GTO-MPC-Based Target Chasing Using a Quadrotor in Cluttered Environments

Lele Xi, Xinyi Wang, Lei Jiao, Shupeng Lai, Zhihong Peng, *Member, IEEE*, and Ben M. Chen, *Fellow, IEEE*

***Abstract*—This article addresses the challenging problem of chasing an escaping target using a quadrotor in cluttered environments. To tackle these challenges, we propose a guided time-optimal model predictive control (GTO-MPC)-based practical framework to generate chasing trajectories for the quadrotor. A jerk limited approach is first adopted to find a time-optimal jerk limited trajectory (JLT), an initial reference for the quadrotor to track, without taking into account surrounding obstacles and potential threats. An MPC-based replanning framework is then applied to approximate the JLT together with the consideration of other issues such as flight safety, line-of-sight maintenance, and deadlock avoidance. Combined with a neural network, the proposed GTO-MPC framework can efficiently generate chasing trajectories that guarantee flight smoothness and kinodynamic feasibility. Our simulation and actual experimental results show that the proposed technique is highly effective.**

***Index Terms*—Model predictive control, motion primitive, target chasing, time optimal, trajectory planning.**

NOMENCLATURE

p	The position on one axis.
v	The velocity on one axis.
a	The acceleration on one axis.
j	The jerk on one axis.
p_q	The coordinates of the quadrotor.
p_t	The coordinates of the escaping target.
$\mathrm{p}_i, i = 1, 2, \ldots$	The coordinates of ith simulated sequential points.
$\mathbf{x}_q$	The state of the quadrotor.
$\mathbf{x}_t$	The state of the escaping target.
$\mathbf{x}_{d_i}, i = 1, 2, \ldots$	The candidate state of the quadrotor.
S_{NN}	Neural network.
$\mathcal{T}_q$	The trajectory of the quadrotor.
$\mathcal{T}_r$	JLT reference.
$\mathcal{X}$	The surrounding environment.
$\mathcal{X}_{\mathrm{free}}$	The obstacle-free region in the surrounding environment.
$\mathcal{M}_{\mathrm{go}}$	The cost-to-go map.

Manuscript received December 11, 2020; revised March 28, 2021 and May 12, 2021; accepted June 9, 2021. Date of publication June 24, 2021; date of current version February 1, 2022. This work was supported in part by the Key Program of National Natural Science Foundation of China (NSFC) under Grant U2013602, and in part by Peng Cheng Laboratory. The work of Xinyi Wang and Ben M. Chen was supported in part by the Research Grants Council of Hong Kong SAR under Grant 14209020. *(Corresponding author: Zhihong Peng.)*

Lele Xi, Lei Jiao, and Zhihong Peng are with the School of Automation, Beijing Institute of Technology, Beijing 100081, China, and also with the State Key Laboratory of Intelligent Control, and Decision of Complex System, Beijing 100081, China (e-mail: 3120170437@bit.edu.cn; 3120185446@bit.edu.cn; peng@bit.edu.cn).

Xinyi Wang and Ben M. Chen are with the Department of Mechanical, and Automation Engineering, Chinese University of Hong Kong, Hong Kong (e-mail: xywangmae@link.cuhk.edu.hk; bmchen@cuhk.edu.hk).

Shupeng Lai is with the Department of Electrical, and Computer Engineering, National University of Singapore, Singapore 117583, Singapore, and also with the Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: eleshla@nus.edu.sg).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIE.2021.3090700.

Digital Object Identifier 10.1109/TIE.2021.3090700

I. INTRODUCTION

AUTONOMOUS quadrotors or vehicles have been widely utilized for target chasing in many situations, in which the primary goal is to persistently chase mobile targets (see, e.g., [1] and [2]). Many of the related applications might only require a robot or drone to autonomously track a mobile target in obstacle free or sparse environments (see, for example, [3]–[6]). However, in cluttered environments, when a target of interest enters into unstructured scenes, the drone would have to simultaneously perform real-time target chasing and obstacle avoidance. As such, chasing a maneuvering target puts forward the need for active reactions to both the motion of a target and obstacle avoidance. The work in [7] presented a probabilistic prediction model with the Markov decision process to fulfill air-to-ground surveillance, but did not consider flight smoothness. Many works formulated the trajectory of quadrotors as polynomial or spline curves for target tracking [8], [9], and solved the problem by quadratic programming (QP) or sequential QP in static or obstacle-free environments. In particular, it is essential to keep the target in the line-of-sight as long as possible, since the motion of a mobile target can be obtained only if the target is in the field of view. Jeon and Kim [10] dealt with a moving target chasing mission of aerial vehicles in a cluttered environment. In the chasing phase, safe corridors and visible waypoints are utilized as hard constraints to complete polynomial trajectory generation through QP without considering the physical limits of the drone, which is insufficient for fast-chasing flight. The work in [11] proposed a vision-based approach to follow a moving target in a smooth but reactive way while avoiding obstacles in cluttered environments. A method of a person following using an indoor mobile robot was proposed in [12]. A concept, following field,

which directly considers visibility information, was introduced in that work to enhance the robot follow the target well.

In many scenarios, such as the pursuit-evasion problem in border antismuggling, the quadrotor should chase a suspected target as quickly as possible to prevent it from escaping. To achieve such chasing missions, differing from the target tracking problem, an autonomous system with a quadrotor should have following capabilities:

1) the trajectory planning module should be able to generate feasible trajectories for the quadrotor in real-time in accordance with the status of the escaping target and surrounding environments;
2) the quadrotor should be able to catch up with the escaping target in the minimum amount of time.

Aside from safety considerations, chasing efficiency, flight smoothness, and line-of-sight maintenance are also critical. It is thus necessary to find feasible flight trajectories for the quadrotor to keep chasing the target efficiently and to improve the robustness to potential safety threats.

In recent years, MPC-based methods have been proven effective for motion planning and trajectory tracking of autonomous vehicles in complex environments (see, e.g., [13]–[16]). The work in [17] dealt with local motion planning for autonomous ground vehicles in unstructured environments with static and dynamic obstacles using a nonlinear MPC-based method. In addition, neural networks (NNs) have become a popular choice as a model for real-time MPC applications because its learned models have efficient and accurate prediction performance [18], and has been widely applied to the planning and navigation problems of autonomous vehicles. The works in [19] and [20] presented a framework of quadrotor for online motion planning with boundary state constrained primitives (BSCPs). They evaluated the trajectory by combining NN and MPC to meet the requirements of real-time applications in cluttered environments. Song and Scaramuzza [21] leveraged the policy search and artificial NN to the powerful online optimization for the MPC problem. A multilayer perceptron policy is combined with an MPC to solve a challenging control problem. A comprehensive survey on motion and task planning for multicopters can be found in [22].

In this article, a guided time-optimal model predictive control (GTO-MPC)-based trajectory generation method for a quadrotor to chase an escaping target in obstacle-rich environments is proposed. More specifically, an MPC technique combining with NN is utilized to find optimal motion primitives for the quadrotor that directly incorporate the static obstacles and predicted intentions of the dynamic obstacles in surrounding environments. The contributions and innovations of this work are as follows.

