Efficient Trajectory Planning for Multiple Non-holonomic Mobile Robots via Prioritized Trajectory Optimization

Juncheng Li, Maopeng Ran, and Lihua Xie, *Fellow, IEEE*

Abstract—In this paper we present a novel approach to efficiently generate collision-free optimal trajectories for multiple non-holonomic mobile robots in obstacle-rich environments. Our approach first employs a graph-based multi-agent path planner to find an initial discrete solution, and then refines this solution into smooth trajectories using nonlinear optimization. We divide the robot team into small groups and propose a prioritized trajectory optimization method to improve the scalability of the algorithm. Infeasible sub-problems may arise in some scenarios because of the decoupled optimization framework. To handle this problem, a novel grouping and priority assignment strategy is developed to increase the probability of finding feasible trajectories. Compared to the coupled trajectory optimization, the proposed approach reduces the computation time considerably with a small impact on the optimality of the plans. Simulations and hardware experiments verified the effectiveness and superiority of the proposed approach.

Index Terms—Multi-robot systems, motion and path planning, collision avoidance.

I. INTRODUCTION

AUTONOMOUS multi-robot systems are attracting significant attention from the industry since they can provide more diverse functionality and efficiency than single-robot systems. Many applications of multi-robots systems require coordination of mobile robots navigating in a complex environment, e.g., material handling in warehouses [1] and drone delivery [2]. In these scenarios, collision-free trajectories connecting initial and final positions for the robots need to be generated, which is referred to as the labeled multi-robot trajectory planning problem [3].

It is challenging to obtain optimal trajectories for a large team of robots in an efficient way. In [4], [5], the optimal trajectory generation problem is formulated as mixed-integer quadratic programming (MIQP) or sequential convex programming (SQP) problems. These methods try to jointly optimize the trajectories of all robots. Though the optimality is guaranteed, the application of these methods is limited to small teams with few obstacles in the environment.

To improve computational efficiency, decoupled planning methods have been proposed. A widely used decoupled scheme for multi-robot trajectory planning is sequential planning [6]. In sequential planning, the trajectory of each robot is decoupled and coordinated by avoiding the previously planned robots. The trajectory optimization methods in [7], [8] utilize sequential planning to decouple inter-robot collision constraints and achieve significant improvement in computational efficiency. These decoupled planning methods are relatively fast but typically suffer from incompleteness. In certain scenarios, a feasible solution for the robots exists but cannot be found.

In obstacle-rich environments, the trajectory planning problem is generally solved using a two-stage pipeline [9], i.e., path finding and trajectory optimization. A great number of works have been done in single-robot cases [10]–[12]. The basic idea of such methods is to first generate a geometric path and then optimize the path to a smooth and dynamically feasible trajectory. This two-stage pipeline can also be applied in the multi-robot trajectory planning problem. In [13]–[15], collision-free discrete paths are first generated for all robots. Then based on the results, a quadratic program (QP) problem is formulated for each robot to generate the optimal trajectory. The multi-robot trajectory planning algorithms using the two-stage pipeline are guaranteed to be complete [13], [14]. Besides, since the path finding stage provides a good initial guess for the multi-robot trajectory optimization, the efficiency of solving the optimization problem is significantly improved by using the two-stage pipeline.

The aforementioned multi-robot trajectory planning algorithms [13]–[15] focus on robots with linear dynamics. In this case, the trajectory optimization can be formulated as a QP problem, which is convex and easy to solve. However, in modern industry, most mobile robots are differential-drive and inherently subject to nonlinear dynamics. When nonlinear dynamics is considered, the trajectory optimization can only be formulated as a general nonconvex nonlinear programming (NLP) problem, which makes the existing multi-robot trajectory planning methods [13]–[15] inapplicable. Currently, the problem of motion planning for multiple differential-drive robots is generally addressed via discrete formulations [16]–[18]. However, since the planned piecewise linear paths contain corner turns that are dynamically infeasible for differential-drive robots, these paths are difficult for the robots to execute.

The multi-robot motion coordination problem can also be

solved in a distributed way. Reactive methods such as buffered Voronoi cells [19] and velocity obstacle (VO) [20], [21] are developed. Besides, in [22], [23], distributed trajectory planning methods are proposed based on the distributed model predictive control (DMPC). In these methods, the robot follows the shortest path to its destination and resolves conflicts in real time when future collisions are detected. Some algorithms have been extended to nonlinear dynamics and applied to non-holonomic robots [24], [25]. Distributed planning methods are computational efficient, but cannot guarantee no deadlock and are poorly suited to problems in maze-like environments.

Based on the above considerations, in this paper, we aim to propose an efficient trajectory planning approach that generates safe, dynamically feasible and near-optimal trajectories for multiple non-holonomic mobile robots. Our approach first uses a multi-robot path planner to generate an initial solution for the problem, and then it is refined into smooth trajectories by solving a trajectory optimization problem. In both stages, the nonlinear motion model of the mobile robots is directly considered to guarantee the feasibility of the trajectories. In particular, we introduce a prioritized trajectory optimization method to improve the computational efficiency such that the algorithm is applicable to large-scale robot teams. To the best of our knowledge, this paper is the first attempt to develop a centralized multi-robot trajectory planning method for non-holonomic mobile robots in continuous space. The main contributions of this work are as follows:

- An efficient multi-robot trajectory planning approach which generates collision-free optimal trajectories for a large team of non-holonomic mobile robots.
- A prioritized optimization method which decouples the multi-robot trajectory optimization problem and improves the computational efficiency significantly.
- Extensive evaluations of the proposed approach via simulations and real-world experiments.

