Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

Homotopic Path Set Planning for Robot Manipulation and Navigation

Jing Huang[1,2], Yunxi Tang[1], and Kwok Wai Samuel Au[1,2]
[1]Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong
[2]Multi-Scale Medical Robotics Center, Hong Kong SAR
Email: {huangjing, yxtang}@mae.cuhk.edu.hk, samuelau@cuhk.edu.hk

***Abstract*—This paper addresses path set planning that yields important applications in robot manipulation and navigation such as path generation for deformable object keypoints and swarms. A path set refers to the collection of finite agent paths to represent the overall spatial path of a group of keypoints or a swarm, whose collective properties meet spatial and topological constraints. As opposed to planning a single path, simultaneously planning multiple paths with constraints poses nontrivial challenges in complex environments. This paper presents a systematic planning pipeline for homotopic path sets, a widely applicable path set class in robotics. An extended visibility check condition is first proposed to attain a sparse passage distribution amidst dense obstacles. Passage-aware optimal path planning compatible with sampling-based planners is then designed for single path planning with adjustable costs. Large accessible free space for path set accommodation can be achieved by the planned path while having a sufficiently short path length. After specifying the homotopic properties of path sets, path set generation based on deformable path transfer is proposed in an efficient centralized manner. The effectiveness of these methods is validated by extensive simulated and experimental results.**

I. Introduction

Simultaneously generating paths for multiple agents finds vast applications in robotics. For instance, planning multiple robots' paths with certain collective properties has been extensively studied in multi-robot navigation tasks like surveillance, formation flight, and collaborative transport [1]-[4]. Manipulative tasks also demonstrate similar needs. Particularly, robotic deformable object manipulation (DOM) remains challenging today. One core reason is that widely populated constrained environments in reality make DOM far more complicated beyond pure control approaches [6], [7] and entail planning [8], [9]. The reliance on models or simulations renders deformation planning not easily applicable. An effective alternative is directly planning spatial paths for deformable objects (DOs). Considering DO keypoints provides a tractable way to depict the object states [5]. Analogous to multi-robot paths, keypoint paths should be specified coordinately. For consistency, a path set here refers to the collection of paths for a robot team or object keypoints.

In comparison to path set planning, multi-trajectory planning is more commonly studied in multi-robot systems to impose time-dependent constraints. Usually cast to spatiotemporally constrained optimization problems, multi-trajectory planning poses high complexity. In contrast, path set planning decouples time to make a more basic and tractable problem.

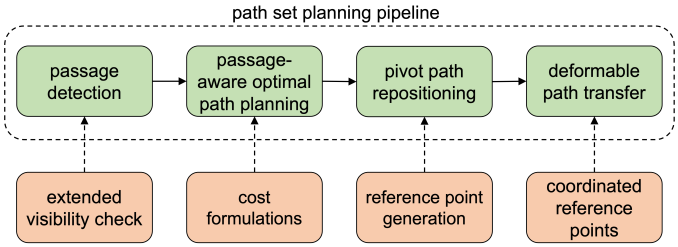


Fig. 1. Block diagram illustrating the overall workflow and main modules in the proposed path set planning pipeline. Blocks in the bottom row are the key components of corresponding modules.

Planned spatial paths can be converted to trajectories readily. This paper addresses path set planning, in particular a general class of homotopic path sets, and presents a systematic planning pipeline as shown in Fig. 1. In obstacle-dense environments, the obstacle distribution is first perceived by identifying valid passages that constrain agent motions. A novel passage identification criterion is proposed which drastically reduces the number of passages and subsequent computations over the original visibility condition. Then, passage-aware optimal path planning is utilized to find a single path trading off classical objectives, e.g., the path length, and free space along the path to accommodate multiple paths. After analyzing the feasible topological properties of path sets, a deformable path-transfer scheme is designed to efficiently generate coordinated path sets from a single agent path.

A recently reported study on path set planning in [5] only adopts DOM setups. In navigation, the concept of virtual tube in [2], [3] shows similarity to path sets in that it contains homotopic swarm trajectories, but standard planners and heavy optimization are relied on to attain trajectories. This paper extends and completes path set planning via a systematic and thorough investigation of core modules of the pipeline. The key contributions can be summarized as

1) A general passage check condition to detect sparse passage distributions in environments. It reduces the passage number dramatically over pure visibility check and helps save computations in planning stages.
2) New cost forms in passage-aware optimal path planning for adjustable planning results. Planners are enabled to optimally trade off path's accessible free space and length over conventional clearance-based methods.
3) Further refined path transfer procedure to better guarantee

transferred paths' feasibility and coordination. Path sets can be generated much more efficiently than approaches of separately planning.

Additionally, we demonstrate extensive simulation and experimental results as well as practical robotic applications to reveal the generality and applicability of our proposals.

II. Related Work

The related work draws equally from manipulation planning, mostly DOM planning, and multi-robot path planning literature. While DOs usually do not permit easily attainable state representations, a feasible path connecting the initial and target DO states can be planned with explicit deformation models. The DO state is usually characterized by extracted geometrical or topological properties. A survey on model-based DOM planning can be found in [8]. Models include the elasticity model [10], the mass-spring model [11], minimal-energy curves for deformable linear objects (DLOs) [12], to determine valid intermediate states, namely samples. Standard planning algorithms, e.g., Probabilistic Roadmap (PRM) and Rapidly-exploring Random Tree (RRT), are utilized to attain a path to the target state. Most works are DO/task-specific, e.g., DLOs, planar DOs, and clothes [13], [14], or ad-hoc elemental action paradigms like folding and bending [15]. The reliance on models or simulations severely hinders the utility of model-based planning in practice.

More relevant to this work is spatial DO path planning for a feasible path connecting initial and target DO configurations in complex environments. Using sampling-based planners, a nominal feasible path can be achieved with models/simulations [16], [17]. For better practicality, DO state prediction, motion planning, and control are interleaved in [18]. An increasing number of approaches unleash the potential of learning methods in DOM planning. In [19], the reliability of planned state-action pairs is enforced by learning a classifier of more reliably feasible state-pairs. The imagined plan, i.e., a sequence of images to the desired goal, is attained by learning a causal InfoGAN [20] of the deformation dynamics and planning in the latent space in [21]. More recently, path sets of feedback points are leveraged as DO motion references in DOM [5], but path set generation and passage processing are simplified, leading to restrictive generality.

Multi-robot path planning is a basic problem when multiple robots are present and has received substantial research in the robotics and AI community, formally named the multi-agent path finding (MAPF) problem. It aims to find feasible paths for all agents while optimizing objectives of the makespan or the sum of individual path costs [22], [23]. Numerous MAPF algorithms are proposed on common abstractions like ignoring agents' kinematics constraints and using discrete grid graphs. Despite the NP-hardness of MAPF [24], sub-optimal solvers, including search-based solvers like hierarchical cooperative A* and its variants [25], [26] as well as rule-based solvers [27], can quickly find all agent paths. Some optimal solvers reduce the problem to standard and more tractable ones, e.g., integer linear programming [28], answer set programming [29] and satisfiability solving [30]. M* dynamically changes the dimensionality and branching factors upon detected conflicts [31]. Conflict-based search adopts a two-level structure to enable fewer state examinations [32]. Other methods such as increasing cost tree search [33] also exist and a comprehensive survey on MAPF is available in [22].

In robotics, realistic conditions in ground and aerial swarms usually render the above methods less applicable. Trajectory planning, optimization, and local motion planning are often jointly considered [1]-[4]. Mix-integer quadratic optimization is formulated for multiple trajectories passing specified intermediate waypoints [34]. Discretized linear temporal logic is employed to depict robot groups' desired behaviors and trajectories are solved as a problem of satisfiability modulo theories [35]. An important alternative for global path planning is sampling-based methods that define the path by a set of safe team configurations. For instance, PRM is used for team formation in [36], [37]. Sampling-based strategies in multi-robot path planning usually need to compute cell decomposition of environments to recognize traversable areas [1], [38]. Recently in [2], [3], the virtual tube is proposed to generate infinite homotopic optimal trajectories efficiently via convex combinations of several optimal vertex trajectories. The tube's topological properties are decided by vertex paths found by RRT* (optimal RRT) that minimize the path length, but tube's accessible free space is not optimized.

III. Passage Detection and Passage-Aware Optimal Path Planning

This section elaborates on prerequisites for path set planning. After a brief problem statement, the extended visibility check condition in passage detection is proposed. Passage-aware optimal path planning is then presented.

A. Homotopic Path Set Planning Problem

Consider a team of K agents $S = [\mathbf{s}_1^\mathsf{T}, ..., \mathbf{s}_K^\mathsf{T}]^\mathsf{T}$ with $\mathbf{s}_i \in \mathbb{R}^3$ denoting the i-th agent's position. In manipulation, the team can be the group of keypoints picked on DOs for deformation description. The *path set*, i.e., the collection of agent paths, is employed as the path embodiment of the team. Given the initial $S_0 = [\mathbf{s}_{0,1}^\mathsf{T}, ..., \mathbf{s}_{0,K}^\mathsf{T}]^\mathsf{T}$ and target $S_d = [\mathbf{s}_{d,1}^\mathsf{T}, ..., \mathbf{s}_{d,K}^\mathsf{T}]^\mathsf{T}$, the aim is to find the path set to encode the team's spatial path in complex obstacle-dense environments. Apart from individual path's properties, the path set needs to fulfill certain constraints as a whole. In particular, we target homotopic path sets in which all paths are homotopic. Denote σ_i the path of $\mathbf{s}_i$, then the path set Σ_S satisfies

$$\Sigma_S = \{\sigma_1, \sigma_2, ..., \sigma_K\},\ \mathcal{H}(\sigma_i, \sigma_j) = \text{True}\ \forall \sigma_i, \sigma_j \quad (1)$$

where $\mathcal{H}(\cdot)$ checks path homotopy. Meanwhile, Σ_S is required to occupy a large free space in complex environments for multiple path accommodation and have short paths as detailed later. Intuitively, homotopic path sets do not allow agents to be split apart by obstacles, a widely applicable homotopy path class in DOM [5] and robot navigation [1], [39], [40], and are also readily extendable to general cases by allowing multiple homotopic path sets [3], [41].

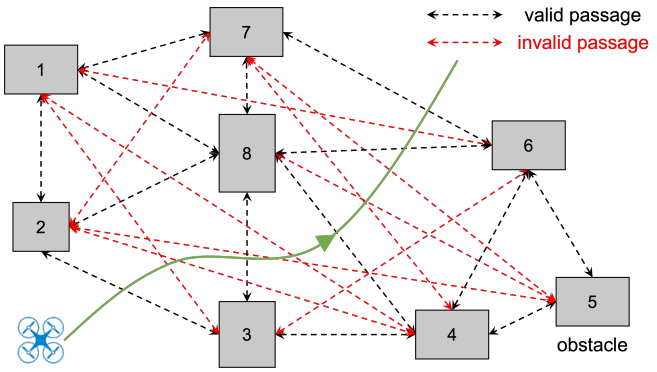


Fig. 2. For simplicity, segments connecting obstacle side centers represent passages formed by two obstacles here. All passages pass the visibility check, but only black ones are useful in free space determination.

B. Extended Visibility Check in Passage Detection

Narrow space easily causes collisions in team motion and manipulation. Thus, sufficient free space along the path set represents a fundamental requirement. As paths are homotopic, it suffices to resort to one agent path for a large free space. An effective way to gauge accessible free space along a path is by checking traversed passages. Unlike passage detection based on computationally intensive bridge tests [42], [43], obstacle distribution is exploited for fast detection. Assume obstacles are separate polyhedrons $\mathcal{E}_i (i = 1, ..., M)$. Each ordered pair $\mathcal{E}_i, \mathcal{E}_j$ $(i < j)$ forms a generic passage denoted as $(\mathcal{E}_i, \mathcal{E}_j)$. A total of $\binom{M}{2}$ such passages exist, but not all are physically valid. The pruning strategy of visibility check is used in [5]. $(\mathcal{E}_i, \mathcal{E}_j)$ is classified as valid only if the passage segment is collision-free with other obstacles. Though visually invalid passages are excluded, a significant downside is that it leaves a large fraction of passages invalid for free space determination in obstacle-dense environments, e.g., Fig. 2.

