Decentralized Trajectory Generation Technique for Multiple Unmanned Multicopter Systems in Cluttered Environments

Xinyi Wang*, Lele Xi, Yizhou Chen, Shupeng Lai, Feng Lin, and Ben M. Chen

ABSTRACT

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 model predictive control (MPC) based gradient-free approach for multiple quadrotors to achieve distributed and asynchronous cooperative motion planning in cluttered environments with the consideration of time consumption. 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. Finally, the reference trajectories are generated using the same BVP solver. Our simulation and experimental results demonstrate the superior performance of the proposed method.

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 high flight performance, as well as minimizing execution time of trajectory for actual implementations.

A wide variety of techniques exist to tackle the multi-quadrotor trajectory generation problem. MPC-based methods have been proven effective for the motion planning of autonomous vehicles in complex environments. The distributed model predictive control (DMPC) approach [1] is developed due to its abilities to handle constraints and achieve good performance for task coordination (see e.g., [2, 3]). The result in Parys and Pipeleers [4] 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 [5] 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., [6, 7]) usually with high trajectory execution time.

Moreover, there are many methods formulating trajectory generation as a non-linear 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 [8]. 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) [9]. In addition, considering the time efficiency, the jerk limited trajectory (JLT) has been proven well suited for quadrotors since it could satisfy the dynamic limits well [10]. It can handle arbitrary initial states for the position, velocity, and acceleration, and resulting in a smooth and time-optimal trajectory from the current state to the next target state.

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

*Email address(es): xywangmae@link.cuhk.edu.hk
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

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 Section 2 some preliminary knowledge and the problem formulation, and in Section 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 Section 4 to validate the proposed technique. Finally, we draw some concluding remarks in Section 5.

2 PROBLEM FORMULATION

We separate the flight control problem into an outer loop and an inner loop. For the quadrotor, the outer loop stabilizes transnational variables and generates a reference signal fed to the inner loop. In our paper, we use the nominal model in [11] to generate the outer loop trajectory, which acts as the reference to the inner loop. 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 [12]. 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 $\mathrm{x}_k = [p_k, v_k, a_k]^{\mathrm{T}}$ 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:

$$\mathrm{x}_k[n+1] = A_k \mathrm{x}_k[n] + b_k u_k[n], \tag{1}$$

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 kinodynamical feasible trajectory $\mathcal{F}_k$ of each quadrotor k. The state vector of each quadrotor is denoted as $\mathrm{x}_k(t) \in \mathcal{X}$ at runtime t. The static obstacles are represented using an occupancy grid map and the dynamic obstacles are defined as a set of convex moving obstacles. It is assumed that we can predict the trajectory of each quadrotor and penalize the relative distance of them to avoid crashes. The free configuration space for a single quadrotor is defined as $\mathcal{X}_{\mathrm{free}}$. All trajectories are considered collision-free if, for each quadrotor, there are no collisions between the quadrotor and environment:

Considering the control objective, we adopt in the following an optimization cost function for generating control input sequence u_k, which has three main components: input variation penalty, desired set position penalty and collision-free penalty:

$$\begin{aligned} \min\ J_k = \int_{t_0}^{t_0+T} & \{w_1 u_k^2(t) + w_2 ||\mathrm{x}_k(t) - g_k||^2 + \\ & w_3 e^{-||d_o^k(t)||} + w_4 ||d_s^k||^2\} dt \\ \text{s.t.} \quad & \dot{\mathrm{x}}_k(t) = f(\mathrm{x}_k(t), u_k) \\ & g(\mathrm{x}_k(t), u_k) < 0 \\ & h(\mathrm{x}_k(t), u_k) = 0 \end{aligned} \tag{3}$$

where w_1, w_2, w_3, w_4 are used to trade off these four penalty. For the constraints of trajectory generation problem, $f(\cdot)$ is the nominal model of the quadrotor, which is a continuous version defined by Eqs. 1 and 2. Inequality constraint $g(\cdot)$ indicates the limit interval of velocity, acceleration and jerk, respectively, for three axes. The boundary state constraint $h(\cdot)$ regulates the triple integrator from an initial state configuration s_k to an assigned goal state g_k. Here we note that the first term is to penalize the square of jerk is to generate smooth trajectories. The second term is to minimize the deviation from the current state to the desired goal state. The third term 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 point. The cost to the goal point of every grid for each quadrotor d_s^k is a value stored on this map. As shown in Figure 1, this collision cost pushes the quadrotor away from static and dynamic obstacles to satisfy the collision-free requirements

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.

