Guidance, Navigation and Control
Vol. 1, No. 2 (2021) 2150007 (20 pages)

DOI: 10.1142/S2737480721500072

Decentralized MPC-Based Trajectory Generation for Multiple Quadrotors in Cluttered Environments

Xinyi Wang[*,§], Lele Xi[†,¶], Yizhou Chen[*,‖], Shupeng Lai[‡,**],
Feng Lin[‡,††] and Ben M. Chen[*,‡‡]

[*]*Department of Mechanical and Automation Engineering,*
The Chinese University of Hong Kong,
Shatin, N.T., Hong Kong

[†]*School of Automation,*
Beijing Institute of Technology,
Beijing, P. R. China

[‡]*Peng Cheng Laboratory, Nanshan,*
Shenzhen, Guangdong, P. R. China
[§]*xywangmae@link.cuhk.edu.hk*
[¶]*xilele.bit@gmail.com*
[‖]*yzchen@mae.cuhk.edu.hk*
[**]*shupenglai@u.nus.edu*
[††]*linfeng@gmail.com*
[‡‡]*bmchen@cuhk.edu.hk*

Received 26 April 2021
Revised 6 May 2021
Accepted 25 May 2021
Published 16 July 2021

Challenges in motion planning for multiple quadrotors in complex environments lie in overall flight efficiency and the avoidance of obstacles, deadlock, and collisions among themselves. In this paper, we present a gradient-free trajectory generation method for multiple quadrotors in dynamic obstacle-dense environments with the consideration of time consumption. A model predictive control (MPC)-based approach for each quadrotor is proposed to achieve distributed and asynchronous cooperative motion planning. First, the motion primitives of each quadrotor are formulated as the boundary state constrained primitives (BSCPs) which are constructed with jerk limited trajectory (JLT) generation method, a boundary value problem (BVP) solver, to obtain time-optimal trajectories. They are then approximated with a neural network (NN), pre-trained using this solver to reduce the computational burden. The NN is used for fast evaluation with the guidance of a navigation function during optimization to guarantee flight safety without deadlock. Finally, the reference trajectories are generated using the same BVP solver. Our simulation and experimental results demonstrate the superior performance of the proposed method.

Keywords: Multi-quadrotor systems; motion planning; motion primitive; model predictive control.

1. Introduction

Trajectory planning for multiple quadrotors is key to execute missions in cluttered environments. In particular, multi-quadrotor tasks are especially challenging due to many decision-making agents sharing the same space. In such settings, the planning algorithms must compute collision-free and goal-oriented trajectories taking into account of the neighboring agents and environment. Furthermore, the requirements of excellent flexibility, flight efficiency and speed for multiple quadrotors system make high demands on planning methods, which should have good computational efficiency and less execution time of trajectory for actual implementations.

The optimization solvers can be classified into two categories, gradient-based and gradient-free optimization, respectively. In practice, the trajectory generation optimization is generally a complex and often non-convex problem, in which gradient-based solvers are easy to get stuck at a local optimal point. It thus requires a good initial guess to alleviate such a drawback (see e.g. Ref. 1). Some optimization problems, on the other hand, cannot be formulated with gradient information. For example, the line-of-sight cost may result in a discontinuous cost function. Gradient-free methods solve such an optimization problem without gradient information and guarantee policy improvement by estimating and compromising among different directions (see e.g. Ref. 2).

In addition to the type of solvers, there are a wide variety of techniques to tackle the multi-quadrotor trajectory generation problem. Sequential convex programming (SCP) has been applied to generate trajectories for multi-agent systems in convex domains.[3] In Ref. 3, Augugliaro *et al.* proposed a local planner by solving an SCP problem sacrificing safety for a real flight system. In Ref. 4, Robinson *et al.* combine sequential planning with nonlinear constraints for nonlinear systems, which provides better scalability by sacrificing optimal solutions. Other applications of SCP are trajectory optimization and target assignment[5] and formation payload transport.[6] Although it shows good performance when planning a small team, the required decoupling process may substantially increase computational complexity with decreased success rate when dealing with a large team of multiple quadrotors.It is used to replace non-convex constraints with convex ones forming as over-constrained optimization problems, thus always fail to find feasible solutions in complex environments. MPC-based methods have been proven effective for the motion planning of autonomous vehicles in complex environments. A centralized nonlinear model predictive control (NMPC) method (see, e.g. Ref. 7) has good theoretical properties such as guaranteed optimality to generate multiple quadrotors trajectories. They formulated the problem as an optimal control using mixed-integer quadratic programs (MIQPs) with integer collision avoidance constraints. However, this approach requires separate integer variables for every face of every obstacle, which causes the mixed-integer formulation intractable. The computational complexities limit its applications merely feasible for small teams of autonomous agents with few obstacles. In order to solve this problem, other methods for distributed motion planning have

been proposed (see, e.g. Refs. 8 and 9). In Ref. 8, Derenick and Spletzer formulated a second-order cone programming problem for multi-quadrotors collision-free planning through static obstacles. This approach triangulates the free 2D space to convexify static obstacle avoidance constraints and computed the optimal motion in formation. Although, in Ref. 9, Alonso-Mora *et al.* used a more general geometric-based optimization approach by computing the convex hull of the formation. The multi-quadrotors need to share information with their neighbors, which limits their applications. Besides, other NMPC methods solving formation control problems were proposed for pursuing evasion and static obstacle avoidance (see, e.g. Refs. 10 and 11).

The distributed model predictive control (DMPC) approach[12] is developed due to its abilities to handle constraints and achieve good performance for task coordination (see e.g. Refs. 13 and 14). The result in Parys and Pipeleers[15] shows that parallel computing can also be combined with this method to reduce the run time when quadrotors are updating their predicted states. In terms of tracking and formation problems, Wang and Ding[16] presented a synchronous DMPC scheme using the estimated information of other quadrotors to avoid collisions. However, with the increasing of environments and task complexity, it is hard to handle state and input constraints well and guarantee trajectory feasibility. Real-time trajectory generation is required for quick adaptation in complex environments, but it remains challenging to implement for robot swarms. Most obstacle avoidance techniques for trajectory generation are either centralized or sub-optimal (see e.g. Refs. 17 and 18) usually with high trajectory execution time.

