International Journal of Intelligent Robotics and Applications (2021) 5:176–185
https://doi.org/10.1007/s41315-021-00183-2

REGULAR PAPER

Decentralized motion planning for multi quadrotor with obstacle and collision avoidance

Xuewei Zhang[1] · Hongming Shen[1] · Guohui Xie[1] · Hanchen Lu[1] · Bailing Tian[1]

Received: 12 November 2020 / Accepted: 18 May 2021 / Published online: 28 May 2021

Abstract
In this paper, a decentralized motion planning approach for multi-quadrotor autonomous navigation is developed in environments populated with obstacles. We propose a priority-based RRT algorithm, which generates an online global path for each quadrotor with obstacle avoidance. Furthermore, the front-end paths uncross by assigning priorities and sharing global states in the swarm. The security and clearance of the trajectory are improved by B-spline optimization, where inter-collision-free is achieved by formulating the collision risk as a penalty term of the cost function. The efficiency of the developed algorithm is verified through numerical simulation.

Keywords Multi quadrotor · Decentralized motion planning approach · Collision avoidance

1 Introduction

In recent years, unmanned aerial vehicles (UAVs) are becoming popular in various applications such as industrial inspection, intelligence, surveillance, search-and-rescue missions (Saikin et al. 2020; Augugliaro et al. 2014; Mohta et al. 2018). To achieve full autonomy in these scenarios, the motion planning module plays an essential role in generating safe and smooth reference motion. However, computational power and payload become the limiting factors when planning a mission with small UAVs. Multi-UAV collaboration is often considered, aiming to coordinate multiple agents to complete the mission within a limited time or resources budget.

Several related works have promoted the development of multi-robot planning. In Fiorini and Shiller (1998), Van den Berg et al. (2008), Van Den Berg et al. (2011), the collision avoidance trajectory of multiple agents can be safely generated. However, these methods assume that there are no other obstacles in the environment. Even considering obstacles in the environment, the information about global maps are fully known (Park et al. 2019; Burri et al. 2015; Usenko et al. 2017). To obtain optimal trajectories for a large team of robots, the trajectory generation problem is solved in a centralized manner (Mellinger et al. 2012; Augugliaro et al. 2012; Tang et al. 2018). Nevertheless, it can not meet the requirements of real-time planning for large-scale robots. Decentralized approaches have been proposed in Liu et al. (2018), while the dynamic model of robots are ignored. To sum up, the motion planning for multi-robot is still an open question.

In this paper, we propose a decentralized motion planning approach for multi-quadrotor autonomous navigation in obstacle-dense previously unknown environments, where the quadrotors can steer away from obstacles and avoid collisions between them. A priority-based RRT algorithm is presented to find a guidance path for each quadrotor, which avoids obstacle and path crossing between multiple UAVs. Then, we take a piecewise polynomial trajectory to smooth the initial path. Finally, the trajectory is represented as a uniform B-spline, which refines carefully the initial polynomial trajectory.

The main contributions of the present work are summarized as follows.

1. A priority-based RRT algorithm is proposed for multi-UAV navigation to find an obstacle-avoidance and inter-vehicle collision-free piecewise line segment path with asymptotic optimality. Then B-spline optimization

✉ Bailing Tian
bailing_tian@tju.edu.cn

Xuewei Zhang
zhangxuewei@tju.edu.cn

[1] Tianjin University, Tianjin, China

(Usenko et al. 2017) is extended for reciprocal collision avoidance with the cost function.

2. A decentralized multi-UAV motion planning approach is proposed for generating a safe, smooth, and dynamically feasible trajectory to each UAV.
3. A multi-UAV simulation environment is established in Gazebo, which can be used to verify the feasibility of the multi-UAV motion planning algorithm. And an extensive simulation evaluation of the proposed method is conducted.

The rest of this paper is given as follows. Section 2 reviews the related works, while system overview and problem formulation are presented in Sect. 3. The developed path searching method and trajectory optimization method are detailed in Sect. 4. The system settings and simulation verification are presented in Sect. 5. The paper is concluded in Sect. 6.

2 Related work

For the problem of motion planning for multi-robot, there have been a lot of researches and applications. In this section, we give a brief overview of prior work in trajectory generation of robot swarm, which can be grouped into two broad approaches, centralized and decentralized trajectory generation.

Centralized trajectory generation In Mellinger et al. (2012), Augugliaro et al. (2012), Chen et al. (2015), the trajectory generation problems are formulated as mixed-integer quadratic programming (MIQP) or sequential convex programming (SCP) problems. However, the application of these methods is limited to small-scale teams due to the high computational overhead. To reduce the computation cost, Hamer et al. (2018) presented a parallel formulation that can fast generate collision-free trajectories for multi-agent. Hönig et al. (2018) proposed a method for multi-robot trajectory planning in known, obstacle-rich environments based on the concept of roadmaps, while it is not efficient enough. Wu et al. (2019) presented a novel approach to address the problem of multi-UAV trajectory planning only in the time domain, the original problem is decoupled into three stages to improve the efficiency. In summary, centralized trajectory generation methods attempt to jointly optimize the trajectories of all robots. Although the optimality and completeness are guaranteed, the complexities grow exponentially with the number of robots.

Decentralized trajectory generation Fiorini and Shiller (1998) pioneered a velocity obstacles (VO) approach, where a set of robot velocities that would result in a collision between the robot and an obstacle moving at a given velocity (RVO) are defined, and collision-avoidance is achieved by selecting robot's velocities out of that set. Van den Berg et al. (2008) presented a new concept called the reciprocal velocity obstacle to overcome the shortcoming of the velocity obstacles approach that results in undesirable oscillatory motions. Liu et al. (2018) explored a search-based motion planning approach that solves the problem of planning in dynamic environments for multi-agent. While those algorithms ignore the dynamic model of robots. Based on the concept of optimal reciprocal collision avoidance (ORCA) and flatness-based model predictive control (MPC), Arul and Manocha (2020) presented a collision avoidance algorithm for navigating a swarm of quadrotors in obstacle-rich scenes. Nevertheless, only the planning was verified without integrating localization and mapping capabilities.