Redundant passages will incur a large computational load in following passage-related procedures in path set planning. To address this, an extended visibility check condition is proposed here to enable a thorough passage check. The shortest segment between obstacles is leveraged as a compact representation of the passage, i.e.,

$$\begin{aligned} (\mathcal{E}_i, \mathcal{E}_j) &= l(\mathbf{p}_i^*, \mathbf{p}_j^*) \\ \text{s. t. } (\mathbf{p}_i^*, \mathbf{p}_j^*) &= \underset{\mathbf{p}_i \in \mathcal{E}_i, \mathbf{p}_j \in \mathcal{E}_j}{\arg\min} \|\mathbf{p}_i - \mathbf{p}_j\|_2 \end{aligned} \tag{2}$$

where $l(\cdot) \subset \mathbb{R}^3$ is the segment connecting two points. In pure visibility check, $\mathcal{V}(\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k)$ returns true if $(\mathcal{E}_i, \mathcal{E}_j)$ is not occluded by $\mathcal{E}_k$ and false otherwise

$$\mathcal{V}(\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k) = \text{False} \;\text{ if }\; \mathcal{E}_k \cap l(\mathbf{p}_i^*, \mathbf{p}_j^*) \neq \emptyset. \tag{3}$$

$(\mathcal{E}_i, \mathcal{E}_j)$ is evaluated as a passage if $\mathcal{V}(\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k)$ is true for all $\mathcal{E}_k, k \neq i, j$. This condition, however, is overly restrictive to filter out invalid passages. For instance, $(\mathcal{E}_2, \mathcal{E}_4)$ in Fig. 2 is marked as a passage under the visibility criterion. Nonetheless, suppose an agent with a certain volume is passing $(\mathcal{E}_2, \mathcal{E}_4)$, its motion is more directly restricted by nearby obstacles $\mathcal{E}_3$ and $\mathcal{E}_8$ in passages $(\mathcal{E}_2, \mathcal{E}_3)$ and $(\mathcal{E}_3, \mathcal{E}_8)$.

Taking the perspective of an agent with isotropic motion directions, $(\mathcal{E}_i, \mathcal{E}_j)$ constrains the agent in a circular area $\mathcal{R}_{i,j}$.

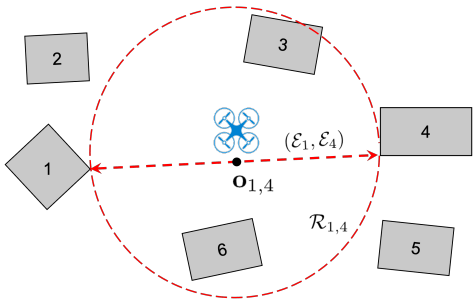


Fig. 3. $(\mathcal{E}_1, \mathcal{E}_4)$ will pass the original visibility check, but cannot pass the extended variant because both $\mathcal{E}_3$ and $\mathcal{E}_6$ intersect $\mathcal{R}_{1,3}$.

The region center $\mathbf{o}_{i,j}$ coincides with the passage center and the diameter $2r_{i,j}$ equals the passage width $\|(\mathcal{E}_i, \mathcal{E}_j)\|_2$. If any other $\mathcal{E}_k$ intersects $\mathcal{R}_{i,j}$, the free space shrinks as exemplified in Fig. 3 for a drone. $(\mathcal{E}_i, \mathcal{E}_j)$ should be classified as invalid since the agent is more directly confined by $\mathcal{E}_k$ when it is in $(\mathcal{E}_i, \mathcal{E}_j)$. This criterion naturally involves the original visibility condition as an extended variant and can be expressed as

$$\mathcal{V}(\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k) = \text{False} \;\text{ if }\; \mathcal{E}_k \cap \mathcal{R}_{i,j} \neq \emptyset. \tag{4}$$

This infers that an agent in cluttered environments is more likely to be blocked by obstacles in its vicinity than obstacles forming the generic passage it is traversing. (4) can also be interpreted by Voronoi diagrams that partition the environment by clearance to obstacles [44], [45], [46]. Specifically, (4) imposes that $(\mathcal{E}_i, \mathcal{E}_j)$ is valid if it only goes through the Voronoi cells associated with $\mathcal{E}_i$ and $\mathcal{E}_j$. In this way, redundant passages are thoroughly excluded, which helps save computations in following passage-related procedures. In 3D maps, passage detection can be performed in height intervals determined by obstacles to maintain a sparse passage structure.

C. Passage-Aware Optimal Path Planning

Passage determination preprocesses the environment before path planning. For each $\mathbf{s}_i \in S$, the path from its start $\mathbf{s}_{0,i}$ to the target $\mathbf{s}_{d,i}$ optimizing a user-defined cost function is obtainable by optimal planers such as sampling-based methods [47]-[49]. Formally, a feasible path is a continuous function $\sigma : [0, 1] \mapsto \mathcal{X}_{free}$ where $\mathcal{X}_{free}$ is the obstacle-free configuration space. The path argument $\tau \in [0, 1]$ is given by path length parameterization by default. As aforementioned, apart from the typical criterion of path length, one core requirement for an agent path is to optimize the accessible free space for the path set. The passages passed by σ from the start $\sigma(0)$ to $\sigma(\tau)$ are stored in an ordered list $P_\sigma(\tau) = \{(\mathcal{E}_i, \mathcal{E}_j), ..., (\mathcal{E}_p, \mathcal{E}_q)\}$. $P_\sigma(\tau, i)$ indexes the i-th passage in $P_\sigma(\tau)$. $\min \|P_\sigma(\tau)\|_2$ returns the minimum passage width in $P_\sigma(\tau)$.

Optimal planners asymptotically find the path that optimizes some properly defined cost. A well-formulated cost is therefore essential to depict the aforementioned planning requirements. However, it is not straightforward since these requirements may be inconsistent and conflicting. A path minimizing the path length need not have sufficient free space along it and vice versa. The cost to trade off them in [5] is

$$f(\sigma) = \text{Len}(\sigma)/f_P(\sigma) \tag{5}$$

Algorithm 1: Update Cost of $\mathbf{s}_{new}$ in A New Edge

Input $\mathbf{s}_{near} \in S_{near}, \mathbf{s}_{new}, P_{valid}, E$;
foreach $(\mathcal{E}_i, \mathcal{E}_j) \in P_{valid}$ **do**
  if $edge(\mathbf{s}_{near}, \mathbf{s}_{new})$ passes $(\mathcal{E}_i, \mathcal{E}_j)$ **then**
    $\sigma' \leftarrow \sigma^*_{near} * edge(\mathbf{s}_{near}, \mathbf{s}_{new})$;
    Compute $f(\sigma'), f_P(\sigma')$;
    if $f(\sigma_{temp}) < f(\sigma^*_{new})$ **then**
      Update $f(\sigma^*_{new}), f_P(\sigma^*_{new})$ as values of σ';
      Update $f_{cur}(\mathbf{s}_{new})$ and parent of $\mathbf{s}_{new}$;
      $E \leftarrow (E \setminus \{edge(\mathbf{s}_{parent}, \mathbf{s}_{new})\}) \cup \{edge(\mathbf{s}_{near}, \mathbf{s}_{new})\}$;

where Len(σ) is the path length. $f_P(\sigma) = \min \|P_\sigma(1)\|_2$ is the minimum passage width passed by σ. Proportional comparison between Len(σ) and $f_P(\sigma)$ is adopted in (5). The composite cost tends to minimize the path length while maximizing the minimum passage width the path passes.

Terms' priorities in (5) are fixed. Adjustable costs are more desirable to enable different path preferences considering the free space requirement varies with problem setups such as the team size and the obstacle density. A weighted cost structure is introduced herein as

$$f(\sigma) = \text{Len}(\sigma) - k_P f_P(\sigma) \tag{6}$$

where $k_P > 0$ acts as the weight of $f_P(\sigma)$ that determines the importance of $f(\sigma)$. Intuitively, the passage width is converted into a generalized and weighted path length subtracted from the true path length in (6). By taking different k_P, the dominance between Len(σ) and f_P changes. k_P selection is problem-specific. In most scenarios, $\|P_\sigma(1)\|_2$ is significantly smaller than Len(σ), k_P should not be too small to bring out the effect of $f_P(\sigma)$ in the cost.

The cost formulations (6) are monotonic under path concatenation (Len($\cdot$) is monotonically increasing, $f_P(\cdot)$ is monotonically non-increasing) and bounded. Therefore, sampling-based optimal planners are guaranteed to find the optimal path asymptotically [48] and RRT* is taken in our implementation. Denote σ^*_{new} the optimal path from the start $\mathbf{s}_{init}$ to the new sample $\mathbf{s}_{new}$. $\mathbf{s}_{new}$ carries attributes of $f(\sigma^*_{new})$, $f_P(\sigma^*_{new})$ and $f_{cur}(\mathbf{s}_{new})$, the passage width passed by the edge from the parent node to $\mathbf{s}_{new}$. $f_P(\sigma^*_{new})$ and $f_{cur}(\mathbf{s}_{new})$ are initialized as large values to indicate no passing of passages. To iteratively find σ^*_{new}, passage passing is checked in every attempt to add the edge between $\mathbf{s}_{new}$ and a near node $\mathbf{s}_{near}$. Attributes are updated accordingly (see Algorithm 1). This procedure is invoked when finding the parent node and rewiring the tree, i.e., `GetParent()` and `Rewire()`. See Algorithm 2 for passage-aware optimal path planning for a single path that integrates the extended visibility check for passage detection and the new cost formulation.

IV. Path Set Homotopy and Path Transfer

This section first discusses the feasibility requirements for path sets regarding paths' homotopic properties. Path transfer is then introduced as the basic primitive for path set generation.

Algorithm 2: RRT*-Based Passage-Aware Optimal Path Planning

$P_{valid} \leftarrow$ `ExtendedVisibilityCheck`$(\mathcal{E}_1, ..., \mathcal{E}_M)$;
$V \leftarrow \{\mathbf{s}_{init}\}$; $E \leftarrow \emptyset$;
for $i = 1, 2, ..., N$ **do**
  $\mathbf{s}_{rand} \leftarrow$ `SampleFree`(δ);
  $\mathbf{s}_{nearest} \leftarrow$ `Nearest`$(G = (V, E), \mathbf{s}_{rand})$;
  $\mathbf{s}_{new} \leftarrow$ `Steer`$(\mathbf{s}_{nearest}, \mathbf{s}_{rand})$;
  if `ObstacleFree`$(\mathbf{s}_{nearest}, \mathbf{s}_{new})$ **then**
    $S_{near} \leftarrow$ `Near`$(G = (V, E), \mathbf{s}_{new}, r_{near})$;
    $V \leftarrow V \cup \{\mathbf{s}_{new}\}$;
    $\mathbf{s}_{min} \leftarrow$ `GetParent`$(S_{near}, \mathbf{s}_{new}, P_{valid})$;
    $E \leftarrow E \cup \{edge(\mathbf{s}_{min}, \mathbf{s}_{new})\}$;
    `Rewire`$(G = (V, E), S_{near}, \mathbf{s}_{min}, \mathbf{s}_{new}, P_{valid})$;
return $G = (V, E)$;

A. Feasibility Requirement for Path Sets in Homotopy

For the feasibility of the path set Σ_S as a collection, paths' homotopic interrelationships are analyzed. In general non-winding scenarios where agents do not wrap around obstacles, the homotopy constraint is imposed on paths. σ_1, σ_2 with identical initial and final positions are path homotopic if there exists a continuous function, i.e., the homotopy, $\psi(\cdot) : [0, 1] \mapsto \Sigma_{free}$, where Σ_{free} is the set of paths in $\mathcal{X}_{free}$, such that $\psi(0) = \sigma_1, \psi(1) = \sigma_2$ and $\psi(x) \in \Sigma_{free}, \forall x \in [0, 1]$. Homotopic paths essentially can continuously deform to one another in $\mathcal{X}_{free}$ but are not easily verifiable in general. For easy verifiability and good generality, straight-line homotopy is imposed on paths. If σ_1, σ_2 are path homotopic, their straight-line homotopy is

$$\psi_{1,2}(x, \tau) = (1 - x)\sigma_1(\tau) + x\sigma_2(\tau), \quad x, \tau \in [0, 1]. \tag{7}$$

σ_1, σ_2 are said to be strong path homotopic if $\psi_{1,2}(x, \tau) \in \mathcal{X}_{free}$ always holds, indicating that the hypersurface swept by $\psi_{1,2}(x, \tau)$ lies in $\mathcal{X}_{free}$. We do not construct strictly homotopic paths with the same endpoints as in [50]. $\sigma_i, \sigma_j \in \Sigma_S$ are said to be strong path homotopic-like if their straight-line homotopy in (7) remains in Σ_{free}.

Any pair of paths in Σ_S are required to be strong path homotopic-like as in [5], equivalent to the uniform visibility condition for quadrotor paths [39], [40], line of sight [50], and visibility deformation of roadmaps [41]. Since agents are not separated by obstacles, paths are homotopic to transform into each other, which is hard to check in high-dimensional spaces. The strong homotopic-like condition is easy to check, although it can be overly constrained in 3D space to exclude feasible situations where paths round obstacles like Fig. 4. This can be further checked by examining if the straight-line homotopy passes the entire obstacle top part.

B. Path Transfer

Generating Σ_S by planning each agent path in a decentralized fashion is complex due to the homotopy constraint.