Moreover, there are many methods formulating trajectory generation as a nonlinear optimization problem that takes smoothness and safety into account. Motion primitives (MPs)-based local planning methods for mobile robots are also frequently applied to abstract the continuous state space.[19] The MPs along the whole trajectory are sampled on the vehicle's boundary state constraints (i.e. jerk), and then generate the actual motion by solving a boundary value problem (BVP).[20] In addition, considering the time efficiency, the jerk limited trajectory (JLT) has been proven well suited for quadrotors.[21] It can handle arbitrary initial position, velocity, and acceleration, resulting in a smooth and time-optimal trajectory from the current state to the next target state. Furthermore, its closed-form solution can also save the computational time.[21]

Inspired by the existing researches, in this paper, we develop a novel trajectory generation method by solving a non-convex optimization problem to achieve distributed cooperative motion planning with considering deadlock and flight efficiency for multiple quadrotors. The main contribution of our work is to propose a decentralized and meta-heuristic, a gradient-free motion planning framework based on MPC which allows for fast trajectory generation for multiple quadrotors in cluttered environments. Unlike gradient-based method, the replanning time is more uniform and thus can improve the success rate. More specifically, we use JLT approximated with NN to reduce the time consumption of trajectories and rapidly generate

trajectories with a less computational burden. Simulation and experimental results show that for different environments and boundary states, our method can generate dynamically feasible trajectories for multiple quadrotors and guide them to achieve their goals in an obstacle-dense environment without deadlocks. The proposed framework is tested using actual quadrotors, and the flight experiments are carried out in cluttered environments with static and dynamic obstacles to verify the planning performance.

The rest of the paper is organized as follows. We present in Sec. 2 some preliminary knowledge and the problem formulation, and in Sec. 3 we propose our MPC-based trajectory generation framework and analyze the detailed techniques of the trajectory generation problem for multiple quadrotors. The experimental results and analysis are then given in Sec. 4 to validate the proposed technique. Finally, we draw some concluding remarks in Sec. 5.

2. Problem Formulation

We consider the trajectory planning problem for a multi-agent quadrotor system with K quadrotors in the 3D space $\mathcal{X} \subset \mathbb{R}^3$. Every quadrotor k with $k \in K$ is assumed to obey the same dynamic limits and its dynamics is differentially flat as adopted in Mellinger and Kumar.[22] The quadrotor dynamics has inputs $\sigma_k = [p_k, \psi_k]^{\mathrm{T}}$, where $p_k \in \mathbb{R}^3$ is the position of the mass center of quadrotor in the world frame, and ψ_k is the yaw angle. This allows us to plan the trajectory of quadrotor k independently using a triple integrator with its position p_k, velocity v_k, acceleration a_k, and jerk j_k, respectively. Let $x_k = [p_k, v_k, a_k]^{\mathrm{T}}$ be the state variable with invariant constraints $v_k \in [v_{\min}, v_{\max}]$, $a_k \in [a_{\min}, a_{\max}]$ and $j_k \in [j_{\min}, j_{\max}]$ which might be different for the horizontal and vertical axes. $u_k = j_k$ is defined as the control input and Δt is the sampling interval. Thus, the discrete-time linear model can be defined as follows:

$$x_k[n+1] = A_k x_k[n] + b_k u_k[n], \tag{1}$$

where

$$A_k = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad b_k = \begin{bmatrix} \Delta t^3/6 \\ \Delta t^2/2 \\ \Delta t \end{bmatrix}. \tag{2}$$

We aim to generate a continuous, smooth, collision-free and kinodynamic feasible trajectory $\mathcal{F}_k : [0, t_{\text{goal}}] \to \mathbb{R}^3$ of each quadrotor k, where t_{goal} represents the terminal time at which the last quadrotor in the team reaches its goal. The state vector of each quadrotor is denoted as $x_k(t) \in \mathcal{X}$ at runtime t. All quadrotors need to reach their destinations from their starting points, i.e. $x_k(0) = s_k$ and $x_k(t_{\text{goal}}) = g_k$, where s_k and g_k are, respectively, the initial and terminal state of each quadrotor.

The cluttered environment is modeled by an occupancy grid map. The static obstacles are denoted by a set $\mathcal{O}^{\text{static}}$ which might contain non-convex obstacles.

The dynamic obstacles are defined as a set of convex moving obstacles and removed from the static map. We denote each moving obstacle i as an area $\mathcal{O}_i(t) \subset \mathcal{X}$ at time t. A set $i \in \mathrm{I} := \{1, \ldots, N\}$ represents all moving obstacles, where N is the number of moving obstacles. Other quadrotors can be regarded as a part of dynamic environment for one specific quadrotor. It is assumed that we can predict the trajectory of each quadrotor and penalize the relative distance of them to avoid crashes. Thus, we can construct an set at time instant t given by

$$\mathcal{O}_t^{\text{dynamic}} = \bigcup_{i \in \{1,\ldots,N\}} \mathcal{O}_i(t). \tag{3}$$

The free configuration space for a single quadrotor is defined as

$$\mathcal{X}_{\text{free}} = \mathcal{X} \backslash (\mathcal{O}^{\text{static}} \cup \mathcal{O}_t^{\text{dynamic}}). \tag{4}$$

All trajectories are considered collision-free if, for each quadrotor, there are no collisions between the quadrotor and environment:

$$x_k(t) \in \mathcal{X}_{\text{free}}, \quad \forall\ 0 \le t \le t_{\text{goal}}. \tag{5}$$

We adopt the following an optimization cost function for generating control input sequence u_k:

$$\begin{aligned} \min J_k = & \int_{t_0}^{t_0+T} \{w_1 u_k^2(t) + w_2 \|x_k(t) - g_k\|^2 + w_3 e^{-\|d_o^k(t)\|} + w_4 \|d_s^k(p_k(t))\|^2\} dt \\ = & \text{ s.t. } \quad \dot{x}_k(t) = f(x_k(t), u_k(t)), \\ & \qquad g(x_k(t), u_k(t)) < 0, \\ & \qquad h(x_k(t), u_k(t)) = 0, \end{aligned} \tag{6}$$

where w_1, w_2, w_3, w_4 are used to trade off these four penalties. For the constraints of trajectory generation problem, $f()$ is the nominal model of the quadrotor defined by Eqs. (1) and (2). Inequality constraint $g()$ indicates the limit interval of velocity, acceleration and jerk, respectively, for three axes. The boundary state constraint $h()$ regulates the triple integrator from an initial state configuration s_k to an assigned goal state g_k.

