IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 6, 2022, accepted January 28, 2022, date of publication January 31, 2022, date of current version February 7, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3148145

Communication-Free MPC-Based Neighbors Trajectory Prediction for Distributed Multi-UAV Motion Planning

ZIJIA NIU[1,2,5], XIAOHU JIA[1,2,5], AND WANG YAO[2,3,4]

[1]School of Mathematical Sciences, Beihang University, Beijing 100191, China
[2]Key Laboratory of Mathematics, Informatics and Behavioral Semantics, Ministry of Education, Beijing 100191, China
[3]Institute of Artificial Intelligence, Beihang University, Beijing 100191, China
[4]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China
[5]Peng Cheng Laboratory, Shenzhen, Guangdong 518055, China

Corresponding author: Wang Yao (yaowang@buaa.edu.cn)

This work was supported by the Science and Technology Innovation 2030-Key Project under Grant 2020AAA0108200.

ABSTRACT In an environment with multiple static obstacles, UAVs usually communicate with each other to avoid collisions during trajectory planning. However, such communication may become infeasible or unreliable due to interference or jam in practice. This paper introduces a neighbors trajectory prediction algorithm based on model predictive control (MPC), which enables each UAV to predict the motion behavior of its neighbors without communication. By solving the MPC model of its neighbors, an UAV can predict their trajectories and then avoid collision with them in the future. To prove the practicability, we integrate the proposed algorithm into distributed model predictive control (DMPC) framework to realize multi-UAV trajectory planning without communication and with static obstacles. The performance of our method is verified by simulation experiments in two scenes.

INDEX TERMS Collision avoidance, model predictive control, neighbors trajectory prediction, path planning for multiple mobile robots or agents.

I. INTRODUCTION

Autonomous navigation and trajectory generation of multi-UAV is very important and widely used in many fields, such as disaster rescue [1], warehouse management [2], multi-view photography [3], smart agriculture [4], and so on. In order to predict the collision with other individuals, UAVs need to communicate their planned future trajectory. In actual scenes, communication jam or interference may occur, which will cause the interaction between UAVs to become unavailable or unreliable. Therefore, it is particularly important for multi-UAV to fly safely without communication.

For the safe and reliable trajectory planning of multi-UAV, there are some traditional distributed approaches: artificial potential field (APF) [5], [6], optimal reciprocal collision avoidance (ORCA) [7], force-based motion planning (FMP) [8], some methods based on swarm intelligence [9], and so on. These methods have their own advantages, but they also have some shortcomings, such as only calculating the state of the next instant. In recent years, reinforcement learning (RL) has been applied to multi-UAV trajectory planning [10]–[12]. UAVs optimize their learning strategies according to the rewards they get from performing each action. However, a bad reward function will make these algorithms difficult to train effectively. In distributed model predictive control (DMPC) [13], each UAV solves an optimization problem in a receding horizon manner to obtain its future trajectory [14]. It has the ability to predict the states of a long period of time in the future. For an individual in multi-UAV, when solving a local optimization problem with collision avoidance constraints, it needs to know the future trajectory of its neighbors. However, when the communication between UAVs is not feasible, this kind of states interaction will face difficulties.

Predicting the motion behavior of other individuals is a good way to avoid collision without communication. Trajectory prediction has been researched through multiple

The associate editor coordinating the review of this manuscript and approving it for publication was Emre Koyuncu.

methods. In [15], a social forces model has been proposed to describe pedestrian motions by simulating attractive and repulsive potentials. A method based on hierarchical reasoning game theory has been proposed to simulate driver and vehicle interactions in traffic under given conditions [16]. In addition, there are methods such as Bayesian formulation [17], interactive Gaussian Processes (IGP) [18] and Linear Trajectory Avoidance (LTA) [19], etc. These methods have their own characteristics, but still have some problems, such as low prediction accuracy or high calculation cost. In recent years, many methods based on deep learning have been used in trajectory prediction. Literature [20] has introduced a model based on long-short term memory networks (LSTM), which incorporated both static obstacles and surrounding pedestrians for trajectory forecasting. Inspired by [20], an interaction-aware trajectory prediction model based on recurrent neural networks (RNN) has been proposed [21]. In [22], a generative adversarial networks (GANs) encoder-decoder framework has been developed for capturing the multi-modality of the future trajectory prediction. The prediction accuracy of learning-based methods is high, but most methods need a large number of samples to train the model. This limits the application of learning-based methods in the field of trajectory prediction.