3 System overview

3.1 System architecture

The decentralized system architecture is proposed for quadrotor swarm motion planning, as shown in Fig. 1. To avoid conflict in the group, the global paths and states are shared. The mapping process in Usenko et al. (2017) is adopted to generate a local map with UAV's moving. Then, the 3-D local map is projected into the 2-D grid map which is stored for global path planning.

Trajectory planning A priority-based RRT algorithm as the front-end of motion planning is proposed in this paper to generate a guidance path. The path can avoid the collision with obstacles in the environment and realize multiple UAVs collision-free in the local range. Then piecewise polynomial trajectory is adopted to smooth the initial path. Finally, We improve clearance and security of the trajectory by uniform B-spline optimization, which incorporates gradient information, collision avoidance constraints between multi-UAV and dynamic constraints.

Simulation environment The ground truth of UAVs can be accurately obtained by using the plugin in the Gazebo due to simulation verification. Besides, there is a VLP-16 lidar plugin with 16 lines on the quadrotor platform to obtain point cloud data, which is fused with the location information to build an occupancy map of the environment. We adopt the method proposed in Usenko et al. (2017) to generate a local environment map, which is a 3-D occupied voxel map. And the projection of the local environment map to the ground as a global map is stored, which is a 2-D map. As the paths traveled by UAVs increase, the global map is constantly expanding, while the UAV only maintains the current local map. For the controller of UAV, the dual-mode cascaded nonlinear controller proposed in our previous work is adopted (Shen et al. 2020), which ensures the hovering accuracy and trajectory tracking performance.

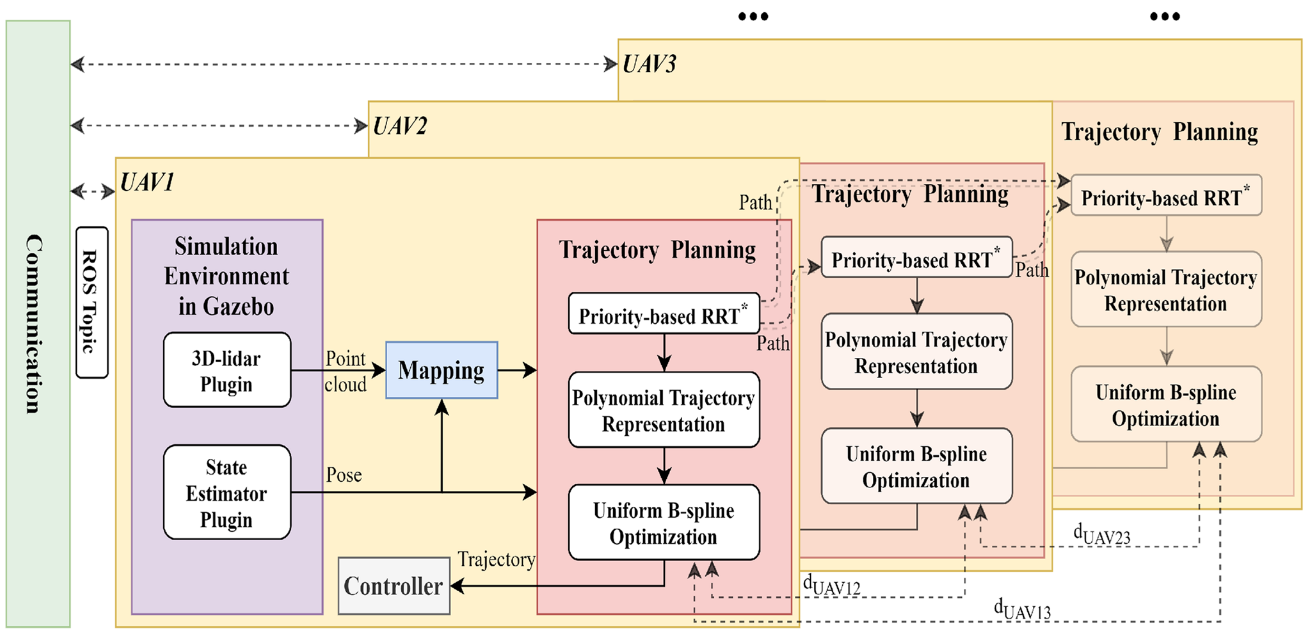

Fig. 1 The complete system architecture. The whole system is mainly composed of two parts, which are simulation environment (localization and mapping, control of a single UAV) and swarm motion planning. Suppose that UAV1 has the highest priority, UAV2 takes the second place, and UAV3 has the lowest priority. The dotted arrows in the figure indicate the information transfer between UAVs. $d_{UAV_{ij}}$ is the distance between UAVi and UAVj

3.2 Problem formulation

Consider a team of $N \in \mathcal{Z}$ quadrotors operating in an obstacle-dense workspace, which is fully unknown. The environment is defined by $\mathcal{W} \subseteq \mathbb{R}^3$, while the obstacles that have been explored in the environment are denoted as $\mathcal{O}$. The free configuration space of the UAVs is given by $\mathcal{F}=\mathcal{W}\backslash\mathcal{O}$. The state of a UAV, which is at position $x \in \mathbb{R}^3$, is defined by $\mathcal{R}(x)$. Let $s_i \in \mathcal{F}$ and $g_i \in \mathcal{F}$ represent the start and goal position of UAVi. To guarantee collision-free in the initial and terminal state for each UAV, $\mathcal{R}(s_i) \cap \mathcal{R}(s_j) = \emptyset$ and $\mathcal{R}(g_i) \cap \mathcal{R}(g_j) = \emptyset$ must be satisfied for all $i \neq j$. For the ith UAV, its trajectory is defined as f_i, where $pos(f_i) \in f_i$ denote the waypoint. There is no collision with obstacles when the following conditions are met:

$$\mathcal{R}(pos(f_i)) \in \mathcal{F}, \ \forall i \tag{1}$$

Furthermore, the collisions between any two UAVs should be avoided:

$$\mathcal{R}(pos(f_i)) \cap \mathcal{R}(pos(f_j)) = \emptyset, \ \forall i \neq j \tag{2}$$

In the swarm trajectory planning problem, we seek to generate kinodynamically feasible trajectories $f_k(k = 1, 2, \dots N)$ for quadrotor swarm, where obstacles and inter-vehicle collisions are avoided simultaneously.