The remaining of this paper is organized as follows. In Section II, we state the problem formulation. In Section III, the proposed approach is described in detail. We evaluate our approach in Section IV and present a real-world experiment in Section V. Finally, Section VI concludes the paper and discusses future work.

II. PROBLEM FORMULATION

Consider a multi-robot system consisting of N mobile robots which operate in a 2-D workspace $\mathcal{W} \subseteq \mathbb{R}^2$. The obstacles in the environment are assumed to be known and denoted as $\mathcal{O}$. The free workspace of the robots is given by $\mathcal{F} = \mathcal{W} \backslash \mathcal{O}$. The collision model of each robot is defined as a circle with radius R. The subset of $\mathcal{W}$ occupied by the body of a robot at position $x \in \mathbb{R}^2$ is denoted by $\mathcal{R}(x)$. For each robot i, a task is assigned to move from its start position $s_i \in \mathcal{F}$ to its goal position $g_i \in \mathcal{F}$. To guarantee that there is no inevitable collision at the start and goal positions, we assume the tasks $\langle s_i, g_i \rangle_{i=1...N}$ must satisfy $\mathcal{R}(s_j) \cap \mathcal{R}(s_k) = \emptyset$ and $\mathcal{R}(g_j) \cap \mathcal{R}(g_k) = \emptyset$ for all $j \neq k$.

A sequence of waypoints which connect the start and goal positions of the robot is denoted by path $p = \{r^k\}_0^{N_p}$, where $r^k = [x^k, y^k, \theta^k]^{\mathrm{T}}$ denotes the kth waypoint on the path and N_p denotes the total number of waypoints. A trajectory ν is path p parameterized by time t. Let $\mathrm{pos}(r)$ denote the 2-D position of waypoint r. For each robot i, its body should not collide with any obstacle in the environment when following its trajectory ν_i, i.e., $\mathcal{R}(\mathrm{pos}(r_i(t))) \subseteq \mathcal{F}$. Besides, the collisions between any two robots should be avoided, i.e., $\mathcal{R}(\mathrm{pos}(r_i(t))) \cap \mathcal{R}(\mathrm{pos}(r_j(t))) = \emptyset, \forall i \neq j$.

Furthermore, the planned trajectory should be dynamically feasible for each robot. The kinematic model of each robot is defined as

$$\dot{z} = f(z, u), \ u \in \mathcal{U}. \tag{1}$$

In this paper, the unicycle model for differential wheeled robots is considered. The state z is defined as $[x, y, \theta]^{\mathrm{T}}$ which consists of the 2-D position and orientation. The robot is controlled by the linear and angular velocity $u = [v, \omega]^{\mathrm{T}}$, and its motion equations are given by

$$\dot{x} = v\cos(\theta), \ \dot{y} = v\sin(\theta), \ \dot{\theta} = \omega. \tag{2}$$

Besides, each robot is limited by its maximum velocity, i.e., $\mathcal{U}_i = \{[v_i, \omega_i]^{\mathrm{T}} : |v_i| < v_i^{\max}, |\omega_i| < \omega_i^{\max}\}$. In this paper, we aim to generate dynamically feasible and optimal trajectories $\nu_1, \ldots, \nu_N$ for a group of mobile robots with nonlinear dynamics, such that each robot can reach its goal, and avoid collisions with obstacles and other robots.

III. METHODOLOGY

The overall architecture of the proposed multi-robot trajectory planning approach is shown in Fig. 1. Firstly, we use a graph-search based method to find shortest collision-free paths for all robots with the non-holonomic constraint (2). Then the safe corridor is constructed around each robot's path, which is a collection of convex polyhedra that models the safe space of the robot. Finally, we formulate a constrained nonlinear optimization problem based on the planned discrete paths and safe corridors. By efficiently solving the problem using prioritized trajectory optimization, safe and near-optimal trajectories for all robots are obtained. The detail of our approach is described in the following subsections.

A. Discrete Path Planning

The workspace of the robots is abstracted as an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where each vertex $v \in \mathcal{V}$ corresponds to a location agents can cover and each edge $(v_i, v_j) \in \mathcal{E}$ corresponds to a path the agents traverse when moving from vertex v_i to v_j. In this work, occupancy map of the environment is transformed into a 2-D grid graph. In general, each vertex and one of its 4 neighbors can form an edge (Fig. 2a). As mentioned before, the state of each robot contains both the position and orientation information. The traditional grid graph only considers the position and neglects the orientation information, so it is not suitable for non-holonomic mobile robots. To handle this problem, we propose a new graph representation for the multi-robot path planning.