In this paper, we propose an algorithm that enables each UAV to predict the future trajectories of its neighbors without communication, and then combine it with DMPC framework to complete the trajectory planning task of multi-UAV. In order to get closer to the actual scene, we also consider the collision avoidance between UAVs and multiple static obstacles, which makes a great increase in the difficulty of neighbors trajectory prediction. Fig.1 gives an overview of the proposed method. Firstly, each UAV obtains the current states of its neighbors through sensors (such as depth cameras and optical flow sensors, etc). Then model predictive control (MPC) is used to establish a receding horizon optimization problem for each neighbor. By solving these optimization problems, an UAV can get the future trajectories of all its neighbors. Finally, the proposed algorithm is integrated into DMPC framework to obtain the future state sequence of each UAV. For an optimization problem, whether it is established by an UAV to predict its neighbor trajectory or its own trajectory, we transform it into a quadratic programming problem. Considering the collision avoidance constraints for neighbors and static obstacles in this problem, we use the predicted first collision to establish soft constraints, thereby reducing the complexity of the model and the difficulty of solving it. We implement the offline sequential form of the above method. In theory, it can also be applied in a distributed form. Through simulation experiments in two specific scenarios, the feasibility and effectiveness of our proposed method are verified. The major contributions of this paper are summarized as follows:

(1) A neighbors trajectory prediction algorithm based on MPC is proposed to overcome the absence of communication;

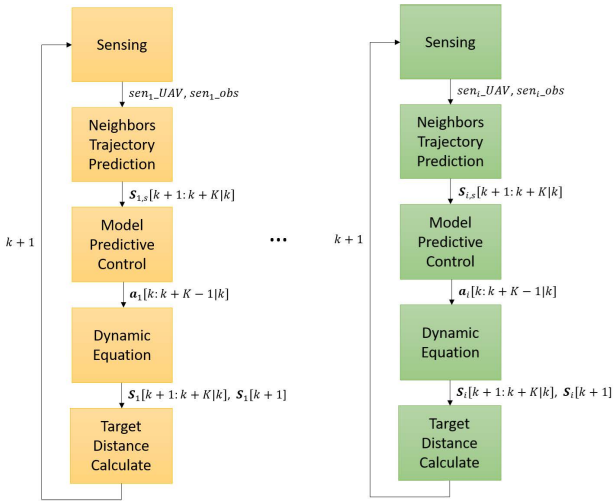

FIGURE 1. The overall framework of the proposed method. It mainly includes neighbors trajectory prediction and trajectory planning based on MPC. sen_i_UAV, sen_i_obs respectively represent other UAVs and obstacles in the sensing area detected by n_i. $S_{i,s}[k+1:k+K|k]$ is defined as the state sequence of a neighbor of n_i at K times in the future. $S_i[k+1:k+K|k]$ and $a_i[k:k+K-1|k]$ are denoted as the state and control sequence of n_i at K times, respectively. $S_i[k+1]$ is the next state of n_i. After trajectory planning, n_i calculates whether its position at the next time reaches the target. If not, proceed to the next round.

(2) The neighbors trajectory prediction algorithm is integrated into DMPC framework to realize the trajectory planning of multi-UAV;

(3) We conducted simulation experiments of the proposed algorithm in two scenes, and the results show that the proposed method has good performance.

The rest of this paper is organized as follows: Section II describes the preliminaries. Section III details the proposed method. The effectiveness of the proposed method is verified by simulation experiments in Section IV. Section V gives the conclusion.

II. PRELIMINARIES

A. UAV MODEL

The multi-UAV set composed of N homogeneous UAVs is defined as $\mathcal{N} = \{n_1, \ldots, n_i, \ldots, n_N\}$, where n_i represents the ith UAV. The downwash air generated by a quadrotor in flight will affect the safety of other UAVs under it. Similar to the literature [23], we model each UAV as an ellipsoid elongated along the vertical axis. An UAV at position $\boldsymbol{p} \in \mathbb{R}^3$ can be represented by a set of points as follows:

$$\xi(\boldsymbol{p}) = \{\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{p} : \|\boldsymbol{x}\|_2 \leq r\}, \tag{1}$$

where $\boldsymbol{Q} = diag(1, 1, r_c)$, $r_c > 1$, and r is the radius of the UAV in the XY plane.