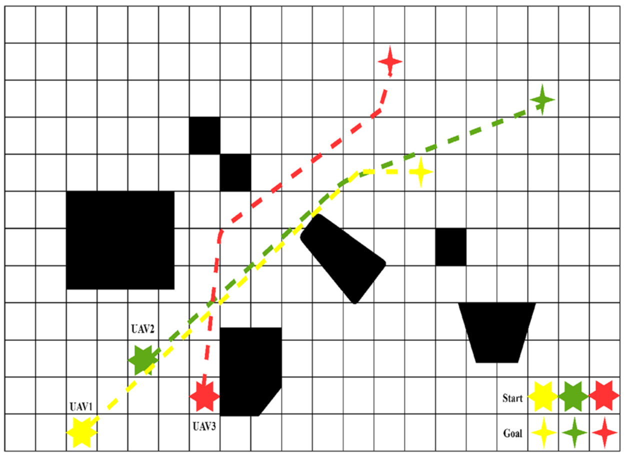

Fig. 2 The RRT algorithm is applied to the front-end of three UAVs motion planning, and the search results are represented by dotted lines with different colors. The black solid blocks represent the obstacles in the environment. The hexagon stars represent the starting points, and the quadrangle stars represent the target points. Yellow, green, and red represent UAV1, UAV2, and UAV3, respectively

4 Trajectory representation and optimization

4.1 Priority-based RRT

Simply applying RRT algorithm to search the initial path for the quadrotor swarm is unsafe. As illustrated in Fig. 2, the

RRT algorithm only considers the collision avoidance between UAV and obstacles in the environment. Based on RRT, this paper proposes a priority-based RRT algorithm, which can generate swarm paths satisfying (1) and (2) by assigning priorities. When a UAV searches a global path, it only conducts collision checking with other UAVs that have higher priority than itself. To reduce unnecessary conflict checking, only when the distance between two UAVs (the distance between the UAVi and UAVj is expressed as $d_{UAV_{ij}}$, e.g.) is less than a certain threshold d_{che}, the conflict check will be triggered. Furthermore, the current UAV only checks the collision path conflict within a certain range to improve the efficiency. For the ith UAV, the range is a circle with the current UAV as the center and r^i_{thr} as the radius.

The procedure of priority-based RRT path generation for multi-UAV is shown in Fig. 3. Take 3 UAVs as an example and suppose their priorities are: UAV1 > UAV2 > UAV3. Firstly, three UAVs independently generate their paths using the RRT algorithm and get other paths with higher priority than itself for conflict checking. UAV1 need not check for path conflicts due to its highest priority, while the other two UAVs need to. If the current UAV finds a path conflict, it will re-search the path until there is no conflict. However, the re-search process may fail within the specified time. Then the higher priority UAV with path conflict will re-generate its path. This process is repeated until all UAVs find safe paths. The final path generation result is shown by the dotted line in Fig. 3a. Although the path of UAV3 conflicts with that of the other two UAVs, it will not re-search the path because the conflict path point is not within the range r^3_{thr}. When three UAVs fly forward for a while, they reach the positions shown in Fig. 3b. At this time, the path of UAV3 crosses the path of UAV1 and is within the conflict checking range of UAV3. UAV3 will re-search for a new path ensuring it will not conflict with the paths of UAV1 and UAV2 within the scope r^3_{thr}.

The whole process of priority-based RRT is summarized in Alg. 1. Line 3–12 is a process of RRT, where n represents the preset maximum number of iterations. When the relative distance between UAVs is less than d_{che}, path collision checking will be triggered (line 13). Then the UAV will check whether its path crosses with others, which have a higher priority, within the range of the circle with its current position as the center and radius x_{init} (line 14). If no path crossing is found, the search process is successful (line 17). Otherwise, search again until success (line 15–16).

Algorithm 1: Priority-based RRT Algorithm

```
    Input: $\mathcal{F}$, $x_{init}$, $x_{goal}$
    Output: A path $\mathcal{T}$ from $x_{init}$ to $x_{goal}$
1   $\mathcal{T}.init()$;
2   while $(!find(x_{init}, x_{goal}))$ do
3       for $i = 1$ to $n$ do
4           $x_{rand} \leftarrow Sample(\mathcal{F})$;
5           $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
6           $x_{new} \leftarrow Steer(x_{rand}, x_{near}, Stepsize)$;
7           if Collision-free with obstacles($x_{new}$) then
8               $\mathcal{T}.rewire()$;
9               $X_{near} \leftarrow Near(\mathcal{T}, x_{new})$;
10              $x_{\min} \leftarrow Chooseparent(X_{near}, x_{near}, x_{new})$;
11              $\mathcal{T}.AddNodetoEdge(x_{\min}, x_{new})$;
12              $\mathcal{T}.rewire()$;
13      if $d_{UAV} < d_{che}$ then
14          Conflict checking;
15          if Conflict within a range $r_{thr}$ then
16              $re - find(x_{init}, x_{goal})$;
17      $connect(x_{init}, x_{goal})$;
18      $Success()$;
```