1) A novel framework named GTO-MPC for a drone to chase an escaping target in obstacle-rich environments is proposed in which a jerk limited trajectory (JLT) is generated to provide a guided time-optimal (GTO) initial reference; and an MPC-based method is applied to approximate the initial reference with the consideration of flight smoothness, flight safety, line-of-sight maintenance, and deadlock avoidance.
2) A gradient-free method using an NN and differential evolution (DE) is proposed to efficiently construct a dynamically evolving single-layer tree that gradually converges to the optimal chasing BSCPs. DE can find the optimal solution to the optimization problem with uniform optimization time, which improves the robustness of quadrotor trajectory planning in cluttered environments.

Compared to the usual MPC method, the proposed GTO-MPC method has better chasing efficiency. The results of actual flight tests with a quadrotor system in cluttered or dynamic environments show that the proposed technique is effective and efficient.

The rest of this article is organized as follows. In Section II, the escaping target chasing problem is formulated along with some background materials that are essential for the development of the result presented in this article. In Section III, the proposed GTO-MPC framework and detailed method are presented, and the results and analysis of simulations and actual flight experiments are given in Section IV. Finally, Section V concludes this article.

II. Problem Formulation

We present in this section, the problem formulation together with necessary background materials.

A nonlinear system is termed to exhibit differential flatness if the system nominal states and control inputs can be written in terms of a set of specific variables (called *flat outputs*) and their derivatives. A quadrotor has a six-dimensional configuration space, and the states and control inputs can be represented as algebraic functions of four carefully selected flat outputs and their derivatives. As described in [23], $\sigma = [x, y, z, \psi]$ is the proper choice of flat outputs, where $\mathrm{p} = [x, y, z]$ is the vehicle's coordinates of the mass center in the global frame, and ψ is the yaw angle. With a properly design inner loop control law, the outer loop of the quadrotor can be treated as a point mass. On each axis of the quadrotor, the outer loop can be simplified as a triple integrator

$$\begin{cases} \dot{p} = v \\ \dot{v} = a \\ \dot{a} = j \end{cases} \tag{1}$$

where p, v, a, and j are its position, velocity, acceleration, and jerk, respectively. In our problem formulation, the jerk j is chosen as the control input. The dynamic model of the outer loop of the drone can be discretized as follows:

$$\mathbf{x}[n+1] = A\mathbf{x}[n] + bu[n] \tag{2}$$

where

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

in which the state $\mathbf{x} = [p, v, a]^{\mathrm{T}}$ includes its position, velocity, and acceleration, and input $u = j$ is its jerk. To take kinodynamic feasibility into count, we can set the invariant constraints as $v \in [v_{\min}, v_{\max}]$, $a \in [a_{\min}, a_{\max}]$, and $j \in [j_{\min}, j_{\max}]$, in which

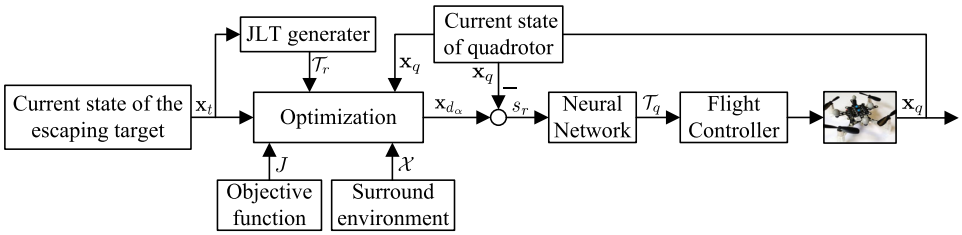


Fig. 1. Overall structure of the proposed method framework.

the dynamic limits might be different for the horizontal and vertical axes.

The objective of this article is to generate a collision-free and smooth trajectory for the quadrotor to chase an escaping target while maintaining the line-of-sight and deadlock avoidance, which can be formulated as the following optimization problem:

$$\begin{aligned}\min \quad & \int_{t_0}^{t_0+T} \Big\{ u^2(t) + w_1 \left(e^{-\|d_1(t)\|} + e^{-\|d_2(t)\|} \right) \\ & + w_2 \|\mathrm{p}_q(t) - \mathrm{p}_j(t)\|^2 + w_3 e^{-\|d_3(t)\|} + w_4 \|d_{\mathrm{go}}\|^2 \Big\} dt \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), u) \\ & \mathrm{p}_q(t) \in \mathcal{X}_{\mathrm{free}} \\ & \mathcal{T}_q^{(k)}(t) \in \Phi_k. \end{aligned} \tag{4}$$

The goal is to search for a feasible state trajectory $\mathbf{x}(t)$, $t \in [t_0, t_0 + T]$. The objective function is divided into the following five parts:

1) the first term is to minimize the norm of the jerk to guarantee smoothness of the flight trajectory while maintaining aggressiveness of the drone;
2) the second term to penalize the closest distances of the quadrotor to the static and dynamic obstacles;
3) the third term to minimize the deviation from the chasing trajectory to the jerk-limited one;
4) the fourth term to penalize the closest distance from the bearing vector, formed by the positions of quadrotor and escaping target, to surrounding obstacles;
5) the fifth term to minimize the cost-to-go (to be introduced in Section III-D5).

For the optimization constraints, $f(\cdot)$ is the nominal dynamic model of the quadrotor in (2); and Φ_k indicates the limit interval of velocity, acceleration, and jerk, respectively, for $k = 1, 2, 3$. $\|\cdot\|$ denotes the usual l_2 norm and w_i, $i = 1, 2, 3, 4$ are the corresponding weight values. The detailed explanation of the cost function is given in Section III-D.

III. Trajectory Generation for Target Chasing

In the following, the GTO-MPC optimization problem formulated in the previous section is utilized to generate feasible trajectories for the drone to chase an escaping target. It is assumed that the relative states of the mobile target and dynamic obstacles can be estimated or predicted, and observations of the surrounding environment are possible.

Algorithm 1: The Optimization Process.

Input: The environment $\mathcal{X}$, The quadrotor's state $\mathbf{x}_q$, and the target's predicted state $\mathbf{x}_t$;
Output: Quadrotor's chasing trajectory $\mathcal{T}_q$;
1: **for** Each planning horizon **do**
2: $\mathcal{T}_r = \text{genJLT}(\mathbf{x}_q, \mathbf{x}_t)$;
3: $\mathcal{M}_{\mathrm{go}} = \text{getCost\_to\_go}(\mathbf{x}_t)$;
4: **for** Each candidate $\mathbf{x}_{d_i}, i = 1, 2, \ldots$ **do**
5: $\mathcal{T}_i = \text{approxTrajectory}(\mathbf{x}_q, \mathbf{x}_{d_i}, S_{\mathrm{NN}})$;
6: $c(\mathcal{T}_i) = \text{evalTrajectory}(\mathcal{T}_i, \mathcal{T}_r, \mathcal{X}, \mathcal{M}_{\mathrm{go}})$;
7: update c^*;
8: **end for**
9: $\alpha = \arg\min(c^*)$;
10: $\mathcal{T}_q = \text{approxTrajectory}(\mathbf{x}_q, \mathbf{x}_{d_\alpha}, S_{\mathrm{NN}})$;
11: Return $\mathcal{T}_q$;
12: **end for**