Firstly, in the graph $\mathcal{G}$, each vertex $v \in \mathcal{V}$ is now three dimensional and corresponds to a robot state consisting of

Discrete Path Planning   Safe Corridor Construction   Trajectory Optimization
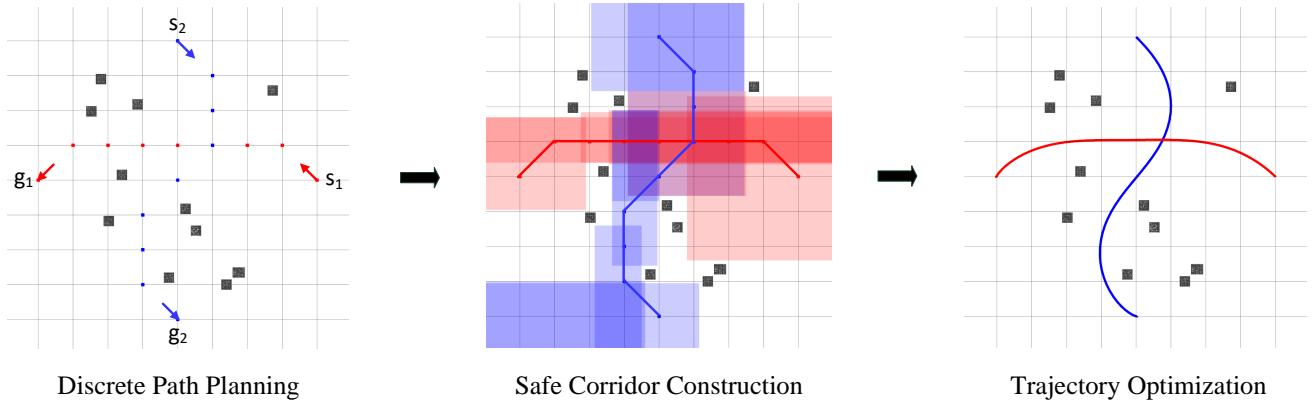
Fig. 1: Overall architecture of the proposed multi-robot trajectory planning approach. Gray squares show the static obstacles in the environment. In this example, the start and goal positions of agent 1 and 2 are assigned as $s_1 = (4, 0)$, $g_1 = (-4, 0)$, $s_2 = (0, 4)$, $g_2 = (0, -4)$. Firstly, we find shortest collision-free paths for the robots. Secondly, safe corridors (red and blue blocks) are constructed along the planned paths. Finally, a constrained trajectory optimization problem is solved to generate smooth and dynamically feasible trajectories for the robots.
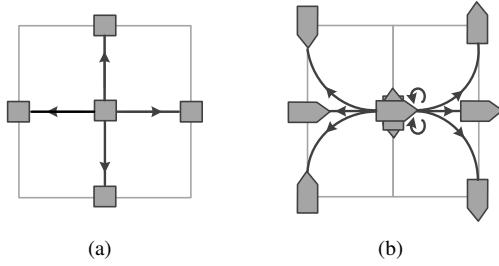

(a)   (b)

Fig. 2: Edges in the grid graph. Fig. 2a shows the traditional 4-connected edges. Considering the kinematic constraint of differential wheeled robots, the edges in the graph are defined by motion primitives, as shown in Fig. 2b.

the position and orientation. The heading of the robot can be chosen from the four main directions, i.e., north, south, east, and west. Let $P(v)$ denote the 2-D position of vertex v, then each vertex should satisfy $\mathcal{R}(P(v)) \subseteq \mathcal{F}$ in order to avoid possible collisions. Secondly, based on the kinematic model (2), we define the possible one-step state transitions of the robot, which is also referred to as motion primitives [26]. There are three available actions for each robot to take at each step, i.e., moving forward, moving backward and turning 90 degrees. Combining these possible actions, we can obtain all the motion primitives which are shown in Fig 2b. Let $\mathcal{A}$ denote the set of all the motion primitives. Each one-step transition in $\mathcal{A}$ is considered as an edge in the graph $\mathcal{G}$. The edge is valid only when no collision is found when any one of the robots traverses that edge.

Since the kinematics of the robot is considered in the graph construction process, the feasibility of the motion primitives needs to be guaranteed. Let D denote the grid size of graph $\mathcal{G}$, ΔT denote the time required for the robot to traverse one edge. Due to the physical limits of each robot, the following condition should be satisfied to guarantee the feasibility of the

diagonal one step transition:

$$v_i^{\max}\Delta T \geq \frac{\pi}{2}D, \quad w_i^{\max}\Delta T \geq \frac{\pi}{2}, \quad \forall i \in \{1, \cdots, N\}. \quad (3)$$

It can be observed that if the diagonal one-step transition is guaranteed to be feasible, then all actions in $\mathcal{A}$ are feasible. Since both ΔT and D in (3) are user-defined parameters, we can construct suitable graph representations for different scales of environments based on the proposed method.

Conflicts may occur when each robot plans its path individually. In the same graph $\mathcal{G}$ which is shared by all robots, two kinds of conflicts are considered, i.e., vertex conflict and edge conflict. At each timestep k, if two robots i and j occupy two vertexes which are too close to each other, it is a vertex conflict denoted by $\langle i, j, k \rangle$. At any time between two consecutive timesteps k_1 and k_2, if two robots are too close to each other when traversing their own edge, it is an edge conflict denoted by $\langle i, j, k_1, k_2 \rangle$. The vertex conflict set at timestep k and edge conflict set at timestep (k_1, k_2) are described as:

$$\text{VCon}(k) = \{\langle i, j \rangle \, | \quad\quad\quad\quad\quad\quad\quad\quad (4a)$$
$$\mathcal{R}(\text{pos}(r_i^k)) \cap \mathcal{R}(\text{pos}(r_j^k)) \neq \emptyset\},$$
$$\text{ECon}(k_1, k_2) = \{\langle i, j \rangle| \quad\quad\quad\quad\quad\quad\quad (4b)$$
$$\mathcal{R}(\text{pos}(r_i(t))) \cap \mathcal{R}(\text{pos}(r_j(t))) \neq \emptyset, \forall t \in (k_1\Delta T, k_2\Delta T)\}.$$

Finding shortest conflict-free paths for multiple agents in the defined graph is known as a multi-agent path finding (MAPF) problem. Solving MAPF optimally is NP-hard [17] and suffers from a scalability problem. In this work, enhanced conflict-based search (ECBS) [27] is leveraged as the multi-robot discrete path planner. This algorithm can obtain a bounded suboptimal solution efficiently and guarantee completeness.