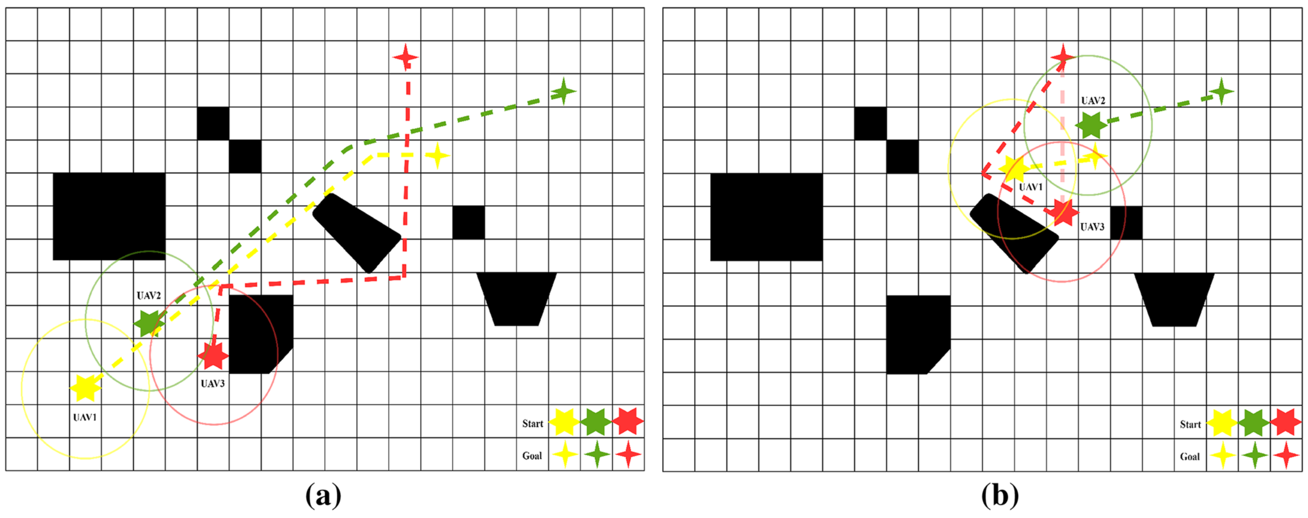

Fig. 3 Schematic diagram of priority-based RRT algorithm. The priority is UAV1>UAV2>UAV3. The radius of yellow, green, and red circles are r^1_{thr}, r^2_{thr}, and r^3_{thr}, respectively. **a** The final path search results of three UAVs. Each UAV uncross paths with others within its conflict checking scope. **b** When UAV3 finds a path conflict with UAV1 in the conflict check range, it will search the path again, and the new path will not conflict with the path of others within the range r^3_{thr}

4.2 Local trajectory planning and reciprocal collision avoidance

For each UAV, a piecewise polynomial is leveraged to represent the trajectory. The trajectory is parameterized to the time variable t in each dimension λ out of x, y, z. The Nth order M-segment trajectory of one dimension can be written as follows (Zhou 2019):

$$p_\lambda(t) = \begin{cases} \sum_{j=0}^{N} \sigma_{1j}(t-T_0)^j & , \quad T_0 \le t \le T_1 \\ \sum_{j=0}^{N} \sigma_{2j}(t-T_1)^j & , \quad T_1 \le t \le T_2 \\ \vdots & , \quad \vdots \\ \sum_{j=0}^{N} \sigma_{Mj}(t-T_{M-1})^j & , \quad T_{M-1} \le t \le T_M \end{cases} \tag{3}$$

where σ_{ij} is the jth order polynomial coefficient of the ith segment of the trajectory. $T_1, T_2, \dots, T_M$ are the end time of each segment, and the total time T_{total} is $T_{total} = T_M - T_0$.

We follow Mellinger and Kumar (2011) to construct the cost function of the piecewise polynomial trajectory. The cost function penalizing the squares of the derivatives of ith segment can be written as:

$$J_i(T_{total}) = \int_0^{T_{total}} \sigma_{i0}(t)^2 + \sigma_{i1}(t)^2 + \sigma_{i2}(t)^2 + \cdots + \sigma_{iN}(t)^2 dt = \boldsymbol{\sigma}^T Q(T_{total})\boldsymbol{\sigma} \tag{4}$$

The M-polynomial segments can be jointly optimized by connecting their cost matrices in a block diagonal way, which is written as follows:

$$J_{total} = \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \vdots \\ \boldsymbol{\sigma}_M \end{bmatrix}^T \begin{bmatrix} Q_1(T_1) & & \\ & \ddots & \\ & & Q_M(T_M) \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \vdots \\ \boldsymbol{\sigma}_M \end{bmatrix} \tag{5}$$

In this paper, we minimize the snap (4th derivative of the position) of the UAV to obtain a smooth trajectory, and the order N of the polynomial is selected as 9. Using the method proposed in Richter et al. (2016), a mapping matrix and a selection matrix are introduced to map the polynomial coefficients to the derivatives at each segment points of the piecewise polynomial. Then the coefficients of the polynomial can be obtained by the closed-form method.

However, the trajectory produced by the polynomial is often close to obstacles and other UAVs since distance information in the free space is ignored, as shown in Fig. 4. Therefore, we improve the security and clearance of the initial trajectory using uniform B-spline optimization for the trajectory function $p(t)$. To initialize the control points, we sample the piecewise polynomial trajectory at equal

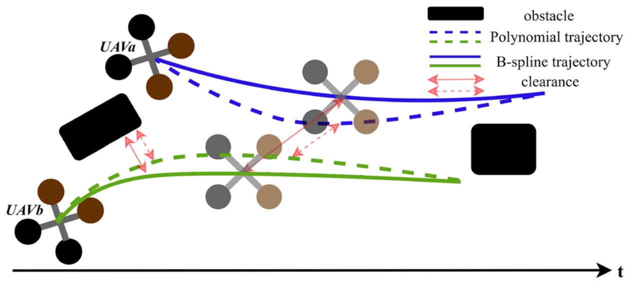

Fig. 4 Polynomial trajectories and B-spline trajectories

intervals. The value of a B-spline of degree $k-1$ can be evaluated using the following equation:

$$p(t)=\sum_{i=0}^{n} p_i B_{i,k}(t) \tag{6}$$

A B-spline trajectory is parameterized by time t, where $t \in [t_{k-1}, t_{m-k+1}]$. t_j $(j=0,1,\ldots,m)$ is the node value, which constitutes the node vector of the B-spline curve. $p_i \in \mathbb{R}^n, (i=[0,1,\ldots n])$ are control points and $m=n+k$, while $B_{i,k}(t)$ are basis functions that can be computed using the DeBoor – Cox recursive formula (Qin 2000). Uniform B-splines have a fixed time interval Δt between their control points, which simplifies computation of the basis functions. To evaluate the position at time $[t \in [t_l, t_{l+1}) \subset [t_{k-1}, t_{m-k+1}]$, we transform time to a uniform representation $u(t)=(t-t_l)/\Delta t$. Then the position can be evaluated using the following matrix representation (Zhou 2019):

$$\boldsymbol{p}(u(t))=\boldsymbol{u}(t)^T \boldsymbol{M}_k \boldsymbol{p} \tag{7}$$

$$\boldsymbol{u}(t)=\left[\,1\ \ u(t)\ \ u^2(t)\ \ \cdots\ \ u^{k-1}(t)\,\right]^T \tag{8}$$

$$\boldsymbol{p}=\left[p_{l-k+1}\ p_{l-k+2}\ p_{l-k+3}\ \cdots\ p_l\right]^T \tag{9}$$

where M_k is a constant matrix determined by k. In the case of our implementation, k is set as 6 since the quintic B-splines is used.

Refer to Usenko et al. (2017), the local replanning problem is represented as an optimization problem with smoothness cost, collision cost, quadratic derivative cost, and derivative limit cost. Moreover, an additional constraint on reciprocal collision avoidance is proposed in this paper. The overall cost function is defined as:

$$Q_{total}=Q_s+Q_o+Q_c+Q_q+Q_l \tag{10}$$

where Q_s is a smoothness cost function. Q_o and Q_c are the cost function of collision with static obstacles and reciprocal collision avoidance, respectively. Q_q is the cost function of the integral over the squared derivatives (acceleration, jerk, snap), while Q_l is a soft constraint on velocity, acceleration, jerk and snap. The detailed expressions of Q_s, Q_o, Q_q, and Q_l can be found in Usenko et al. (2017).

The cost function of reciprocal collision avoidance penalizes the trajectory points whose distance among drones is within a certain threshold d_{UAV}. Assuming there are N UAVs in total. For the ith UAV, the cost is:

$$Q_c^i=\begin{cases}\sum_{j=0,j\neq i}^{N} \lambda_c^{ij}\int_{t_{k-1}}^{t_{m-k+1}} e^{\frac{d_{UAV}{}^2-d_{UAV_{ij}}{}^2}{\tau}}dt & d_{UAV_{ij}} \le d_{UAV}\\ 0 & d_{UAV_{ij}} > d_{UAV}\end{cases} \tag{11}$$

where $d_{UAV_{ij}}$ is the distance between the ith and the jth UAV, λ_c^{ij} is a weighting parameter. In this paper, we take the same value for λ_c^{ij} no matter what the values of i and j are. τ controls the rising speed of the function. The cost is negatively correlated to the distance value. The cost function of reciprocal collision avoidance is non-convex. Nevertheless, the local minimum can be avoided due to the guidance path generated from priority-based RRT.

To allow for changes in the global paths, we adopt a receding horizon planning strategy for running the local replanning algorithm in real-time. As shown in Fig. 5, the

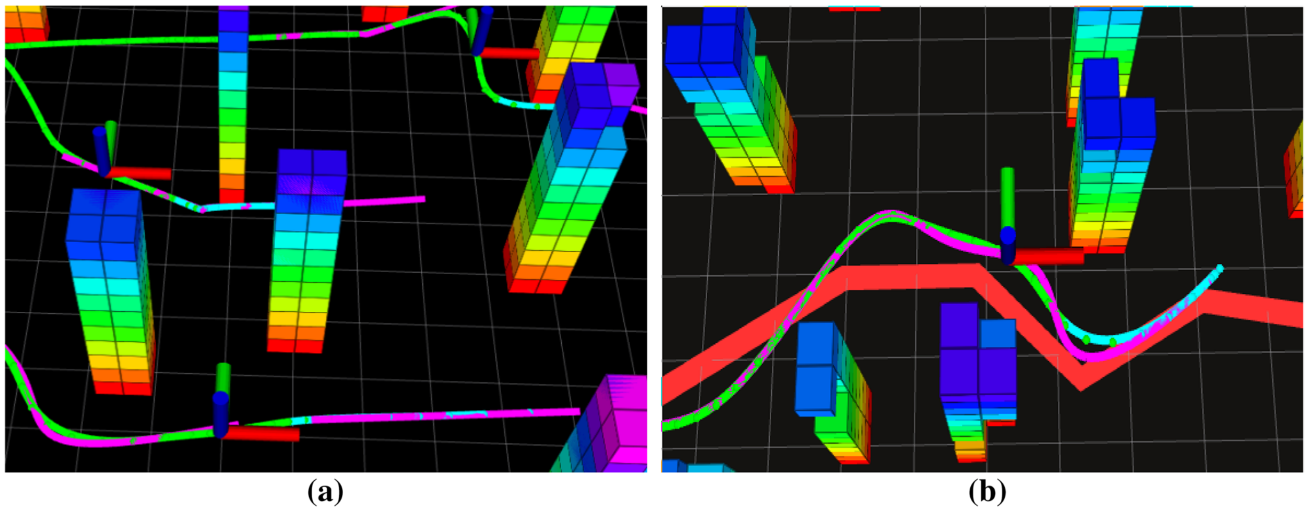

Fig. 5 The local planning strategy. The red geometry path indicates the initial path searched by the priority-based RRT algorithm and the pink curve is the polynomial trajectory. The cyan-blue curve and the green curve is the execution trajectory and planning trajectory, respectively

