventional applications assume simple, predefined paths, for a robotic system employing pushing manipulation, the route itself becomes a primary challenge. A path is not merely a geometric line but a dynamically feasible trajectory governed by complex kinematic and frictional constraints. Furthermore, the problem is compounded because a relay point is not just a location but a full pose; the object's final orientation determines which surfaces are available for the subsequent robot, defining the feasibility of the handover itself. This elevates routing from a simple connection task to a

H. Zhi and D. Navarro-Alarcon are with the Department of Mechanical Engineering, The Hong Kong Polytechnic University (PolyU), Kowloon, Hong Kong. hui1225.zhi@connect.polyu.hk, dnavar@polyu.edu.hk

F. Bertoncelli is with Autonomous Robotics Research Center, Technology Innovation Institute, Abu Dhabi, UAE. filippo.bertoncelli@tii.ae

A. Ferrando and L. Sabattini are with the University of Modena and Reggio Emilia, Italy. angelo.ferrando@unimore.it, lorenzo.sabattini@unimore.it

Chenguang Yang is with Department of Computer Science, University of Liverpool, Liverpool, United Kingdom. cyang@ieee.org

core optimization component, deeply intertwined with the full pose of the RPs.

Traditionally, this routing problem is solved in two stages: first, a collision-free geometric path is found using search-based (e.g., A* [19]) or sampling-based (e.g., RRT* [20]) planners; second, this path is smoothed into a time-parameterized trajectory using methods like polynomial curves [21] or B-splines [22]. These methods fail to explicitly enforce the continuous curvature constraints essential for stable pushing. While alternatives like Dubins' paths [23] respect curvature, their rigidity offers limited flexibility in cluttered environments and they do not consider the full pose of the object at the handover point. To address this, our framework integrates the complete pushing kinematics—including curvature limits and the final handover pose—directly into the trajectory optimization. This ensures that every generated path is not only dynamically feasible but also strategically aligned for a successful robot-to-robot transfer.

Our main contributions in this work are as follows:

- A novel framework for relay-based transport that uses nonprehensile pushing to physically execute handovers between simple robots.
- A constraint-aware trajectory optimization method generating dynamically feasible pushing paths by explicitly considering curvature limits and the final handover pose.
- Real-world validation of the complete framework on a physical multi-robot system across varied environments.

II. Problem Statement and Dynamic Modeling

This paper investigates the transport of polyhedral parcels through a narrow, cluttered environment using M homogeneous differential-drive robots. We employ a relay-based pushing strategy where robots sequentially transfer the parcel at designated RPs. The parcel's polyhedral shape is leveraged at each handoff, allowing the pushing surface to be changed to best suit the next trajectory segment.

The problem is defined within a 2D bounded environment with static obstacles $\mathcal{O}$. Given the parcel's initial configuration $q_s = [x_s, y_s]^{\mathrm{T}}$ and a target goal $q_g = [x_g, y_g]^{\mathrm{T}}$, our objective is to find the sequence of relay points and generate the spatio-temporal trajectories that ensure a stable, collision-free transport. As shown in Fig. 3, our proposed framework addresses this by first determining the locations of the RPs and then planning the detailed push and pickup trajectories between them.

The analysis relies on two key assumptions: (1) contacts are rigid and governed by Coulomb friction with no wheel slip; (2) motions are quasi-static, rendering inertial forces negligible.

The kinematic model of the mobile robot is expressed as:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{1}$$

where the velocity v_r and angular velocity ω_r are the control inputs. During pushing, the robot and the manipulated object are assumed to be relatively static, i.e., $\dot{x}_r = \dot{x}_o$, $\dot{y}_r = \dot{y}_o$, and $\dot{\theta}_r = \dot{\theta}_o$. The pushed object model is shown in Fig. 2. Following [24], the line contact between the end-effector and

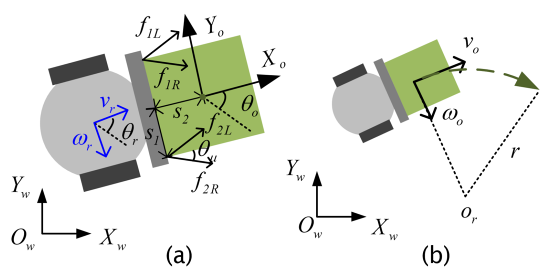

Fig. 2. Schematic representation of the pushed object. (a) The analysis of the contact force between the object and the end effector planner. (b) The force experienced by the object during motion along a circular trajectory.

the object is modeled as contact at the line's endpoints. Each contact force is decomposed into two components aligned with the edges of the friction cone. The angle between each component and the contact normal is:

$$\theta_\mu = \tan^{-1}\mu \tag{2}$$

where $\mu > 0$ is the friction coefficient of the interacting surfaces. The vector of contact forces is defined as:

$$f_c = [f_{1R}, f_{1L}, f_{2R}, f_{2L}]^T \tag{3}$$

For simplicity, the analysis is conducted in the object's body frame, $\sum_o$. The total external wrench $w \in \mathbb{R}^3$ is applied at the object's center of mass. Under the quasi-static assumption, the object's motion can be described using an ellipsoidal approximation of the limit surface, $S(w) = \frac{1}{2}w^{\mathrm{T}}Hw$. The matrix $H \in \mathbb{R}^{3\times3}$, which represents this approximation, can be estimated from the object's geometry, mass, and the friction coefficient μ [25]. Based on the principle of maximal dissipation [26], the object's instantaneous velocity is normal to the limit surface for a given wrench, which implies:

$$[\dot{x}_o, \dot{y}_o, \dot{\theta}_o]^T = Hw = HGf_c \tag{4}$$

where (x_o, y_o, θ_o) is the object's pose in its body frame. This equation relates the object's motion to the contact forces via the grasp matrix $G \in \mathbb{R}^{3\times4}$. For a rectangular object, G is defined as:

$$G = \begin{bmatrix} \cos\theta_\mu & -\sin\theta_\mu & (-s_1\cos\theta_\mu + s_2\sin\theta_\mu) \\ \cos\theta_\mu & \sin\theta_\mu & (-s_1\cos\theta_\mu - s_2\sin\theta_\mu) \\ \cos\theta_\mu & -\sin\theta_\mu & (s_1\cos\theta_\mu + s_2\sin\theta_\mu) \\ \cos\theta_\mu & \sin\theta_\mu & (s_1\cos\theta_\mu - s_2\sin\theta_\mu) \end{bmatrix}^T \tag{5}$$

where s_1 is half the length of the end-effector and s_2 is the distance from the end-effector's midpoint to the object's origin O_o. From (4), motion constraints can be derived from the condition that contact forces must lie within the friction cone, meaning each component of f_c must be non-negative. For motion along an arc of radius r and angular velocity ω_c, equation (4) can be rewritten in the robot's body frame as:

$$f_c = (HG)^+ \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{\theta}_0 \end{bmatrix} = (G^{\mathrm{T}}H^{\mathrm{T}}HG)^{-1}G^{\mathrm{T}}H^{\mathrm{T}} \begin{bmatrix} \omega_c r \\ 0 \\ \omega_c \end{bmatrix} \tag{6}$$

where ω_c and r are signed to indicate the direction of rotation. Here, $(HG)^+$ is the pseudo-inverse of HG. The constraint on r is thus:

$$(HG)^+[|r|, 0, 1]^{\mathrm{T}} \geq 0 \tag{7}$$

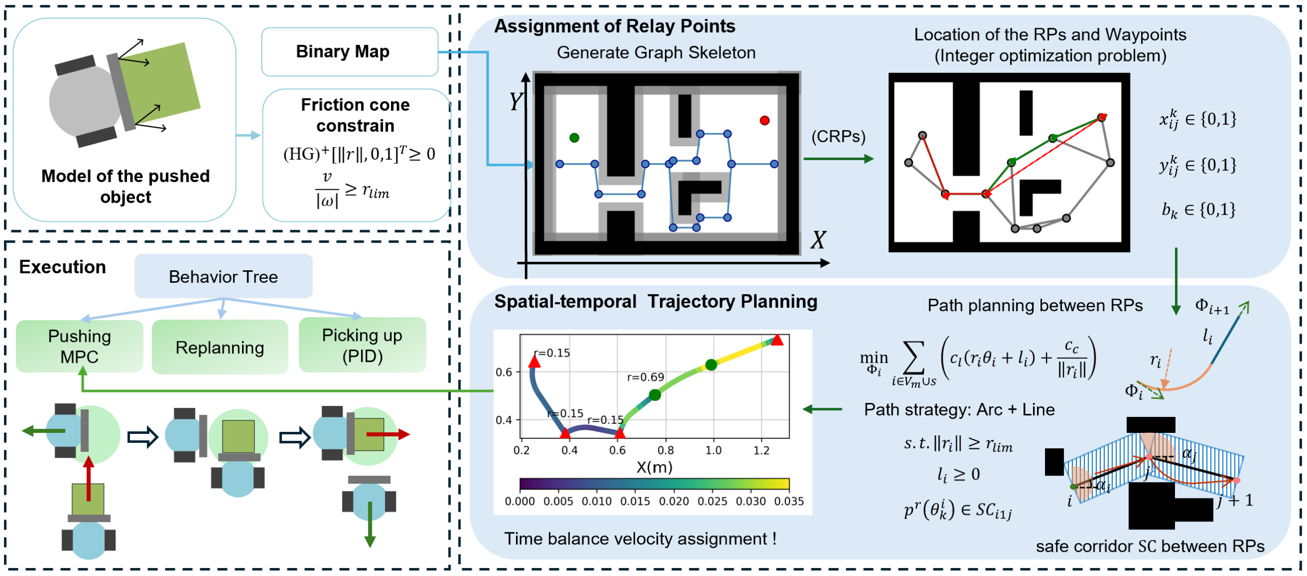
Model of the pushed object
Binary Map
Friction cone constrain
$(HG)^+[\|r\|,0,1]^T \geq 0$
$\frac{v}{|\omega|} \geq r_{lim}$
Assignment of Relay Points
Generate Graph Skeleton
(CRPs)
Location of the RPs and Waypoints (Integer optimization problem)
$x_{ij}^k \in \{0,1\}$
$y_{ij}^k \in \{0,1\}$
$b_k \in \{0,1\}$
Execution
Behavior Tree
Pushing MPC
Replanning
Picking up (PID)
Spatial-temporal Trajectory Planning
Path planning between RPs
$\min_{\Phi_i} \sum_{i \in V_m \cup s} \left(c_l(r_i\theta_i + l_i) + \frac{c_c}{\|r_i\|} \right)$
Path strategy: Arc + Line
$s.t. \|r_i\| \geq r_{lim}$
$l_i \geq 0$
$p^r(\theta_k^i) \in SC_{i1j}$
safe corridor SC between RPs
Time balance velocity assignment !
X(m)

Fig. 3. Overview of the proposed framework, which consists of two main stages: (i) offline assignment of RPs and trajectory planning, and (ii) online execution with handover synchronization.

which defines the minimum feasible turning radius r_{lim}. This lower bound on curvature ensures that planned paths respect contact and friction constraints.

At any instant, the object's motion can be viewed as a rotation about an Instantaneous Center of Rotation o_r with an instantaneous turning radius of $v_o/|\omega_o|$ (see Fig. 2 (b)). To ensure the entire trajectory is physically realizable, this instantaneous radius must never be smaller than the minimum feasible radius. Thus, the controller and feasible trajcotories also need to satisfy the constraint:

$$\frac{v_o}{|\omega_o|} \geq r_{\text{lim}} \tag{8}$$

III. Relay Points Assignment

The first stage of our framework determines the locations of RPs by analyzing a topological skeleton of the environment, specifically a directed Reeb graph. This structure is ideal because it compactly encodes the free-space topology and inherently identifies high-clearance corridors and critical geometric events (e.g., junctions, bends).

We leverage this structure based on the principle that the most effective RPs are situated in high-clearance areas that coincide with these topological events. Handoffs at such locations can simplify subsequent path segments and reduce overall path complexity. Our method, therefore, consists of two steps: (a) we first generate a set of Candidate Relay Points (CRPs) from the directed Reeb graph, and (b) we then solve an integer optimization problem to select the optimal subset of these candidates that will serve as the final RPs and intermediate waypoints for the transport task.

A. Candidate Relay Points Generation

As detailed in Algorithm 1, we generate CRPs from a directed Reeb graph that forms the environment's skeleton. We construct the graph using the height function $f(x, y) = x$,

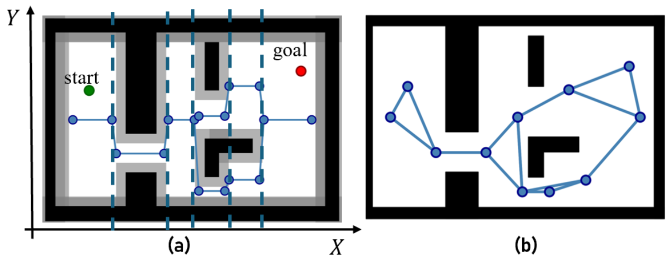

Fig. 4. (a) A schematic view of the Reeb Graph of a rectangular environment with static obstacles. (b) The resulting set of Candidate Relay Points (CRPs) and their connections after applying Algorithm 1.

which collapses regions of free space with the same x-coordinate into level sets [27]. The graph's vertices, V_o, mark the critical points where the free-space topology changes (Fig. 4 (a)), and its edges are directed by increasing x-coordinate. Prior to construction, obstacles are inflated by a safety margin d_s.

The raw Reeb graph then undergoes a multi-step refinement process to produce a more practical set of CRPs (shown as Algorithm 1). First, to simplify the graph, nearby connected nodes (distance $< r_b$) are merged by replacing them with their midpoint, provided all resulting edge reconnections remain collision-free. Next, the graph is pruned by removing redundant nodes whose neighbors can be directly connected with a collision-free edge. Conversely, to create valuable shortcuts, new collision-free edges are added between any two nodes within a specified radius. Finally, connectivity to the start and goal configurations is ensured. An example of the resulting CRPs graph is shown in Fig. 4 (b).

B. Location of the RPs and waypoints

The selection of RPs is formulated as an integer optimization problem that seeks to minimize a composite cost of path

Algorithm 1 Generate CRPs

```
    Input: Environment $En$ with Obstacles $\mathcal{O} = \{O_i\}$, Start Configuration $p_s$, Goal Configuration $p_g$
    Output: Directed Skeleton Graph $G_s = \{V_s, E_s\}$
1:      Expansion Obstacles $O_i$ with Safe Margin $d_s$
2:      Calculate the original Reeb Graph $G_o = \{V_o, E_o\}$
3:      Combine two near nodes by their middle point
4:      Delete Redundant Nearby Nodes
5:      Add collision-free edges between nearby nodes
6:      Connect start $p_s$ and goal $p_g$ to the graph
```

length and turning angles. This dual objective is essential: pushing manipulation is sensitive to path curvature, and hand-offs at RPs allow for a change in the pushing face, which directly influences the effective turning angle. Furthermore, since not all candidate RPs are directly connected in the graph G_s, intermediate waypoints are required to form feasible path segments between them.

To quantify the path length objective, we define an estimated distance matrix $L_E \in \mathbb{R}^{n\times n}$ based on the graph G_s. The distance d_{ij} between two nodes i and j is their Euclidean distance if a direct edge exists ($l_{ij} = 1$), and infinite otherwise:

$$d_{ij} \triangleq \begin{cases} \|p_i - p_j\|, l_{ij} = 1 \\ +\infty, l_{ij} = 0 \end{cases} \tag{9}$$

To model the turning cost, we first define a 3D tensor $C_E \in \mathbb{R}^{n\times n\times n}$ that stores the geometric angle for every possible three-node sequence (i, j, z). The element c_{ijz} is the angle formed by the vectors $\vec{p_i p_j}$ and $\vec{p_j p_z}$, representing the turn at node j, as shown in Fig. 5 (b). A penalty angle of 2π is assigned if the path $i \to j \to z$ is not connected in G_s.

$$c_{ijz} \triangleq \begin{cases} \arccos\left[\dfrac{(p_j - p_i)\cdot(p_z - p_j)}{\|p_j - p_i\|\|p_z - p_j\|}\right], & l_{ij} = 1 \wedge l_{jz} = 1 \\ 2\pi, & \text{otherwise} \end{cases} \tag{10}$$

Crucially, the geometric angle c_{ijz} can be modified by changing the pushing face at the RP j, as shown in Fig. 5 (a). For an object like a cube, a 90° change of face corresponds to a corrective rotation of $\delta = \pi/2$. This allows us to define two adjusted angular deviation tensors, C_E^l and C_E^r, which represent the effective turning angle after applying this rotation in either direction:

$$c_{ijz}^l \triangleq \begin{cases} c_{ijz} - \delta, & l_{ij} = 1 \wedge l_{jz} = 1 \\ 2\pi, & \text{otherwise} \end{cases} \tag{11}$$

Similar to the right angular deviation matrix $C_E^r \in \mathbb{R}^{n\times n\times n}$, which has elements $c_{ijz}^r \in [-\pi, \pi]$, is defined:

$$c_{ijz}^r \triangleq \begin{cases} c_{ijz} + \delta, & l_{ij} = 1 \wedge l_{jz} = 1 \\ 2\pi, & \text{otherwise} \end{cases} \tag{12}$$

Selecting the optimal pushing face (and thus the corresponding angle cost) at each RP is a key decision variable in our optimization, critical for minimizing path deviation and improving manipulation precision.