A. Method Framework

The overall structure of the proposed framework and optimization process are depicted in Fig. 1 and Algorithm 1, respectively. For the optimization process given in Algorithm 1, the function $\text{genJLT}(\mathbf{x}_q, \mathbf{x}_t)$ generates the time-optimal JLT reference from state $\mathbf{x}_q$ to $\mathbf{x}_t$. The function $\text{getCost\_to\_go}(\mathbf{x}_t)$ returns a cost-to-go map $\mathcal{M}_{\mathrm{go}}$ that guides the quadrotor to the state $\mathbf{x}_t$. The function $\text{approxTrajectory}(\mathbf{x}_q, \mathbf{x}_d, S_{\mathrm{NN}})$ approximates the trajectory by the NN S_{NN}. The function $\text{evalTrajectory}(\mathcal{T}_q, \mathcal{T}_r, \mathcal{X}, \mathcal{M}_{\mathrm{go}})$ evaluates the trajectory $\mathcal{T}_q$ against the reference $\mathcal{T}_r$, surrounding environment $\mathcal{X}$, and cost-to-go map $\mathcal{M}_{\mathrm{go}}$.

The two main steps in each replanning cycle of the GTO-MPC method are the following.

1) First, a time-optimal JLT is generated from the current state of the quadrotor to the predicted state of the target without considering safety threats in the surrounding environment (Line 2 of Algorithm 1), resulting in an unnecessarily aggressive but time-optimal formulation.
2) An MPC-based framework is then adopted to primarily minimize the deviation from the chasing trajectory to the JLT, and take flight safety, line-of-sight maintenance, and deadlock avoidance into consideration. In the optimization process, as shown in Fig. 2, all trajectories are approximated by the pretrained NN, which explicitly guarantees the smoothness and kinodynamic feasibility (Line 5). The samples are then evaluated against the surrounding environment and position of the escaping

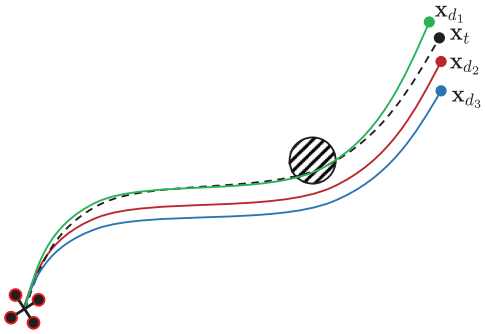


Fig. 2. Illustration of candidate selection. As illustrated, the green, red, and blue curves indicate trajectories generated by three candidates $\mathbf{x}_{d_1}$, $\mathbf{x}_{d_2}$, and $\mathbf{x}_{d_3}$, respectively, and the black dash line represents the JLT. Comparatively and clearly, the red curve is the best one among the three trajectories.

target (Line 6), which results in the best solution (Lines 9–11).

B. Time-Optimal Trajectory

The JLT has been proven well suited for quadrotors as it could satisfy the maximum thrust and body rate limits [24]. It can handle arbitrary initial states for the position, velocity, and acceleration, and achieve a smooth blending from the current state to the next target state. Meanwhile, it can handle changes in the limited conditions, allowing replanning at any time. The time-optimal JLTs can be generated in the three dimensional (3-D) space to satisfy the constraints on velocity, acceleration, and jerk. The time-optimal JLT generation problem can be formulated as

$$\begin{aligned}
\min \quad & t_{\text{end}} \\
\text{s.t.} \quad & p(0) = p_0, \quad p(t_{\text{end}}) = p_{\text{ref}} \\
& v(0) = v_0, \quad v(t_{\text{end}}) = v_{\text{ref}} \\
& a(0) = a_0, \quad a(t_{\text{end}}) = a_{\text{ref}} \\
& \dot{p}(t) = v(t) \\
& \dot{v}(t) = a(t) \\
& \dot{a}(t) = j(t) \\
& -v_{\max} \le v(t) \le v_{\max} \\
& -a_{\max} \le a(t) \le a_{\max} \\
& -j_{\max} \le j(t) \le j_{\max}
\end{aligned} \tag{5}$$

where $v(t)$, $a(t)$, and $j(t)$ denote the velocity, acceleration, and jerk, respectively, along any axis in 3-D space. $j_{\max}$, $a_{\max}$, and $v_{\max}$ represent the maximum value of jerk, acceleration, and velocity, respectively. A closed-form solution to the aforementioned problem has been given in [24], in which it is proven that the jerk can only be taken as $\pm j_{\max}$ or zero for the time-optimal trajectory of the unmanned aerial vehicle. As shown in Fig. 3, both acceleration and deceleration phase run at the maximum jerk of opposite sign, resulting in a rectangular-shaped jerk profile.

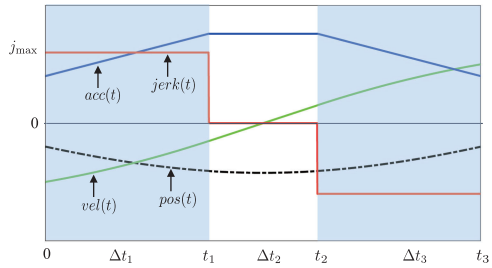


Fig. 3. Canonical third-order profile comprising three phases. Δt_1, Δt_2, and Δt_3 are the time intervals with $j_{\max}$, zero, and $j_{\min}$, respectively.

In this article, a stable approach [25] is adopted to find the time-optimal trajectory of the quadrotor from an initial state to a target state. A binary search method, which has been shown to be efficient and stable for quadrotors, is applied to search for a switching time that separates two different phases. It should be noted that the final velocity and acceleration are supposed to be zero throughout the article.

C. Construction of BSCPs

The motion of a quadrotor can be formulated as BSCPs and solved by the standard boundary-value problem solvers or controllers. Local motion planning for a quadrotor is considered herein as an MPC problem that searches for the input $u(t)$ and state $\mathbf{x}(t)$ of the quadrotor's dynamics, i.e., (2) to minimize the following optimization problem:

$$J = \min_{\theta} \int_{t_0}^{t_0+T} R(\theta)dt + \phi(\theta) \tag{6}$$

with updated quadrotor and surrounding environment information when the goal state $\mathbf{x}_d = [\theta, 0, 0]$ is given, and where R and ϕ represent the running and terminal costs resulting from the goal position θ, respectively. A model-based reinforcement learning method, offering a powerful method of searching for the optimal controller of systems with nonlinear and uncertainty, is adopted to obtain the optimal action once the relative state is given [26]. The relative state is defined as

$$s_r = \mathbf{x}_q - \mathbf{x}_d = [p_q - \theta, v_q, a_q] \tag{7}$$

where $\mathbf{x}_q = [p_q, v_q, a_q]$ is the initial state of the quadrotor. To reach $\mathbf{x}_d$, one should regulate s_r to zero. In the proposed implementation, an offline model-based dynamic programming (DP) method is applied to find the optimal action u (jerk) with the given state. The value function is updated as follows:

$$Q(s_r[n]) = r(s_r[n], u[n]) + Q(s_r[n+1]) \tag{8}$$

where $r(s_r[n], u[n])$ is the instantaneous cost which is given as

$$r(s_r, u) = s_r^{\mathrm{T}} Q_w s_r + \lambda u^2 + Q_c(s_r, u) \tag{9}$$

where Q_w is the corresponding weight matrix and

$$\begin{aligned}
Q_c(s_r, u) = {} & w_v \mu^2(v, v_{\min}, v_{\max}) + w_a \mu^2(a, a_{\min}, a_{\max}) \\
& + w_j \mu^2(j, j_{\min}, j_{\max})
\end{aligned} \tag{10}$$