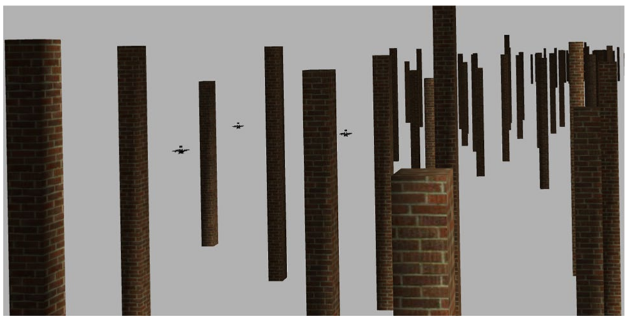

Fig. 6 Multi-UAV simulation environment in Gazebo

planning trajectory is divided into the execution part and planning part. For each iteration, we set the planning time horizon as $[t_{\min} - 2\Delta t, t_{\max})$. This horizon is decided by the number of control points that need to be optimized. The execution horizon is set as $[t_{\min} - 3\Delta t, t_{\min} - 2\Delta t)$. Then, after local replanning, the execution part of the local optimal trajectory is sent to the controller.

5 Simulation verification

5.1 System settings

The multi-UAV motion planning method proposed in this paper is implemented in C++11 with a general non-linear optimization solver NLopt. And all algorithms are executed using the robot operating system (ROS) in the Ubuntu16.04 system and run with a desktop, which has a six-core Intel i5-8500 processor running at 3.00 GHz with 8 GB RAM. The 3-D lidar module, which outputs the dense point cloud map, runs at 10 Hz and the local planner (B-spline curve) module runs at 2 Hz. The generated reference trajectory is input to the controller, and the frequency is 100 Hz. It is worth noting that the parameter settings of all drones are the same. A multi UAV simulation environment in the Gazebo simulation platform of the ROS system is built, as shown in Fig. 6.

5.2 Comparative testing

Comparison of front-end We compare the proposed front-end path searching algorithm with RRT. In the comparison, the starting point, endpoint, and the test environment of each UAV are set to be the same. The results are shown in Fig. 7. Compared with RRT, the paths obtained by priority-based RRT are more secure with taking a little more time since they uncross each other within the scope of their respective conflict checks. Benefit from the front-end paths not crossing, the computation burden from reciprocal collision avoidance constraint in the B-spline optimization process will be reduced.

Comparison of back-end A comparison of our back-end and method (Usenko et al. 2017) is conducted. The velocity limit is $1.5m/s$. For fairness, we use the same guidance path generated by priority-based RRT as the front-end. The comparison results are shown in Fig. 8. When the guidance paths of the two UAVs are close, the optimization method proposed in Usenko et al. (2017) may lead to collision (Fig. 8a, b). Since reciprocal collision as a constraint is considered, the resulting trajectories optimized by our approach are safer (Fig. 8c, d). In addition, an inverse point-to-point transition is designed to make inter-vehicle collision avoidance inevitable during flight. The simulation results are summarized in Table 1. Despite the sacrifice of a few milliseconds and costs, reciprocal collision avoidance can be guaranteed and the aggressiveness of the trajectories is maintained.

5.3 Simulation flight test

Flight tests of three UAVs in the Gazebo simulation environment are conducted. In the test, all UAVs are planned

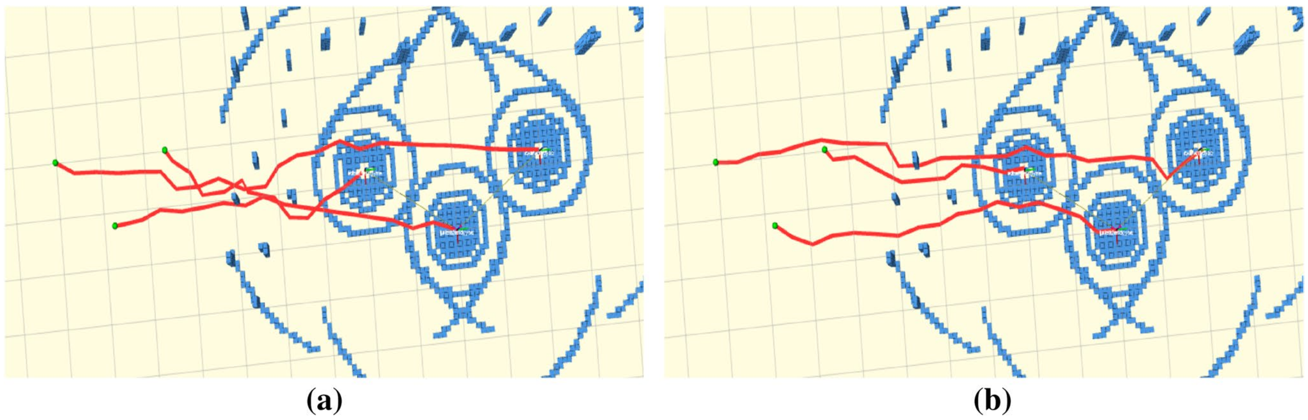

Fig. 7 **a** The initial paths of three UAVs are searched by RRT algorithm. **b** The initial paths of three UAVs are searched by priority-based RRT algorithm, where the side length of each grid is set to 1 m, $r^1_{thr} = r^2_{thr} = r^3_{thr} = 5m$. The average search time is 12.425 and 19.707 ms, respectively

Table 1 Comparison of trajectory optimization

Method	Trajectory optimization time (ms)				Total cost	Average velocity (m/s)
	UAV1	UAV2	UAV3	Average		
Method (Usenko et al. 2017)	3.866	5.647	4.199	4.571	55.299	1.076
Proposed	5.742	8.272	5.871	6.628	184.067	0.988

online in a distributed manner, and they are commanded to fly to their respective targets. Each UAV obtains the pose information of the other two UAVs so that the relative distance information between the UAVs can be calculated. State estimation, mapping, control, and trajectory generation are all performed online without any prior information about the environments. Representative figures which record the flight are shown in Figs. 9 and 10, where each UAV generates a safe and smooth trajectory and travels through complex environments autonomously. In Figs. 9 and Fig. 10, we assume that the top trajectory is generated by UAV1, the middle is generated by UAV2, and the bottom is generated by UAV3. The cyan-blue curve and the green curve is the execution trajectory and planning trajectory, respectively.