Then, the assignment of RPs is formulated as an integer optimization problem with following three sets of binary variables:

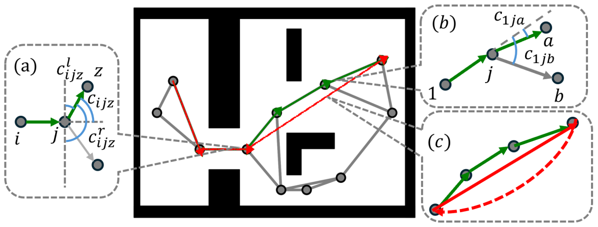

Fig. 5. Illustration of the relay point and waypoint selection problem. (a) Turning angle calculation at a waypoint. (b) Effect of push face selection on turning angle at a relay point. (c) Virtual loop Path formulation.

- $x_{ij}^k = 1$ when the k^{th} segment is from relay point i to j; otherwise, $x_{ij}^k = 0$.
- $y_{ij}^k = 1$ when the waypoint i is connected to point j and belongs to segment k; otherwise, $y_{ij}^k = 0$.
- $b_k = 0$ when the pushing face changes to the left one at the end of the segment k; $b_k = 1$ when it changes to the right one.

where segment $k \in K = \{1, 2, \cdots, M\}$ is assigned to each mobile robot, $i \in V_s$, $j \in V_s$. Given that M mobile agents are used, there are M distinct segments linking the start, goal, and $M - 1$ RPs. The goal of the optimization process is to find the locations and sequence of relay points, waypoints, and the changes of pushing faces to minimize the cost function:

$$\begin{aligned} Q = {} & \underbrace{a_1 \sum_{k\in K}\sum_{i\in V_s}\sum_{j\in V_s} y_{ij}^k l_{ij} d_{ij}}_{①} \\ & + a_2 \Bigg[\underbrace{\sum_{k\in K}\sum_{i\in V_s}\sum_{j\in V_s}\sum_{z\in V_s} \|c_{jiz}\| y_{ji}^k y_{iz}^k l_{ji} l_{iz}}_{②} \\ & \underbrace{+ \sum_{k\in K, k\neq M}\sum_{j\in V_s}\sum_{z\in V_s}\sum_{r\in V_s} y_{zj}^k y_{jr}^{k+1} l_{zj} l_{jr} \times (b_k c_{j,z,r}^l + (1 - b_k) c_{j,z,r}^r)}_{③} \Bigg] \end{aligned} \tag{13}$$

where $a_1 \in \mathbb{R}$, $a_2 \in \mathbb{R}$ are the weights of the distance term and angular deviation term. The first term ① is the sum of the Euclidean distance; the second term ② is the cumulative angular deviation at waypoints; the third term ③ is the angular deviation resulting from push-face reorientations at RPs. The choice of turning direction at RPs is modeled by the binary variables b_k. It's noted that only $M-1$ times the push surface changes are needed. The constraints are formulated as follows:

- Unique flow of each segment connecting two adjacent RPs:

$$\sum_{i\in V_s}\sum_{j\in V_s} x_{ij}^k = 1, \quad \forall k \in K \tag{14}$$

- Constrains the length of each segment k to be within the range $[L_{\min}, L_{\max}]$:

$$L_{\min} \le \sum_{i\in V_s}\sum_{j\in V_s} y_{ij}^k l_{ij} d_{ij} \le L_{\max}, \quad \forall k \in K \tag{15}$$

- The first segment departs from p_s:

$$\sum_{i\in V_s} x^1_{si}=1, \qquad \sum_{i\in V_s} y^1_{si} l_{si}=1 \tag{16}$$

- The last segment ends at p_g:

$$\sum_{i\in V_s} x^M_{ig}=1, \qquad \sum_{i\in V_s} y^M_{ig} l_{ie}=1 \tag{17}$$

- The flow balance constraints for each RP:

$$\sum_{i\in V_s} x^k_{ij}=\sum_{z\in V_s} x^{k+1}_{jz}, \quad \forall j,\forall k\cup k\neq M \tag{18}$$

- The flow conservation constraints for each segment:

$$\sum_{i\in V_s} y^k_{ij} l_{ij}+\sum_{i\in V_s} x^k_{ji}=\sum_{i\in V_s} y^k_{ji} l_{ji}+\sum_{i\in V_s} x^k_{ij}, \forall j,k \tag{19}$$

  Each route may include additional waypoints in addition to the two end RPs. In Fig. 5 (c), considering the route connecting the two end RPs (represented by the red solid line) as the anti-path (depicted by the red dashed arc) creates a loop, ensuring the segment remains connected by waypoints. The constraint (19) gives the flow balance of the waypoints in the segment k.
- The consistency constraints of x^k_{ij} and y^k_{ij} for each segment:

$$\sum_{j\in V_s}\sum_{i\in V_s}\sum_{z\in V_s} x^k_{ij} y^k_{iz} l_{iz}=1, \quad \forall k\in K \tag{20}$$

$$\sum_{i\in V_s}\sum_{j\in V_s}\sum_{z\in V_s} x^k_{ij} y^k_{zj} l_{zj}=1, \quad \forall k\in K \tag{21}$$

- The flow of waypoints is no more than one:

$$\sum_{i\in R}\sum_{k\in K} y^k_{ij} l_{ij}\leq 1, \quad \forall j \tag{22}$$

$$\sum_{j\in R}\sum_{k\in K} y^k_{ij} l_{ij}\leq 1, \quad \forall i \tag{23}$$

The problem described in (13) is a multi-objective integer optimization problem. It involves two objectives with incommensurable units: total distance (Term ①, in meters) and cumulative angular deviation (Terms ② and ③, in radians). To create a unified objective function, we employ a weighted sum method where each objective is first normalized.

Normalization is achieved using the Utopia and Nadir points for each objective, a standard technique for multi-objective optimization [28]. Then, the composite objective function is formulated to minimize a weighted sum of the normalized objectives:

$$\bar{Q}=a_1\frac{① - f^d_U}{(f^d_N-f^d_U)}+a_2\frac{②+③ - f^a_U}{(f^a_N-f^a_U)} \tag{24}$$

where f^d_N and f^d_U are the Nadir and Utopia points of distance objective, f^a_N and f^a_U are the Nadir and Utopia points of angular deviation objective. The non-negative weights a_1 and a_2 are used to control the trade-off between the two normalized objectives.

The formulated Integer Programs (IPs) are solved using the state-of-the-art commercial solver Gurobi [29]. To deal with the multiplication of more than two variables that are not supported in Gurobi, it is possible to define auxiliary variables v^k_{zjr}, which are also binary variables, and add the constraints:

$$v^k_{zjr}=y^k_{zj} y^{k+1}_{jr} l_{zj} l_{jr}, \quad \forall z,j,r,k \tag{25}$$

The optimization results will give the set of waypoints V_w, the set of RPs V_{RP}, their sequences and the pushing face changed at each RP. It is noted that RPs are also waypoints, i.e., $V_{RP}\in V_w$.

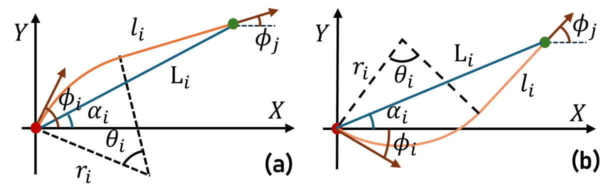

Fig. 6. Geometric path synthesis between two waypoints using an arc-line primitive. The path's shape is determined by the waypoints' relative positions and orientations, resulting in (a) a right turn or (b) a left turn.

IV. Spatial-Temporal Trajectory Planning

With the sequence of waypoints and rendezvous points (RPs) established, we generate the connecting spatial-temporal trajectory. This process involves three phases: Geometric Path Planning to create a curvature-constrained path, Time-Parameterization to ensure dynamic feasibility, and Online Handover Synchronization at RPs to enable seamless object transfers.