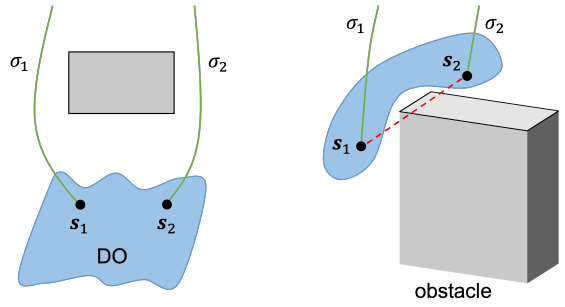


Fig. 4. In the left 2D case, DO point path 1 and 2 are not homotopic and thus infeasible. In the right 3D case, though path 1 and 2 are not strong path homotopic-like, the shown DO pose is feasible.

Coordination among paths is also hard to achieve. Paths in a homotopic path set share an identical passage passing list in 2D, i.e., $P_{\sigma_i} = P_{\sigma_j}, \forall\, \sigma_i, \sigma_j \in \Sigma_S$, or have limited differences in 3D. Therefore, if one path in Σ_S is planned, other paths can be generated by transferring from it. This flow fulfills homotopy by construction and is computationally efficient. S_0 and S_d are essentially two clusters with different agent distributions. A path set from S_0 to S_d connects the two clusters while meeting the above constraints. Suppose $\mathbf{s}_p \in S$ has a planned path σ_p from $\mathbf{s}_{0,p}$ to $\mathbf{s}_{d,p}$, then for each point $\mathbf{s}_i \in S$, the path transferred from σ_p is

$$\sigma_{t,i}(\tau_i) = \sigma_p(\tau_p) + \mathbf{v}_{p\to i}(\tau_p, \tau_i) \tag{8}$$

where a general path transfer form is utilized compared to [5]. It permits different path arguments in σ_p, $\sigma_{t,i}$, and a varying transfer vector to enable more flexible transfer.

The planned path σ_p in path transfer is conceptually similar to the generator curve in [3]. The corresponding agent, termed *pivot*, can be picked arbitrarily since following transfer procedures are invariant to pivot. Designate p the chosen pivot index. The pivot path σ_p is found by the passage-aware optimal path planner. When transferring σ_p to other agents, forward and backward transfer are proposed in [5] by assigning $\mathbf{v}_{p\to i}$ as $\mathbf{v}_{0,p\to i} = \mathbf{s}_{0,i} - \mathbf{s}_{0,p}$ and $\mathbf{v}_{d,p\to i} = \mathbf{s}_{d,i} - \mathbf{s}_{d,p}$, respectively, where extra postprocessing of path concatenation is required. To resolve this, these two transfer paradigms are combined in (8) to be

$$\sigma_{t,i}(\tau) = \sigma_p(\tau) + (1-\tau)\mathbf{v}_{0,p\to i} + \tau \mathbf{v}_{d,p\to i} \tag{9}$$

where $\tau_i = \tau_p = \tau$, $\mathbf{v}_{0,p\to i}$ and $\mathbf{v}_{d,p\to i}$ are linearly interpolated to compose a varying transfer vector along $\sigma_{t,i}$. In this way, $\sigma_{t,i}(0) = \mathbf{s}_{0,i}$, $\sigma_{t,i}(1) = \mathbf{s}_{d,i}$ with no need for path postprocessing. Performing (9) for each agent in S leads to a transferred path set $\Sigma_t(S_0, S_d, \sigma_p)$. Σ_t is strong homotopic-like if the hyperplane swept by the varying transfer vector $\mathbf{v}_{p\to i}$ keeps collision-free. As constrained environments are present, such an assumption usually fails, which entails refined processing to reach the final path set.

V. Path Set Generation Using Path Transfer

In this section, a path set generation scheme for constrained environments is proposed incorporating two refined steps on [5]: 1) pivot path planning and repositioning, and 2) coordinated deformable path transfer.

A. Pivot Path Planning and Repositioning

1) Repositioning Reference Points Determination: Before planning the pivot path σ_p, a pivot selection criterion minimizing the transfer vector magnitude is introduced as

$$p = \arg\min_{1\le i\le K} \max_{1\le j\le K} \left(\|\mathbf{s}_{0,i} - \mathbf{s}_{0,j}\|_2, \|\mathbf{s}_{d,i} - \mathbf{s}_{d,j}\|_2\right) \tag{10}$$

which limits path transfer to a tunnel centered at σ_p with a radius as small as possible. In Σ_t, transferred paths can percolate obstacles easily since σ_p is often close to obstacles to reduce its length in (5) or (6). σ_p thus needs to be repositioned to a more reasonable configuration while preserving $P_{\sigma_p}(1)$. After planning σ_p, Σ_t is obtained for checking passage intersections. Usually, it is sufficient to consider obstacles in $P_{\sigma_p}(1)$, but this may miss some nearby obstacles. For completeness, a distance filter is first applied to register obstacles near σ_p. Denote the distance between σ_p and $\mathcal{E}_i$ as

$$d(\sigma_p, \mathcal{E}_i) = \min_{\tau\in[0,1], \mathbf{p}_i\in\mathcal{E}_i} \|\sigma_p(\tau) - \mathbf{p}_i\|_2. \tag{11}$$

A threshold $\lambda = \max_{1\le i\le K} \max(\|\mathbf{s}_{0,i} - \mathbf{s}_{0,p}\|_2, \|\mathbf{s}_{d,i} - \mathbf{s}_{d,p}\|_2)$ is set to rule out obstacles not in σ_p's vicinity. The remaining obstacles are divided into two categories: obstacles in passages traversed by σ_p, termed passage obstacles, and isolated obstacles otherwise. Formally, define $\mathcal{E}_{near} = \{\mathcal{E}_i \,|\, d(\sigma_p, \mathcal{E}_i) \le \lambda\}$, $\mathcal{E}_P = \{\mathcal{E}_i \,|\, (\mathcal{E}_i, \mathcal{E}_j) \in P_{\sigma_p}(1) \text{ or } (\mathcal{E}_j, \mathcal{E}_i) \in P_{\sigma_p}(1)\, \exists\, \mathcal{E}_j\}$. Passage obstacles are $\mathcal{E}_P^{near} = \mathcal{E}_{near} \cap \mathcal{E}_P$. Isolated obstacles are $\mathcal{E}_{iso} = \mathcal{E}_{near} \setminus \mathcal{E}_P^{near}$.

The relative position of σ_p to nearby obstacles determines how σ_p should be locally repositioned. The passage list $P_{\sigma_p}(1)$ is updated to only contain passages with obstacles in $\mathcal{E}_{near}$. $P_{\sigma_p}(1,i) = (\mathcal{E}_j, \mathcal{E}_k)$ is discarded if both $\mathcal{E}_j$ and $\mathcal{E}_k$ are not in $\mathcal{E}_{near}$. The intersection segment between Σ_t and $P_{\sigma_p}(1,i)$ is characterized by the chord, denoted as $\Sigma_t \cap P'_{\sigma_p}(1,i)$, with its length being

$$\|\Sigma_t \cap P'_{\sigma_p}(1,i)\|_2 = \max_{1\le k,j\le K} \|\sigma_{t,k}(\eta_{k,i}) - \sigma_{t,j}(\eta_{j,i})\|_2 \tag{12}$$

where $P'_{\sigma_p}(1,i)$ signifies the entire straight line on which $P_{\sigma_p}(1,i)$ lies. $\sigma_{t,k}(\eta_{k,i})$ is the intersection point between $\sigma_{t,k}$ and $P'_{\sigma_p}(1,i)$. The ordered intersection points between σ_p and $P_{\sigma_p}(1)$, $\{\sigma_p(\eta_{p,1}), \sigma_p(\eta_{p,2}), ..., \sigma_p(\eta_{p,n})\}$, constitute reference points of σ_p. Two overlapping possibilities between $\Sigma_t \cap P'_{\sigma_p}(1,i)$ and $P_{\sigma_p}(1,i)$ exist: $\Sigma_t \cap P'_{\sigma_p}(1,i)$ is completely contained in $P_{\sigma_p}(1,i)$ or otherwise. If $\Sigma_t \cap P'_{\sigma_p}(1,i)$ falls inside $P_{\sigma_p}(1,i)$, the intersection point $\sigma_p(\eta_{p,i})$ just remains unchanged for anchoring. In situations otherwise, the chord is in collision and σ_p should be modified locally. If $\|\Sigma_t \cap P'_{\sigma_p}(1,i)\|_2 \le \|P_{\sigma_p}(1,i)\|_2$, $P_\sigma(1,i)$ is sufficiently wide. σ_p can be simply translated along $P_{\sigma_p}(1,i)$ to move the chord into $P_{\sigma_p}(1,i)$. Denote $\sigma_p^*(\eta_{p,i})$ the adjusted reference point, it can be given as

$$\sigma_p^*(\eta_{p,i}) = \sigma_p(\eta_{p,i}) + \mathbf{d}_{\underline{\delta},i} \tag{13}$$

where $\mathbf{d}_{\underline{\delta},i}$ is the translation along $P_{\sigma_p}(1,i)$. Note that there is one chord end outside $P_{\sigma_p}(1,i)$, be it $\sigma_q(\eta_{q,i})$. After translation, the end is moved to a point $\mathbf{p}_{\underline{\delta},i}$ on $P_{\sigma_p}(1,i)$ with a preset clearance $\underline{\delta}$ to obstacles. The shift is $\mathbf{d}_{\underline{\delta},i} = \mathbf{p}_{\underline{\delta},i} - \sigma_q(\eta_{q,i})$ as illustrated in Fig. 5 and Fig. 6.

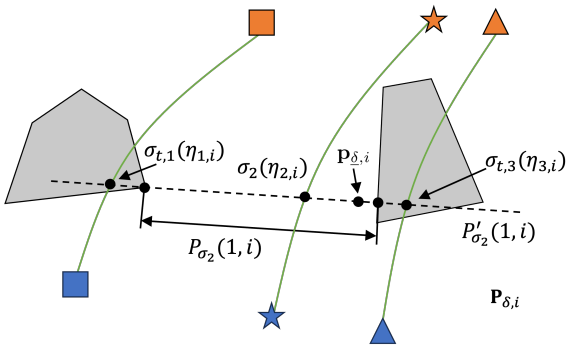


Fig. 5. In the shown passage, σ_2 is the pivot path. $\sigma_{t,1}, \sigma_{t,3}$ are transferred from σ_2 and both collide with obstacles. To tackle this, σ_2 is repositioned first. $\sigma_{t,1}, \sigma_{t,3}$ are then locally deformed.

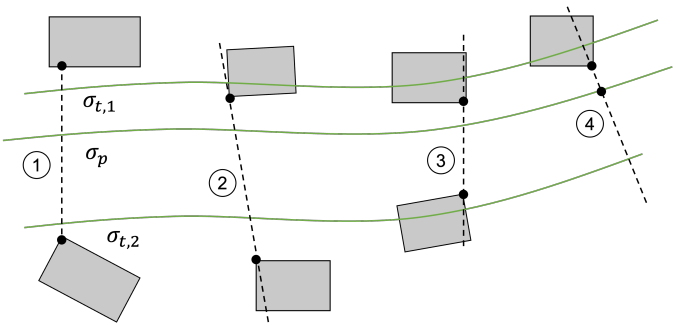


Fig. 6. Different situations when attaining reference points for the pivot path σ_p. 1. No need for repositioning. 2. Translate σ_p along the passage segment. 3. Reposition σ_p to proportionally compress the chord. 4. Push σ_p away from an isolated obstacle.

$\|\Sigma_t \cap P'_{\sigma_p}(1,i)\|_2 > \|P_{\sigma_p}(1,i)\|_2$ is more constrained. The chord cannot be placed within the passage segment by translation. Repositioning σ_p now aims for a coordinated distribution of all transferred paths. The strategy to overlap centers of $\Sigma_t \cap P'_{\sigma_p}(1,i)$ and $P_{\sigma_p}(1,i)$ is utilized in [5]. Its problem is that it may cause extremely unbalanced path distributions. To avoid this, proportionally placing intersection points on $P_{\sigma_p}(1,i)$ according to the relative distribution of $\{\sigma_{t,1}(\eta_{1,i}), \sigma_{t,2}(\eta_{2,i}), ..., \sigma_{t,K}(\eta_{K,i})\}$ provides more reasonable reference points. $\sigma_p^*(\eta_{p,i})$ now is

$$\sigma_p^*(\eta_{p,i}) = \mathbf{p}_{\underline{\delta},i} + r_i(\sigma_p(\eta_{p,i}) - \sigma_q(\eta_{q,i})) \tag{14}$$

where $r_i = (\|P_{\sigma_p}(1,i)\|_2 - 2\underline{\delta})/\|\Sigma_t \cap P'_{\sigma_p}(1,i)\|_2$ is the scaling ratio. $\mathbf{p}_{\underline{\delta},i}$ is the new chord end. As $\|\Sigma_t \cap P'_{\sigma_p}(1,i)\|_2 > \|P_{\sigma_p}(1,i)\|_2$, at least one chord end is outside of $P_{\sigma_p}(1,i)$ to be taken as $\sigma_q(\eta_{q,i})$. For isolated obstacles, we only consider the case in which they collide with some path or lie between paths. As such, $\sigma_p(\tau_{\mathcal{E}_i})$ is translated along the direction of $\sigma_p(\tau_{\mathcal{E}_i}) - \mathbf{p}_i^*$, where $\sigma_p(\tau_{\mathcal{E}_i})$ is the minimum distance projection of $\mathcal{E}_i$ on σ_p in (11). $\mathbf{p}_i^*$ is the optimal point on $\mathcal{E}_i$. This is analogous to (13) by replacing $P'_{\sigma_p}(1,i)$ with the line of $\sigma_p(\tau_{\mathcal{E}_i}) - \mathbf{p}_i^*$.