6 Conclusion and future work

In this paper, a decentralized motion planning approach for multi-UAV autonomous navigation is developed. The online motion planning problem for multi-UAV is decoupled as a front-end path searching and a back-end nonlinear trajectory optimization. A priority-based RRT, as a front-end, is proposed in this paper to generate an online global path for each quadrotor without reciprocal chiasmata and no collision with

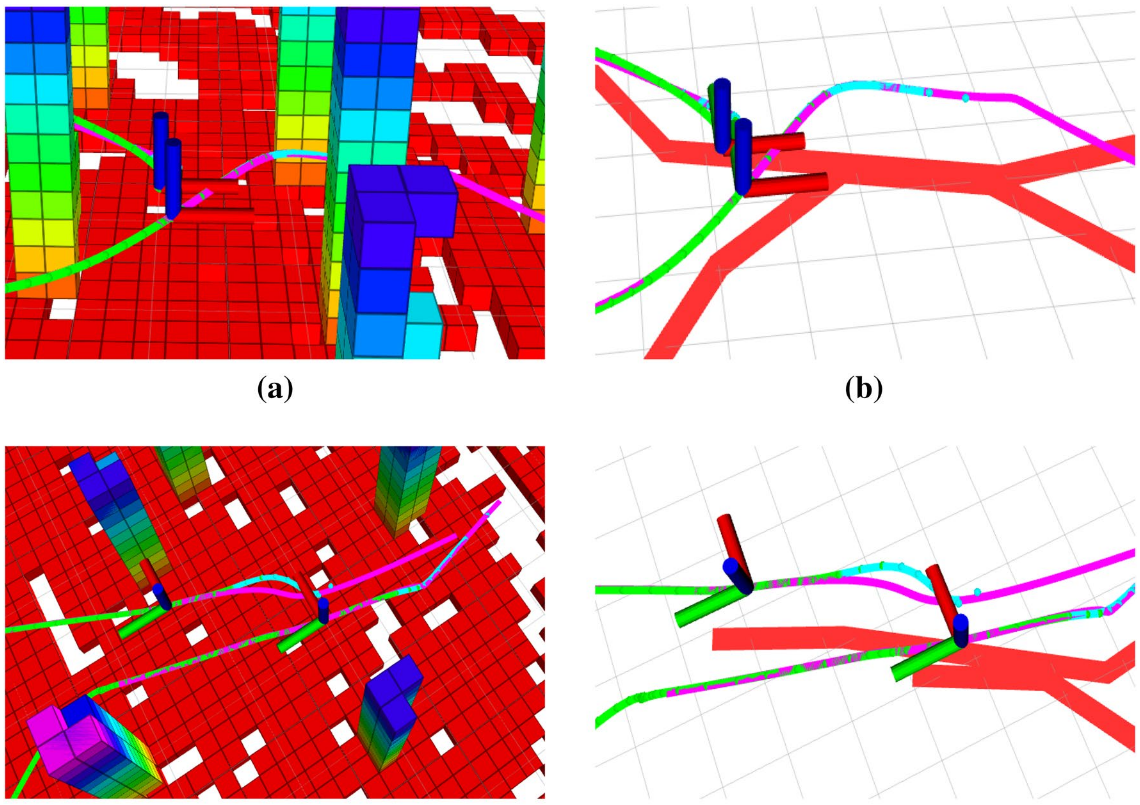
(a) (b) (c) (d)

Fig. 8 Trajectory optimization results. The red, pink, cyan-blue, and green curves represent the guidance path, polynomial trajectory, execution trajectory, and planning trajectory respectively. To make the trajectories clearly visible, the voxels of obstacles in **b** and **d** are not displayed. **a**, **b** The trajectories are optimized by the method (Usenko et al. 2017). **c**, **d** The trajectories are optimized by our method

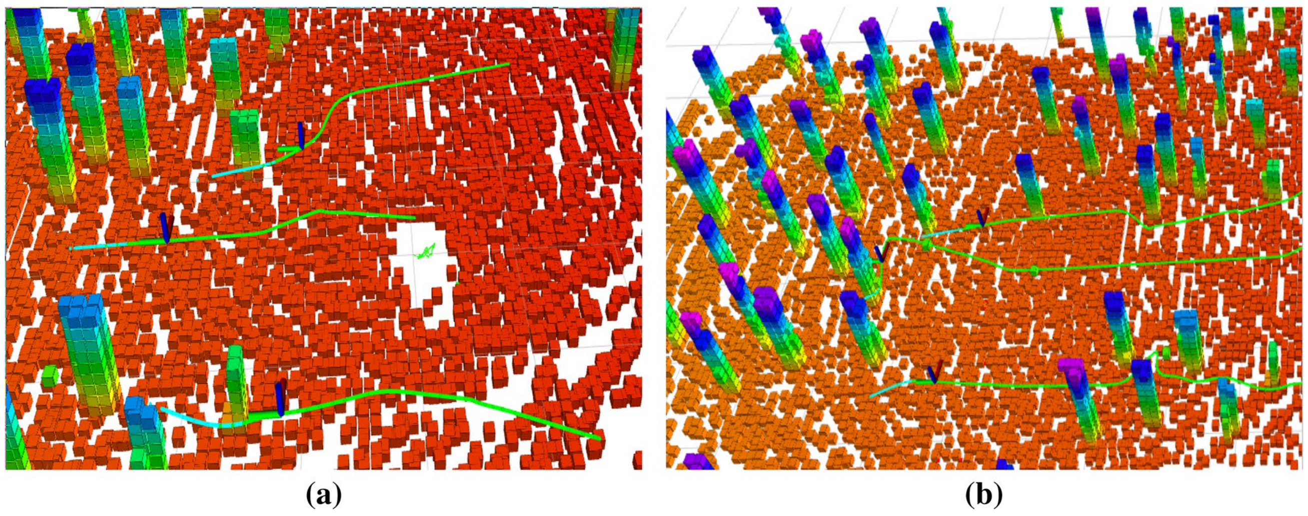

Fig. 9 Three UAVs fly in simulation. **a** UAV1 and UAV3 can successfully avoid obstacles. **b** UAV1 and UAV2 can successfully avoid the inter-vehicle collision