Each UAV is assumed to satisfy double integral dynamics. Let $\boldsymbol{p}_i \in \mathbb{R}^3, \boldsymbol{v}_i \in \mathbb{R}^3, \boldsymbol{a}_i \in \mathbb{R}^3$ denote the position, velocity, and acceleration of the UAV n_i respectively. The dynamic equation of n_i is

$$\boldsymbol{p}_i[k+1] = \boldsymbol{p}_i[k] + d\boldsymbol{v}_i[k] + \frac{d^2}{2}\boldsymbol{a}_i[k]$$

$$\boldsymbol{v}_i[k+1] = \boldsymbol{v}_i[k] + d\boldsymbol{a}_i[k], \tag{2}$$

where k is a discrete time point and d is a discrete time step. $\boldsymbol{s}_i = \left[\boldsymbol{p}_i^T, \boldsymbol{v}_i^T\right]^T \in \mathbb{R}^6$ and $\boldsymbol{u}_i = \boldsymbol{a}_i \in \mathbb{R}^3$ represent the state quantity and control quantity of UAV n_i respectively. Then (1) can be written as

$$\boldsymbol{s}_i[k+1] = \boldsymbol{A}\boldsymbol{s}_i[k] + \boldsymbol{B}\boldsymbol{u}_i[k], \tag{3}$$

where $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I} & d\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{6\times6}$, $\boldsymbol{B} = \begin{bmatrix} \frac{d^2}{2}\boldsymbol{I} \\ d\boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{6\times3}$. $\boldsymbol{I} \in \mathbb{R}^{3\times3}$, $\boldsymbol{0} \in \mathbb{R}^{3\times3}$ denote the 3×3 identity matrix and 3×3 zero matrix respectively.

B. MUTIL-UAV COLLISION AVOIDANCE

For an insight on collision avoidance approaches of UAVs, the reader is referred to [24]. If UAV n_i wants to avoid collision with n_j at the time point k, the following collision avoidance constraint should be satisfied:

$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_i[k] - \boldsymbol{p}_j[k])\|_2 \geq 2r_u, \tag{4}$$

where $\boldsymbol{p}_i[k]$ and $\boldsymbol{p}_j[k]$ represent the positions of n_i and n_j at time point k, respectively. $2r_u$ is the minimum distance between two UAVs on the XY plane, where r_u is slightly larger than the radius of one UAV.

Suppose that the set of M static obstacles in the environment is $\mathcal{O} = \{O_1, \ldots, O_m, \ldots, O_M\}$. $\boldsymbol{p}_{i,m}[k]$ is defined as the closest point to UAV n_i on the mth obstacle O_m at time point k, namely:

$$\boldsymbol{p}_{i,m}[k] = \arg\min_{\boldsymbol{p}\in O_m} \|\boldsymbol{p} - \boldsymbol{p}_i[k]\|_2. \tag{5}$$

If n_i wants to avoid collision with O_m at the time point k, it needs to satisfy the following collision avoidance constraint:

$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_i[k] - \boldsymbol{p}_{i,m}[k])\|_2 \geq r_u. \tag{6}$$

Regarding the acquisition of $\boldsymbol{p}_{i,m}[k]$, it is assumed that it can be obtained by the sensors of n_i. Or according to the literature [25], if the obstacle O_m has a hyperplane boundary, and the boundary has a unit normal $\boldsymbol{a}_m \in \mathbb{R}^3$ and passes through the point $\boldsymbol{y}_m \in \mathbb{R}^3$, then the point on O_m that is closest to n_i can be determined by

$$\boldsymbol{p}_{i,m}[k] = \boldsymbol{P}\boldsymbol{p}_i[k] + (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}_m, \tag{7}$$

where $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{a}_m\boldsymbol{a}_m^T$. If the obstacle O_m can be simplified as a closed sphere with radius r_m centered at $\boldsymbol{y_m} \in \mathbb{R}^3$, then the point on O_m that is closest to n_i can be determined by

$$\boldsymbol{p}_{i,m}[k] = \mu\boldsymbol{p}_i[k] + (1-\mu)\boldsymbol{y}_m, \tag{8}$$

where $\mu = r_m/\|\boldsymbol{p}_i[k] - \boldsymbol{y}_m\|_2$.