TABLE I
DETAILS OF THE NN

	Quadrotor horizontal	Quadrotor vertical
NN structure	64-128-128-41 MLP	64-128-128-41 MLP
Training set size	4×10^5	4×10^5
Test set size	1×10^5	1×10^5
Batch size	32	32
Epoch	12	12
Average Mean Squared Error	5.4×10^{-4}	5.32×10^{-4}

* MLP stands for the multilayer perception.

with $\mu(k_1, k_2, k_3) = \max(k_2 - k_1, 0) + \max(k_1 - k_3, 0)$, $k_1, k_2, k_3 \in \mathbb{R}$. w_v, w_a, w_j are the weighting factors of velocity, acceleration, and jerk, and are set as 0.5, 0.1, and 2.0, respectively. For the value function, value iteration turns the Bellman optimality into an iterative assignment [26]. The value iteration is ended once the minimum value function for each state is achieved when

$$Q^*(s_r[n]) = \min_{u[q] \in \mathcal{J}} r(s_r[n], u[n]) + \gamma Q^*(s_r[n+1]) \quad (11)$$

is satisfied for all possible s_r with discount factor γ. The result is a lookup table for the optimal policy $\pi(s_r)$, of which the output is optimal action (jerk). Then, the chosen jerks determine a unique trajectory of the quadrotor to regulate the quadrotor system in (2) from a given state s_r to the origin.

In actual applications, it is unrealistic to calculate the maximum expected future rewards for action at each state for high-dimensional systems using a lookup table. In the proposed approach, the trajectory is approximated and evaluated with a pretrained NN, S_{NN}, which is described in detail later. Similar to the work in [19], the proposed approach includes an offline training stage and an online MPC stage. In the offline process, the mapping from $s_r = \mathbf{x}_q - \mathbf{x}_d$ to the chasing trajectory is learned to limit the size of S_{NN}. By discretizing the relative position $p(t)$ at 20 Hz for 2 s into the future and cascading the result with $u_i, i = 1, 2, \ldots 40$. The input of S_{NN} is a 3×1 vector determined by the size of s_r, whereas its output is a 41×1 vector formed by all elements of $p(t)$ and one smoothness factor. PyTorch with the Adam optimizer is applied to train S_{NN} on a CPU-based platform. The relevant details of S_{NN} can be found in Table I. The horizontal and vertical motions are approximated using two different NNs due to the vehicle's physical properties.

D. Feasibility Analysis

To take flight safety, smoothness, and chasing performance into account, we must enforce several optimization terms when performing the chasing task. Benefiting from the instantaneous cost in the iteration process as proposed in Section III-C, the physical limits on the thrust, body rate, and speed can be directly guaranteed. The extra feasibility conditions are described as follows.

1) Smoothness: Smooth motion and task performance should be considered simultaneously to meet the requirements for real applications. Smooth response from the vehicle can be quantified as an integral function of jerk or snap, which are the third- and fourth-order derivatives of position, respectively. In the proposed method, jerk cost, adversely affecting the efficiency of the chasing flight, is considered to minimize the total control efforts for executing the chasing task. The smoothness penalty term along the chasing trajectory can be expressed as

$$J_{\mathrm{smooth}} = \min \int_{t_0}^{t_0+T} u^2(t) dt \quad (12)$$

where $u(t)$ is the input (jerk) of the controller in (2). As described in [23], the higher the jerk cost, the more aggressive the trajectory.

2) Flight Safety: Typically in dynamic and cluttered environments, chasing an escaping target puts forward the need for active reactions to both the target's motion and obstacle avoidance along the whole chasing process. To handle the collision of potential threats, the flight safety cost is penalized as a soft constraint

$$J_{\mathrm{safety}} = \min \int_{t_0}^{t_0+T} e^{-||d_1(t)||} + e^{-||d_2(t)||} dt \quad (13)$$

where $d_1(t)$ and $d_2(t)$ are the closest Euclidean distances from the quadrotor to the static and dynamic obstacles, respectively, at runtime t. In this article, the surrounding environment is represented as a 3-D grid map to achieve static obstacle avoidance. For each grid, the closest distance from the node to the obstacle can be efficiently obtained in a preprocessed Euclidean distance transform (EDT) map, in which the grid index and the closest distance are stored. In terms of dynamic obstacle avoidance, the Euclidean distance between the quadrotor and mobile obstacles is penalized to achieve collision avoidance.

3) Contouring Control: As shown in Fig. 4, to pursue the escaping target as quickly as possible, the proposed method should minimize the deviation from the chasing trajectory to the initial reference, i.e., the time-optimal JLT. The deviation of contouring control is penalized as

$$J_{\mathrm{contouring}} = \min \int_{t_0}^{t_0+T} ||\mathrm{p}_q(t) - \mathrm{p}_j(t)||^2 dt \quad (14)$$

where $\mathrm{p}_q(t)$ and $\mathrm{p}_j(t)$ represent the coordinates of the quadrotor and the corresponding state of the JLT at runtime t, respectively. By penalizing the aforementioned objective function, the chasing trajectory will be obtained under the guidance of the time-optimal JLT in the relaxing constraints.

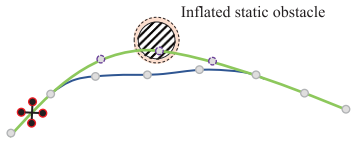


Fig. 4. Representation of the contouring control. The dotted circles are the waypoints of the JLT, while the solid circles represent the waypoints generated by the GTO-MPC method.

4) Line-of-Sight Maintenance: Conventional onboard sensors, such as camera and radar, have limited sensing range and detection angle. In general scenarios, the motion of the mobile target can be obtained only if the target is in the line-of-sight. Therefore, it is essential to keep the target detectable as long as possible to tackle the persistent chasing problem. It may be noted

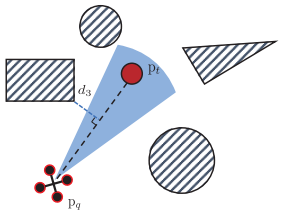


Fig. 5. Illustration of line-of-sight constraint in the 2-D case. The blue sector region represents the field of view of the onboard sensor. p_q and p_t are the coordinates of quadrotor and escaping target, respectively.

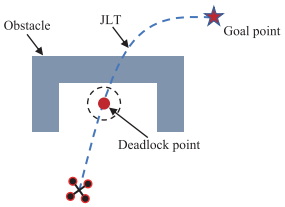


Fig. 6. Illustration of deadlock while tracking a JLT.

that the line-of-sight is represented herein as a sector-shaped sensing domain, but not limited to that.