ECBS works in two-levels. At the high-level, a binary constraint tree (CT) is constructed to resolve the detected conflicts. At the low-level, optimal paths for individual agents are planned which are consistent with their own constraints. In the beginning, the root node of the CT contains no constraints and the low-level search returns an initial solution.
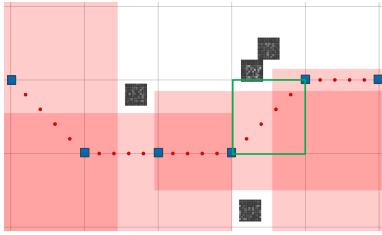
Fig. 3: Construction of the safe corridor. Blue squares show an original discrete path, which cannot be directly used to construct the safe corridor. The green box shows a failure case. Red dots show a sampled path. By expanding each point of this path in x and y axes, the safe corridor can be constructed successfully, which is shown as red blocks.

The solution is checked for conflicts in chronological order by (4). Suppose a vertex conflict $\langle i, j, k \rangle$ is found, then two child nodes are generated to resolve this conflict. The first node adds a constraint for agent i to avoid staying at r_i^k at timestep k. The second node adds a constraint for agent j to avoid staying at r_j^k at timestep k. An edge conflict $\langle i, j, k_1, k_2 \rangle$ can be resolved in a similar way. Each time a CT node is created, the low-level search for the agent with added constraint is executed, which returns a new solution. ECBS performs a best-first search on the CT where nodes are ordered by their costs. The expansion of the CT continues until a solution without any conflict is found. By implementing a focal search with conflict heuristic in both high-level and low-level, ECBS solves the MAPF problem efficiently.

Since the relative distance between robots will be checked at each timestep in the trajectory optimization process, the length of each robot's discrete path should be the same. In this case, we denote the length of the longest path as M. For each agent i, we append its goal position g_i at the end of its path p_i until the length of p_i is M.

B. Safe Corridor Construction

The planned discrete path of each robot is denoted by $p = \{r^k\}_0^M$, where each waypoint r^k is in the free space $\mathcal{F}$. The kth line segment in the path is denoted by $I^k = \langle r^{k-1} \rightarrow r^k \rangle$. We generate a convex polyhedron around each line segment in the path to construct a valid safe corridor. The generated convex polyhedron around the line segment I^k is denoted as $\mathcal{S}^k$. To ensure an agent in the polyhedron $\mathcal{S}^k$ does not collide with any obstacle in the environment, the following condition should be satisfied:

$$\mathcal{R}(a) \cap \mathcal{O} = \emptyset, \ \forall a \in \mathcal{S}^k. \tag{5}$$

The collection of these convex polyhedra constitutes the safe corridor, which is denoted by $\mathrm{SC}(p) = \{\mathcal{S}^k \,|\, k = 1, \ldots M\}$. Note that the safe corridor needs to be sequentially connected, and hence satisfies the following condition:

$$\mathcal{S}^k \cap \mathcal{S}^{k+1} \neq \emptyset, \ \forall k \in \{1, \cdots, M-1\}. \tag{6}$$

An axis-search method inspired from [15] is leveraged in this work to build the safe corridor. The safe corridors are expanded in both x and y axes until the maximum possible

---

Algorithm 1 Safe corridor construction

Require: discrete path $p = \{r^k\}_0^H$
1: **function** SAFE CORRIDOR (p)
2:     **for** $k \leftarrow 1$ to H **do**
3:         **if** $r^k \in \mathcal{S}^{k-1}$ **then**
4:             $\mathcal{S}^k \leftarrow \mathcal{S}^{k-1}$
5:         **else**
6:             $\mathcal{S}^k \leftarrow r^k$
7:             $\mathcal{D} \leftarrow \{+x, -x, +y, -y\}$
8:             **while** $\mathcal{D} \neq \emptyset$ **do**
9:                 **for** d in $\mathcal{D}$ **do**
10:                     $\widetilde{\mathcal{S}}^k \leftarrow$ expand $\mathcal{S}^k$ in the direction d
11:                     **if** $\mathcal{S}^k \subseteq \mathcal{F}$ **then**
12:                         $\mathcal{S}^k \leftarrow \widetilde{\mathcal{S}}^k$
13:                     **else**
14:                       $\mathcal{D} \leftarrow \mathcal{D} \backslash d$
15:     **return** $\mathrm{SC}(p) = \{\mathcal{S}^k | k = 1, \cdots, H\}$

---

free space has been covered. To ensure the continuity of the safe corridors, $\mathcal{S}^k$ should be initialized to be the set containing the consecutive waypoints r^{k-1} and r^k. In our approach, since the motion primitives are included in the discrete path planning stage, the planned paths consist of horizontal, vertical and diagonal line segments. However, the initial safe corridor around a diagonal line segment may not be completely within the free space. Fig. 3 shows an example.