2) Repositioning Pivot Path: A series of reference points have been obtained along σ_p. σ_p is repositioned iteratively between every two consecutive reference points in a linear interpolation manner. For $\eta_{p,i} \le \tau \le \eta_{p,i+1}$, the new path position $\sigma_p^*(\tau)$ is given as

$$\begin{aligned}\sigma_p^*(\tau) = \sigma_p(\tau) &+ \frac{\tau - \eta_{p,i}}{\eta_{p,i+1} - \eta_{p,i}}(\sigma_p^*(\eta_{p,i+1}) - \sigma_p(\eta_{p,i+1})) \\ &+ \frac{\eta_{p,i+1} - \tau}{\eta_{p,i+1} - \eta_{p,i}}(\sigma_p^*(\eta_{p,i}) - \sigma_p(\eta_{p,i})).\end{aligned} \tag{15}$$

σ_p^* is the repositioned σ_p. The start and final points of σ_p need to be inserted to establish an augmented list of reference points $\{\sigma_p(0), \sigma_p^*(\eta_{p,1}), ..., \sigma_p^*(\eta_{p,n}), \sigma_p(1)\}$. Since the shift magnitude on a passage segment is small compared to the total path length, the path segment in (15) is feasible in general. As for the repositioning procedure in 3D space, the chord evolves into the convex hull formed by intersection points with the passage plane. Without considering floating obstacles, the shift of $\sigma_p(\eta_{p,i})$ is restricted in the direction parallel to the ground and rules are set similarly to (13) and (14). One inherent problem of chords is that they may poorly represent how the entire path set passes through the passage and lead to nonsmooth path segments, e.g., when the passage is nearly parallel to the local path. Inspired by the virtual tube [2], [3], we propose a geometrical approach for chord determination, which uses the normal direction of the pivot path to get the chord and then rotates it back to the passage segment. Details are given in Appendix A.

B. Coordinated Deformable Path Transfer

After attaining σ_p^*, the next step generates rest agent paths. Although σ_p^* gets further optimized based on σ_p, directly transferred paths from σ_p^* via (9) may still be infeasible in constrained environments. To effectively convert a locally infeasible path into a feasible one, a path-deforming scheme similar to the path-guided optimization avenue in [39], [40] is leveraged. The core idea is to use proper reference points to tailor infeasible path parts to feasible paths through path deformation. Firstly, a new path set Σ_t^* is regenerated by transferring σ_p^* to rest agents as (9). Passage intersection check is conducted as before. If adjustment is needed, the reference point for each transferred path is given as

$$\sigma_{t,j}^*(\eta_{j,i}) = \sigma_p^*(\eta_{p,i}) + r_i(\sigma_{t,j}(\eta_{j,i}) - \sigma_p^*(\eta_{p,i})) \tag{16}$$

where $\sigma_{t,j}^*(\eta_{j,i})$ is the fixed reference point. The intersection point of the repositioned σ_p^* is fixed as the reference point. This follows the proportional distribution in (14) to compress the chord. With all these reference points, each transferred path is deformed in the same manner as (15).

The deformed transferred path $\sigma_{t,j}^*$ need not be collision-free because obstacle sizes may not be negligible in practice. The key to addressing this is providing more reference points near narrow passages than those on the passage segments. This can be achieved by introducing translated passage segments on obstacle vertices as shown in Fig. 16 and more details are provided in Appendix B. The overall path set generation pipeline is outlined in Algorithm 3. The completeness of the entire scheme is ensured by the completeness of the optimal planner backbone for pivot path planning. Only linear time complexity w.r.t. the discrete path node number exists to obtain a transferred path and the resulting paths are strong path homotopic by construction.

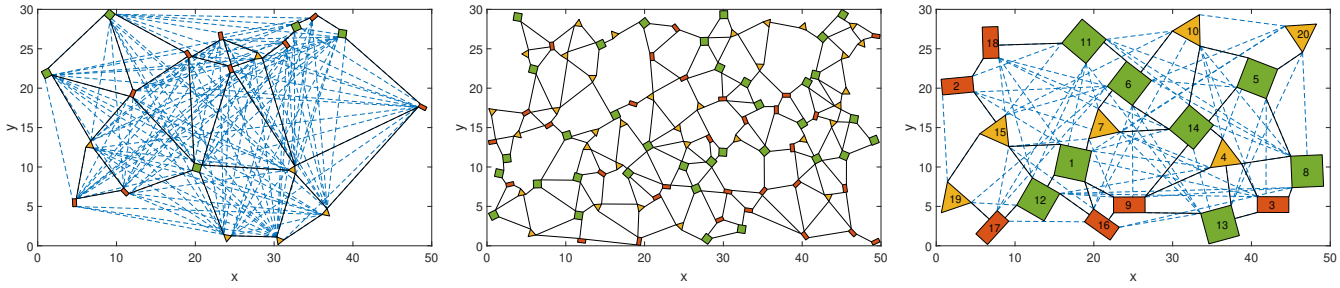


Fig. 7. Examples of passage detection results with different obstacle distributions. Dashed blue segments are passages after the visibility check. Solid black segments are passages after the extended visibility check. (a) and (b) have the same obstacle side length of one with 20 and 100 obstacles respectively. Dense dashed segments are not plotted in (b) for clarity. (c) 20 obstacles with an obstacle side length of four.

VI. Experimental Results

The proposed path set planning pipeline together with core modules is implemented and tested in various conditions. This section presents comprehensive evaluation results. Core code is updated at *https://github.com/HuangJingGitHub/HPSP*.

A. Passage Detection by Extended Visibility Check

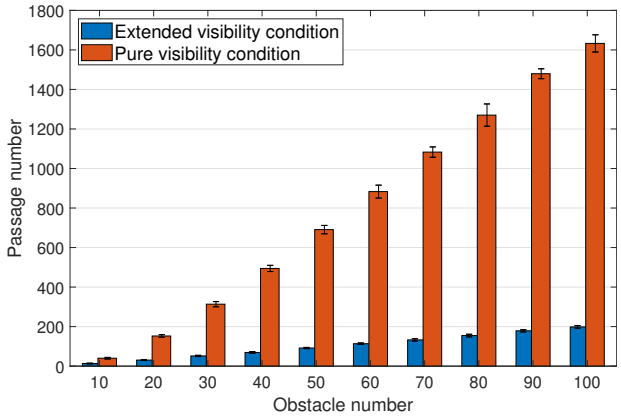
As a key upstream module, passage detection determines passages for the following path set planning. The experiment aims to investigate how the visibility condition and the extended variant affect the resulting passages. Different setups of obstacle shapes, sizes, and densities are tested to enrich passage variations. For each obstacle number equally spaced from 10 to 100, the passage number is averaged over 10 random obstacle distributions (map size: 50×30) of random shapes (squares, regular triangles, and rectangles with an aspect ratio of 2:1) and poses. The obstacle size is controlled by the side length which is set as one here to accommodate more obstacles (see Fig. 7). The statistical results in Fig. 8(a) show that the combinatorial quadratic increase of the passage number w.r.t. the obstacle number is reduced to significant linear relations by two visibility conditions, which dramatically brings down the valid passage number. Both conditions have coefficients of determination larger than 0.99 after linear regression. The extended visibility condition, however, has a much smaller passage number increase rate (~ 2.1 vs. 15.0). The ratios of the passage number using the visibility condition to that using the extended version has a mean of 0.158, suggesting that only a small fraction of passages remain after further checking the extended visibility.

Next, the obstacle number is fixed as 20, and side lengths are equally spaced from 0.5 to 5 to change obstacle sizes. Obstacle distributions are still randomly generated in 10 tests for each side length. Unlike the pure visibility check, Fig. 8(b) indicates that the extended visibility check is not sensitive to obstacle size changes. The passage number via the visibility check decreases significantly and nearly linearly as obstacles expand, but passage numbers after the extended visibility check present small variations. The passage number increases slightly rather than decreases as obstacles get larger. This counterintuitive phenomenon can be attributed to the fact that when obstacle sizes grow, the passage segment length $l(\mathbf{p}_i^*, \mathbf{p}_j^*)$ shrinks twice as fast as the third obstacle $\mathcal{E}_k$'s distance to the passage center $\mathbf{o}_{i,j}$, making $\mathcal{E}_k \cap \mathcal{R}_{i,j} \neq \emptyset$ easier to meet in (4), leading to insignificant passage number rises in Fig. 8(b). Predictably, the two conditions' differences will be unnoticeable if obstacle sizes are sufficiently large. Finally, Fig. 9 shows an example of passage detection in a 3D map. The passage distribution varies in height intervals divided by obstacle heights to get a sparse result. The passage distribution can be retrieved efficiently by indexing height as the key.

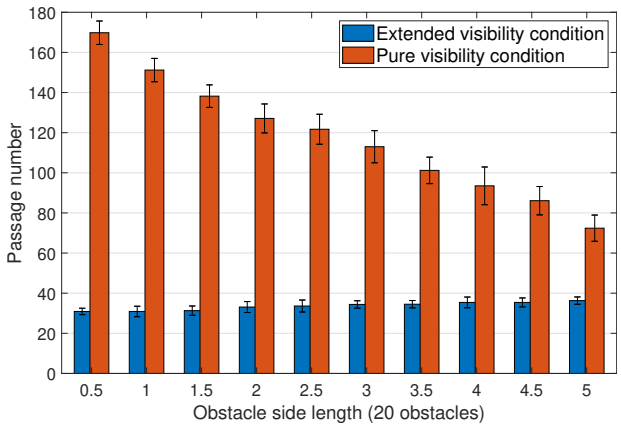
B. Passage-Aware Optimal Path Planning Results

This part showcases passage-aware optimal path planning (PAOPP) results. We aim to investigate two major aspects: the influence of cost formulations on planned paths and computational performance differences brought by two visibility checks in path planning. Despite many available efficient planner implementations such as the open motion planning library [51] and the navigation toolbox in MATLAB, it is not straightforward to incorporate passage-related functions and customized costs into existing frameworks due to the lack of related interfaces. RRT* planner is thus implemented separately with all subroutines in C++. Obstacle-related functionalities, including two types of visibility check for passage detection, passage segment positioning, and passage passing check for path segments, are packaged to be invoked readily. These make cost forms and parameters easily configurable when instantiating a planner.

As depicted in Fig. 10, different cost formulations are tested in path planning: the ratio cost in (5) and the weighted cost in (6) with different weights ($k_P = 1, 10$, and 100 respectively to change the preference). Various numbers of obstacles are randomly distributed. Passage detection is conducted using the extended visibility check and environment boundaries are treated as extra obstacles. The planning problem is constant across tests to find an optimal path from the top left to the bottom right corner as in Fig. (10). It is observed that paths can vary with cost choices. For the weighted cost, the path length $\text{Len}(\sigma)$ dominates the cost when $k_P = 1$. Thus, the resulting paths (green paths) prioritize minimizing $\text{Len}(\sigma)$, similar to the most typical setup in optimal path planning. In comparison, when $k_P = 100$, the minimum traversed passage width $f_P(\sigma)$ largely determines the cost. Planned paths (red paths) try to

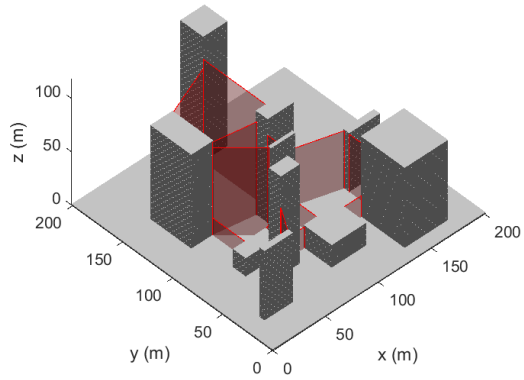


(a)

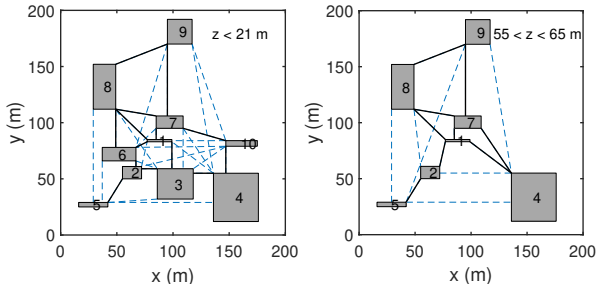


(b)

Fig. 8. Statistical results of passage numbers detected by two check conditions in different setups. (a) Passage numbers with different numbers of obstacles (obstacle side length is one). (b) Passage numbers with the same number of obstacles but different obstacle sizes.