Here we note that the control objective, the cost function has three main components: (i) The first term (an input variation penalty) is to penalize the square of jerk to generate smooth trajectories. The total control efforts should be minimized to meet the requirements for real flight applications; (ii) The second term (a desired set position penalty) is to minimize the deviation from the current state to the desired goal state; (iii) finally, the last term (a collision-free penalty) is flight safety cost of moving obstacles to handle the collision of potential threats. Differing from the geometrical constraints, we can obtain the relative closest distance $d_o^k(t)$ at time t

between the predictive trajectories of quadrotor and moving obstacles. Besides, for flight safety cost of static and non-convex obstacles, we calculate a potential function over a Euclidean Distance Transform (EDT) map giving the quadrotor global vision to avoid suffering from deadlock. The cost to the goal point of every grid for each quadrotor $d_s^k(p_k(t))$ is a value stored on the map. As shown in Fig. 1, this collision

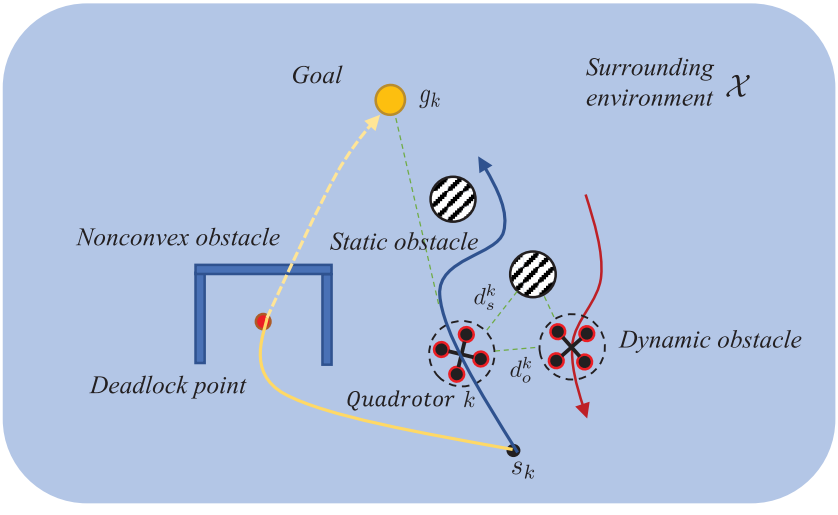


Fig. 1. The flight safety of quadrotor k.

cost pushes the quadrotor away from static and dynamic obstacles to satisfy the requirement of Eq. (5).

By solving the optimization problem, a locally optimal sequence of commands during each predictive horizon can be attained to drive each quadrotor to reach its desired destination while avoiding collisions. Based on above optimization process, we can find feasible trajectories for this decentralized and asynchronous multiple quadrotors system in cluttered environments.

3. Framework of Multi-quadrotor Trajectory Generation

In this section, we present a novel trajectory generation framework for multiple quadrotors in dynamic obstacle-dense environments. A synchronous and decentralized non-convex optimization procedure based on MPC is proposed to achieve distributed cooperative motion planning with considering deadlock and flight efficiency. The overall structure of our proposed method is depicted in Fig. 2, which consists of offline training Neural Network process and MPC-based optimization.

The proposed framework consists of the following two stages:

(1) Training NN stage: We construct the BSCPs, a trajectory library, designed through JLT generation method, a BVP solver, with the given start and goal

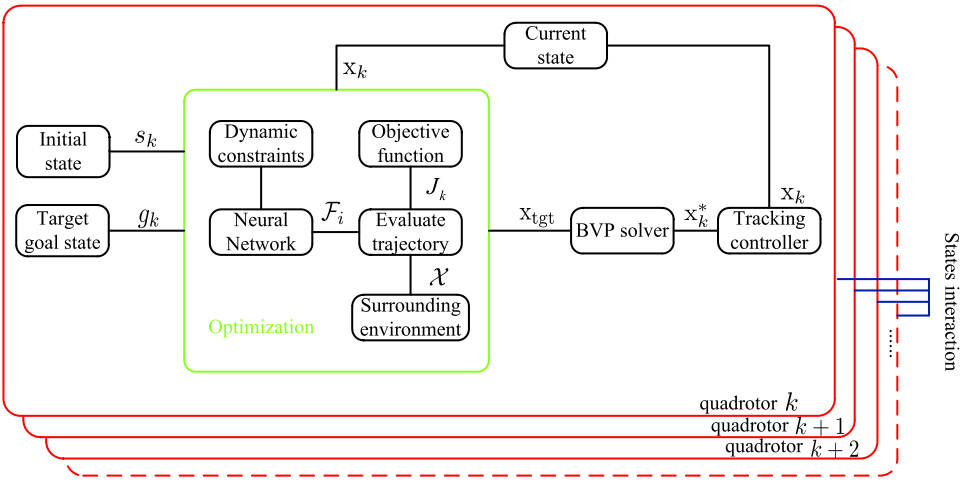


Fig. 2. The overall structure of the proposed method for multi-quadrotor motion planning.

states. Thus, an NN can be pre-trained to approximate the generated BSCPs to save the time consumption.

(2) MPC stage: Given the objective function J_k, we can fast evaluate all candidate end states of quadrotors which are approximated by means of NN with consideration of the surrounding environments $\mathcal{X}$. After this optimization process, we can obtain the end-state constraints x_{tgt} among these candidates to minimize the cost function J_k given in Eq. (6) at each step. In addition, we also add guidance map to help the quadrotors to avoid deadlock. After obtaining x_{tgt}, a series of reference state x_k^* can be generated through JLT and only the first few states will be given to the low level dynamic controller. Thus, recomputing the system state and a current state x_k can be returned for the next planning horizon.

3.1. *Construction of BSCPs using JLT*

In this subsection, we present detailed procedures for the offline training stage. To prepare for training data, we use a BVP solver associated with the JLT generation method to create a certain range of trajectories of quadrotors in the state space, which are formulated as BSCPs.

The JLT method can solve the trajectory optimization problem and obtain an analytical solution under the condition of limited computing power. Specifically, it can compute the time-optimal trajectory to a set goal $x(t_{\text{end}}) = [p_{\text{ref}}, v_{\text{ref}}, a_{\text{ref}}]$ for the quadrotor system in Eqs. (1) and (2) with arbitrary initial state values $x(0) = [p_0, v_0, a_0]$ and with physical limits $v_{\max}, a_{\max}, j_{\max}$. The jerk profile is given as $j \in \{j_{\min}, j_{\max}, 0\}$ and the state variables $x(t)$ can be obtained by integrating the jerk profile $j(t)$ at time index t. The time-optimal JLT generation problem can then be

formulated as

$$\begin{aligned}
\min \quad & t_{\text{end}} \\
\text{s.t.} \quad & p(0) = p_0, \quad p(t_{\text{end}}) = p_{\text{ref}}, \\
& v(0) = v_0, \quad v(t_{\text{end}}) = v_{\text{ref}}, \\
& a(0) = a_0, \quad a(t_{\text{end}}) = a_{\text{ref}}, \\
& \dot{p}(t) = v(t), \\
& \dot{v}(t) = a(t), \\
& \dot{a}(t) = j(t), \\
& -v_{\max} \leq v(t) \leq v_{\max}, \\
& -a_{\max} \leq a(t) \leq a_{\max}, \\
& -j_{\max} \leq j(t) \leq j_{\max}.
\end{aligned} \tag{7}$$