To solve this problem, firstly we divide each line segment in the path into h equal parts. The new sampled path now consists of $H = 1 + h(M - 1)$ waypoints. Then we initialize S^k to be the point r^k, and expand it in x and y axes until the maximum possible free space is covered. The safe corridor initialized in this way is completely within the free space. To guarantee the continuity of the safe corridor, the minimum safe regions around two consecutive waypoints need to be overlapped, which means that the following condition should be satisfied:

$$\frac{\sqrt{2}D}{h} < 2R_i, \ \ \forall i \in \{1, \cdots, N\}, \tag{7}$$

where R_i is the radius of the collision model of the robot i. Additionally, if the point r^k is found to be within the previous safe region $\mathcal{S}^{k-1}$, the safe corridor expansion process is not needed and the convex polygon $\mathcal{S}^k$ should be exactly the same as $\mathcal{S}^{k-1}$. The whole safe corridor construction process of each robot's path is summarized in Alg. 1.

C. Trajectory Optimization Problem

The discrete paths are refined into smooth and feasible trajectories in the trajectory optimization phase. For each robot, we assign a time $t^k = k\Delta t$ to each waypoint in its sampled path $p = \{r^k\}_0^H$ so that the path becomes a reference trajectory $\nu^r = \{r^k, t^k\}_0^H$. As mentioned before, each line segment in the original path is divided into h equal parts, so $\Delta t = \Delta T / h$. Based on the reference trajectory ν^r and the constructed safe corridor $\mathrm{SC}(p)$, we aim to obtain the optimal trajectory $\nu = \{z^k, t^k\}_0^H$ for each robot such that the group

of robots can reach their goals and avoid any collision. The nonlinear optimization problem is formulated as:

$$\text{minimize} \sum_{i=1}^{N}(\sum_{j=1}^{H-1} \Delta u_i^{j\,\mathrm{T}} P \Delta u_i^j + \sum_{j=1}^{H} \widetilde{z}_i^{j\,\mathrm{T}} Q \widetilde{z}_i^j) \tag{8a}$$

$$\text{s.t.} \quad z_i^0 = s_i, \; z_i^H = g_i \qquad \forall i \tag{8b}$$

$$z_i^{j+1} = f(z_i^j, u_i^j), \qquad \forall i, j \tag{8c}$$

$$\text{pos}(z_i^j) \in \mathcal{S}_i^j, \qquad \forall i, j \tag{8d}$$

$$u_i^j \in \mathcal{U}_i, \qquad \forall i, j \tag{8e}$$

$$\left\| \text{pos}(z_{i_1}^j) - \text{pos}(z_{i_2}^j) \right\| \geq R_{i_1} + R_{i_2}, \; \forall i_1, i_2, j, i_2 \neq i_1 \tag{8f}$$

where $\Delta u_i^j = u_i^j - u_i^{j-1}$, $\widetilde{z}_i^j = z_i^j - r_i^j$, and $P \in \mathbb{R}^{2\times 2}$ and $Q \in \mathbb{R}^{3\times 3}$ are two positive definite weighting matrices. The objective function consists of two parts. In the first part, differences between two successive control inputs are penalized for the smoothness of the trajectories. In the second part, since the reference trajectory ν^r of each robot is already a feasible solution, to keep the feasibility of the nonlinear optimization problem (8), we penalize the deviation between the optimal trajectory and the reference trajectory. The state transition at each timestep is determined by (8c), where the nonlinear model f is the discrete-time version of (2). The position of each waypoint on the trajectory is limited by the constructed safe corridor in the constraint (8d). The control inputs of each robot is limited to physically admissible values in (8e). Moreover, the safe distance between each pair of robots is limited by (8f).

D. Prioritized Trajectory Optimization

Problem (8) is not efficiently solvable for large-scale robot teams. Therefore, we propose an efficient prioritized trajectory optimization method to solve the problem. The robots are first divided into some groups with unique priorities. Then the trajectory optimization subproblem is solved sequentially from the highest-priority group to the lowest-priority group. In each iteration, trajectories of the current group of robots are optimized under the constraint that they must avoid collisions with all the higher-priority robots. The trajectory optimization process runs relatively fast using this decoupled framework. However, if the priority is not carefully defined based on the specific scenarios, prioritized optimization may lead to infeasible subproblems for lower-priority robots due to lack of consideration by higher-priority robots. To handle this problem, a novel grouping and priority assignment strategy is proposed to enable the algorithm to find a near-optimal solution with a higher probability.

The main concern of the prioritized trajectory optimization is the inter-robot constraint (8f). Since the optimal trajectories are close to the reference ones according to the cost function (8a), we can analyze the inter-robot constraints based on the reference trajectory $\nu_1^r, \ldots, \nu_N^r$. At each timestep t, we search for a triple of robots $(i_1, i_2, i_3)^t$ which satisfies:

$$\left\| \text{pos}(r_a^t) - \text{pos}(r_b^t) \right\| \leq D_{\text{th}}, \; \forall a, b \in \{i_1, i_2, i_3\}, b \neq a \tag{9}$$