C. MODEL PREDICTIVE CONTROL

1) THE PREDICTION MODEL OF UAV

According to the principle of DMPC [13], [26], the flight time of UAVs is discretized. Assuming that the current time point is k, each UAV wants to predict its states at K future time points. Let $\boldsymbol{s}_i[k+t|k] = [\boldsymbol{p}_i[k+t|k]^T, \boldsymbol{v}_i[k+t|k]^T]^T \in \mathbb{R}^6$ denotes the state of UAV n_i at the tth time point after instant k, $t \in \{1, 2, \ldots, K\}$. The following formula can be deduced from (3):

$$\boldsymbol{s}_i[k+t|k] = \boldsymbol{A}\boldsymbol{s}_i[k+t-1|k] + \boldsymbol{B}\boldsymbol{u}_i[k+t-1|k]. \tag{9}$$

Let $\boldsymbol{P}_i = [\boldsymbol{p}_i[k+1|k]^T, \boldsymbol{p}_i[k+2|k]^T, \ldots, \boldsymbol{p}_i[k+K|k]^T]^T \in \mathbb{R}^{3K}$, $\boldsymbol{U}_i = [\boldsymbol{u}_i[k|k]^T, \boldsymbol{u}_i[k+1|k]^T, \ldots, \boldsymbol{u}_i[k+K-1|k]^T]^T \in \mathbb{R}^{3K}$ denote the position and control sequence of UAV n_i at K time points in the future, respectively. $\boldsymbol{S}_i^0 = \boldsymbol{s}_i[k|k]$ is the initial state of instant k, then an affine function from $\boldsymbol{U}_i$ to $\boldsymbol{P}_i$ can be obtained through the recurrence relation in (9):

$$\boldsymbol{P}_i = \boldsymbol{A}_0\boldsymbol{S}_i^0 + \boldsymbol{\Phi}\boldsymbol{U}_i. \tag{10}$$

Let $\boldsymbol{C} = \begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\end{bmatrix} \in \mathbb{R}^{3\times6}$, the matrix $\boldsymbol{A}_0 \in \mathbb{R}^{3K\times6}$ in (10) is defined as

$$\boldsymbol{A}_0 = \begin{bmatrix}(\boldsymbol{CA})^T & (\boldsymbol{CA}^2)^T & \ldots & (\boldsymbol{CA}^{\boldsymbol{K}})^T\end{bmatrix}^T. \tag{11}$$

The matrix $\boldsymbol{\Phi} \in \mathbb{R}^{3K\times3K}$ in (10) is defined as

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{CB} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{CAB} & \boldsymbol{CB} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA^{K-1}B} & \boldsymbol{CA^{K-2}B} & \cdots & \boldsymbol{CB} \end{bmatrix}. \tag{12}$$

Similarly, we can get the affine function from $\boldsymbol{U}_i$ to $\boldsymbol{V}_i$:

$$\boldsymbol{V}_i = \boldsymbol{A}_0'\boldsymbol{S}_i^0 + \boldsymbol{\Phi}'\boldsymbol{U}_i, \tag{13}$$

where $\boldsymbol{A}_0'$ and $\boldsymbol{\Phi}'$ have the same matrix form as $\boldsymbol{A}_0$ and $\boldsymbol{\Phi}$, except that the matrix $\boldsymbol{C}' = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\end{bmatrix} \in \mathbb{R}^{3\times6}$ instead of $\boldsymbol{C}$ is brought into them.

2) OBJECTIVE FUNCTION

The objective function for each UAV needs to balance the following three aspects: (1) The control effort. (2) The change range of control quantity between two instants. (3) The distance between the planned position point and target point. It is established as follows:

$$\begin{aligned} J_i = \sum_{t=0}^{K-1} (&\|\boldsymbol{u}_i[k+t|k]\|_{\boldsymbol{R_1}}^2 + \|\boldsymbol{u}_i[k+t|k] \\ &- \boldsymbol{u}_i[k+t-1|k]\|_{\boldsymbol{R_2}}^2) + \|\boldsymbol{p}_i[k+K|k] - \boldsymbol{p}_{fi}\|_{\boldsymbol{R_3}}^2, \end{aligned} \tag{14}$$