The algorithm will calculate the covered area of the velocity that equals the desired change in position. The time distribution can be divided into the ascent phase, descent and cruise phases, respectively. The problem is to determine the velocity profile and switching times subject to the state and input constraints. The procedure to determine the shape of the velocity profile is shown in Fig. 3. First, it is to accelerate velocity to the maximum with constant jerk $-j_{\max}$, and then to decelerate the speed to zero with $j_{\max}$ to determine whether the end position exceeds the desired position. If the end position is undershooting compared with desired position, i.e. the triangular case in Fig. 4, then we use bisection searching algorithm given in Ref. 24 to calculate the acceleration time t_{ascent} and deceleration time t_{descent}. Otherwise, it is corresponding to the trapezoidal case in Fig. 4, we can also use the position area to determine the acceleration time t_{ascent}, cruise time t_{cruise} and deceleration time t_{descent}. A closed-form and smooth trajectory can be obtained through the above method.

After generating the BSCPs, we then implement the trajectory approximation and evaluation system using a pre-trained NN in Ref. 20, which is called S_{NN} to save computational time for real flight implementations. The inputs to the network are 9×10^5 randomly samples which consist of both the initial and goal states. The corresponding trajectory generated from the JLT as output is a 41×1 vector. The detailed information of the network architecture is shown in Table 1. The accuracy of the NN meets our requirements with an average error of 0.00052 compared with the BVP solver. We should note that the purpose of using the NN approximation is to reduce computational burdens for real implementations. We have chosen a simple and classic network structure, i.e. MLP, as it proves to yield sufficiently accurate approximations for our needs. Other neural networks should also work well, but might require longer training time.

3.2. *Construction of guided map*

At the MPC stage, we use a guidance map to predict information of future states navigating the quadrotor to choose an optimal action. We follow the work of

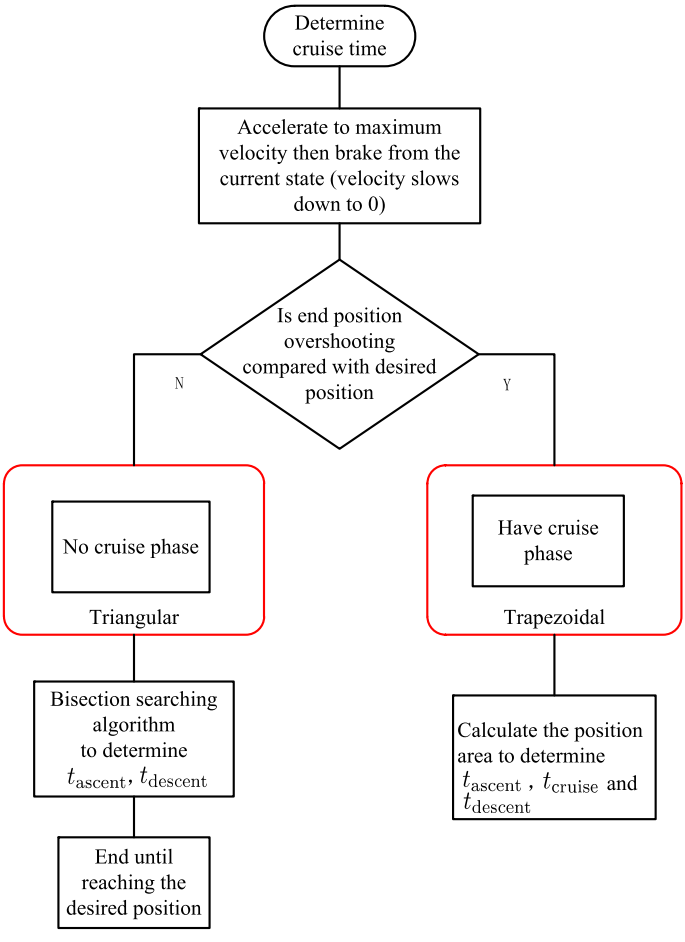
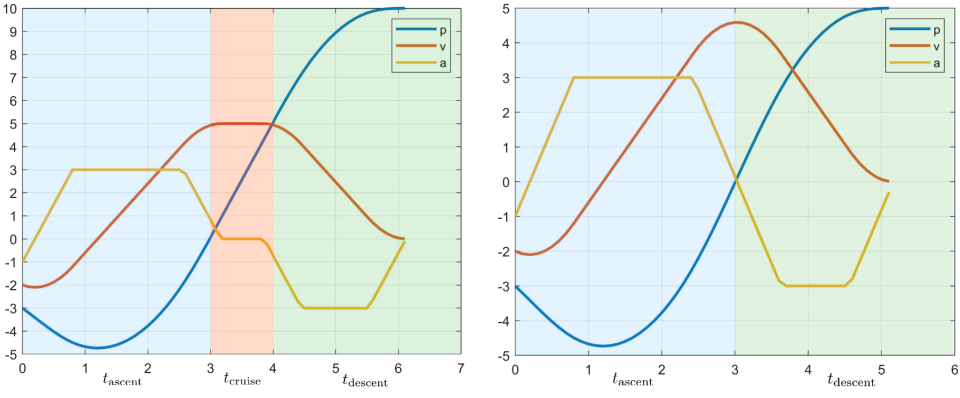


Fig. 3. Selection correct canonical velocity profile type.

Lau *et al.*[23] to use an EDT module to represent a surrounding environment. The closest distance of the static obstacles and index for every cell can be efficiently obtained through this map. In this subsection, we propose a method to calculate a potential function from a set of goals to the current positions of the quadrotor combined with an EDT map. We use the Dijkstra algorithm to calculate the cost value c_g to the goal point for each cell p_i, which is the definition of the potential function over the state space. Index i is associated with the revolution of the map. The potential function has the ability to push the quadrotor away from the obstacles and guide it to the goal. Besides, it is quite suitable for the quadrotor, a holonomic system, as this type of robot can freely move in any direction. After constructing the EDT map and potential function, we can combine them together, so the cost value of a cell $\mathcal{M}(p_i)$ consisting of a cost-to-goal value c_g and a safety value c_s with



(a) Trapezoidal case (b) Triangular case

Fig. 4. Three types of velocity profile.

Table 1. Illustration of network architecture used in this work.