The obstacle region is represented as $\mathcal{X}_{\text{obs}}$ where the Euclidean distance function $\phi(\mathrm{p}) = 0$, while $\phi(\mathrm{p}) > 0$ for the obstacle-free region $\mathcal{X}_{\text{free}}$. As illustrated in Fig. 5, the bearing vector is defined as $\vec{b}(\mathrm{p}_q, \mathrm{p}_t) = \mathrm{p}_q - \mathrm{p}_t$, formed by the coordinates of drone and escaping target. To maintain the line-of-sight for robust chasing, one should maximize the closet distance between the bearing vector $\vec{b}(\mathrm{p}_q, \mathrm{p}_t)$ to the nearest obstacles. Benefiting from the fast voxel traversal algorithm, the grid indexes and closest distance can be efficiently obtained. To maintain the line-of-sight as long as possible, this constraint is formulated as

$$J_{\text{visible}} = \min \int_{t_0}^{t_0+T} e^{-||d_3(t)||} dt \tag{15}$$

where $d_3(t)$ is the closest distance from bearing vector to the nearest obstacles at runtime t.

5) Deadlock Avoidance: In many cluttered scenarios, for example, as shown in Fig. 6, the replanning planner may fall into local minima or result in a deadlock while carrying out JLT tracking. Both local minima and deadlock are undesirable scenarios for quadrotors or other mobile robots. Many work [27] have inflated surrounding obstacles to convex free spaces, such as circles or ellipses, which is convenient but limited merely for local motion planning. To tackle this problem, following [28], Dijkstra's algorithm is used in this article to construct a cost-to-go map $\mathcal{M}_{\text{go}}$ to overcome such a deadlock problem and directly guide the quadrotor to the goal point. In the $\mathcal{M}_{\text{go}}$, each grid contains the remaining distance d_{go} (cost-to-go) from the current grid to the goal point with considering static obstacles information. Therefore, in the evaluation process of optimization, the cost-to-go map $\mathcal{M}_{\text{go}}$ that directly minimizes the norm of d_{go}, should be applied to avoid the deadlock or getting stuck.

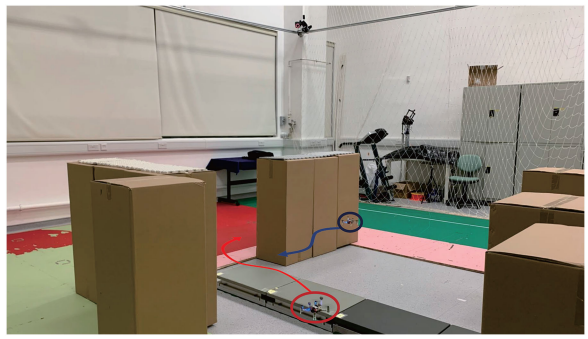


Fig. 7. Real flight scene.

E. Trajectory Replanning Strategy

As described in Section III-A, the entire optimization problem, considering the static or dynamic obstacles, is nonconvex, and hence, difficult to solve or demonstrate its convergence. Aside from that, the line-of-sight cost mentioned in Section III-D4 results in a discontinuous cost function, which indicates that it is impractical to solve the problem using gradient-based optimization methods. To efficiently solve such an optimization problem, herein the BSCPs are combined with the DE algorithm for local replanning.

DE is a gradient-free technique capable of efficiently finding the solution even when the optimization function is not continuous and has a more stable replanning time to find a feasible solution. The detailed DE process is given in Algorithm 2. First, the population Θ with random positions in the search space and relevant parameters are initialized (Lines 1–6). Each θ in Θ represents an end state of the quadrotor. Within each iteration, for each $\theta_{i,k}^{j}$ in population, mutation and crossover are carried out to yield a new agent $u_{i,k}^{j}$ (Lines 9–12). During the evaluation process, the predicted trajectory is obtained with the pretrained S_{NN}, and evaluated against the cost function J with the target's state $\mathbf{x}_t$, surrounding environment $\mathcal{X}$, and the cost map $\mathcal{M}_{\text{go}}$ (Lines 13–14). Then, the origin $\theta_{i,k}$ is replaced with $u_{i,k}$ if cost $c(u_{i,k})$ is less than $c(\theta_{i,k})$. Meanwhile, the global best is replaced with $u_{i,k}^{j}$ if the cost $c(u_{i,k})$ is less than $c(\theta^*)$ (Lines 15–22). With the progress of iterations, the search tree is gradually converged, and the best end state θ^* can be obtained. DE has better global search ability and convergence speed than many heuristic algorithms, such as particle swarm optimization and genetic algorithm.

IV. Simulation and Experimental Results

Simulation and experimental results of the proposed method are presented in this section to demonstrate its feasibility, efficiency, and robustness.

A. Experimental Setting

The relevant real flight experiments are carried out in an indoor Vicon environment, which is shown in Fig. 7. The diagram of the experimental system is illustrated in Fig. 8. The `Vicon Tracker` obtains the markers information on a PC and provides

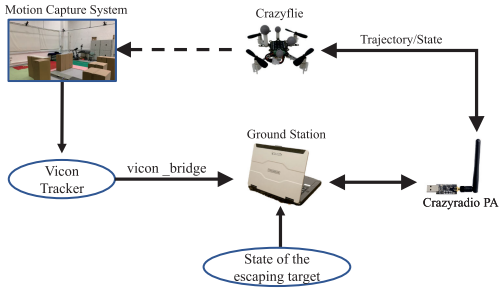


Fig. 8. Diagram of the experimental system.

Algorithm 2: DE Algorithm.

Input: Environment $\mathcal{X}$, Quadrotor's initial state $\mathbf{x}_q$, Target's state $\mathbf{x}_t$;
Output: Best end state θ^*;

$k \leftarrow 1$, η, CR (Initialization);
for $i = 1$ to size(Θ) **do**
  for $j = 1$ to Dimension=3 **do**
    $\theta_{i,k}^j = \theta_{\min}^j + \text{rand}(0,1) \cdot (\theta_{\max}^j - \theta_{\min}^j)$;
  end for
end for
for $m = 1$ to $iter_{\max}$ **do**
  for $i = 1$ to size(Θ) **do**
    for $j = 1$ to Dimension=3 **do**
      $v_{i,k}^j$ =Mutation$(\theta_{i,k}^j, \eta)$;
      $u_{i,k}^j$ =Crossover$(\theta_{i,k}^j, v_{i,k}^j, CR)$;
    end for
    $c(u_{i,k}^j) = J(S_{\text{NN}}(\mathbf{x}_q, u_{i,k}^j), \mathbf{x}_t, \mathcal{X}, \mathcal{M}_{\text{go}})$;
    $c(\theta_{i,k}) = J(S_{\text{NN}}(\mathbf{x}_q, \theta_{i,k}), \mathbf{x}_t, \mathcal{X}, \mathcal{M}_{\text{go}})$;
    if $c(u_{i,k}) < c(\theta_{i,k})$ **then**
      $\theta_{i,k} \leftarrow u_{i,k}$;
      if $c(u_{i,k}) < c(\theta^*)$ **then**
        $\theta^* \leftarrow u_{i,k}$;
      end if
    else
      $\theta_{i,k} \leftarrow \theta_{i,k}$;
    end if
  end for
  $k \leftarrow k + 1$;
end for

localization and obstacle sensing. A Crazyflie, a small, versatile quadrotor, was used as the experimental platform. The Crazyflie can survive high-speed crashes due to its low inertia and poses little risk to humans. The online chasing trajectory planning is done wirelessly on a laptop with an Intel I7 CPU, and the generated chasing trajectories, which include position, velocity, and acceleration are sent to the Crazyflie via a Crazyradio PA, a 2.4-GHz USB radio that transmits up two megabits per second in 32-byte packets.[1]