3 FRAMEWORK OF MULTI-QUADROTOR TRAJECTORY GENERATION

In this section, a synchronous and decentralized nonconvex 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 Figure 2, which consists of offline training NN process and MPC-based optimization. The proposed framework consists of the following two stages:

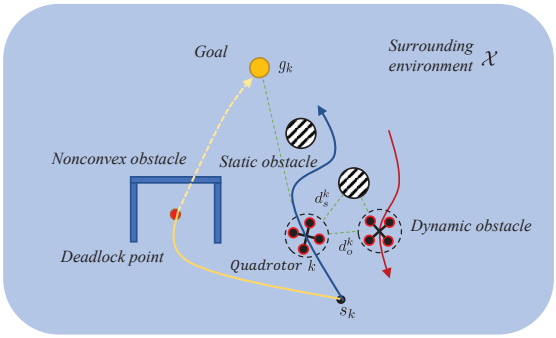


Figure 1: The flight safety of quadrotor k.

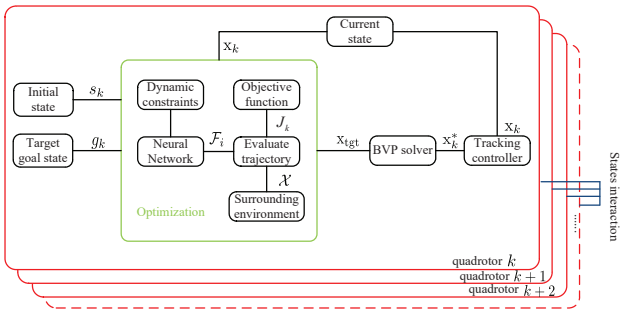


Figure 2: The overall structure of the proposed method for multi-quadrotor motion planning.

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 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 $\mathrm{x}_{\mathrm{tgt}}$ among these candidates to minimize the cost function J_k given in Equation 3 at each step. In addition, we also add guidance map to help the quadrotors to avoid deadlock. After obtaining $\mathrm{x}_{\mathrm{tgt}}$, 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.

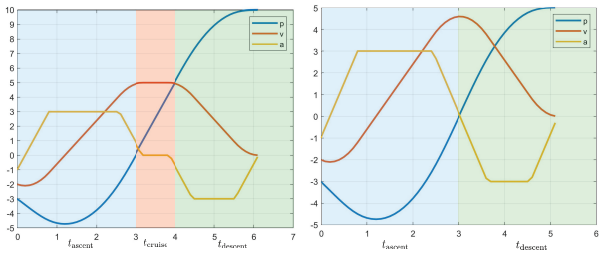
3.1 Construction of BSCPs using JLT

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 $\mathrm{x}(t_{\mathrm{end}}) = [p_{\mathrm{ref}}, v_{\mathrm{ref}}, a_{\mathrm{ref}}]$ for the quadrotor system in Eqs. 1 and 2 with arbitrary initial state values $\mathrm{x}(0) = [p_0, v_0, a_0]$ 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 $\mathrm{x}(t)$ 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

$$\begin{aligned}
\min \quad & t_{\mathrm{end}} \\
\text{s.t.} \quad & p(0) = p_0, \quad p(t_{\mathrm{end}}) = p_{\mathrm{ref}}, \quad \dot{p}(t) = v(t) \\
& v(0) = v_0, \quad v(t_{\mathrm{end}}) = v_{\mathrm{ref}}, \quad \dot{v}(t) = a(t) \\
& a(0) = a_0, \quad a(t_{\mathrm{end}}) = a_{\mathrm{ref}}, \quad \dot{a}(t) = j(t) \\
& -v_{\max} \le v(t) \le v_{\max} \\
& -a_{\max} \le a(t) \le a_{\max} \\
& -j_{\max} \le j(t) \le j_{\max},
\end{aligned} \tag{4}$$

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. 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 Figure 3, then we use bisection searching algorithm given in [13] to calculate the acceleration time t_{ascent} and deceleration time t_{descent}. Otherwise, it is corresponding to the trapezoidal case in Figure 3, we can also use the position area to determine the acceleration time t_{ascent}, cruise time t_{cruise} and deceleration time t_{descent}.