A. Geometric Path Planning with Curvature and Corridor Constraints

We synthesize curvature-bounded paths between successive waypoints using an arc–line primitive constrained to lie within safe corridors extracted from the skeleton.

The path between any two waypoints consists of a constant-curvature arc followed by a straight line, as illustrated in Fig. 6. The geometry of this path is uniquely determined by the orientations at the start and end waypoints. The total turning angle of the arc, $\theta_i\in[-\pi,\pi]$, is the difference between the final and initial orientations:

$$\theta_i=\phi_j-\phi_i \tag{26}$$

where $\phi_i\in[-\pi,\pi]$ and $\phi_j\in[-\pi,\pi]$ are the orientation of waypoint i and its subsequent waypoint j. By decomposing the path into x and y directions, the path can be obtained by following the equations:

$$\begin{cases} L_i\cos\alpha_i=l_i\cos\phi_j+r_i\cos(\phi_i-\frac{\pi}{2})-r_i\cos(\frac{\pi}{2}-\phi_j)\\ L_i\sin\alpha_i=l_i\sin(\phi_j)+r_i\sin(\phi_i-\frac{\pi}{2})+r_i\sin(\frac{\pi}{2}-\phi_j)\end{cases} \tag{27}$$

where L_i and α_i are the distance and angle between waypoints i and j, respectively. The path parameters are the arc radius r_i and the straight-line length l_i. Solving (27) for these parameters yields the explicit solution:

$$\begin{bmatrix} l_i\\ r_i\end{bmatrix}=\begin{bmatrix}\frac{L_i(\cos(\alpha_i-\phi_i)-\cos(\alpha_i-\phi_j))}{\cos(\phi_i-\phi_j)-1}\\ -\frac{L_i\sin(\alpha_i-\phi_j)}{\cos(\phi_i-\phi_j)-1}\end{bmatrix} \tag{28}$$

However, the choice of orientations ϕ_i and ϕ_j is constrained by the path geometry. The feasibility depends on whether the initial orientation ϕ_i is greater or less than α_i, the direct angle to the next waypoint. These two cases correspond to a right turn ($r_i > 0$) or a left turn ($r_i < 0$), as shown in Fig. 6.

If $\phi_i > \alpha_i$ (a right turn, Fig. 6 (a)), the final orientation ϕ_j is constrained to:

$$2\alpha_i - \phi_i \le \phi_j \le \alpha_i \tag{29}$$

Conversely, if $\phi_i < \alpha_i$ (a left turn, Fig. 6 (b)), the constraint on ϕ_j is:

$$\alpha_i \le \phi_j \le 2\alpha_i - \phi_i \tag{30}$$

The boundary cases are notable: when $\phi_j = 2\alpha_i - \phi_i$, the path consists solely of an arc ($l_i = 0$), and when $\phi_j = \phi_i$, the path degenerates to a straight line ($r_i \to \infty$). For implementation, the conditional constraints in (29) and (30) can be combined into a single set of inequalities:

$$\begin{cases} (\phi_i - \alpha_i)(\phi_j - \alpha_i) \le 0 \\ (\phi_i - \alpha_i)(\phi_i + \phi_j - 2\alpha_i) \ge 0 \end{cases} \tag{31}$$

And the constraints of r_i and l_i are

$$l_i \ge 0, \qquad \|r_i\| \ge r_{\text{lim}} \tag{32}$$

To guarantee collision avoidance, we introduce safe corridors that confine the path to obstacle-free regions. For each straight, collision-free edge (i, j) in the graph G_s, we construct a rectangular safe corridor aligned with the edge. The corridor's width is defined by the distance to the nearest obstacles, as depicted in Fig. 7 (a). By ensuring that the curved portions of the path remain within these corridors, the entire trajectory is guaranteed to be collision-free.

To enforce this, we discretize the arc segment of the path between waypoints i and j and require each point $p^r(\theta_k^i)$ to lie within the corresponding safe corridor SC_{ij}:

$$p^r(\theta_k^i) = [r_i \cos\theta_k^i - x_i^o, r_i \sin\theta_k^i - y_i^o]^{\mathrm{T}} \in SC_{ij} \tag{33}$$

where $\theta_k^i = \phi_i + \frac{k\theta_i}{n_k}$ for $k = 1, \dots, n_k$, and the arc's center (x_i^o, y_i^o) is determined by r_i and ϕ_i. For easier computation, we transform this constraint into a local frame $X_{ij}Y_{ij}$, where waypoint i is the origin and the edge ij aligns with the x-axis (Fig. 7(b)). Equation (33) then becomes:

$$\begin{bmatrix} 0 \\ b_i^l \end{bmatrix} \le \begin{bmatrix} \cos\alpha_i & \sin\alpha_i \\ -\sin\alpha_i & \cos\alpha_i \end{bmatrix} p^r(\theta_k^i) \le \begin{bmatrix} L_{ij} \\ b_i^u \end{bmatrix} \tag{34}$$

where $b_i^u \in \mathbb{R}$ and $b_i^u \in \mathbb{R}$ are the upper bound and lower bound of the safe corridor under the frame $X_{ij}Y_{ij}$.

The safe corridors impose geometric constraints on the orientation ϕ_k at each node. As illustrated in Fig. 7 (a), the feasible orientation is confined to the angular sector formed by the intersection of adjacent corridors. For a standard waypoint i, the orientation ϕ_i must align with both the incoming (α_{i-1}) and outgoing (α_i) edges. However, at a Relay Point (RP), the ability to switch the pushing face shifts the constraint relative to the outgoing edge.

By introducing an offset angle $\delta_i \in \{0, \pi/2, -\pi/2\}$ to represent this face switch (where $\delta_i = 0$ for standard waypoints, and $\pm\pi/2$ for right/left turns at RPs), these geometric limits are unified as:

$$\max(\alpha_{i-1}, \alpha_i + \delta_i) - \frac{\pi}{2} \le \phi_i \le \min(\alpha_{i-1}, \alpha_i + \delta_i) + \frac{\pi}{2} \tag{35}$$

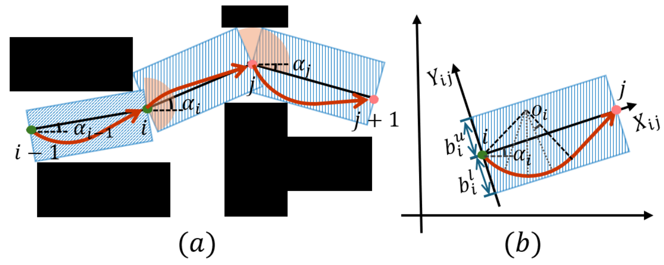

Fig. 7. Safe corridors (blue rectangles) constrain the planned path to collision-free regions. The orientation at each waypoint (green points) is constrained by the intersection of adjacent corridors.

These constraints define a feasible search space for the path geometry. The optimal orientations ϕ_i at each waypoint are found by solving the following optimization problem, which minimizes a cost function that balances path length and curvature:

$$\begin{aligned} \min_{\{\phi_i\}} \quad & Q' = \sum_{i \in V_w \cup \{s\}} \left(c_l(|r_i\theta_i| + l_i) + \frac{c_c}{|r_i|} \right) \\ \text{s.t.} \quad & \text{Eqs. (28), (31), (32), (34), (35)} \end{aligned} \tag{36}$$

where c_l and c_c are weights for path length and curvature, respectively. The problem is non-convex, and a feasible initial guess is difficult to find, often causing standard solvers to fail. To overcome this, we employ a two-stage approach. First, a Genetic Algorithm (GA) [30] finds a high-quality, feasible initial solution. This solution is then used to warm-start a gradient-based solver (IPOPT [31]), which refines the path to a local optimum. This two-stage optimization yields a feasible and optimized geometric path connecting all waypoints.

B. Time-Parameterization and Kinematic Feasibility

Once the geometric path is defined, the next step is to generate a time-parameterized trajectory that is both dynamically feasible and time-efficient. The goal is to assign a velocity profile to the path that respects the robot's physical limits while minimizing total transport time and balancing the workload across the relay team.

The geometric path, composed of arc and line segments, is time-parameterized by discretizing each segment and optimizing the time duration for each subsegment. The optimization variables are $\Delta t_c^{(i,j)}$ and $\Delta t_l^{(i,j)}$, representing the time to traverse the j-th subsegment of the i-th arc and line, respectively.