	Horizontal coordinate	Vertical coordinate
NN structure	64–128–128–41 MLP*	64–128–128–41 MLP*
Training set size	9×10^5	9×10^5
Test set size	3×10^5	3×10^5
Batch size	32	32
Epoch	15	15
Average Mean Squared Error	5.73×10^{-4}	4.64×10^{-4}

Note: *MLP stands for the multi-layer perception.

consideration of the closest distance to obstacles can be calculated as

$$\mathcal{M}(p_i) = c_s(p_i) + \alpha_g c_g(p_i), \tag{8}$$

where α_g is a weight factor that trades off the relative importance of the cost-to-go value, as seen in Fig. 5. It can guide the local planner to choose appropriate actions u^* to the lowest-cost path of a map at every point during execution, which considers both environments and goal information, thus avoids falling into local minimal, as seen in Eq. (7).

$$u^* = \underset{u}{\operatorname{argmin}}\, \mathcal{M}(p_i). \tag{9}$$

3.3. *Trajectory generation strategy*

We combine the BSCPs with a gradient-free technique, i.e. the particle swarm optimization (PSO), for trajectory planning. This method is capable of finding a feasible solution for each quadrotor in the team to either non-convex or non-continuous problem without reaching the maximum iterations. The other advantage of such meta-heuristic algorithms is that the per iteration time is quite uniform thus avoids

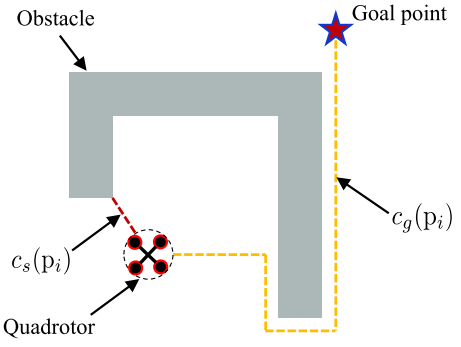


Fig. 5. The illustration of deadlock avoidance.

failure greatly. The detailed optimization process is shown in Algorithm 1. First, calculate the map $\mathcal{M}_k$ using the method in the previous subsection (Line 2). Then, perform random initialization of particles in the search space (Line 3). Each particle

Algorithm 1. Kinodynamic MPC planner using Particle Swarm Optimization

Input: The surrounding environment $\mathcal{X}$, each quadrotor's state x_k, and corresponding target's goal state g_k;

Output: Trajectory of each quadrotor $\mathcal{F}_k$;

1: **for** Each planning horizon **do**
2: $\mathcal{M}_k = \text{getGuidedMap}(g_k, \mathcal{X})$;
3: ParticleInitialization();
4: **for** Each x_{tgt_i}, $i = 1, 2, \ldots$ **do**
5: $\delta_i = \delta_i + \omega_1 * (x^*_{\text{tgt}_i} - x_{\text{tgt}_i}) + \omega_2 * (x^g_{\text{tgt}} - x_{\text{tgt}_i})$
6: $x_{\text{tgt}_i} = x_{\text{tgt}_i} + \delta_i$
7: $\mathcal{F}_i = \text{approxTrajectory}(x_k, x_{\text{tgt}_i}, S_{\text{NN}})$;
8: $\text{cost}_i = J_k(\mathcal{F}_i, \mathcal{M}_k)$
9: **if** $\text{cost}_i < \text{cost}^*_i$ **then**
10: $\text{cost}^*_i = \text{cost}_i$
11: $x^*_{\text{tgt}_i} = x_{\text{tgt}_i}$
12: **end if**
13: **if** $\text{cost}_i < \text{cost}^g$ **then**
14: $\text{cost}^g = \text{cost}_i$
15: $x^g_{\text{tgt}} = x_{\text{tgt}_i}$
16: **end if**
17: **end for**
18: $x_{\text{tgt}} = x^g_{\text{tgt}}$
19: $\mathcal{F}_k = \text{JLTTrajGeneration}(x_k, x_{\text{tgt}})$;
20: Return $\mathcal{F}_k$;
21: **end for**

represents a terminal state constraint for one planning horizon. Next, update the velocity δ_i and position x_{tgt_i} of particles according to the equation in this algorithm (Lines 5–6), where ω_1 and ω_2 are acceleration constants and $x^*_{\text{tgt}_i}$ and x^g_{tgt} represent the best position experienced by the particle i and the best position experienced by the particles group for the previous interactions. After updating the state of particles, the trajectory $\mathcal{F}_i$ can be approximated using S_{NN} for fast evaluation. The objective function will consider the information of trajectory and the surrounding environment to get the cost value cost_i (Lines 7–8). We can then update the local best and global best position of particles (Lines 9–16) to get the optimal end state, where cost^*_i represents local best cost value and cost^g is the corresponding global best value. The cost function J_k given in Sec. 2 handles the various tasks such as internal collisions among quadrotors, avoidance of deadlock and static obstacles, and navigation to the goal target. For the internal collision, we treat it as a dynamic obstacle avoidance problem by using the NN to predict the trajectories of other quadrotors and add them to the cost function. The feasible solution x_{tgt} will be found after multiple interactions, which is used in a BVP solver to obtain a $\mathcal{F}_k$ of each quadrotor (Lines 18–20).

4. Simulation and Experimental Results

In this section, we present our simulation and experimental results to demonstrate the feasibility and robustness of the proposed framework of motion planning for multi-quadrotors in dynamic cluttered environments. The BSCPs and NN used for the MPC-based trajectory generation are those given in Sec. 3. The PSO algorithm is used to solve the multi-objective optimization that includes the penalty term consisting of the square of a jerk, which results in a smoother response and saves the control efforts of multi-quadrotors. We evaluate different settings for the parameters in our planner and compare its performance for different tasks against the state-of-the-art motion planners in the literature.

4.1. *Implementation details*

As shown in Fig. 6, we use Crazyflie, a small, versatile quadcopter, as the experimental platform to verify the proposed method for motion planning of multi-quadrotors. The real flight experiments are carried out in an indoor VICON environment. The VICON system provides localization and obstacles sensing for each quadrotor. The planning is done on a laptop with an Intel I7 CPU, and the generated trajectories that includes position, velocity, and acceleration are sent to Crazyflies via a Crazyradio PA.

The following are three scenarios conducted through simulations and experiments:

- Scenario 1: Each quadrotor must reach its destination while avoiding obstacles in cluttered environments containing several pillars.

(a) Crazyflie (b) Experimental environment

Fig. 6. The flight platform and experimental environment.