where $\boldsymbol{p}_i[k+K|k]$ is the last position in n_i's position sequence of future K time points and $\boldsymbol{p}_{fi}$ is n_i's target point. According to (10) and $\boldsymbol{U}_i$, the objective function can be transformed into a quadratic form:

$$\begin{aligned} J_i = \boldsymbol{U}_i^T(\boldsymbol{R}_1 + \boldsymbol{\Gamma}^T\boldsymbol{R}_2\boldsymbol{\Gamma} + \boldsymbol{\Phi}^T\boldsymbol{R}_3\boldsymbol{\Phi})\boldsymbol{U}_i + 2(\boldsymbol{S}_i^{0T}\boldsymbol{A}_0^T\boldsymbol{R}_3\boldsymbol{\Phi} \\ - \boldsymbol{P}_{fi}^T\boldsymbol{R}_3\boldsymbol{\Phi} - \boldsymbol{U}_i^{'T}\boldsymbol{R}_2\boldsymbol{\Gamma})\boldsymbol{U}_i, \end{aligned} \tag{15}$$

where $\boldsymbol{R}_1 = diag(\boldsymbol{R}'_1, \ldots, \boldsymbol{R}'_1)$, $\boldsymbol{R}_2 = diag(\boldsymbol{R}'_2, \ldots, \boldsymbol{R}'_2)$, $\boldsymbol{R}_3 = diag(\boldsymbol{0}, \ldots, \boldsymbol{R}'_3) \in \mathbb{R}^{3K\times 3K}$ are all positive definite and block-diagonal matrices, they are the weights of the three penalties. $\boldsymbol{R}'_1, \boldsymbol{R}'_2, \boldsymbol{R}'_3 \in \mathbb{R}^{3\times 3}$ are all positive definite diagonal matrices. The matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{3K\times 3K}$ is defined as follows:

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix}, \tag{16}$$

the matrix $\boldsymbol{U}'_i \in \mathbb{R}^{3K}$ is defined as follows:

$$\boldsymbol{U}'_i = \begin{bmatrix} \boldsymbol{u}_i[k-1]^T & \boldsymbol{0}^T_{3\times 1} & \cdots & \boldsymbol{0}^T_{3\times 1} \end{bmatrix}^T, \tag{17}$$

where $\boldsymbol{u}_i[k-1]^T$ represents the control quantity of UAV n_i at the last time of current time k, and $\boldsymbol{0}^T_{3\times 1} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$. $\boldsymbol{P}_{fi} = \begin{bmatrix} \boldsymbol{0}^T_{3\times 1} & \boldsymbol{0}^T_{3\times 1} & \cdots & \boldsymbol{p}^T_{fi} \end{bmatrix}^T \in \mathbb{R}^{3K}$ is the vector of n_i's target point.

3) CONSTRAINTS AND MODEL

Due to the physical characteristics, space limitations, and collision avoidance requirements of UAV, the state and control constraints of UAV n_i are as follows:

$$\boldsymbol{U}_{min} \le \boldsymbol{U}_i \le \boldsymbol{U}_{max}, \tag{18}$$
$$\boldsymbol{P}_{min} \le \boldsymbol{P}_i \le \boldsymbol{P}_{max}, \tag{19}$$
$$\boldsymbol{V}_{min} \le \boldsymbol{V}_i \le \boldsymbol{V}_{max}, \tag{20}$$
$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_i[k+t|k] - \boldsymbol{p}_j[k+t|k])\|_2 \ge 2r_u, \tag{21}$$
$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_i[k+t|k] - \boldsymbol{p}_{O_m}[k+t|k])\|_2 \ge r_u, \tag{22}$$
$$\forall j \ne i \in \mathcal{N}; \quad \forall O_m \in \mathcal{O}; \ \forall t \in \{1, 2, \ldots, K\}, \tag{23}$$

where $\boldsymbol{U}_{min}, \boldsymbol{U}_{max}, \boldsymbol{P}_{min}, \boldsymbol{P}_{max}, \boldsymbol{V}_{min}, \boldsymbol{V}_{max} \in \mathbb{R}^{3K}$ are the boundary values of control, position and velocity, respectively. After the above constraints are processed by using (10), (13) and linearization methods, the model of n_i can be established:

$$\begin{aligned} &\min_{\boldsymbol{U}_i} J_i = \frac{1}{2}\boldsymbol{U}_i^T \boldsymbol{H} \boldsymbol{U}_i + \boldsymbol{f}_i \boldsymbol{U}_i \\ &s.t. \ \boldsymbol{\Pi} \boldsymbol{U}_i \le \boldsymbol{b}_i. \end{aligned} \tag{24}$$