(a) Trapezoidal case (b) Triangular case

Figure 3: Three types of velocity profile.

After generating the BSCPs, we then implement the trajectory approximation and evaluation system using a pre-

trained NN in [9], which is called S_{NN} to save computational time for real flight implementations.

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 Lau et al. [14] 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 $\mathcal{M}(\mathrm{p}_i)$ consisting of a cost-to-goal value c_g and a safety value c_s with consideration of the closest distance to obstacles can be calculated as:

$$\mathcal{M}(\mathrm{p}_i) = c_s(\mathrm{p}_i) + \alpha_g c_g(\mathrm{p}_i) \tag{5}$$

where α_g is a weight factor that trades off the relative importance of the cost-to-go value, seen Figure 4. 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, seen Equation 6.

$$u^* = \arg\min_{u} \mathcal{M}(\mathrm{p}_i) \tag{6}$$

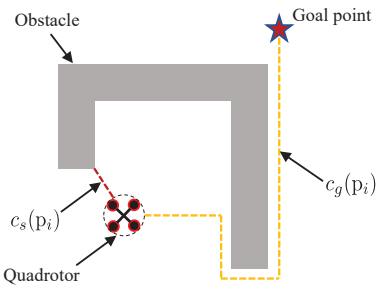


Figure 4: The illustration of deadloack avoidance.

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 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 represents a terminal state constraint for one planning horizon. Next, update the velocity δ_i and position $\mathrm{x}_{\mathrm{tgt}_i}$ of particles according to the equation in this algorithm (Lines 5–6), where ω_1 and ω_2 are acceleration constants and $\mathrm{x}^*_{\mathrm{tgt}_i}$ and $\mathrm{x}^g_{\mathrm{tgt}}$ 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 9–10) to get the optimal end state, where $cost^*_i$ represents local best cost value and $cost^g$ is the corresponding global best value. The feasible solution $\mathrm{x}_{\mathrm{tgt}}$ will be found after multiple interactions, which is used in a BVP solver to obtain a $\mathcal{F}_k$ of each quadrotor (Lines 12–14).

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: $\mathcal{M}_k$ = getGuidedMap(g_k, $\mathcal{X}$);
3: ParticleInitialization();
4: **for** Each $\mathrm{x}_{\mathrm{tgt}_i}$, $i = 1, 2, \ldots$ **do**
5: $\delta_i = \delta_i + \omega_1 * (\mathrm{x}^*_{\mathrm{tgt}_i} - \mathrm{x}_{\mathrm{tgt}_i}) + \omega_2 * (\mathrm{x}^g_{\mathrm{tgt}} - \mathrm{x}_{\mathrm{tgt}_i})$
6: $\mathrm{x}_{\mathrm{tgt}_i} = \mathrm{x}_{\mathrm{tgt}_i} + \delta_i$
7: $\mathcal{F}_i$ = approxTrajectory($\mathrm{x}_k, \mathrm{x}_{\mathrm{tgt}_i}$,$S_{\text{NN}}$);
8: $cost_i = J_k(\mathcal{F}_i, \mathcal{M}_k)$
9: Update local best $cost^*_i$, $\mathrm{x}^*_{\mathrm{tgt_i}}$
10: Update global best $cost^g$, $\mathrm{x}^g_{\mathrm{tgt}}$
11: **end for**
12: $\mathrm{x}_{\mathrm{tgt}} = \mathrm{x}^g_{\mathrm{tgt}}$
13: $\mathcal{F}_k$ =JLTTrajGenerotion($\mathrm{x}_k, \mathrm{x}_{\mathrm{tgt}}$);
14: Return $\mathcal{F}_k$;
15: **end for**

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. 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

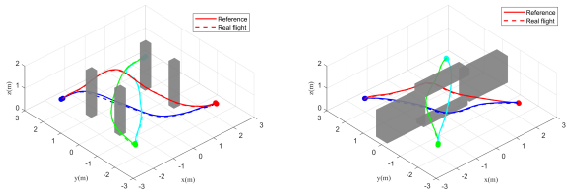
We use Crazyflie, a small, versatile quadcopter, as the experimental platform to verify the proposed method for motion planning of multi-quadrotors. The real flight experiments are carried out in an indoor VICON environment. The VICON system provides localization and obstacles sensing for each quadrotor. The 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.
- 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 Section 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.