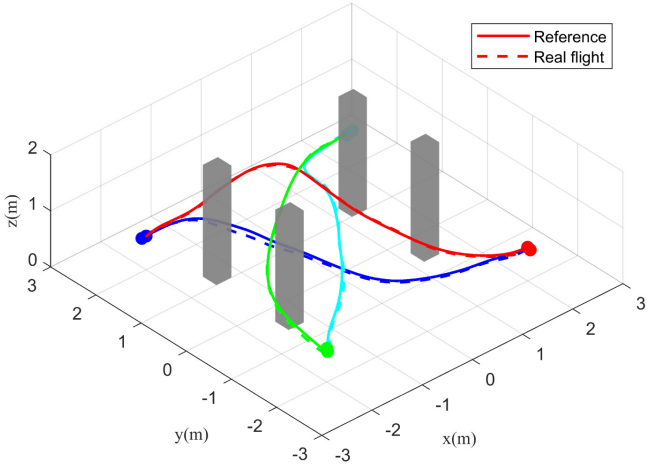
- Scenario 2: Each quadrotor must reach its destination while passing through a bridge and guaranteeing the flight safety simultaneously in the vertical axis.
- Scenario 3: Each quadrotor must reach its destination while avoiding deadlock in cluttered environments containing non-convex obstacles.

4.2. *Simulation and flight experiment*

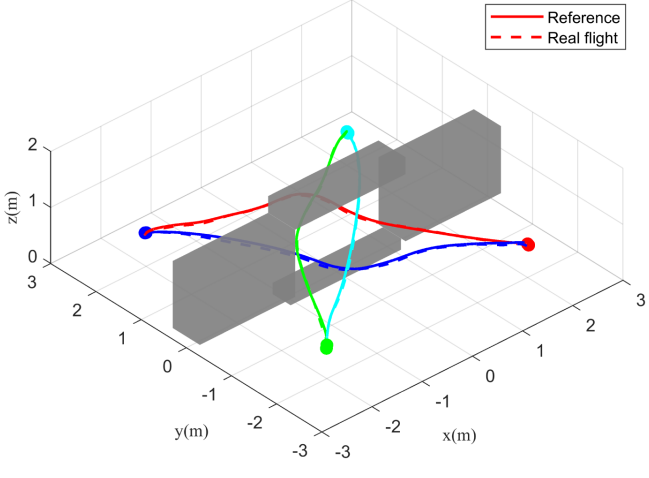
To obey the limited physical performance, the kinodynamic feasibility constraints are specified on the velocity, acceleration, and jerk with $v = [2.0, 2.0, 1.0]$, $a = [3.0, 3.0, 2.0]$, and $j = [5.0, 5.0, 5.0]$, which consist of the maximum horizontal limit, minimum and maximum vertical limits, respectively. The environment is represented as a 3D grid map and is then processed to the guided map in Sec. 3. For the PSO algorithm, 40 candidates of population are iterated over 20 times to find the best candidate solution. The model predictive planning along the process is executed at 5 Hz with a prediction horizon of 2 s.

Figure 7 shows cases that the four quadrotors start from their initial positions to their antipodals while avoiding collision with other quadrotors and static obstacles in the surrounding environment. All the quadrotors avoid the obstacles and converge to their antipodal positions in about 4.8 s. The maximum speed reached is 1.99 m/s with an average speed of 0.93 m/s.

As it can be clearly seen in Fig. 8, the six quadrotors q_i, $i = 1, 2, \ldots, 6$, carry out the antipodal position swapping in the environment containing several pillars and non-convex obstacles which may result in local minima in optimizing the objective function. In Fig. 8(a), Some exciting results can be found that quadrotors, benefiting from the potential function, reach their goal points, and avoid the deadlock simultaneously. As a contrast, in Fig. 8(b), the quadrotor q_5 falls into local minima, and it cannot reach its goal successfully without the potential function. Both scenarios show that multi-quadrotors can achieve tasks successfully. Detailed flight experiments can be found in the video clips at `https://youtu.be/QgHfa2dgvv8`.



(a) An environment with several pillars



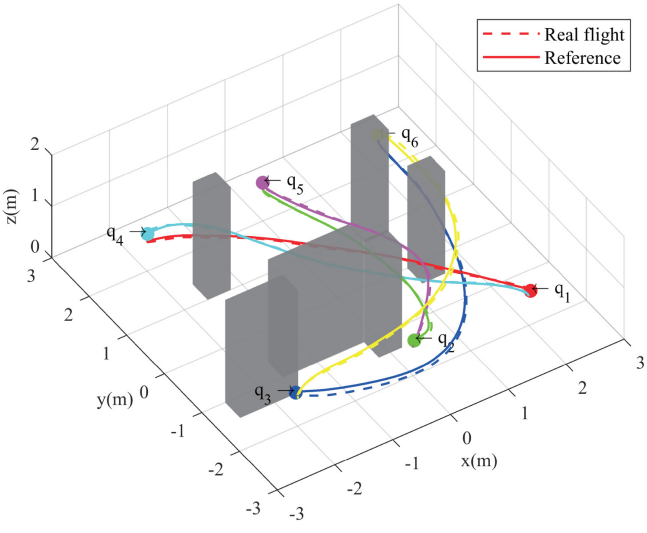
(b) An environment with bridge openings

Fig. 7. Quadrotor trajectories during antipodal position swapping. In both (a) and (b), the solid circles denote four quadrotors and the solid curves denotes their trajectories, respectively.

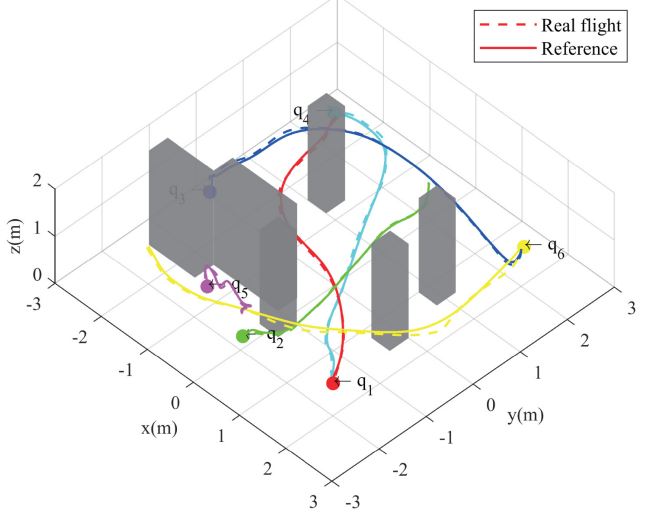
4.3. *Comparison with other multi-quadrotor trajectory planning method*

We compare our method with the methods in Augugliaro *et al.*,[3] Park *et al.*[18] and Lai *et al.*[20] on a 10 m × 10 m × 2.5 m obstacle-free environment using 8 quadrotors and the same initial and terminal states. Detailed simulation results are shown in Fig. 9 and Table 2.