(a)



(b)

Fig. 9. 3D passage detection example. (a) Valid passages are transparent red planes. (b) Height-varying passage distributions in two height intervals.

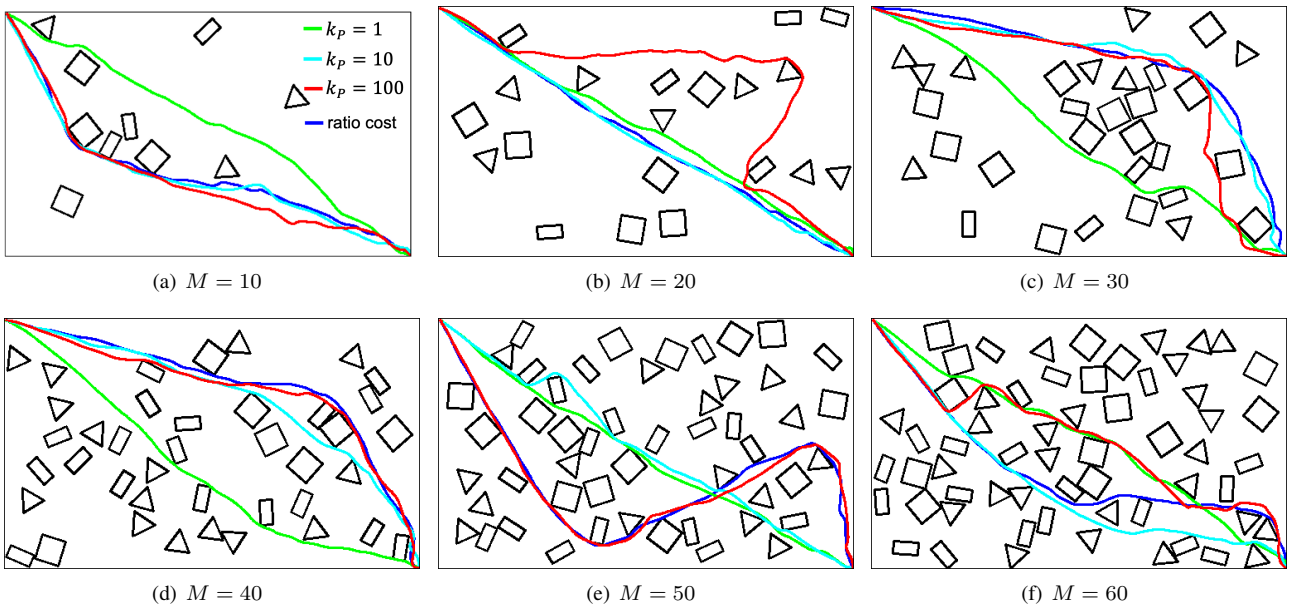
TABLE I
AVERAGE TIME OF PASSAGE-AWARE OPTIMAL PATH PLANNING AND MAX-CLEARANCE PATH PLANNING TESTS (ms)

Obstacle Num.	10	20	30	40	50	60
MC-time	327× 9.0	619× 9.0	740× 9.0	916× 9.0	1361× 8.9	1345× 8.7
MC-sample	868	1440	2087	2594	3627	4223
PAOPP-pure	1995	4172	6342	8312	10161	11405
PAOPP-ext	**882**	**1640**	**2401**	**3331**	**4212**	**5119**

avoid going through narrow passages. In between, $k_P = 10$ balances the two items shown by cyan paths. The value range of k_P to effectively alter two factors' precedence is problem-related as k_P varies to make $\text{Len}(\sigma)$ and $k_P f_P(\sigma)$ comparable. Overall, the ratio cost (paths are in blue) behaves similarly to adopting a moderate k_P in the weighted cost, making it a balanced cost in most cases.

To quantitatively measure how the extended visibility check improves planning efficiency, passages are identified by two check conditions respectively for PAOPP, i.e., PAOPP-pure and PAOPP-ext. Planning efficiency is gauged by the planning time with a maximum valid sample number $N = 10$k. The path cost is weighted ($k_P = 10$) and the start and goal remain unchanged. All computations are run on a PC with Ubuntu 18.04, Intel Core i9-7980XE CPU@2.60 GHz×36, and 64 GB of RAM, with no specifications for acceleration. For the same problem, paths found with two passage check results have the same passage traversal list and similar path costs. Thus, they can be regarded as equivalently optimal paths. See Appendix D for theoretical analyses of two checks' equivalence in PAOPP. The planning time differs much in Table I. For each obstacle number (the side length is three), 10 different planning tests are run. The planning time is shown to reduce $58.7\,\%$ on average. Roughly, the extended visibility check helps save more than half the planning time. Note that despite the significant time saving, its reduction is not as dramatic as the passage number reduction in Fig. 8(a) because passage-related operations only take a fraction of planning computations.

Next, PAOPP is further compared to the conventional max-clearance path planning (MCPP) method. MCPP finds the shortest path with the maximum clearance (MC) to all obstacles and represents a classical avenue to balance path's accessible free space and length. To get the MC, a binary search for MC in planning is performed as in Algorithm 4. The lower and upper bounds are half the minimum and maximum passage width respectively. For completeness, passage widths are detected by the pure visibility check. Two failure criteria for an MC value exist: return false if the planner fails to find a path after a given time (MCPP-time) or a given sample number (MCPP-sample). In Table I, MCPP-time reports the one-round planning time when MC is found within an error of $0.5\times$ the total planning trail number. MCPP-sample reports the total planning time and the max sample number is $N_{MC} = 20$k.



(a) $M = 10$ (b) $M = 20$ (c) $M = 30$

(d) $M = 40$ (e) $M = 50$ (f) $M = 60$

Fig. 10. Path planning results using different cost formulations. The obstacle number in the environment ranges from 10 to 60. Blue paths correspond to the cost in (5). Green, cyan, and red paths correspond to the cost in (6) with $k_P = 1, 10$, and 100, respectively.

MCPP-time turns out to be the slowest. Due to the fast planning failure check, MCPP-sample is faster than PAOPP with sparse passages by $13.4\,\%$ on average. However, its inherent drawbacks are that the planned result is not adjustable and its performance deteriorates if N_{MC} is large. See Appendix C for more implementation details and results.

C. Path Set Generation Results

Built on the modules above, the set generation scheme is implemented and tested. Given the initial S_0 and final S_d, a pivot path is first planned via PAOPP. $k_P = 10$ is utilized in pivot path planning for a balanced cost. The top row in Fig. 11 illustrates the directly transferred path set Σ_t (in blue) and the repositioned pivot path σ_p^* (in green). Though the planned pivot path σ_p goes through passages associated with large free space, directly transferred paths collide with obstacles easily. This is because the passage passing positions of σ_p, namely intersections with passages, are obtained to minimize the path length, which commonly makes σ_p located in obstacles' vicinity. Thereby, repositioning σ_p is required while preserving its passed passages to better accommodate transferred paths. In experiments, the geometrical approach for chord determination is utilized. Reference points on passage segments are given from chords' locations and σ_p is deformed piecewise to σ_p^* as (15).

It is reinforced that filtering out invalid passages is not only for computational efficiency but also necessary. Dense passages after pure visibility check make path set intersections overly redundant. The extended variant attains a sparse distribution of passages more suitable for repositioning σ_p and deformable path transfer. The repositioned σ_p^* is in a configuration more likely to lead to feasible transferred paths. In less restrictive cases with fewer obstacles like Fig. 11(a) and Fig. 11(d), paths directly transferred from σ_p^* are feasible. In general, transferred paths from σ_p^* may collide with obstacles, and coordinated deformable path transfer is required. As in Fig. 11(f), there exist narrow passages that cannot accommodate all directly transferred paths. Coordinated reference points are given for each path to guide path deformation, making the path set go through narrow passages freely. In scenarios where obstacles are dense and of significant sizes, translated passage segments can be introduced to density reference points for feasible deformed paths. See Appendix B for more examples with translated passages.

TABLE II
AVERAGE PATH SET PLANNING TIME USING PATH SET TRANSFER AND SEPARATELY PLANNING METHOD (ms)

Path Number	3	6	9	12	15	18
SP ($M = 10$)	4442	9501	14653	20083	25393	31105
PT ($M = 10$)	**840**	**882**	**845**	**879**	**863**	**867**
SP ($M = 20$)	5809	11972	17743	24630	28956	33411
PT ($M = 20$)	**1660**	**1681**	**1653**	**1665**	**1661**	**1658**
SP ($M = 30$)	7242	13259	21332	28087	34020	40727
PT ($M = 30$)	**2480**	**2433**	**2498**	**2450**	**2452**	**2272**

For efficiency evaluation, the benchmark method of separately planning (SP) each path is compared with the proposed scheme based on path transfer (PT). In the SP method, σ_p is also first planned using PAOPP. Then the rest paths are planned separately with the homotopy constraint to σ_p. Specifically, samples are restricted to be close to σ_p and are obstacle-free to σ_p in the neighborhood. Table II outlines the average path set planning time when obstacle number and path number vary in 10 random setups. As is observed, the SP method has a significantly larger time cost that increases nearly linearly with the path number. Conversely, the PT method time is not sensitive to the path number at all. Its dominant cost solely comes from σ_p planning in PAOPP. Other operations in PT

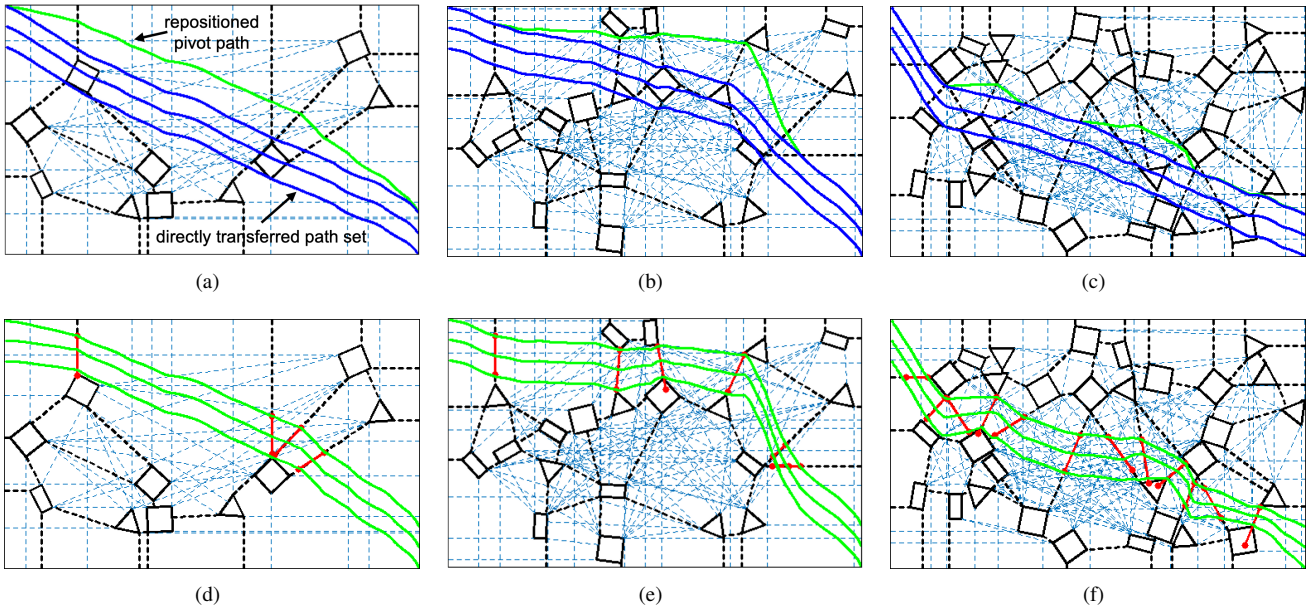


Fig. 11. Path set generation results. Black dashed lines are passages after the extended visibility check. Light blue dashed lines are passages after the pure visibility check. (a)-(c) show the directly transferred path sets in blue and the repositioned pivot paths in green. (d)-(f) show corresponding path sets obtained via deformable path transfer from repositioned pivot paths. Red segments depict the chords of Σ_t^* transferred from the repositioned σ_p^*.

(e.g., pivot path reposition, path transfer) are linear to the path resolution, i.e., path node number, which is almost negligible compared to the cost of invoking the planner once. Moreover, coordination constraints are hard to impose in SP. Thus, the PT method is much more advantageous in homotopic path set planning. See Appendix C for more results.