Algorithm 2 Grouping and priority assignment

Require: reference trajectories $\nu_1^r \ldots \nu_N^r$
1: **function** PRIORITY ASSIGNMENT($\{\nu_i^r\}_{i=1\ldots N}$)
2:     **for** $t \leftarrow 1$ to H **do**
3:         $(i_1, i_2, i_3)^t \leftarrow$ find triple satisfies (9)
4:         Insert (i_1, i_2, i_3) into L
5:     **while** $L \neq \emptyset$ **do**
6:         $e \leftarrow$ most common element in L
7:         Append e to the group list G
8:         **for** l in L **do**
9:             **for** robot m in e **do**
10:                 **if** robot m in l **then**
11:                     Remove m from l
12:         **for** $n \leftarrow 1$ to N **do**
13:             **if** n not in G **then**
14:                 Append n to G
15:     **return** G

---

where $D_{\text{th}} = \sqrt{2}D$ is a threshold. Each triple represents a situation where three robots are close to each other at a specific timestep. In the discrete path planning stage, only when four robots are located at the four vertexes of a grid cell at the same time, we can find a situation where four agents simultaneously satisfy (9). Therefore, in this case, we only analyze inter-robot collisions among three robots rather than four or more robots.

If the robots in a triple are assigned different priorities, the higher priority robots will plan their trajectories first. Then these trajectories are considered as hard constraint in the optimization problem of the low priority robots. In this case, the feasibility of the original problem may be lost. Therefore, the three robots in a triple should form a group and be assigned the same priority. The list of all triples satisfying (9) is denoted by L.

If the number of robots in the workspace is large, L will contain a great number of elements and thus a robot may be included in different triples. In this case, the robots cannot be grouped directly. Since each triple represents a situation where three robots are close to each other, if one triple repeats many times in L, that means the trajectories of these three robots are highly coupled. A group of robots with higher coupling effect should be assigned with a higher priority, so we propose a priority assignment algorithm based on the number of occurrences of each triple.

In the beginning, the most common element in L is selected as the first-priority group. Since the robots in this new group may be included in other triples in L, we remove all of them from L and obtain a new list L'. As a result, each element of L' may contain three, two, or just one robot. Then we continue to select the most common element in the new list L' as the second-priority group. The process is repeated until the list is empty. Finally, we need to check the completeness of the algorithm. If a robot does not have any significant coupling with other robots, then it will not be included in the original list L. In this case, the robot is regarded as a single-robot group and assigned the lowest priority. Using the proposed algorithm, all robots in a team can be grouped and assigned
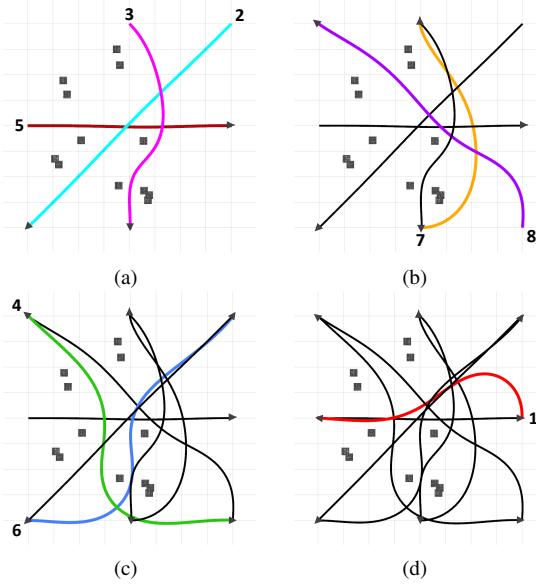
Fig. 4: An example of the prioritized trajectory optimization. In each iteration, the current group of robots need to avoid all higher-priority robots. The trajectories of the current group of robots and the higher-priority robots are depicted as colored lines and black lines, respectively.

priorities successfully and completely. The whole process is summarized in Alg. 2.

As mentioned before, next we formulate and solve the problem (8) for each group sequentially from high-priority to low-priority. Fig. 4 shows an example of the overall prioritized trajectory optimization process. Firstly, we group the 8 robots using Alg. 2. The groups are listed from high-priority to low-priority, i.e. $[(2, 3, 5), (7, 8), (4, 6), (1)]$. In each iteration of the prioritized trajectory optimization, in addition to the inter-robot constraints inside the current group, inter-robot constraints between the current group of robots and the optimized higher-priority robots should also be considered to ensure no collisions. The optimized trajectories of each group of robots are shown in Figs. 4a–4d.

IV. SIMULATIONS

A. Implementation Details

The proposed algorithms are implemented in C++ and executed on a laptop running Ubuntu 16.04 with Intel i5-6300HQ @2.30GHz CPU and 12GB of RAM. We use OctoMap [28] to represent the occupancy map of the environment and an interior-point nonlinear programming solver IPOPT [29] to solve the trajectory optimization problem. Our code is released as an open-source package[1].

In the simulation, the radius of the collision model of each robot is set to $R = 0.15$m and the velocities of each robot are limited by $v^{\max} = 1$m/s, $\omega^{\max} = 1$rad/s. We plan the discrete paths in the grid graph with grid size $D = 1$m. In this case, the time interval ΔT is set to 1.6s so that the condition (3) is

[1]https://github.com/LIJUNCHENG001/multi_robot_traj_planner

TABLE I: Computation time

Number of agents	Discrete path planning (s)	Safe corridor construction (s)	Trajectory optimization (s)	Total (s)
4	0.001	0.008	0.142	0.151
8	0.002	0.019	0.360	0.381
16	0.006	0.031	1.042	1.079
24	0.326	0.046	2.526	2.898
32	2.175	0.062	3.853	6.090

satisfied. Besides, in the safe corridor construction, each line segment is divided into $h = 5$ equal segments.