(1) Safety Ratio: The safety ratio is defined as $\min d_o^k/r_c$ for all k, where r_c is an expanding radius of a quadrotor. We have tested the flight safety 20 times and obtain



(a) With potential function

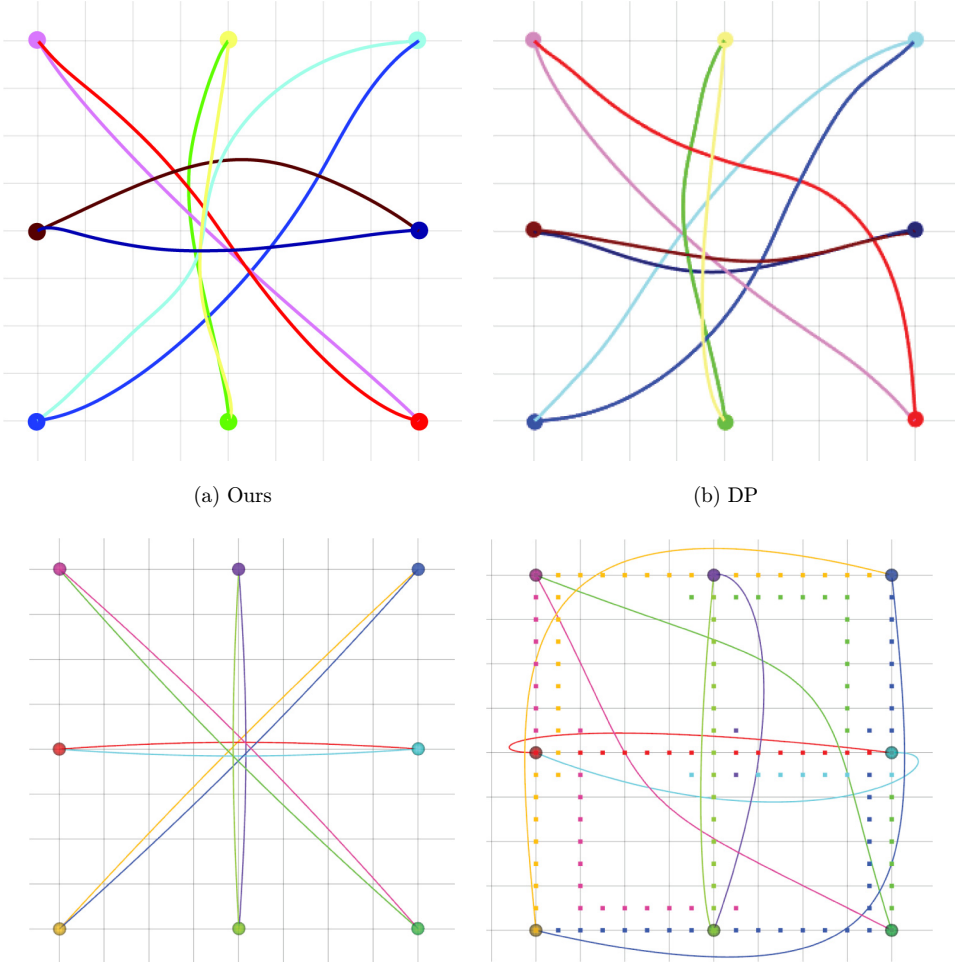


(b) Without potential function

Fig. 8. The 3D navigation of multi-quadrotors in cluttered environments with non-convex obstacles.

the minimum safety ratio of 1.16, which is still over 1, avoiding collisions with obstacles. Obviously, the SCP-based method sacrifices the safety of a real flight system.

(2) Total Flight Time: our method performs better in terms of flight efficiency than the algorithm in Park *et al.*[18] work as it uses JLT to get the time-optimal trajectory, thus having a shorter total flight time.



(a) Ours (b) DP

(c) Algorithm in Ref. 3 (d) Algorithm in Ref. 18

Fig. 9. Comparison of flight performance of different work.

(3) Time Per Iteration: For each planning horizon, the average time of our work is 0.095 s for each quadrotor, which satisfies the real flight requirements. Compared with the DP method in Lai *et al.*,[20] this method has better performance in terms of

Table 2. Summary of flight performance.

Method	Safety ratio	Time per iteration	Total flight time
Our Method	1.19	0.095	56.4
DP Method[20]	1.16	0.132	58.4
SCP ($h = 0.34$ s)[3]	0.92	16.2	—
Algorithm in Ref. 18	1.01	—	75.61

reducing the computational burden. Besides, as the number of quadrotors increases, the computational time will only fluctuate slightly due to the quadrotor system is distributed.

5. Conclusion

In this paper, we have presented a decentralized MPC-based trajectory planning method of multi-quadrotors in cluttered environments. The motion primitives along the flight trajectory are generated using the jerk limited method with given initial and goal states and approximated by a pre-trained NN during the optimization process. Furthermore, a gradient-free algorithm, differential evolution, has been applied to find the best solution to meet the requirements of flight safety, efficiency, and kinodynamic feasibility. The proposed method has been tested with simulations and real flight experiments on multi-quadrotors. Experimental results have demonstrated that the proposed method has excellent performance for motion planning of multi-quadrotors in dynamic cluttered environments.

Acknowledgments

This work is supported in part by the Research Grants Council of Hong Kong SAR (Grant No. 14209020), and in part by the Peng Cheng Laboratory.

References

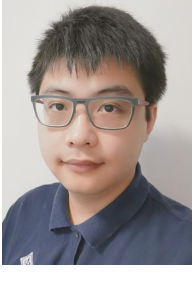
1. H. Bharadhwaj, K. Xie and F. Shkurti, Model-predictive planning via cross-entropy and gradient-based optimization, in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, Vol. 120 (2020), pp. 277–286.
2. A. Pan, W. Xu, L. Wang and H. Ren, Additional planning with multiple objectives for reinforcement learning, *Knowl-Based Syst.* **193**, 105392 (2020).
3. 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* (Vilamoura-Algarve, Portugal, 2012), pp. 1917–1922.
4. 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 Robot. Autom. Lett.* **3**, 1215–1222 (2018).
5. D. Morgan, S. Chung, and F. Y. Hadaegh, Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and model predictive control, *AIAA J.*, 0599 (2015).
6. J. Alonso-Mora, S. Baker and D. Rus, Multi-robot navigation in formation via sequential convex programming, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Hamburg, Germany, 2015), pp. 4634–4641.
7. 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* (Saint Paul, MN, USA, 2012), pp. 477–483.
8. J. C. Derenick and J. R. Spletzer, Convex optimization strategies for coordinating large-scale robot formations, *IEEE Trans. Robot.* **23**, 1252–1259 (2007).