In the chasing scenario, as shown in Fig. 7, a quadrotor is tasked to chase the given 3-D predicted states of the simulated escaping target as quickly as possible while avoiding the static

[1][Online]. Available: https://www.bitcraze.io/

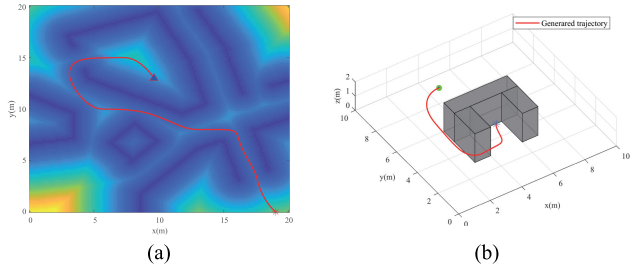


Fig. 9. Illustration of MPC-based trajectory generation of a quadrotor in obstacle-rich environments. (a) Motion planning in an EDT map. The red star and blue triangle are the start and goal points, respectively. (b) The illustration of motion planning in an environment with nonconvex obstacles. The blue star and green circle are the start and goal points, respectively.

TABLE II
COMPARISON BETWEEN NN AND FORWARD SIMULATION

	Average Time Consumption	Average error
NN approximation	26 μs	0.004
Forward Simulation	3.2 ms	—

* The comparison is done on an Intel I7 CPU and repeated ten times.

obstacles equipped in the experimental space or another mobile quadrotor acting as the dynamic obstacle moving in the experimental space. The kinodynamic feasibility constraints of the drone on the chasing trajectory's velocity, acceleration, and jerk are set as $v_c = [2.5, -0.5, 1.0]$, $a_c = [2.0, -0.5, 2.0]$, and $j_c = [5.0, -2.0, 3.0]$, respectively, each consisting of the maximum horizontal limit and the minimum and maximum vertical limits. The weight values are set as follows: $w_1 = 1.2, w_2 = 1.5$, $w_3 = 8$, and $w_4 = 0.5$. Inspired by the barrier method, we could increase the weight value of each corresponding term to yield a better performance. For example, we could increase w_2 for more accurate tracking. As shown in Fig. 9(a), the environment is represented as the aforementioned 3-D EDT map, in which the cost of each grid can be efficiently obtained. Unlike the gradient-based method, a smooth cost map is not necessary in the method proposed in this article, which decreases the computational burden. Moreover, deadlock is fully taken into account to improve the robustness to unstructured environments. In Fig. 9(b), the quadrotor is assigned to fly in an environment in which the vehicle may fall into a state of being stuck or deadlocked. The vehicle moves from a slot to its destination without encountering difficulties under the guidance of the cost-to-go map $\mathcal{M}_{\text{go}}$.

B. Performance Evaluation

With the efficiency of the NN approximation, the time consumption is much less than that of the forward process, which is shown in Table II. Moreover, for the ease of references, forward simulation and NN prediction are also conducted. The results are plotted and compared in Fig. 10. The initial states of the quadrotor are $\text{p}_0 = [0, 0]$, $\text{v}_0 = [1, 2]$, and $\text{a}_0 = [-1, -1]$, while the goal states are set as $\text{p}_g = [5, 4]$, $\text{v}_g = [0, 0]$, and $\text{a}_g = [0, 0]$. The NN prediction approximates forward simulation with an average error of 0.004, which indicates that the pretrained NN meets the requirements very well.

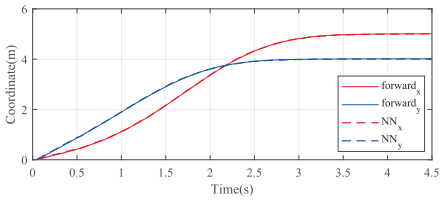


Fig. 10. NN prediction versus forward simulation of the quadrotor for trajectory generation.

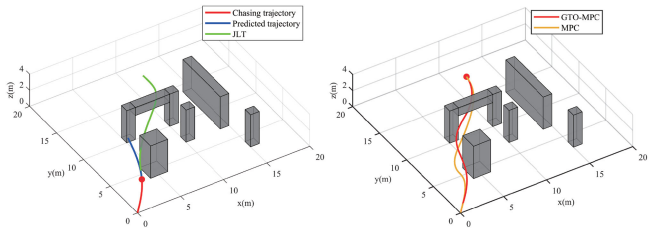


Fig. 11. Illustration of GTO-MPC and MPC-based trajectory generation of the quadrotor in a cluttered environment. (a) t = 3.2 s. (b) terminal.

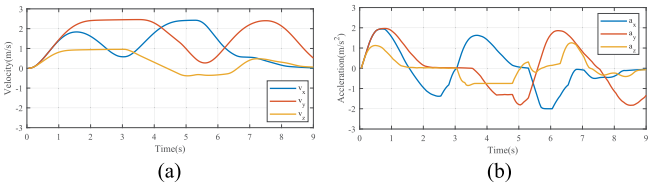


Fig. 12. Resultant velocities and accelerations. (a) Velocity. (b) Acceleration.

TABLE III
COMPARISON BETWEEN MPC AND GTO-MPC BASED METHODS

	Total flight time		Compuation time
	Longer trajectory	Shorter trajectory	
MPC [19]	9.6s	4.8s	34 ms
GTO-MPC (ours)	8.2s	3.8s	42 ms

In the target chasing simulations, first, the efficiency of the proposed method, i.e., reaching a given goal point as quickly as possible in an obstacle-rich environment, is verified. In the scenario depicted in Fig. 11, the quadrotor is tasked to fly in a $20 \times 20 \times 4$ m space containing four different pillars and one bridge hole. The following simulations are done using the MPC [19] and proposed GTO-MPC, separately, with a prediction horizon of $T = 2$ s and a replanning frequency of 5 Hz in MATLAB with an I7 CPU. In the optimization process, 20 candidates of the population are iterated over 15 times in the aforementioned two simulations. The resulting velocities and accelerations are plotted in Fig. 12, in which it can be verified that the quadrotor tracks the JLT and guarantees kinodynamic feasibility, flight safety, and smoothness simultaneously. The comparison between the MPC and the proposed GTO-MPC method is given in Table III. The results are acquired as the average of ten consecutive runs with the same initial state and two different goal states, resulting in two trajectories of different lengths. It is obvious that the GTO-MPC based method always takes less time to reach the given goal state with a slightly longer computational time of each replanning while guaranteeing flight efficiency and chasing performance.