B. Computational Efficiency and Solution Quality

Material handling in warehouses is the main target application of our developed approach. Here, we construct a warehouse environment to evaluate the performance of the proposed approach. The environment has a size of 10m $\times$ 12m and contains 6 shelves of size 3m $\times$ 0.6m. The start and goal positions of each robot are randomly assigned at the boundary of the environment or at the pick-up points near the shelves. Based on the environment settings, we set the suboptimal bound of ECBS as 1.5 to fulfill the requirement on computation efficiency. Fig. 5 shows an example of 32 robots navigating in the simulation environment. We conduct the simulations for 40 times and calculate the average computation time, which is shown in Table I. The scalability of our approach is shown to be very good according to the results. The proposed approach takes about 6.1s to complete the whole trajectory planning process for 32 mobile robots.

In this paper, the method which directly solves the original large-scale trajectory optimization problem (8) is referred to as coupled trajectory optimization method. As shown in Fig. 6a, the proposed prioritized optimization method shows much better computational efficiency compared to the coupled trajectory optimization method. Specifically, if the number of agents is 4 or less, the two methods achieve comparable performance. As the number of agents becomes larger, the coupling effect between the robots becomes stronger. Since the proposed method decouples the large-scale optimization problem, the computational efficiency is improved significantly. The coupled optimization takes about 27.2s to solve the 32-agents case while our method only takes about 3.9s. It can be observed that the runtime of the proposed method increases almost linearly. We mention that prioritized optimization uses a decoupled framework, so it typically leads to inferior solutions compared to the coupled optimization. As shown in Fig. 6b, the total cost of the proposed method is on average 3.7% higher than that of the coupled optimization, which is relatively small and acceptable in practice.

C. Success Rate

To avoid infeasible subproblems generated in the prioritized trajectory optimization, a novel grouping and priority assignment strategy is also developed in this paper. Fig. 7 shows the success rate of the prioritized optimization using two different grouping strategies. Both methods can achieve 100% success rate when the number of agents is small. As expected, as the

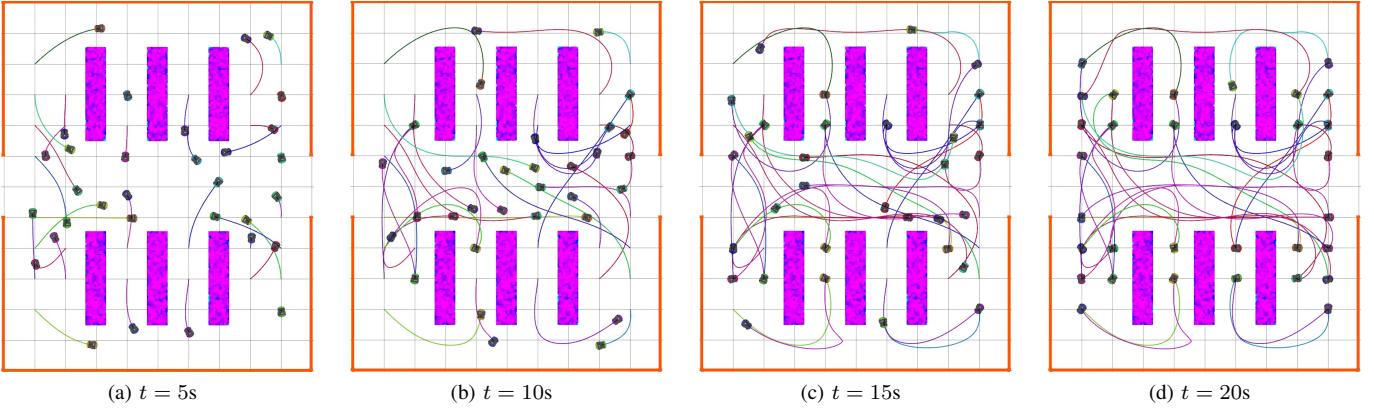(a) $t = 5$s   (b) $t = 10$s   (c) $t = 15$s   (d) $t = 20$s

Fig. 5: Trajectories of 32 agents in a warehouse environment.
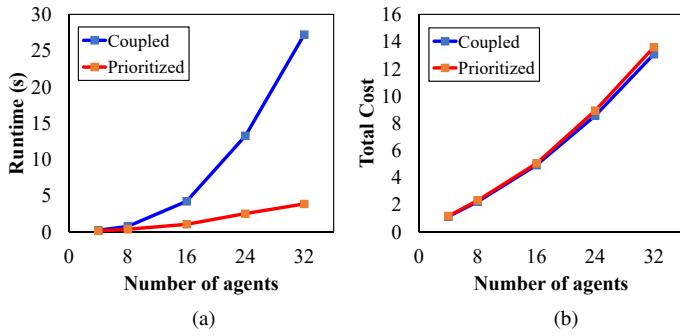

(a)   (b)

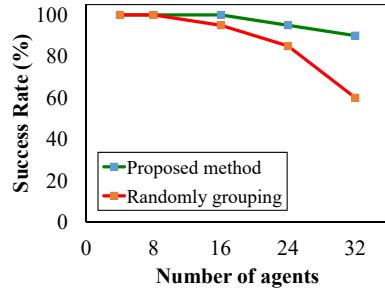Fig. 6: Comparison of the proposed approach and the coupled trajectory optimization.

Fig. 7: Success rate of the trajectory planning using different grouping strategies.