9. J. Alonso-Mora, E. Montijano, M. Schwager and D. Rus, Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus, in *2016 IEEE International Conference on Robotics and Automation* (Stockholm, Sweden, 2016), pp. 5356–5363.
10. D. H. Shim, H. J. Kim and S. Sastry, Decentralized nonlinear model predictive control of multiple flying robots, in *42nd IEEE International Conference on Decision and Control* (Maui, HI, USA, 2003), pp. 3621–3626.
11. Z. Chao, L. Ming, Z. Shaolei and Z. Wenguang, Collision-free UAV formation flight control based on nonlinear MPC, in *2011 International Conference on Electronics, Communications and Control* (Ningbo, China, 2011), pp. 1951–1956.
12. M. Debord, W. Hnig and N. Ayanian, Trajectory planning for heterogeneous robot teams, in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Madrid, Spain, 2018), pp. 7924–7931.
13. J. Alonso-Mora *et al.*, Optimal reciprocal collision avoidance for multiple non-holonomic robots, in *Distributed Autonomous Robotic Systems*, 2013, pp. 203–216.
14. H. Rezaee and F. Abdollahi, A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots, *IEEE Trans. Ind. Electron.* **61**, 347–354 (2014).
15. E. Camponogara, D. Jia, B. H. Krogh and S. Talukdar, Distributed model predictive control, *IEEE Control Syst. Mag.* **22**, 44–52 (2002).
16. R. V. Parys and G. Pipeleers, Distributed model predictive formation control with inter-vehicle collision avoidance, in *2017 11th Asian Control Conference* (Gold Coast, QLD, 2017), pp. 2399–2404.
17. P. Wang and B. Ding, A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance, *Int. J. Control* **87**, 52–63 (2014).
18. 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* (Paris, France, 2020), pp. 434–440.
19. M. Lan, S. Lai, T. H. Lee and B. M. Chen, A survey of motion and task planning techniques for unmanned multicopter systems, *Unmanned Syst.* **9**, 165–198 (2021).
20. S. Lai, M. Lan and B. M. Chen, Model predictive local motion planning with boundary state constrained primitives, *IEEE Robot. Autom. Lett.* **4**, 3577–3584 (2019).
21. R. Haschke, E. Weitnauer and H. Ritter, On-line planning of time-optimal, jerk-limited trajectories, in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Nice, France, 2008), pp. 3248–3253.
22. D. Mellinger and V. Kumar, Minimum snap trajectory generation and control for quadrotors, in *2011 IEEE International Conference on Robotics and Automation* (Shanghai, China, 2011), pp. 2520–2525.
23. B. Lau, C. Sprunk and W. Burgard, Efficient grid-based spatial representations for robot navigation in dynamic environments, *Rob. Auton. Syst.* **21**, 1116–1130 (2013).
24. S. Lai, M. Lan, Y. Li and B. M. Chen. Safe navigation of quadrotors with jerk limited trajectory, *Frontiers Inf. Technol. Electron. Eng.* **20**, 107–119 (2019).

Xinyi Wang received her B.E. degree from Xiamen University, Xiamen, China, in 2019. She is currently working towards a Ph.D. degree in the Department of Mechanical and Automation Engineering at the Chinese University of Hong Kong. Her current research interests include motion planning, multi-agent cooperation, and optimization.

Lele Xi is currently working towards his Ph.D. degree in control science and engineering with the School of Automation, Beijing Institute of Technology, Beijing, China. He was involved in research works as a Visiting Ph.D. Student with the Department of Mechanical and Automation, the Chinese University of Hong Kong. His current research interests include motion planning, target tracking, and reinforcement learning, specifically in the area of aerial robotics.

Yizhou Chen is currently a Ph.D. student in the Department of Mechanical and Automation Engineering at the Chinese University of Hong Kong. His research focus is on robotic tasking and acting, which enable unmanned systems to reason and learn strategy to accomplish missions given by humans automatically. He is also interested in employing parallel computing techniques in applications like robotic mapping and planning.



Shupeng Lai received his Bachelor of Engineering from Nanyang Technological University in 2012 and his Ph.D. from the National University of Singapore in 2016. His research interests include motion planning, nonlinear model predictive control, multi-agent systems, and aerial systems.

Feng Lin received his B.E. degree in computer science and control, and his M.E. degree in system engineering from Beihang University, Beijing, China, in 2000 and 2003, respectively. He received his Ph.D. degree in computer & electrical engineering from the National University of Singapore in 2011. He was the recipient of the Best Application Paper Award, 8th World Congress on Intelligent Control and Automation, Jinan, China (2010). Dr. Lin has worked as a Senior Research Scientist in the Temasek Laboratories @ NUS, and a Research Assistant Professor in the Department of Electrical and Computer Engineering, National University of Singapore from 2011 to 2019. Dr. Lin is currently working as an Associate Research Scientist at Peng Cheng Laboratory and a Chief Technology Officer of Shenzhen MicroHiggs Technologies since 2019. His main research interests are unmanned aerial vehicles, vision-aided control and navigation, target tracking, robot vision as well as embedded vision systems. He has served on the editorial board for *Unmanned Systems*.

Ben M Chen received his B.S. degree in mathematics and computer science from Xiamen University, China, in 1983, M.S. degree in electrical engineering from Gonzaga University, USA, in 1988, and Ph.D. degree in electrical and computer engineering from Washington State University, USA, in 1991. He is currently a Professor of Mechanical and Automation Engineering at the Chinese University of Hong Kong. He was a Provost's Chair Professor in the Department of Electrical and Computer Engineering, National University of Singapore, where he held various academic positions from 1992 to 2018. He was an Assistant Professor at the State University of New York at Stony Brook, in 1992–1993. His current research interests are in unmanned systems, robust control and control applications.

Dr. Chen is an IEEE Fellow and a Fellow of the Academy of Engineering, Singapore. He has authored/co-authored more than 450 journal and conference articles, and a dozen research monographs in control theory and applications, unmanned systems and financial market modeling. He had served on the editorial boards of a dozen international journals including *IEEE Transactions on Automatic Control* and *Automatica*. He currently serves as an Editor-in-Chief of *Unmanned Systems*. Dr. Chen has received a number of research awards. His research team has actively participated in international UAV competitions and won many championships.