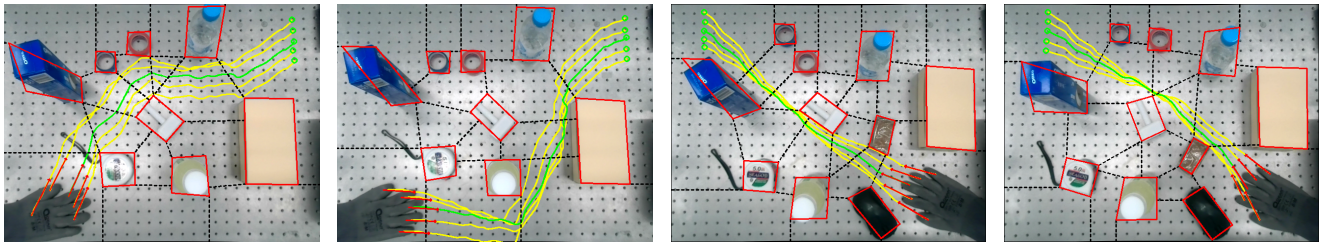
D. Application Case Study

The last part showcases some common applications of path set planning in robot manipulation and navigation. The first kind of task is DO path planning in complex environments, similar to [5], but in much more cluttered setups. As shown in Fig. 12, a manipulation site is emulated with various objects randomly placed on the table. The robot needs to move a deformable grip glove on this table while avoiding collisions. Due to the restriction of tall objects like vertically placed boxes and bottles, manipulation is confined near the table surface, for which a path reference of the glove is required. To do this, several keypoints are picked on the glove to depict the glove state. In Fig. 12, these points are mainly distributed on fingers and the middle one acts as the pivot by (10). The target S_d is vertically aligned. Obstacles are segmented as bounding quadrangles and path set planning is conducted in the image space. Fig. 12 demonstrates different test results. As can be seen, the planned path sets have wide free space along them with sufficiently short lengths, making them good path references for glove movement. With planned path sets, subsequent manipulation can be conducted as a constrained path set tracking problem as in [5].

The second task is swarm navigation in 3D maps. In Fig. 13, aerial robot swarms ($K = 5, 10$) move as a team in city environments where buildings of various sizes and heights are populated. To reach the target, the swarm needs to fly through narrow spaces among buildings. Robots are not allowed to fly subsequently as a queue or vertically aligned when passing narrow space and they are initially aligned horizontally in a line. The target is a different and distant formation. Robots' positions change in 3D, including the flight height. Passage detection is first performed to establish the passage distribution in height intervals formed by buildings. The pivot can be selected arbitrarily and its planned path selects wide passages in Fig. 13(a) and Fig. 13(c). Then pivot path repositioning and coordinated deformable path set transfer are performed to generate the final path set in Fig. 13(b) and Fig. 13(d). Path deformation on passages is restricted in the x-y plane. Since robots' initial and target positions are vertices of convex hulls, methods in [3] will have to plan paths for each robot with no optimization of accessible free space along paths.

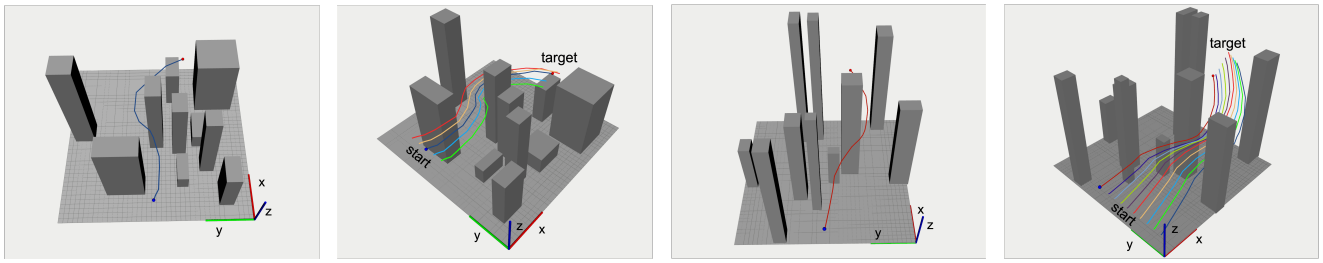
VII. Conclusion

This paper presented a systematic pipeline of homotopic path set planning for substantial robotics applications. The extended visibility check was first proposed to attain sparse passage distributions over pure visibility check. Passage-aware optimal path planning compatible with sampling-based optimal planners was designed with adjustable path costs. It could balance the path length and accessible free space more flexibly over clearance-based planning methods. Path set generation based on path transfer was then proposed. Techniques such as proportional reference point distribution, geometrical chord determination, and translated passage segments provided a better guarantee of the resulting paths' feasibility and coordination. The scheme's effectiveness was validated in different experiments. Future work includes more detailed passage descriptions as an area or space. Considering dynamic environments and imposing constraints among agents such as reducing path intersections are also important directions.



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

Fig. 12. Path set planning in DO manipulation site with random daily objects (bottles, boxes, tapes, etc.). There are eight objects in (a) and (b), ten in (c) and (d). Black dashed lines are detected passages. Planned path sets for the glove comprise yellow and green paths (pivot paths).



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

Fig. 13. Path set planning examples for swarms ($K = 5, 10$) in 3D maps. (a), (c) Passage-aware optimal path planning returns pivot paths with large accessible free space. (b), (d) Corresponding final path sets after pivot path repositioning and coordinated deformable path transfer.

REFERENCES

[1] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[2] P. Mao and Q. Quan, "Making robotics swarm flow more smoothly: A regular virtual tube model," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 4498-4504.

[3] P. Mao, R. Fu, and Q. Quan, "Optimal virtual tube planning and control for swarm robotics," *The International Journal of Robotics Research*, 2023.

[4] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme and N. Ayanian, "Trajectory Planning for Quadrotor Swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856-869, 2018.

[5] J. Huang, X. Chu, X. Ma, and K. W. Samuel Au "Deformable object manipulation with constraints using path set planning and tracking," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4671-4690, 2023.

[6] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[7] V. E. Arriola-Rios et al., "Modeling of deformable objects for robotic manipulation: A tutorial and Review," *Frontiers in Robotics and AI*, vol. 7, 2020.

[8] P. Jiménez, "Survey on model-based manipulation planning of deformable objects," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 154–163, 2012.

[9] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, 2021.

[10] F. Lamiraux and L. E. Kavraki, "Planning paths for elastic objects under manipulation constraints," *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 188–208, 2001.

[11] R. Gayle, W. Segars, M. Lin, and D. Manocha, "Path planning for deformable robots in complex environments," in *Proc. Robotics: Science and Systems*, 2005.

[12] M. Moll and L. E. Kavraki, "Path planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 625–636, 2006.

[13] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1141–1150, 2007.

[14] A. Doumanoglou *et al.*, "Folding clothes autonomously: A complete pipeline," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1461–1478, 2016.

[15] L. Lu and S. Akella, "Folding cartons with fixtures: A motion planning approach," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 4, pp. 346–356, 2000.

[16] O. Burchan Bayazit, Jyh-Ming Lien, and N. M. Amato, "Probabilistic roadmap motion planning for deformable objects," in *Proc. IEEE International Conference on Robotics and Automation*, 2002, pp. 2126-2133.

[17] R. Gayle, M. C. Lin, and D. Manocha, "Constraint-based motion planning of deformable robots," in *Proc. IEEE International Conference on Robotics and Automation*, 2005, pp. 1046-1053.

[18] D. Mcconachie, A. Dobson, M. Ruan, and D. Berenson, "Manipulating deformable objects by interleaving prediction, planning, and control," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 957–982, 2020.

[19] D. Mcconachie, T. Power, P. Mitrano, and D. Berenson, "Learning when to trust a dynamics model for planning in reduced state spaces," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3540–3547, 2020.

[20] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Conference on Neural Information Processing Systems*, 2016.

[21] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar, "Learning robotic manipulation through visual planning and acting," in *Proc. Robotics: Science and Systems*, 2019.

[22] A. Felner et al., "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Proc. International Symposium on Combinatorial Search*, 2017, pp. 29–37.

[23] R. Stern et al., "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proc. International Symposium on Combinatorial Search*, 2019, pp. 151–158.

[24] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proc. AAAI Conference on Artificial Intelligence*, 2013, pp. 1443–1449.

[25] D. Silver, "Cooperative pathfinding," in *Proc. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005, pp. 117–122.

[26] Z. Bnaya and A. Felner, "Conflict-oriented windowed hierarchical cooperative A*", in *Proc. IEEE International Conference on Robotics and Automation*, 2014, pp. 3743-3748.

[27] M. Khorshid, R. Holte, and N. Sturtevant, "A polynomial-time algorithm for non-optimal multi-agent pathfinding," in *Proc. International Symposium on Combinatorial Search*, 2011, pp. 76–83.

[28] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs", in *Proc. IEEE International Conference on Robotics and Automation*, 2013, pp. 3612-3617.

[29] E. Erdem, D. G. Kisa, U. Oztok and P. Schüller, "A general formal framework for pathfinding problems with multiple agents," in *Proc. AAAI Conference on Artificial Intelligence*, 2013, pp. 290-296.

[30] P. Surynek, "Towards optimal cooperative path planning in hard setups through satisfiability solving," in *Proc. The Pacific Rim International Conference on Artificial Intelligence*, 2012, pp. 564-576.

[31] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3260-3267.

[32] G. Sharon, R. Stern, A. Felner and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40-66, 2015.

[33] G. Sharon, R. Stern, M. Goldenberg and A. Felner, "The increasing cost tree search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 195, pp. 470-495, 2013.

[34] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[35] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia, "Automated composition of motion primitives for multi-robot systems from safe LTL specifications," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1525-1532.

[36] T. D. Barfoot and C. M. Clark, "Motion planning for formations of mobile robots," *Robotics and Autonomous Systems*, vol. 46, no. 2, pp. 65–78, 2004.

[37] A. Krontiris, S. Louis, and K. E. Bekris, "Multi-level formation roadmaps for collision-free dynamic shape changes with non-holonomic teams," in *Proc. IEEE International Conference on Robotics and Automation*, 2012, pp. 1570-1575.

[38] N. Ayanian, V. Kallem, and V. Kumar, "Synthesis of feedback controllers for multiple aerial robots with geometric constraints," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3126-3131.

[39] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time UAV replanning using guided gradient-based optimization and topological paths," in *Proc. IEEE International Conference on Robotics and Automation*, 2020, pp. 1208–1214.

[40] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[41] L. Jaillet and T. Simeon, "Path deformation roadmaps: Compact graphs with useful cycles for motion planning," *The International Journal of Robotics Research*, vol. 27, no. 11–12, pp. 1175–1188, 2008

[42] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1105-1115, 2005.

[43] P. Guo, C. Sun and Q. Li, "Obstacle avoidance path planning in unstructured environment with narrow passages," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 11, pp. 4632-4643, 2023.

[44] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *Proc. IEEE International Symposium on Voronoi Diagrams in Science and Engineering*, 2007, pp. 38–47.

[45] H. Niu, A. Savvaris, A. Tsourdos, and Z. Ji, "Voronoi-visibility roadmap-based path planning algorithm for unmanned surface vehicles," *The Journal of Navigation*, vol. 72, no. 04, pp. 850–874, 2019.

[46] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14413–14423, 2020.

[47] S. Karaman, M.R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. IEEE International Conference on Robotics and Automation*, 2011, pp. 1478-1483.

[48] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846-894, 2011.

[49] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2997-3004.

[50] J. M. Esposito and T. W. Dunbar, "Maintaining wireless connectivity constraints for swarms in the presence of obstacles," in *Proc. IEEE International Conference on Robotics and Automation*, 2006, pp. 946-951.

[51] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics and Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

APPENDIX A
CHORD DETERMINATION VIA PATH SET'S LOCAL GEOMETRY

The chord obtained via the intersection check of the passage line and the path set is not always informative for repositioning the path set. Regarding the path set as a tube, a chord makes a good reference when it aligns well with the cross-section of the tube as exemplified in Fig. 14. To better describe how the path set passes a passage, instead of intersections with the passage line, the cross-section nearby can be utilized. Specifically, the intersection between the pivot path σ_p and passage $P_{\sigma_p}(1,i)$ is $\sigma_p(\eta_{p,i})$. The intersection $\Sigma_t \cap \mathcal{N}(\sigma_p, \eta_{p,i})$ is first found, where $\mathcal{N}(\sigma_p, \eta_{p,i})$ represents the pivot path's normal line at $\sigma_p(\eta_{p,i})$ available by the local path's geometrical properties. Similarly to (12), the chord on $\mathcal{N}(\sigma_p, \eta_{p,i})$ has the following length form

$$\|\Sigma_t \cap \mathcal{N}(\sigma_p, \eta_{p,i})\|_2 = \max_{1\le k,j\le K} \|\sigma^N_{t,k}(\eta_{k,i}) - \sigma^N_{t,j}(\eta_{j,i})\|_2 \tag{A1}$$

where $\sigma^N_{t,k}(\eta_{k,i})$ is the intersection point between $\mathcal{N}(\sigma_p, \eta_{p,i})$ and the transferred path $\sigma_{t,k}$. Since paths' shifts are on the passage. Then $\Sigma_t \cap \mathcal{N}(\sigma_p, \eta_{p,i})$ is reflected on $P_{\sigma_p}(1,i)$, which can be done by rotating $\Sigma_t \cap \mathcal{N}(\sigma_p, \eta_{p,i})$ about the point $\sigma_p(\eta_{p,i})$ to $P_{\sigma_p}(1,i)$. The rotation direction is the acute angle between $P_{\sigma_p}(1,i)$ and $\mathcal{N}(\sigma_p, \eta_{p,i})$ to preserve the relative distribution of intersection points. Roughly speaking, looking in the pivot path's forward direction at $\sigma_p(\eta_{p,i})$, if an intersection point $\sigma^N_{t,k}(\eta_{k,i})$ is on the left (right) side of $\sigma_p(\eta_{p,i})$ on $\mathcal{N}(\sigma_p, \eta_{p,i})$. It should be on the left (right) side of $\sigma_p(\eta_{p,i})$ after rotation on $\mathcal{P}_{\sigma_p}(1,i)$. The chord obtained by rotation now depicts the intersection point distribution on the passage and the following steps keep the same.