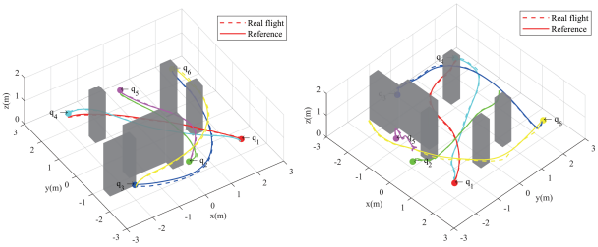
Figure 5 shows cases that the four quadrotors start from their initial positions to their antipodals while avoiding collision with other quadrotors and static obstacles. All the quadrotors avoid the obstacles and converge to their antipodal positions in about 4.8 s. The maximum speed reached is 1.99 m/s with an average speed of 0.93 m/s.



(a) An environment with several pillars (b) An environment with bridge openings

Figure 5: 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.

As it can be clearly seen in Figure 6, the six quadrotors q_i, $i = 1, 2, \cdots, 6$, 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 Figure 6(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 Figure 6(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) With potential function (b) Without potential function

Figure 6: The 3D navigation of multi-quadrotors in cluttered environments with non-convex obstacles.

4.3 Comparison with Other Method

We compare our method with methods in Augugliaro et al. [15], Park et al. [7] and Lai et al. [9] on a 10m×10m×2.5m obstacle-free environment using 8 quadrotors and the same boundary states. Detailed results are shown in Table 1.

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 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. [7] work as it uses JLT to get the time-optimal trajectory, thus having a shorter total flight time.

3) Time Per Iteration: For each planning horizon, the average time of our work is 0.095s for each quadrotor, satisfying real flight requirements. Compared with the DP method in Lai et al. [9], this method has better performance in terms of 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.

Table 1: Summary of flight performance.

Method	Safety Ratio	Time Per Iteration	Total Flight Time
Our Method	1.19	0.095	56.4
DP Method [9]	1.16	0.132	58.4
SCP ($h = 0.34$ s) [15]	0.92	16.2	—
Algorithm in [7]	1.01	—	75.61

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.

ACKNOWLEDGEMENTS

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] M. Rufli J. Alonso-Mora, A. Breitenmoser and et al. Optimal reciprocal collision avoidance for multiple non-holonomic robots. *Distributed Autonomous Robotic Systems*, pages 203–216, 2013.

[2] H. Rezaee and F. Abdollahi. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. *IEEE Transactions on Industrial Electronics*, 61:347–354, 2014.

[3] B. H. Krogh E. Camponogara, D. Jia and S. Talukdar. Distributed model predictive control. *IEEE Control Systems Magazine*, 22:45–52, 2002.

[4] R. V. Parys and G. Pipeleers. Distributed model predictive formation control with inter-vehicle collision avoidance. In *2017 11th Asian Control Conference*, 2017.

[5] P. Wang and B. Ding. A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance. *International Journal of Control*, 87:52–63, 2014.

[6] W. Hönig M. Debord and N. Ayanian. Trajectory planning for heterogeneous robot teams. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[7] I. Jang J. Park, J. Kim 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*, 2020.

[8] T.H. Lee M. Lan, S. Lai and B. M. Chen. A survey of motion and task planning techniques for unmanned multicopter systems. *Unmanned Systems*, 9:165–198, 2021.

[9] M. Lan S. Lai and B. M. Chen. Model predictive local motion planning with boundary state constrained primitives. *IEEE Robotics and Automation Letters*, 4:3577–3584, 2019.

[10] E. Weitnauer R. Haschke and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[11] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen. Teach-repeat-replan: a complete and robust system for aggressive flight in complex environments. *IEEE Transactions on Robotics*, 36:1526–1545, 2020.

[12] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, 2011.

[13] S. Lai, M. Lan, Y. Li, and B. M. Chen. Safe navigation of quadrotors with jerk limited trajectory. *Frontiers Inf Technol Electronic Eng*, 20:107–119, 2019.

[14] C. Sprunk B. Lau and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 21:1116–1130, 2013.

[15] A. P. Schoellig F. Augugliaro 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*, 2012.