The optimization problem is formulated to minimize a weighted sum of the total execution time and the variance of the workload among the robots, ensuring a balanced and efficient transport:

$$\min_{\{\Delta t\}} \quad \sum_{k=1}^{M} T_k + \alpha \sum_{k=1}^{M} (T_k - T_{\text{avg}})^2 \tag{37}$$

where $T_k = \sum_{j=1}^{S_{c,k}} \Delta t_c^{(k,j)} + \sum_{j=1}^{S_{l,k}} \Delta t_l^{(k,j)}$ is the total execution time for robot k, T_{avg} is the average time across all robots, and α is a weighting factor. This minimization is subject to constraints derived from the robot's kinematics and dynamics.

The robot's maximum wheel velocity and acceleration impose limits on the motion along both straight lines and arcs.

For straight-line motion, the pushing force is aligned with the direction of travel, so the friction cone constraint is naturally satisfied. The limits are therefore dictated by the robot's maximum wheel velocity and acceleration. The constraints on the linear velocity v_l^i and acceleration a_l^i are:

$$0 \le v_l^i \le \omega_{\text{max}} r_w, \qquad 0 \le a_l^i \le a_{\text{max}} r_w \tag{38}$$

where ω_{max} and a_{max} are the maximum angular velocity and maximum angular acceleration of the wheels of which the radius is r_w.

For the motion along the arc, the similar constraints of a_c^i and ω_c^i are related to the raduis r_i of the arc, which is as follows:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \le \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} = \begin{bmatrix} \frac{1}{r_w} & \frac{l_w}{2r_w} \\ \frac{1}{r_w} & -\frac{l_w}{2r_w} \end{bmatrix} \begin{bmatrix} \omega_c^i r_i \\ \omega_c^i \end{bmatrix} \le \begin{bmatrix} \omega_{\text{max}} \\ \omega_{\text{max}} \end{bmatrix} \tag{39}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \le \begin{bmatrix} a_r \\ a_l \end{bmatrix} = \begin{bmatrix} \frac{1}{r_w} & \frac{l_w}{2r_w} \\ \frac{1}{r_w} & -\frac{l_w}{2r_w} \end{bmatrix} \begin{bmatrix} a_c^i r_i \\ a_c^i \end{bmatrix} \le \begin{bmatrix} a_{\text{max}} \\ a_{\text{max}} \end{bmatrix} \tag{40}$$

where l_w is the distance between the robot's wheels. The constraints governing motion along arcs and straight lines differ significantly.

Therefore, the constraints of optimization problem (37) are detailed as follows:

Velocity Constraints: With the path geometry (l_i, r_i) fixed, the velocity limits from (38) and (39) translate into lower bounds on the time duration for each subsegment. An upper bound, t_{max}, is also imposed to prevent excessively slow movements and constrain the search space. This yields the following constraints on the time durations $\Delta t_c^{(i,j)}$ and $\Delta t_l^{(i,j)}$:

$$\frac{\theta_i}{S_{c,i}\omega_{\text{max}}^{c,i}} \le \Delta t_c^{(i,j)} \le t_{\text{max}}, \quad \forall (i,j) \in V_c \tag{41}$$

$$\frac{l_i}{S_{l,i}\omega_{\text{max}} r_w^l} \le \Delta t_l^{(i,j)} \le t_{\text{max}}, \quad \forall (i,j) \in V_l \tag{42}$$

where θ_i and l_i are the total angular displacement and length of the i-th path segment, respectively. V_c and V_l are the sets of all arc and line subsegments. The maximum angular velocity for arc i is $\omega_{\text{max}}^{c,i}$.

Acceleration Constraints: To enforce the robot's acceleration limits, the change in velocity between consecutive subsegments must be bounded. The average acceleration is approximated at the midpoint between subsegments $j-1$ and j. This yields the following constraints for arc and line segments, respectively:

$$-a_{\text{max}}^c \le \frac{2\theta_i}{S_{c,i}} \frac{\Delta t_c^{(i,j-1)} - \Delta t_c^{(i,j)}}{\Delta t_c^{(i,j)} \Delta t_c^{(i,j-1)} (\Delta t_c^{(i,j)} + \Delta t_c^{(i,j-1)})} \le a_{\text{max}}^c \tag{43}$$

$$-a_{\text{max}}^l \le \frac{2l_i}{S_{l,i}} \frac{\Delta t_l^{(i,j-1)} - \Delta t_l^{(i,j)}}{\Delta t_l^{(i,j)} \Delta t_l^{(i,j-1)} (\Delta t_l^{(i,j)} + \Delta t_l^{(i,j-1)})} \le a_{\text{max}}^l \tag{44}$$

where α_{max}^c and a_{max}^l are the maximum angular and linear accelerations, respectively.

Kinematic Constraints: The stable pushing constraints (7) and (8), introduced in Section II, are inherently satisfied by the geometric path planner (Section IV-A). The use of arc-line primitives with a minimum turning radius ensures these conditions are met. Therefore, they do not require re-evaluation during time-parameterization.

Initial and Terminal Constraints: To ensure the robot starts and ends its trajectory at zero velocity, the time durations for the initial and final subsegments are constrained by the robot's maximum acceleration and deceleration capabilities.

$$\Delta t_c^{(i,0)} \ge \sqrt{\frac{2\theta_i}{S_{c,i}\alpha_{\text{max}}^c}}, \qquad \Delta t_l^{(i,S_{l,i}-1)} \ge \sqrt{\frac{2l_i}{S_{l,i}a_{\text{max}}^l}}. \tag{45}$$

Velocity Continuity Constraints: To ensure smooth transitions between different segment types (e.g., line-to-arc or arc-to-line), the change in velocity must not exceed the robot's acceleration limits. This is enforced by constraining the average acceleration during the transition:

$$|v_{c,\text{start}} - v_{l,\text{end}}| \le a_{\text{max}} \cdot \Delta t_{\text{trans}} \tag{46}$$

$$|v_{l,\text{start}} - v_{c,\text{end}}| \le a_{\text{max}} \cdot \Delta t_{\text{trans}} \tag{47}$$

where v_{start} and v_{end} are the velocities at the start and end of the respective segments, and Δt_{trans} is the duration of the transition.

Solving the optimization problem (37) yields a discrete sequence of time-parameterized states. To generate a continuous reference trajectory for the robot's controller, we apply cubic spline interpolation to these states. The resulting spline is then sampled at a uniform frequency to provide the target commands for the low-level tracking controller.

C. Online Handover Synchronization at RPs

During execution, short-horizon timing adjustments are made to synchronize the receiving robot with the delivering robot's actual arrival at the RP. This ensures a smooth handover and is achieved by solving a real-time replanning optimization problem formulated as:

$$\begin{aligned} \min_{\{\Delta t\}} \quad & \left(\sum_j (\Delta t_c^{(i,j)} + \Delta t_l^{(i,j)}) - T_{\text{target}}^i \right)^2 \\ \text{s.t.} \quad & \text{Eqs. } (41) - (47) \\ & T_{\text{min}}^i \le \sum_j (\Delta t_c^{(i,j)} + \Delta t_l^{(i,j)}) \le T_{\text{max}}^i \end{aligned} \tag{48}$$

where T_{target}^i is the target arrival time for the i-th robot. It is calculated based on the preceding robot's estimated arrival time T_{arrival}^{i-1}, a handover buffer $\Delta t_{\text{handover}}$, and a safety margin for trajectory generation Δt_{safe}: $T_{\text{target}}^i = T_{\text{arrival}}^{i-1} + \Delta t_{\text{handover}} - \Delta t_{\text{safe}}$. The bounds T_{min}^i and T_{max}^i define a feasible arrival window, guaranteeing that the receiving robot is in position before the parcel arrives. This formulation ensures precise temporal coordination between robots in the relay chain.

The optimization for the pickup trajectory is subject to the same kinematic constraints as the pushing phase—velocity

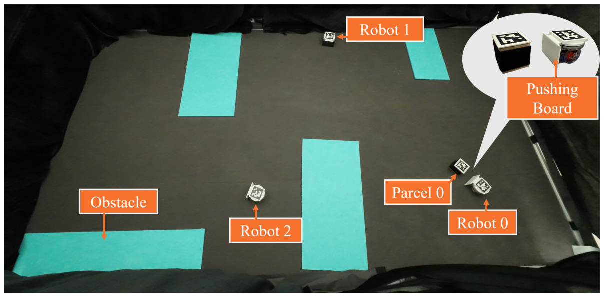

Fig. 8. Experimental robotic platform used to validate our methodology.