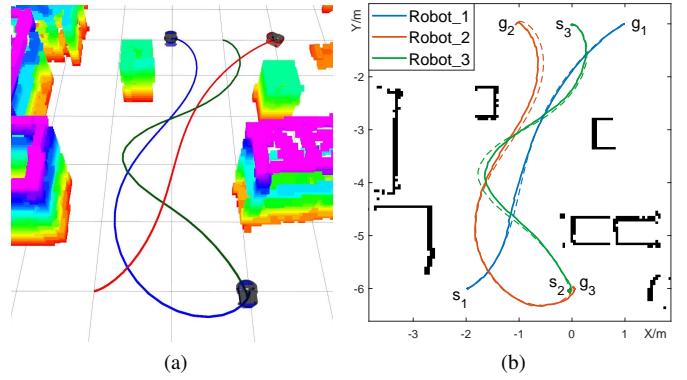Fig. 8: Three robots navigating in the real environment.

(a)   (b)

Fig. 9: Experiment results. Fig. 9a shows the 3D map of the environment and the planned trajectory. Fig. 9b shows the trajectory tracking results, where dash and solid lines represent the planned and real trajectories, respectively.

number of agents increases from 8 to 32, the success rate of the method using randomized grouping decreases from 100% to 60%. However, because we assign priorities to the robots based on the specific scenario, the proposed method achieves higher success rates for all cases in the test environment. In the 32-agent case, the success rate of the proposed method is on average 90%.

V. EXPERIMENT

We conduct a real multi-robot navigation experiment with 3 Pioneer 3-AT robots in a 7m × 8m indoor testing area, which is shown in Fig. 8. One robot is equipped with a Velodyne VLP-16 3D lidar, and each of the other two robots is equipped with a Hokuyo 2D lidar which has a maximum

range of 30m. At the beginning, the map of the test environment is constructed using a 3D lidar odometry and mapping method, LeGO-LOAM [30]. We construct the 3D map because obstacles of different heights need to be considered in the trajectory planning algorithm to guarantee safety. The radius of the collision model of each robot is set to $R = 0.3$m and the velocities of each robot are limited by $v^{\max} = 0.6$m/s, $\omega^{\max} = 0.6$rad/s. Besides, the time interval ΔT is set to 2.65s in the experiment and all the other parameters remain the same as described in Sec IV-A. The planned trajectories in the experiment are shown in Fig. 9a. We upload the

trajectories to the three robots and use a MPC-based trajectory tracking control method [31] to execute the trajectories. The localization of the robots is obtained through AMCL ROS package. As shown in Fig. 9b, the team of robots can track their reference trajectories accurately and complete the task without any collision. The experimental video is available at https://youtu.be/GRl3LM8xBUQ.

VI. Conclusion

In this paper, we presented an efficient trajectory planning algorithm for multiple non-holonomic robots navigation in obstacle-rich environments. The trajectory planning problem is decoupled as a front-end path searching and a back-end nonlinear trajectory optimization. We adopt a multi-agent path searching method to find collision-free time-optimal initial paths, which are further refined into smooth and dynamically feasible trajectories. A prioritized trajectory optimization method is proposed to improve the scalability of the back-end algorithm. We split the team of robots using a novel grouping and priority assignment strategy, and then solve the optimization problem sequentially. The effectiveness and superiority of the proposed method are validated in both simulations and real-world experiments.

In future work, we plan to integrate our work with distributed multi-agent navigation methods, so that the trajectory of each robot can be replanned online to handle unknown dynamic obstacles in the environment.

References

[1] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectrum*, vol. 45, no. 7, pp. 26–34, 2008.

[2] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2016.

[3] S. Tang and V. Kumar, "Safe and complete trajectory generation for robot teams with higher-order dynamics," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1894–1901.

[4] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 477–483.

[5] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1917–1922.

[6] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835–849, 2015.

[7] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5954–5961.

[8] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, "An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1215–1222, 2018.

[9] W. Ding, W. Gao, K. Wang, and S. Shen, "An efficient b-spline-based kinodynamic replanning framework for quadrotors," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1287–1306, 2019.

[10] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[11] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 344–351.

[12] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[13] S. Tang, J. Thomas, and V. Kumar, "Hold or take optimal plan (hoop): A quadratic programming approach to multi-robot trajectory generation," *The International Journal of Robotics Research*, vol. 37, no. 9, pp. 1062–1084, 2018.

[14] W. Hönig, J. A. Preiss, T. 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.

[15] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 434–440.

[16] J. Yu and D. Rus, "An effective algorithmic framework for near optimal multi-robot path planning," in *Robotics Research*. Springer, 2018, pp. 495–511.

[17] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of mapf schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[18] J. Motes, R. Sandström, H. Lee, S. Thomas, and N. M. Amato, "Multi-robot task and motion planning with subtask dependencies," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3338–3345, 2020.

[19] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[20] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 1928–1935.

[21] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Springer, 2011, pp. 3–19.

[22] C. E. Luis and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.

[23] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[24] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[25] J. Alonso-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.

[26] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 2172–2179.

[27] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *7th Annual Symposium on Combinatorial Search*, 2014.

[28] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[29] 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.

[30] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4758–4765.

[31] J. Li, M. Ran, H. Wang, and L. Xie, "MPC-based unified trajectory planning and tracking control approach for automated guided vehicles," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, 2019, pp. 374–380.