TABLE IV
COMPARISON WITH OTHER METHODS

	Re-planning	deadlock avoidance	Safety	Time-optimal
JLT [24]	No	No	No	Yes
MPC [19]	Yes	Yes	Yes	No
LMPCC [17]	Yes	No	Yes	No
GTO-MPC (ours)	Yes	Yes	Yes	Yes

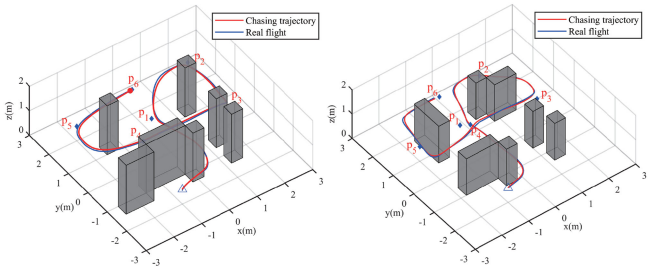


Fig. 13. Relevant chasing trajectories in two different scenarios. The blue rhombuses represent the sequential points. (a) Target chasing in an environment with non-convex obstacles. (b) Target chasing in an environment with considering line-of-sight maintenance.

The comparison of the proposed GTO-MPC and three other methods is shown in Table IV. Specifically, replanning means that the drone has the ability to plan several times as it flies, which improves its ability to react properly to potential threats. Time-optimal represents the consideration of the time-optimal property. As shown in Table IV, GTO-MPC is the only method that simultaneously replans, avoids deadlock, is able to handle static and dynamic obstacle avoidance, and satisfies the time-optimal property.

To verify the proposed method in actual implementation, simulations and flight experiments are also conducted in a $6 \times 6 \times 2$ m 3-D space (the physical limit of the VICON facility used) with pillars and other nonconvex obstacles. In the scenario depicted in Fig. 13(a), the quadrotor is commanded to chase the points p_i, $i = 1, 2, \ldots, 6$, the sequential predicted coordinates of a simulated escaping target, as quickly as possible with the guidance of the JLT. In the scenario shown in Fig. 13(b), the chasing task considering line-of-sight maintenance is performed in a different environment. As shown clearly, the quadrotor remains on the chasing task while maintaining deadlock avoidance and keeping the target in the line-of-sight as long as possible. Thus, it is proved once again that the proposed method can automatically avoid static and dynamic obstacles when chasing a simulated escaping target (see the online available experiment video). In the flight experiments, the maximum speed is approximately 1.9 m/s with an average speed of 1.2 m/s, and the quadrotor is always capable of chasing the given simulated points safely and quickly.

The relevant comparisons of replanning time between the proposed method and MPC [19], and Faster [29] are displayed in Fig. 14. All simulations run in C++ with an Intel Core I7 CPU. For DE algorithm in the proposed method, 20 candidates

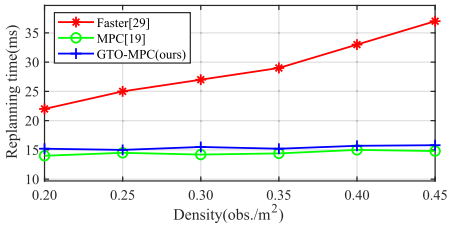


Fig. 14. Results of the replan time comparisons between the proposed method and Faster [29], and MPC [19].

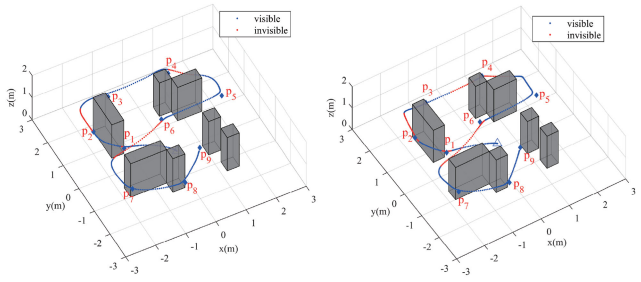


Fig. 15. Illustration of the ablation study. (a) The escaping target chasing with considering visibility. (b) The escaping target chasing without considering visibility.

of the population are iterated over 15 times, which, on average, takes 15 ms to find the best candidate solution. Compared to Faster [29], the proposed method has more uniform replanning time, which is essential for online planning of the quadrotor in cluttered environments. Meanwhile, it has almost the same replanning time of MPC [19] with shorter trajectory time. To improve the stability of solutions, in our framework, the chasing trajectory generated in the online MPC stage will be rechecked against the kinodynamic limits and surrounding environments. If the checking fails, the quadrotor will continue to execute the trajectory from the previous MPC cycle.

Implementing the same system with different methods, e.g., [10] and [12], in terms of visibility in the process of chasing an escaping target is almost unfeasible. Given that, ablation study with the proposed method by evaluating the visibility length of time along the whole chasing task is conducted to prove the utility of our work. In this context, as described in Section III-D4, the visible time t_{visible} can be measured by calculating the duration of time when the closest distance from the bearing vector $\vec{b}(\mathrm{p}_q, \mathrm{p}_t)$ to surrounding obstacles is no less than a threshold of 0.05 m. One can efficiently obtain the visibility information of the chasing trajectory at each runtime t benefiting from the simplicity and efficiency of the ray-tracing algorithm. To evaluate whether the quadrotor chases the mobile target well or not, the total time cost t_{total} and visible time t_{visible} are measured, which indicate how long the quadrotor maintains chasing the target without breaking the visibility constraint. The relevant results carried in an obstacle-rich environment are shown in Fig. 15 and some comparison items are list in Table V.

It is clear that the proposed method has a larger ratio of t_{visible} to t_{total}. This result means the quadrotor that generates the trajectory with the proposed method has more time on the visible trajectory compared to the entire chasing time t_{total}. In view of this, our proposed method is characterized by robustness to the visibility in target chasing mission.

TABLE V
COMPARISONS OF THE ABLATION STUDY

	t_{total}	t_{visible}	ratio
With visibility	19.2 s	15.8 s	83.1%
Without visibility	20.0 s	14.1 s	70.5%

* The comparison is repeated five times.

V. CONCLUSION

In this article, we proposed an effective GTO-MPC-based trajectory planning method for a quadrotor to chase an escaping target in obstacle-rich and dynamic environments. The motion primitives along the chasing trajectory were generated using an offline DP approach, and then, approximated by an NN during the online optimization process, which explicitly guarantees smoothness and kinodynamic feasibility. Furthermore, a gradient-free algorithm was applied to find the best possible solution to track the generated jerk limited trajectory while maintaining the overall flight safety, line-of-sight maintenance, and deadlock avoidance. The performance of the proposed method in terms of chasing trajectory time, optimization time consumption, and the ratio of visible time to total chasing time was analyzed. The experimental results had successfully demonstrated that the proposed technique yields an excellent performance. Our future work includes the incorporation of predictions of the escaping target and dynamic obstacles to further improve the overall performance.