(41)-(42), acceleration, and continuity (46)-(47). However, since the robot is unladen, it can leverage higher velocity and acceleration limits to expedite its return. To ensure real-time performance, the optimizer is warm-started with the solution from the pre-planned pushing trajectory, which significantly accelerates convergence.

V. Experimental Validation

We validated our multi-robot relay system on a 1.4 m $\times$ 2.8 m board (Fig. 8). A team of differential-drive e-puck2 robots, equipped with front boards, pushed 5 cm $\times$ 5 cm $\times$ 5 cm wooden cubes. An overhead camera localized all robots and parcels using ArUco markers. With floor-object friction ($\mu_f \approx 0.3$) and robot-object friction ($\mu \approx 0.8$), the minimum turning radius was approximately 0.2 m, ensuring stable pushing. The system was implemented in ROS2, with all communication handled via topics at 10 Hz.

A. Assignment of RPs and Trajectory Planning

In this section, We tested our relay point (RP) assignment and trajectory planning in a four-robot scenario within a cluttered environment, as shown in Figure 9. First, Candidate Relay Points (CRPs, gray points in (b)) are generated from a Reeb Graph (Algorithm 1). An optimal set of RPs (connected by red arrows) and the pushing direction at each handoff are then determined by solving the mixed-integer program in (24). The objective function minimizes total distance and angular deviation, which avoids sharp turns by enabling the pushing face to be switched at RPs.

Subsequently, a smooth geometric path (blue curves in (c)) is planned that respects all curvature and safety constraints (Section IV-A). From this path, the final spatial-temporal trajectories are computed by solving (37), which successfully balances the duration for each robot to approximately 50s.

To validate the versatility of our proposed relay point assignment and trajectory planning method, we conducted tests across six diverse maps with varying obstacle distributions, as shown in Fig. 10. In each scenario, a multi-robot team was employed to deliver parcels from its start to its goal position, confirming our approach's adaptability to varied environments.

Notably, case (d) showcases the temporal cooperative capabilities of the relay system, featuring a continuous loop where four robots pass two parcels among themselves. The final two cases, (e) and (f), further demonstrate the method's

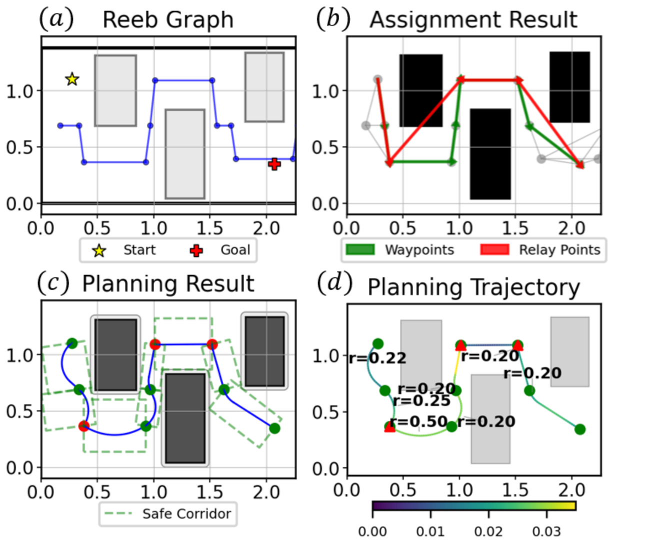

Fig. 9. Results of the assignment and trajectory planning stages for a cluttered environment: (a) The environment with obstacles, start, and goal points. (b) Selected RPs (red arrows) and waypoints (green arrows) from the candidate set (gray points). (c) The planned geometric paths. (d) The final time-parameterized trajectories, color-coded by velocity. All axes are in meters; velocity is in m/s.

effectiveness in relatively large-scale, cluttered environments. In every test, the method successfully assigned RPs and planned feasible trajectories while adhering to the kinematic constraints of the pushing manipulation.

A significant outcome is that the planned trajectories are predominantly straight, which simplifies the requirements for the pushing controller and enhances the overall robustness of the relay system. The autonomous execution of these plans is governed by the runtime architecture detailed below.

B. Behavior Tree and Execution

To translate the offline plans into real-time actions, each robot runs the same decentralized behavior tree (BT) shown in Fig. 12. Blackboards are used to share local state (Parcel ID, RPs, trajectories, Parcel states, Robot states). The communication between robots and location system are via ros2 topics. The main loop is composed of following two parts:

Action Execution: This involves a sequence of three primary actions: 1) *Pushing Sequence:* The robot first waits for the preceding robot to complete its push. It then executes a precisely timed approach trajectory, defined by a Bézier curve and tracked by a PID controller, to align perfectly with the parcel for a smooth handover. Upon arrival, it pushes the parcel to the next relay point using a model predictive controller from [32]. This controller enforces the constraint $\frac{v}{|\omega|} \geq r_{\text{lim}}$ to guarantee pushing stability.

2) *Replanning Sequence:* Concurrently, the robot navigates to a safe standby position while replanning its next trajectory. This replanning process uses the estimated completion time published by the previous robot to minimize handover latency and prevent collisions. Executing these actions in parallel is crucial for efficient and safe operation.

3) *Pickup Sequence:* Once replanning is complete, the robot executes the new trajectory to return to its designated relay

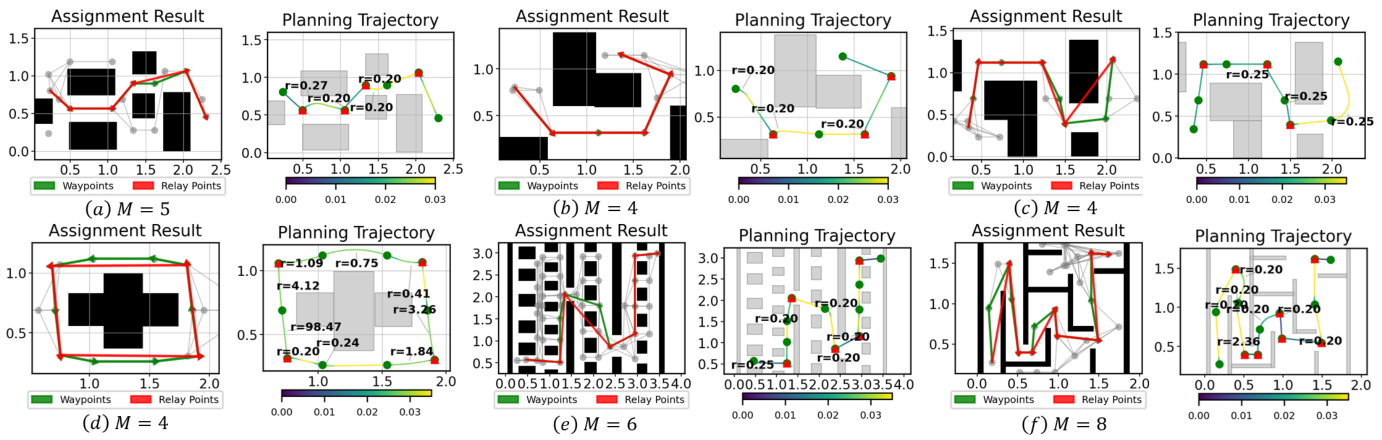

Fig. 10. The assignment and trajectory planning result of other six different maps. M represents the number of robots. All case with the same r limitations(0.2m).

point in anticipation of the next parcel. As this movement does not involve parcel manipulation, the robot is subject to fewer kinematic constraints. We can therefore employ a simple PID controller that allows for higher travel speeds, reducing the return time.

Completion Check: Upon completing its push, the robot verifies that the parcel has exited its workspace and then prepares for the next one by incrementing the parcel ID stored in the Blackboard by one.

The system's overall performance is illustrated in Fig. 12. The time visualization demonstrates the efficiency of the relay system. While a single parcel requires approximately $214\,s$ for delivery, four parcels are transported in only $649\,s$. This sub-linear scaling is achieved by having multiple parcels in transit simultaneously, a significant advantage over single-robot architectures. The handover process is streamlined by a fixed $5\,s$ approach time, ensuring efficient transfers.

The box plot in Fig. 12 validates the effectiveness of our time-parameterization method for workload balancing, showing nearly equal pushing times with low variance across the robots. The one exception is Robot 2, whose shorter pushing time is attributed to its comparatively short path, where the physical sizes of the robot and parcel are non-negligible. The data also confirms that a robot's pickup time is consistently shorter than the preceding robot's push time. For example, Robot 3's $21\,s$ pickup is well within Robot 2's $33\,s$ push duration. This critical timing relationship guarantees that each robot can return to its relay point before the next parcel arrives, ensuring the continuous flow of the system.