APPENDIX B
ADDING TRANSLATED PASSAGE SEGMENTS ON OBSTACLES

Adding translated passage segments aims to provide more references for path collision avoidance in obstacle-dense setups. For example, $\sigma_{t,3}$ in Fig. 5 is deformed to shift left. If the only reference point on $P_{\sigma_2}(1,i)$ is placed too close to the right obstacle, the repositioned path is still in collision and this is not explicitly addressed in [5]. To handle this, the passed passage segment by the pivot path is translated to obstacle vertices to make *translated passage segments*. As illustrated in Fig. 16, $P_{\sigma_p}(1,i)$ is translated to vertices constructing this passage. $P_{\sigma_p}(1,i) = (\mathcal{E}_j, \mathcal{E}_k)$ with $\mathbf{p}^*_j \in \mathcal{E}_j, \mathbf{p}^*_k \in \mathcal{E}_k$ minimizing the distance in (2). There exists a translated passage segment starting at a vertex $\mathbf{v}^E_j \in \mathcal{E}_j$ if

$$\mathbf{v}^E_j + \delta \frac{\mathbf{p}^*_k - \mathbf{p}^*_j}{\|\mathbf{p}^*_k - \mathbf{p}^*_j\|_2} \in \mathcal{X}_{free} \tag{A2}$$

for some small $\delta > 0$. $\mathbf{p}^*_k - \mathbf{p}^*_j$ characterizes the direction from $\mathcal{E}_j$ to $\mathcal{E}_k$ along $P_{\sigma_p}(1,i)$. For vertices on $\mathcal{E}_k$, the direction reverses. (A2) essentially defines a ray starting at $\mathbf{v}^E_j$. The other end of the ray is assigned by detecting its collision with other obstacles or boundaries. Next, reference points of $\sigma_{t,i}$ on translated passage segments are found where the same procedure as in Section V is adopted.

In practice, there may be multiple passage segments associated with an obstacle. For an obstacle vertex not contained in a passage, its nearest passage can be selected. Moreover, to avoid too dense passage segments, $P_{\sigma_p}(1,i)$ can be translated to only one obstacle in $\mathcal{E}_j$ and $\mathcal{E}_k$. Lastly, a translated passage can be filtered away by considering its distance to existing passages on the obstacle. Specifically, $d^*(\mathbf{v}^E_j, \mathbf{v}^E_{near})$ is the distance of $\mathbf{v}^E_j$ to an existing passage end $\mathbf{v}^E_{near}$ on $\mathcal{E}_j$ in the orthogonal direction of $\mathbf{p}^*_k - \mathbf{p}^*_j$, i.e.,

$$d^*(\mathbf{v}^E_j, \mathbf{v}^E_{near}) = \|(\mathbf{v}^E_j - \mathbf{v}^E_{near})^\mathsf{T} \mathbf{n}^\perp_{k,j}\|_2. \tag{A3}$$

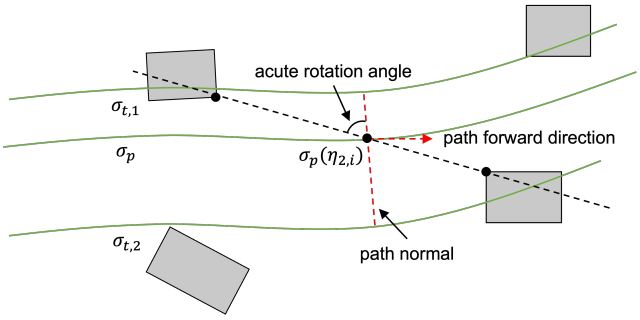
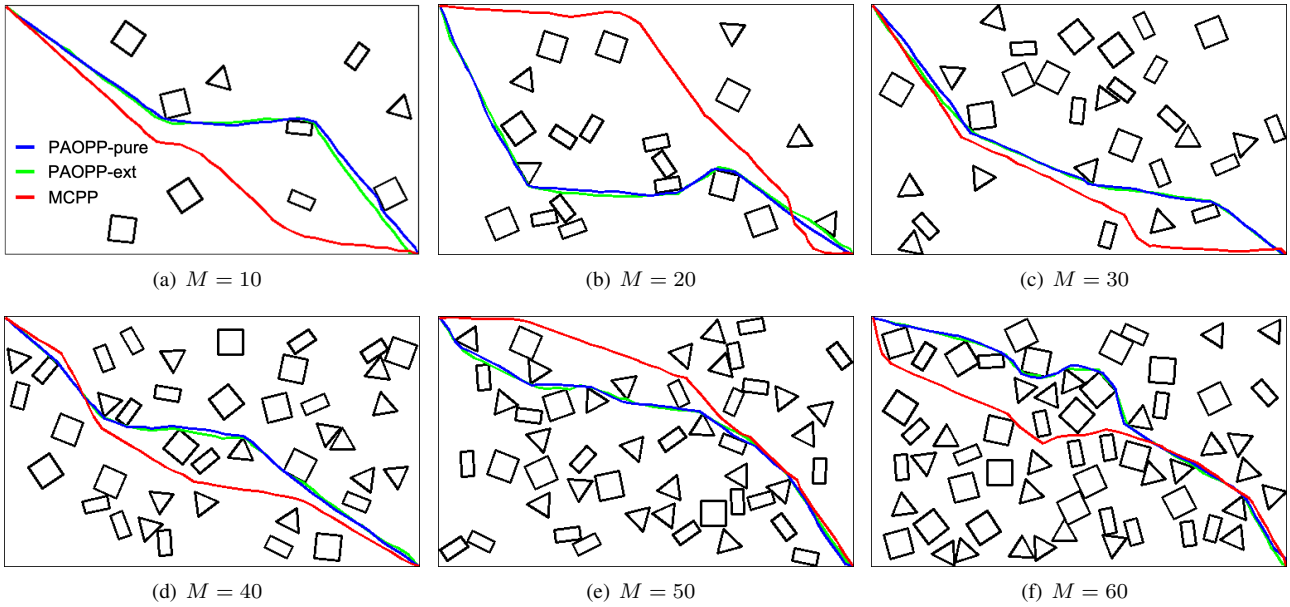


Fig. 14. In this example, the chord on the passage line poorly characterizes how paths should be moved. Paths' intersections on the normal line of the pivot path better describe the path set's local geometry.

Algorithm 3: Path Set Generation

$\Sigma_t \leftarrow \emptyset, \Sigma_S \leftarrow \emptyset$;
$\mathbf{s}_p \leftarrow$ (10);
$\sigma_p \leftarrow$ PAOPP in Algorithm (2);
$\Sigma_t \leftarrow$ (9) using σ_p;
foreach $P_{\sigma_p}(1,i) \in P_{\sigma_p}(1)$ **do**
  $\{\sigma_{t,1}(\eta_{1,i}), ..., \sigma_p(\eta_{p,i}), ..., \sigma_{t,K}(\eta_{K,i})\} \leftarrow \Sigma_t \cap P'_{\sigma_p}(1,i)$;
  $\|\Sigma_t \cap P'_{\sigma_p}(1,i)\|_2 \leftarrow$ (12);
  if $\|\Sigma_t \cap P'_{\sigma_p}(1,i)\|_2 > \|P_{\sigma_p}(1,i)\|_2$ **then**
    $\sigma^*_p(\eta_{p,i}) \leftarrow$ (14);
  else if $(\Sigma_t \cap P'_{\sigma_p}(1,i)) \nsubseteq P_{\sigma_p}(1,i)$ **then**
    $\sigma^*_p(\eta_{p,i}) \leftarrow$ (13);
  else
    $\sigma^*_p(\eta_{p,i}) \leftarrow \sigma_p(\eta_{p,i})$;
$\{\eta_{p,1}, \eta_{p,2}, ...\} \leftarrow \{0\} \cup \{\eta_{p,1}, \eta_{p,2}, ...\} \cup \{1\}$;
foreach $[\eta_{p,i}, \eta_{p,i+1}]$ **do**
  $\sigma^*_p(\tau) \leftarrow$ (15);
$\Sigma^*_t \leftarrow$ (9) using σ^*_p;
foreach $P_{\sigma_p}(1,i) \in P_{\sigma_p}(1)$ **do**
  $\{\sigma^*_{t,1}(\eta_{1,i}), ..., \sigma^*_p(\eta_{p,i}), ..., \sigma^*_{t,K}(\eta_{K,i})\} \leftarrow \Sigma^*_t \cap P'_{\sigma_p}(1,i)$ and reference points generation;
foreach $\mathbf{s}_i \in S$ AND $\mathbf{s}_i \neq \mathbf{s}_p$ **do**
  $\sigma^*_{t,i} \leftarrow$ Reposition $\sigma_{t,i}$ as (15);
  $\Sigma_S \leftarrow \Sigma_S \cup \{\sigma^*_{t,i}\}$;
return $\Sigma_S \cup \{\sigma^*_p\}$;



(a) $M = 10$ (b) $M = 20$ (c) $M = 30$

(d) $M = 40$ (e) $M = 50$ (f) $M = 60$

Fig. 15. Path planning results using different planners. The obstacle number ranges from 10 to 60. Blue paths are found by PAOPP-pure. Green paths are found by PAOPP-ext. $k_P = 50$ in two PAOPP planners. Red paths are found by MCPP.

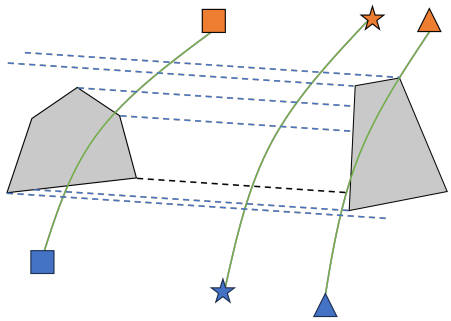


Fig. 16. Transferred paths from the repositioned σ_p^* may be infeasible. The shown paths shifted left from the setup in Fig. 5 as σ_p is shifted left to σ_p^*, but infeasible paths exist. Blue dashed lines are translated passage segments starting from obstacle vertices.

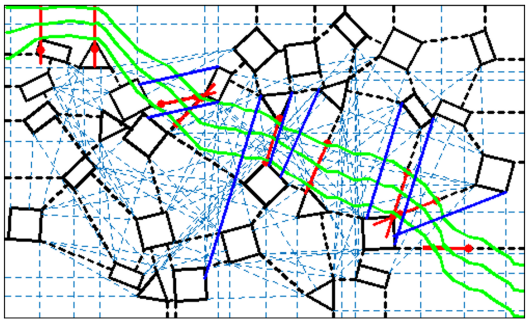


Fig. 17. Example of adding translated passages. Translated passage segments (blue segments) are only on obstacles on one side of the pivot path to avoid overly dense passage segments.

$\mathbf{n}_{k,j}^{\perp}$ is the unit vector orthogonal to $\mathbf{p}_k^* - \mathbf{p}_j^*$. If $d^*(\mathbf{v}_j^E, \mathbf{v}_{near}^E)$ is too small, $P_{\sigma_p}(1, i)$ is not translated to the vertex $\mathbf{v}_j^E$. An example of added passage segments is shown in Fig. 17.

Appendix C
Extended Experimental Results

More comparative results between PAOPP and MCPP, path set generation using SP and PT methods are reported here. Fig. 10 shows examples of path planning results using MCPP and PAOPP with two passage detection strategies ($k_p = 50$). PAOPP-pure (using passages detected by the pure visibility check) and PAOPP-ext (using passages detected by the extended visibility check) find similar paths passing the same passages. Thus, paths have similar path costs and can be regarded as equivalently optimal paths. This empirically validates that the sparse passage distribution after the extended visibility check in PAOPP-ext will generally have the same optimal path as PAOPP-pure. The equivalence of PAOPP-ext and PAOPP-pure on an easy condition is analyzed in Appendix D. Paths found by MCPP may or may not be homotopic to those in PAOPP. Due to the clearance constraint, the planned path keeps the MC from obstacles. For better feasibility, the clearance to map boundaries is not included in MCPP. MCPP implementation is outlined in Algorithm 4.