Generally, there are two ways for UAV n_i to obtain the future trajectories of other UAVs ($\boldsymbol{p}_j[k+t|k]$ in (21)). One is through communication. UAVs communicate their planned future trajectories to others at each time step, but when the number of UAVs is large, this method consumes a lot. In addition, communication is not always available and reliable in practice. Another method is n_i to treat other UAVs as moving at a constant velocity and calculate their future trajectories according to their current state. However, this method can be inaccurate and may lead to unsafe trajectory planning [27]. In this paper, we propose an MPC-based algorithm for each UAV to predict its neighbors' future trajectories without communication.

III. METHOD

In this section, we introduce in detail the proposed neighbors trajectory prediction algorithm and how to integrate it into DMPC framework to achieve the trajectory planning of multi-UAV.

A. PROBLEM FORMULATION

The goal of our works is to generate collision-free trajectories of multi-UAV from their initial locations to their target locations under an environment with static obstacles and without communication. In this situation, UAVs can not obtain the future trajectories of other individuals through information interaction. We propose an algorithm in which an UAV predicts the future state sequences of neighbors and judges whether collisions will occur.

Before describing our proposed algorithm, the following reasonable assumptions are made:

(1) Each UAV has a sensing area. Assuming that the sensing area is a sphere with radius d_s, each UAV can obtain the states (positions and velocities) of other individuals in its sensing area by its own sensors;

(2) Each UAV has an inter-UAV safe area and an UAV-obstacle safe area. They are spheres with radii of d_u and d_o respectively, and d_u, d_o, d_s satisfy $d_u < d_o \ll d_s$. Only when other individuals or obstacles are in its safe area, one UAV will consider collision avoidance;

(3) Due to the homogeneity of UAVs, they have the same size of sensing area, inter-UAV safe area, and UAV-obstacle safe area.

Based on the above assumptions, the UAVs' sensing area and safe areas are shown in Fig. 2 (The mappings of 3D areas on 2D).

B. NEIGHBORS TRAJECTORY PREDICTION

For an UAV n_i, the set of neighbors it senses in its safe area at current instant is defined as $\Theta_i = \{ne_i^1, ne_i^2, \ldots, ne_i^g\}$ $(1 \le g < N)$. For any neighbor $ne_i^s \in \Theta_i$ $(1 \le s \le g)$, n_i uses quadratic programming to solve an MPC problem about ne_i^s to obtain its future state sequence.

The collisions ne_i^s may encounter are considered. According to the second assumption in Section III-A, the sensing radius of UAV is much larger than the safe radius, so it can be counted that ne_i^s's neighbors set $\Theta_{i,s} = \{ne_{i,s}^1, ne_{i,s}^2, \ldots, ne_{i,s}^l\}$ can be sensed by n_i. We regard ne_i^s's neighbors as individuals moving at a constant velocity. If at the time point $k + t_1 (1 \le t_1 \le K-1)$ that after the current instant k, there exists a set $\Theta_{i,s}^{k+t_1} \subseteq \Theta_{i,s}$, which satisfies the following constraint for any $ne_{i,s}^q \in \Theta_{i,s}^{k+t_1}$:

$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_{i,s}[k+t_1] - \boldsymbol{p}_{i,s,q}[k+t_1])\|_2 \le 2r_u, \tag{25}$$

we call ne_i^s collides with its neighbors at $k+t_1$. The obstacles set sensed by n_i at instant k is represented as $\mathcal{O}_i^k \subseteq \mathcal{O}$. If at the time point $k + t_2 (1 \le t_2 \le K-1)$ that after the instant k, there exists a set $\mathcal{O}_{i,s}^{k+t_2} \subseteq \mathcal{O}_i^k$, which satisfies the following