The experiments depicted in Figure 9 and Figure 10 (a)–(d) were all conducted in a real-world setting. Due to constraints of the experimental platform, the cases illustrated in sub-figures (e)–(f) were not physically realized. Comprehensive video documentation of the successful trials is available as supplementary material online.[1].

Figure 11 presents snapshots of the pushing process for a single parcel. In the figure, red arrows denote a robot in the pushing state, while green arrows indicate the parcel has arrived and the robot is beginning its return. All experimental trials resulted in the successful, collision-free delivery of multiple parcels

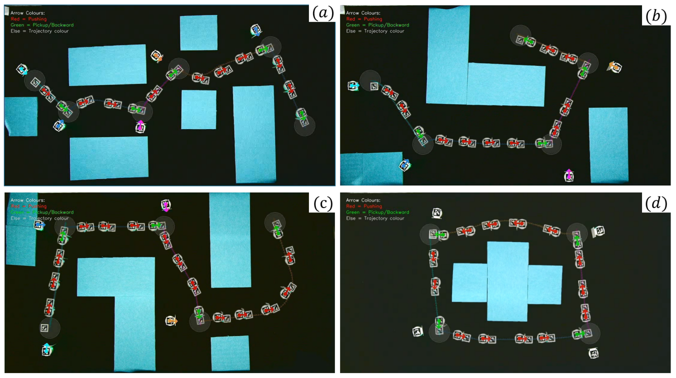

Fig. 11. Snapshots from real-world experiments, showing the successful transport of parcels through the environments from Fig. 10 (a)-(d).

Figure 13 presents a detailed analysis of the pushing and return trajectories for parcel 1, where trajectory segments are denoted by a starting circle and an ending square. The pushing trajectory, depicted in Figure 13 (a), is notably smooth and features gentle curvature, confirming that the generated plans are physically feasible for pushing manipulation. This feasibility is further substantiated by visual snapshots in Figure 13 (b), which show the robot maintaining continuous contact with the parcel. The picking up trajectories in Figure 13 (c) demonstrate that robots retract to the relay point at an increased velocity post-delivery, thereby improving the temporal efficiency of the handover cycle. The real completion time was less than the reference time, attributed to the parallel execution of the replanning algorithm and the robot's "move back to safe position" action. This outcome is an expected consequence of the system's design for parallel processing and does not signify a deficiency in the trajectory following controller.

C. Comparison

To validate the effectiveness of the proposed trajectory planning method, a comparative analysis was conducted against two baseline approaches: (1) connecting waypoints (including RPs) with Bézier curves (WPs+Bézier), and (2) smoothing a

[1]Supplementary videos: https://vimeo.com/1140644540

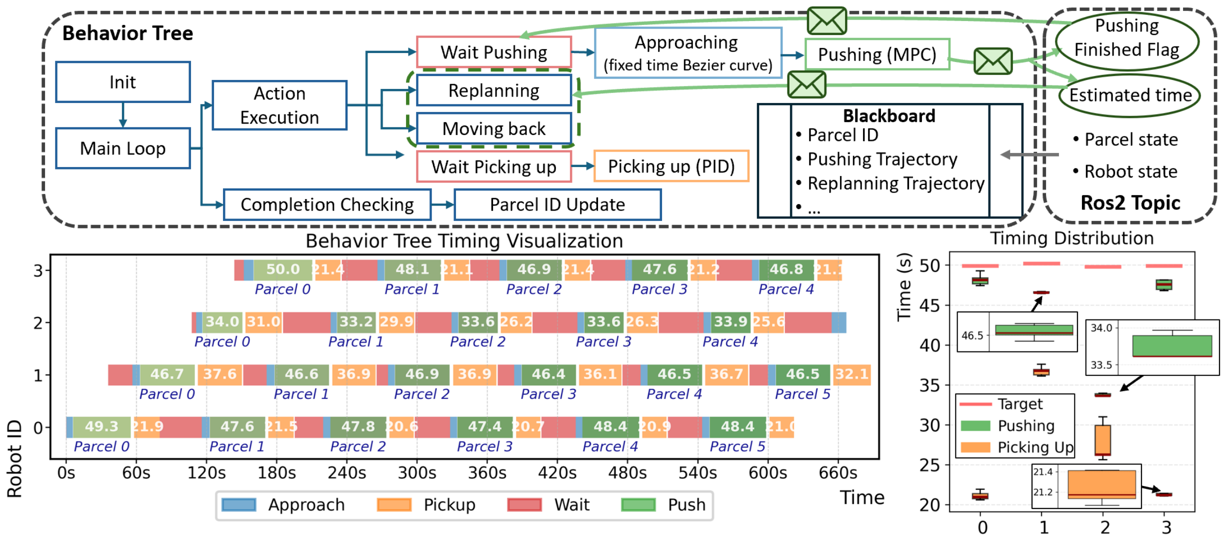

Fig. 12. The behavior tree for each robot of the relay system and the time visualization of the behavior tree execution in the experiments of case of Fig. 9.

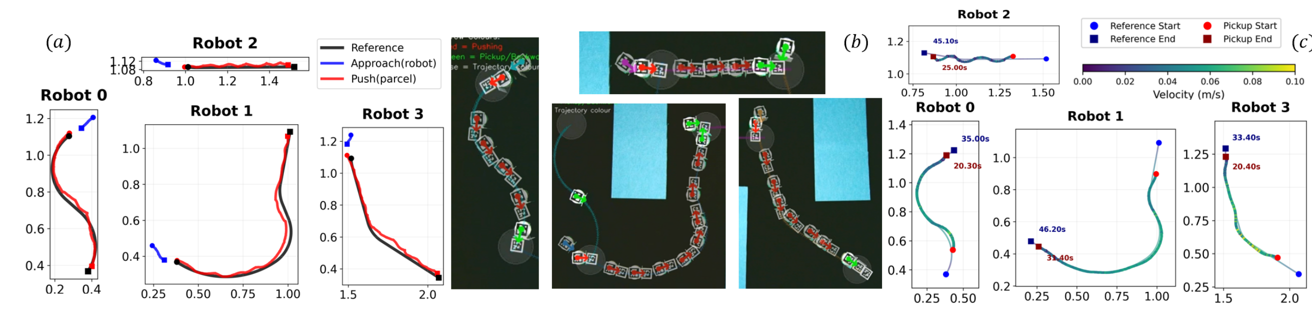

Fig. 13. The experiments result of Fig. 9 case of pushing parcel 1: (a) The pushing tajectory; (b) The snapshots of the pushing process; (c) The pickup trajectories, color-coded by velocity.

path generated by the Rapidly-exploring Random Tree (RRT) algorithm using Bézier curves (RRT+Bézier). The evaluation used minimum turning radius (r), path length (l), and collision status as metrics. As summarized in Table I, the proposed method consistently generated the shortest path length (highlighted in blue) while successfully satisfying all curvature and collision constraints. In contrast, the baseline methods frequently violated the minimum turning radius (values in red) or produced paths with collisions. Notably, in the more complex environments (cases (e) and (f)), the baselines failed to find a feasible path, underscoring the limitations of conventional methods in cluttered environments.

TABLE I
PERFORMANCE COMPARISON.

	Proposed			WPs+Bézier			RRT+Bézier		
Case	r	l	Col.	r	l	Col.	r	l	Col.
Fig. 9	0.2	1.74	No	0.35	3.76	No	0.11	2.88	No
Fig. 10 a	0.2	2.09	No	0.186	2.94	Yes	0.07	2.72	No
Fig. 10 b	0.2	2.46	No	0.51	3.05	Yes	0.36	2.63	No
Fig. 10 c	0.25	2.93	No	0.19	3.81	No	0.29	3.63	No
Fig. 10 e	0.2	6.46	No	0.13	7.50	Yes	-	-	-
Fig. 10 f	0.2	4.15	No	0.008	12.79	Yes	0.058	5.27	Yes

D. Limitations

The proposed method is subject to two primary limitations. First, the feasibility of the solution is sensitive to the specific map configuration and the number of robots. Given the strict manipulation constraints, a feaisble solution is not guaranteed to be found in all scenarios. Besides, the system's performance does not necessarily improve as more robots are added. Second, the computational scalability of the underlying MIP formulation is constrained by the capabilities of the solver, particularly for complex cases such as (e) and (f), which are prone to reaching the solver's limits.