REFERENCES

[1] M. Zhang, X. Liu, D. Xu, Z. Cao, and J. Yu, "Vision-based target-following guider for mobile robot," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9360–9371, Dec. 2019.
[2] M. Mueller, G. Sharma, N. Smith, and B. Ghanem, "Persistent aerial tracking system for UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1562–1569.
[3] C. C. Olsen, K. Kalyanam, W. P. Baker, and D. L. Kunz, "Maximal distance discounted and weighted revisit period: A utility approach to persistent unmanned surveillance," *Unmanned Syst.*, vol. 7, no. 4, pp. 215–232, May. 2019.
[4] D. Zheng, H. Wang, W. Chen, and Y. Wang, "Planning and tracking in image space for image-based visual servoing of a quadrotor," *IEEE Trans. Ind. Electron.*, vol. 65, no. 4, pp. 3376–3385, Apr. 2018.
[5] L. Zhang *et al.*, "Vision-based target three-dimensional geolocation using unmanned aerial vehicles," *IEEE Trans. Ind. Electron.*, vol. 65, no. 10, pp. 8052–8061, Oct. 2018.
[6] Y. Kim and B. K. Kim, "Time-optimal trajectory planning based on dynamics for differential-wheeled mobile robots with a geometric corridor," *IEEE Trans. Ind. Electron.*, vol. 64, no. 7, pp. 5502–5512, Jul. 2017.
[7] S. Dutta and C. Ekenna, "Air-to-ground surveillance using predictive pursuit," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 8234–8240.
[8] J. Chen and S. Shen, "Using a quadrotor to track a moving target with arbitrary relative motion patterns," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 5310–5317.
[9] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous flight for detection, localization, and tracking of moving targets with a small quadcopter," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1762–1769, Jul. 2017.
[10] B. F. Jeon and H. J. Kim, "Online trajectory generation of a MAV for chasing a moving target in 3D dense environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2019, pp. 1115–1121.

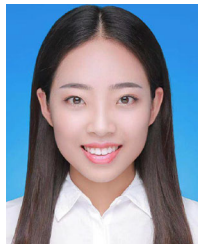
[11] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3725–3732, Oct. 2018.
[12] H. Shin and S. E. Yoon, "Optimization-based path planning for person following using following field," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2020, pp. 11352–11359.
[13] T. Howard, M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Model predictive motion planning: Several key developments for autonomous mobile robots," *IEEE Robot. Automat. Mag.*, vol. 21, no. 1, pp. 64–73, Mar. 2014.
[14] C. Liu and J. K. Hedrick, "Model predictive control-based target search and tracking using autonomous mobile robot with limited sensing domain," in *Proc. Amer. Control Conf.*, May 2017, pp. 2937–2942.
[15] G. Williams *et al.*, "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2017, pp. 1714–1721.
[16] C. Shen, Y. Shi, and B. Buckham, "Trajectory tracking control of an autonomous underwater vehicle using Lyapunov-based model predictive control," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5796–5805, Jul. 2018.
[17] B. Brito, B. Floor, L. Ferranti, and J. A. Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019.
[18] I. Lenz, R. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," *Robot., Sci. Syst.*, 2015, doi: 10.15607/RSS.2015.XI.012.
[19] S. Lai, M. Lan, and B. M. Chen, "Model predictive local motion planning with boundary state constrained primitives," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3577–3584, Oct. 2019.
[20] Y. Zhou *et al.*, "Towards autonomy of micro aerial vehicles in unknown and GPS-denied environments," *IEEE Trans. Ind. Electron.*, vol. 68, no. 8, pp. 7642–7651, Aug. 2021.
[21] Y. Song and D. Scaramuzza, "Learning high-level policies for model predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7629–7636.
[22] 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.*, vol. 9, no 2, pp. 165–198, 2021.
[23] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotor," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2011, pp. 2520–2525.
[24] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 3248–3253.
[25] S. Lai, M. Lan, Y. Li, and B. M. Chen, "Safe navigation of quadrotors with jerk limited trajectories," *Front. Inf. Technol. Electron. Eng.*, vol. 20, no 1, pp. 107–119, Jan. 2019.
[26] L. Busoniu *et al.*, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annu. Rev. Control*, pp. 1367–5788, vol. 46, pp. 8–28, Oct. 2018.
[27] R. Tallamraju, S. Rajappa, M. J. Black, K. Karlapalem, and A. Ahmad, "Decentralized MPC based obstacle avoidance for multi-robot target tracking scenarios," in *Proc. IEEE Int. Symp. Safety, Secur., Rescue Robot.*, Aug. 2018, pp. 1–8.
[28] P. Florence, J. Carter, and R. Tedrake, "Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps," in *Proc. Int. Workshop Algorithmic Foundations Robot.*, pp. 304–319, 2016.
[29] J. Tordesillas, B. T. Lopez and J. P. How, "FASTER: Fast and safe trajectory planner for flights in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2019, pp. 1934–1940.

Lele Xi received the B.E. degree in automation and M.Eng. degree in control engineering from the Hebei University of Science and Technology, Shijiazhuang, China, in 2014 and 2017, respectively. He is currently working toward the Ph.D. degree in control science and engineering with the School of Automation, Beijing Institute of Technology, Beijing, China.

He is currently involved in research works as a Visiting Ph.D. Student with the Department of Mechanical and Automation, Chinese University of Hong Kong, Hong Kong. His current research interests include motion planning, target tracking, and reinforcement learning, specifically in the area of aerial robotics.

Xinyi Wang received the B.E. degree from Xiamen University, Xiamen, China, in 2019. She is currently working toward the Ph.D. degree with the Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Hong Kong.

Her current research interests include motion planning, multiagent cooperation, and optimization.



Lei Jiao received the B.E. degree in automation from Shandong University, Shandong, China, in 2016, and the M.Eng. degree in control engineering in 2018 from the Beijing Institute of Technology, Beijing, China, where she is currently working toward the Ph.D. degree in control science and engineering with the School of Automation.

Her current research interests include intelligent optimization; multiagent cooperation and decision making; and differential game.

Shupeng Lai received the Bachelor of Engineering degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2012, and the Ph.D. degree in integrative sciences and engineering from the National University of Singapore, Singapore, in 2016.

He is currently an Adjunct Assistant Professor from the National University of Singapore. His research interests include motion planning, nonlinear model predictive control, multiagent systems, and aerial systems.

Zhihong Peng (Member, IEEE) received the B.S. degree from the Hunan University of Science and Technology, Xiangtan, China, in 1995, and the Ph.D. degree from Central South University, Changsha, China, in 2000.

She held one postdoctoral appointment with the Beijing Institute of Technology, Beijing, China, where since 2012, she has been a Professor. Her current research interests include intelligent information processing, multiagent cooperation, and optimization and decision.

Ben M. Chen (Fellow, IEEE) received the B.S. degree in mathematics and computer science from Xiamen University, Xiamen, China, in 1983, the M.S. degree in electrical engineering from Gonzaga University, Spokane, WA, USA, in 1988, and the Ph.D. degree in electrical and computer engineering from Washington State University, Pullman, WA, USA, in 1991.

He is currently a Professor of Mechanical and Automation Engineering with the Chinese University of Hong Kong, Hong Kong. He was a Provost's Chair Professor with 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 with the State University of New York at Stony Brook, in 1992–1993. He has authored/coauthored more than 450 journal and conference articles, and a dozen research monographs in control theory and applications, unmanned systems, and financial market modeling. His current research interests include unmanned systems, robust control, and control applications.

Dr. Chen is a Fellow of the Academy of Engineering, Singapore. 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 for *Unmanned Systems*. He was the recipient of a number of research awards. His research team has actively participated in international UAV competitions and won many championships in the contests.