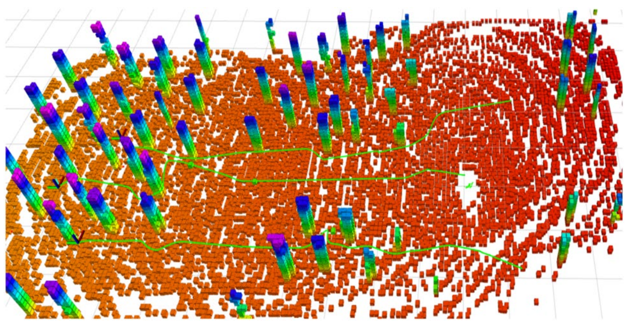

Fig. 10 Complete global map and trajectories with obstacle and collision avoidance of the three UAVs

obstacles. Then uniform B-spline curve, as a back-end, is adopted to improve the security and clearance of the trajectory with the initial value of polynomial sampling trajectory. To further improve the security, reciprocal collision as a constraint is added to the cost function to be optimized. The efficiency of the developed algorithm is validated in Gazebo.

In future work, we plan to transplant the algorithm into the fully autonomous UAV platform for an experiment in the real-world.

References

Arul, S.H., Manocha, D.: DCAD: decentralized collision avoidance with dynamics constraints for agile quadrotor swarms. IEEE Robot. Autom. Lett. **5**(2), 1191–1198 (2020)

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

Augugliaro, F., Lupashin, S., Hamer, M., et al.: The flight assembled architecture installation: cooperative construction with flying machines. IEEE Control Syst. Mag. **34**, 46–64 (2014)

Burri, M., et al.: Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1872–1878 (2015)

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

Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. **17**, 760–772 (1998)

Hamer, M., Widmer, L., Dandrea, R.: Fast generation of collision-free trajectories for robot swarms using GPU acceleration. IEEE Access **7**, 6679–6690 (2018)

Hönig, W., Preiss, J.A., Kumar, T.K.S., et al.: Trajectory planning for quadrotor swarms. IEEE Trans. Robot. **34**, 856–869 (2018)

Liu, S., Mohta, K., Atanasov, N., et al.: Towards search-based motion planning for micro aerial vehicles. arXiv:1810.03071 (2018)

Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. In: IEEE International Conference on Robotics and Automation, pp. 2520–2525 (2011)

Mellinger, D., Kushleyev, A., Kumar, V.: Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In: IEEE International Conference on Robotics and Automation, pp. 477–483 (2012)

Mohta, K., Watterson, M., Mulgaonkar, Y., et al.: Fast, autonomous flight in GPS-denied and cluttered environments. J. Field Robot. **35**, 101–120 (2018)

Park, J., et al.: Efficient multi-agent trajectory planning with feasibility guarantee using relative Bernstein polynomial. arXiv:1909.10219 (2019)

Qin, K.: General matrix representations for B-splines. Vis. Comput. **16**, 177–186 (2000)

Richter, C., Bry, A., Roy, N.: Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: Robotics Research, pp. 649–666. Springer, Cham (2016)

Saikin, D.A., Baca, T., Gurtner, M., et al.: Wildfire fighting by unmanned aerial system exploiting its time-varying mass. IEEE Robot. Autom. Lett. **5**, 2674–2681 (2020)

Shen, H., Zhang, X., Lu, H., et al.: State estimation and control for micro aerial vehicles in GPS-denied environments. In: IEEE International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1070–1075 (2020)
Tang, S., Thomas, J., Kumar, V.: Hold or take optimal plan (hoop): a quadratic programming approach to multi-robot trajectory generation. Int. J. Robot. Res. **37**(9), 1062–1084 (2018)
Usenko, V., von Stumberg, L., Pangercic, A., et al.: Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 215–222 (2017)
Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE International Conference on Robotics and Automation, pp. 1928–1935 (2008)
Van Den Berg, J., et al.: Reciprocal n-body collision avoidance. In: Robotics Research, pp. 3–19 (2011)
Wu, W., Gao, F., Wang, L., et al.: Temporal Scheduling and Optimization for Multi-MAV Planning. Hong Kong University of Science and Technology, Clear Water Bay (2019)
Zhou, B., et al.: Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robot. Autom. Lett. (RAL) **4**(4), 3529–3536 (2019)

Xuewei Zhang received the B.S. degree from Hebei University of Technology, Tianjin, China, in 2019. He is currently working toward the Master's degree in the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. His research interests include motion planning and its application in micro-aerial vehicles.

Hongming Shen received the B.S. degree in 2015. He is currently a Ph.D. student in the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. His research interests include simultaneous localization and mapping and control in micro-aerial vehicles.

Guohui Xie received the B.S. degree from Zhengzhou University, Zhengzhou, China, in 2019. He is currently working toward the Master's degree in the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. His research interests include simultaneous localization and mapping and its application in vehicles.

Hanchen Lu received his B.S. degree from Central South University in 2015. He is currently a Ph.D. student in the School of Electrical and Information Engineering, Tianjin University. His main research interests include formation design and control of micro-aerial vehicles.

Bailing Tian (M'11) received the B.S., M.S. and Ph.D degrees in automatic control from Tianjin University, Tianjin, China, in 2006, 2008 and 2011, respectively. He was an academic visitor at the School of Electrical and Electronic Engineering, University of Manchester from June 2014 to June 2015. He is currently a professor at the School of Electrical and Information Engineering, Tianjin University. His main research interests include finite time control and integrated guidance and control for vehicle.