VI. Conclusion

In this paper, we have presented a comprehensive framework for a multi-robot relay system designed to transport parcels through cluttered environments via a series of pushing manipulations. The framework encompasses the assignment of RPs, trajectory planning that adheres to kinematic constraints, and real-time synchronization during handovers. Experimental results validate the effectiveness of our approach in achieving efficient and reliable parcel delivery. Future work will focus on enhancing the robustness of the system and the multi-goal delivery tasks.

References

[1] X. Cao *et al.*, "Hma-sar: Multi-agent search and rescue for unknown located dynamic targets in completely unknown environments," *IEEE Robot. Autom. Lett.*, vol. 9, no. 6, pp. 5567–5574, 2024.

[2] M. Mishra, P. Poddar, R. Agrawal, J. Chen, P. Tokekar, and P. Sujit, "Multi-agent deep reinforcement learning for persistent monitoring with sensing, communication, and localization constraints," *IEEE Trans. Autom. Sci. Eng.*, vol. 22, pp. 2831–2843, 2024.

[3] Q. Shi, M. Liu, S. Zhang, and X. Lan, "Reinforcement learning for multi-agent path finding in large-scale warehouses via distributed policy evolution," *IEEE Robot. Autom. Lett.*, 2025.

[4] J. Zhao, K. Zhu, H. Hu, X. Yu, X. Li, and H. Wang, "Formation control of networked mobile robots with unknown reference orientation," *IEEE/ASME Trans. Mechatron.*, vol. 28, no. 4, pp. 2200–2212, 2023.

[5] S. Wang *et al.*, "An efficient distributed task allocation method for maximizing task allocations of multirobot systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 21, no. 3, pp. 3588–3602, 2023.

[6] Z. Feng *et al.*, "An overview of collaborative robotic manipulation in multi-robot systems," *Annu. Rev. Control*, vol. 49, pp. 113–127, 2020.

[7] S. Kim, H. Seo, J. Shin, and H. J. Kim, "Cooperative aerial manipulation using multirotors with multi-dof robotic arms," *IEEE/ASME Trans. Mechatron.*, vol. 23, no. 2, pp. 702–713, 2018.

[8] A. Skrynnik, A. Andreychuk, A. Borzilov, A. Chernyavskiy, K. Yakovlev, and A. Panov, "Pogema: A benchmark platform for cooperative multi-agent pathfinding," *arXiv:2407.14931*, 2024.

[9] Z. Amin *et al.*, "Relay network design with direct shipment and multi-relay assignment," *Ann. Oper. Res.*, vol. 328, no. 2, pp. 1585–1614, 2023.

[10] B. Ji, J. Huang, Y. Wang, K. Song, C. Li, C. Han, and H. Wen, "Multi-relay cognitive network with anti-fragile relay communication for intelligent transportation system under aggregated interference," *IEEE Trans. Intell. Veh.*, vol. 24, no. 7, pp. 7736–7745, 2023.

[11] F. Ruggiero *et al.*, "Nonprehensile dynamic manipulation: A survey," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1711–1718, 2018.

[12] G. Sun, R. Zhou, Z. Ma, Y. Li, R. Groß, Z. Chen, and S. Zhao, "Mean-shift exploration in shape assembly of robot swarms," *Nat. Commun.*, vol. 14, no. 1, p. 3476, 2023.

[13] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *Int. J. Robot. Res.*, vol. 36, no. 9, pp. 1000–1021, 2017.

[14] J. Zeng, A. M. Gimenez, E. Vinitsky, J. Alonso-Mora, and S. Sun, "Decentralized aerial manipulation of a cable-suspended load using multi-agent reinforcement learning," *arXiv:2508.01522*, 2025.

[15] C. Fan *et al.*, "Modal planning for cooperative non-prehensile manipulation by mobile robots," *Appl. Sci.*, vol. 9, no. 3, p. 462, 2019.

[16] F. Bertoncelli, M. Selvaggio, F. Ruggiero, and L. Sabattini, "Task-oriented contact optimization for pushing manipulation with mobile robots," in *2022 IEEE Int. Conf. Intell. Robots Syst.*, pp. 1639–1646.

[17] F. Bertoncelli and L. Sabattini, "Streamlining object pushing: Behavior tree-based coordination of control and planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 5203–5210.

[18] H. Üster *et al.*, "Strategic design and analysis of a relay network in truckload transportation," *Transp. Sci.*, vol. 45, no. 4, pp. 505–523, 2011.

[19] C. Li, X. Huang, J. Ding, K. Song, and S. Lu, "Global path planning based on a bidirectional alternating search a* algorithm for mobile robots," *Comput. Ind. Eng.*, vol. 168, p. 108123, 2022.

[20] S. Klemm *et al.*, "Rrt*-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE Int. Conf. Robot. Biomimetics (ROBIO)*. IEEE, pp. 1670–1677.

[21] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 478–485, 2020.

[22] R. Van Hoek, J. Ploeg, and H. Nijmeijer, "Cooperative driving of automated vehicles using b-splines for trajectory planning," *IEEE T. Intell. Veh.*, vol. 6, no. 3, pp. 594–604, 2021.

[23] Y. Lin and S. Saripalli, "Path planning using 3d dubins curve for unmanned aerial vehicles," in *2014 Int. Conf. Unmanned Aircr. Syst. (ICUAS)*. IEEE, pp. 296–304.

[24] J. Xie and N. Chakraborty, "Rigid body dynamic simulation with line and surface contact," in *2016 IEEE Int. Conf. Simul. Model. Program. Auton. Robots (SIMPAR)*. IEEE, pp. 9–15.

[25] S. H. Lee and M. Cutkosky, "Fixture planning with friction," 1991.

[26] S. Goyal *et al.*, "Planar sliding with dry friction part 1. limit surface and moment function," *Wear*, vol. 143, no. 2, pp. 307–330, 1991.

[27] G. Reeb, "Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique [on the singular points of a completely integrable pfaff form or of a numerical function]," *C. R. Acad. Sci. Paris*, vol. 222, pp. 847–849, 1946.

[28] L. He, H. Ishibuchi, A. Trivedi, H. Wang, Y. Nan, and D. Srinivasan, "A survey of normalization methods in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 25, pp. 1028–1048, 2021.

[29] L. Gurobi Optimization. The leader in decision intelligence technology - gurobi optimization. [Online]. Available: https://www.gurobi.com/

[30] S. Sivanandam, S. Deepa, S. Sivanandam, and S. Deepa, *Genetic algorithms*. Springer, 2008.

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[32] F. Bertoncelli, "Planar pushing: a study of non-prehensile manipulation with single and multiple mobile robots," Ph.D. dissertation, University of Modena and Reggio Emilia, 2022, https://hdl.handle.net/11380/1277918.

Hui Zhi has been pursuing a Ph.D. degree in mechanical engineering at The Hong Kong Polytechnic University, Kowloon, Hong Kong, since 2022. Her current research interests include multi-agent formation control, nonprehensile manipulation and motion planning.

Filippo Bertoncelli is a Senior Researcher at Autonomous Robotics Research Center, Technology Innovation Institute (TII), Abu Dhabi, UAE. He received the Ph.D. degree in industrial innovation engineering from the University of Modena and Reggio Emilia, Italy, in 2022. His research interests include multirobot systems, decentralized control, mobile robotics, multirobot cooperative manipulation, and non-prehensile manipulation.

Angelo Ferrando is an Assistant Professor at the University of Modena and Reggio Emilia. His work focuses on formal methods, runtime verification, and the engineering of reliable autonomous and multi-agent systems. His publications span formal verification, runtime monitoring, and agent-based software engineering.

Chenguang Yang received the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010. He is currently a Full Professor and the Leader of the Robot Teleoperation Group, Bristol Robotics Laboratory, Bristol, U.K. His research interests include human–robot interaction and intelligent system design.

Lorenzo Sabattini is Associate Professor at the University of Modena and Reggio Emilia, Italy. His main research interests include robot control, multi-robot systems, human-robot interaction, decentralized estimation and control, and physiological signal elaboration. He received the B.Sc. and M.Sc. degrees in mechatronic engineering from the University of Modena and Reggio Emilia, Italy, in 2005 and 2007, respectively, and the Ph.D. degree in control systems and operational research from the University of Bologna, Italy, in 2012.

David Navarro-Alarcon received the Ph.D. degree in mechanical and automation engineering from The Chinese University of Hong Kong, in 2014. Since 2017, he has been with The Hong Kong Polytechnic University, where he is currently an Associate Professor with the Department of Mechanical Engineering. His research interests include robotics and control systems. He currently serves as an Associate Editor of IEEE Transactions on Robotics.