The comparisons between path set generation results using SP and PT are demonstrated in Fig. 19. In addition to the significant runtime differences reported in Table II, Fig. 19 provides intuitive comparisons of planned path sets. The main difference is that though the homotopy constraint is enforced in the SP method, coordination among paths is hard to achieve and paths overlap in most cases. To address this, explicit rules need to be specified to coordinate paths or postprocessing will be required to refine paths. In contrast, path sets generated by the PT method usually occupy wider spaces in narrow passages to place paths, which makes it more suitable for multiple path generation in obstacle-dense environments. In addition, the PT method allows for specifications of coordination requirements by adjusting reference point distributions in passages, making it quite flexible.

Algorithm 4: Max-Clearance Path Planning (MCPP)

Input $\mathbf{s}_0, \mathbf{s}_d$;
$P_{valid} \leftarrow \texttt{PureVisibilityCheck}(\mathcal{E}_1, ..., \mathcal{E}_M)$;
$MC_{lower} \leftarrow \min \|P_{valid}\|_2/2$;
$MC_{upper} \leftarrow \max \|P_{valid}\|_2/2$;
$\sigma_{MC} \leftarrow \emptyset$;
while $MC_{upper} - MC_{lower} > MC_{err}$ **do**
  $MC \leftarrow MC_{lower} + (MC_{upper} - MC_{lower})/2$;
  $\sigma_{MC} \leftarrow \texttt{PlanWithClearance}(\mathbf{s}_0, \mathbf{s}_d, MC)$;
  if $\sigma_{MC} \neq \emptyset$ **then**
    $MC_{lower} \leftarrow MC$;
  else
    $MC_{upper} \leftarrow MC$;
return σ_{MC};

APPENDIX D
CONDITION OF EQUIVALENCE BETWEEN PAOPP-PURE AND PAOPP-EXT

One fundamental problem of the extended visibility check is whether it guarantees the optimality of planned paths in PAOPP problems. In other words, the problem is if paths planned by PAOPP-ext are guaranteed to be the same as those planned by PAOPP-pure. Here we show that the equivalence between PAOPP-ext and PAOPP-pure can be achieved via simple processing. For simplicity, only the 2D case is discussed here, but the derivation extends to 3D space readily. The elementary case is illustrated in Fig. 18. The passage $(\mathcal{E}_1, \mathcal{E}_2)$ passes the pure visibility check but will be discarded in the extended visibility check due to $\mathcal{E}_3$. $(\mathcal{E}_1, \mathcal{E}_3)$ and $(\mathcal{E}_2, \mathcal{E}_3)$ are valid. Considering the region $\mathcal{A}_{123}$ enclosed by obstacles and passage segments, any non-winding path σ can be categorized into one of the following four types:

1) σ does not pass $\mathcal{A}_{123}$.
2) σ is entirely inside $\mathcal{A}_{123}$.
3) σ passes through $\mathcal{A}_{123}$.
4) One endpoint of σ is inside $\mathcal{A}_{123}$ and the other is not.

Our discussion is restricted to PAOPP problems in which $f_P(\sigma)$, the minimum passage width passed by σ, is used in the path cost like (5) and (6). For type 1 and 2, $f_P(\sigma)$ is intact. For type 3, the discarding of $(\mathcal{E}_1, \mathcal{E}_2)$ will not affect the update of $f_P(\sigma)$ before and after passing $\mathcal{A}_{123}$. Specifically, note that $\|(\mathcal{E}_1, \mathcal{E}_2)\|_2 > \|(\mathcal{E}_1, \mathcal{E}_3)\|_2$, $\|(\mathcal{E}_1, \mathcal{E}_2)\|_2 > \|(\mathcal{E}_2, \mathcal{E}_3)\|_2$. Then if σ passes $(\mathcal{E}_1, \mathcal{E}_2)$ and $(\mathcal{E}_1, \mathcal{E}_3)$ (the passing order does not matter), the update rule of $f_P(\sigma)$ is

$$f_P(\sigma^2) = \min(f_P(\sigma^1),\ \|(\mathcal{E}_1,\ \mathcal{E}_3)\|_2) \tag{A4}$$

where σ^1, σ^2 represent σ before entering $\mathcal{A}_{123}$ and after leaving $\mathcal{A}_{123}$, respectively. If σ passes $(\mathcal{E}_1, \mathcal{E}_2)$ and $(\mathcal{E}_2, \mathcal{E}_3)$, $f_P(\sigma)$ is updated as

$$f_P(\sigma^2) = \min(f_P(\sigma^1),\ \|(\mathcal{E}_2,\ \mathcal{E}_3)\|_2). \tag{A5}$$

Finally, if σ passes $(\mathcal{E}_1, \mathcal{E}_3)$ and $(\mathcal{E}_2, \mathcal{E}_3)$, $f_P(\sigma^2)$ is

$$f_P(\sigma^2) = \min(f_P(\sigma^1), \min(\|(\mathcal{E}_1,\ \mathcal{E}_3)\|_2, \|(\mathcal{E}_2,\ \mathcal{E}_3)\|_2). \tag{A6}$$

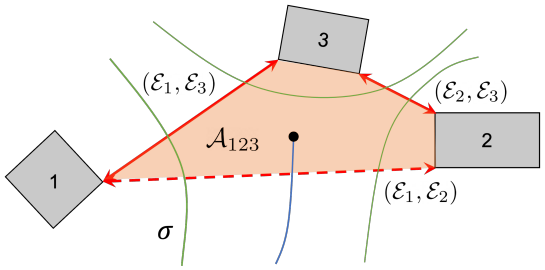


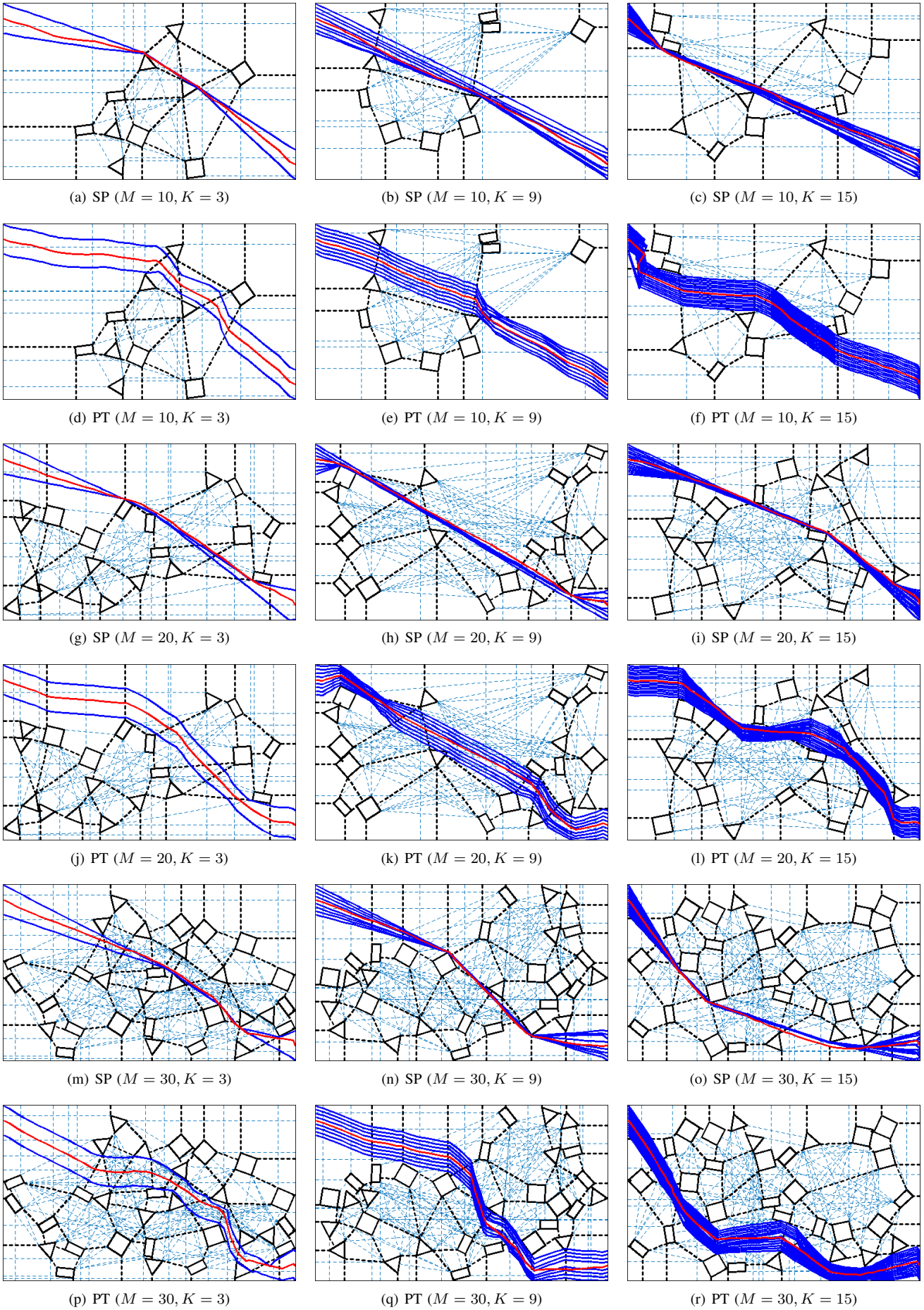
Fig. 18. Only type 3 and 4 are illustrated here. $(\mathcal{E}_1, \mathcal{E}_2)$ can pass the visibility check but cannot pass the extended variant. Green paths pass through $\mathcal{A}_{123}$ across different passages. The blue path has one endpoint inside $\mathcal{A}_{123}$.

In all cases, $f_P(\sigma^2)$ is independent from $(\mathcal{E}_1, \mathcal{E}_2)$. Therefore, discarding $(\mathcal{E}_1, \mathcal{E}_2)$ will not affect the update of the path cost $f(\sigma)$. This result holds iteratively for the entire environment. As a result, optimal planners return the path with the optimal cost, namely the optimal path.

For type 4, however, PAOPP-ext and PAOPP-pure may return different results. The fundamental reason is that in PAOPP-ext, path costs of paths passing $(\mathcal{E}_1, \mathcal{E}_2)$ may not be updated in type 4. Assume the target position $\sigma(1)$ is within $\mathcal{A}_{123}$, in PAOPP-pure, $f_P(\sigma^2_{1,2})$ after σ passes $(\mathcal{E}_1, \mathcal{E}_2)$ is

$$f_P(\sigma^2_{1,2}) = \min(f_P(\sigma^1_{1,2}), \|(\mathcal{E}_1,\ \mathcal{E}_2)\|_2) \tag{A7}$$

where the subscript of σ signifies which passage σ passes. In PAOPP-ext, however, $f_P(\sigma^2_{1,2}) = f_P(\sigma^1_{1,2})$ is not updated. If $f_P(\sigma^1_{1,2}) > \|(\mathcal{E}_1,\ \mathcal{E}_2)\|_2$ happens, $f_P(\sigma^2_{1,2}) = \|(\mathcal{E}_1,\ \mathcal{E}_2)\|_2$ is correctly updated in PAOPP-pure, but $f_P(\sigma^2_{1,2}) = f_P(\sigma^1_{1,2})$ in PAOPP-ext and a smaller cost is wrongly adopted (suppose other conditions except $f_P(\sigma)$ are the same). This may further lead to a different choice of the optimal path among $\sigma^2_{1,2}, \sigma^2_{1,3}$ and $\sigma^2_{2,3}$. The same also applies to situations where the start of σ is in $\mathcal{A}_{123}$. Note that such a situation is rare and only locally affects the path's beginning or ending parts. A simple way to address this is to preserve passages enclosing the start and target positions in PAOPP-ext. To sum up, PAOPP-ext and PAOPP-pure are equivalent in finding the optimal path in type 1 to 3. In type 4, the equivalence can be ensured by preserving passages enclosing the start and target positions.



(a) SP ($M = 10, K = 3$) (b) SP ($M = 10, K = 9$) (c) SP ($M = 10, K = 15$)

(d) PT ($M = 10, K = 3$) (e) PT ($M = 10, K = 9$) (f) PT ($M = 10, K = 15$)

(g) SP ($M = 20, K = 3$) (h) SP ($M = 20, K = 9$) (i) SP ($M = 20, K = 15$)

(j) PT ($M = 20, K = 3$) (k) PT ($M = 20, K = 9$) (l) PT ($M = 20, K = 15$)

(m) SP ($M = 30, K = 3$) (n) SP ($M = 30, K = 9$) (o) SP ($M = 30, K = 15$)

(p) PT ($M = 30, K = 3$) (q) PT ($M = 30, K = 9$) (r) PT ($M = 30, K = 15$)

Fig. 19. Path set generation results in different setups using SP and PT methods. Red paths are pivot paths in SP and repositioned pivot paths in PT. Other agent paths are in blue.