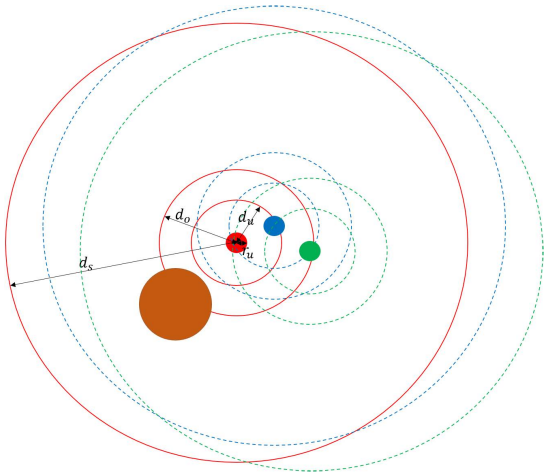

FIGURE 2. The sensing area and safe areas of UAVs (The mappings of 3D areas on 2D). The red, green and blue circles represent three UAVs respectively. The brown circle represents a static obstacle. r_u is sightly larger than the radius of one UAV. d_s, d_o and d_u represent the radius of sensing area, inter-UAV safe area, and UAV-obstacle safe area respectively. According to the second assumption in Section III-A, d_s is much larger than two safe radii. Due to the limitation of the layout, it is only schematic here, not strictly proportional.

constraint for any $O_{i,s}^m \in \mathcal{O}_{i,s}^{k+t_2}$:

$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_{i,s}[k+t_2]-\boldsymbol{p}_{i,s,m}[k+t_2])\|_2 \le r_u, \tag{26}$$

we call ne_i^s collides with obstacles at $k+t_2$. $\boldsymbol{p}_{i,s,m}[k+t_2]$ is the point closest to ne_i^s on obstacle $O_{i,s}^m$ at instant $k+t_2$. In (25) and (26), $\boldsymbol{p}_{i,s}[k+t_1]$ and $\boldsymbol{p}_{i,s}[k+t_2]$ are the predicted positions of ne_i^s at $k+t_1$ and $k+t_2$ respectively, which are calculated by UAV n_i. If ne_i^s is in the safe area of n_i at instant $k-1$, then $\boldsymbol{p}_{i,s}[k+t_*]$ can be obtained from the state sequence of ne_i^s at K future time points calculated by n_i at instant $k-1$. If ne_i^s is not in the safe area of n_i at instant $k-1$, or the current time is the initial time, then n_i treats ne_i^s as a constant velocity individual and calculates its position at $K-1$ time points in the future, so as to obtain $\boldsymbol{p}_{i,s}[k+t_*]$.

According to the description above, n_i can get the instants when ne_i^s collides with other UAVs or static obstacles. Similar to the literature [14], in order to simplify the model, we select the instant when ne_i^s first collides to constrain its state, this instant is assumed as $k+t_0$. In addition, we adopt the soft constraint to make the planning problem more solvable. The future collisions of ne_i^s can be divided into the following three categories:

(1) The UAV ne_i^s only collides with other UAVs at instant $k+t_0$. This occurs in three cases: 1) there are no obstacles in the safe area of ne_i^s; 2) there are obstacles in the safe area of ne_i^s, but there will be no collision in the $K-1$ future time points; 3) the collision time between ne_i^s and obstacles is after $k+t_0$. Assuming that the set of UAVs that collide with ne_i^s at $k+t_0$ is $\Theta_{i,s}^{k+t_0} \subseteq \Theta_{i,s}$, then for any $ne_{i,s}^q \in \Theta_{i,s}^{k+t_0}$, ne_i^s needs to satisfy the following constraint:

$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_{i,s}[k+t_0|k]-\boldsymbol{p}_{i,s,q}[k+t_0])\|_2 \ge 2r_u-\varepsilon_{i,s,q}^{(1)}, \tag{27}$$

where $\varepsilon_{i,s,q}^{(1)} > 0$ is a new variable to relax the constraint.

In order to linearize (27), Taylor expansion is used at $\boldsymbol{p}_{i,s}[k+t_0]$, where $\boldsymbol{p}_{i,s}[k+t_0]$ is the position of ne_i^s at the future instant $k+t_0$, which calculated by n_i in advance. This is one of the methods for linearizing collision avoidance constraints mentioned in Section II-C 3). The result is as follows:

$$(\boldsymbol{\omega}_{i,s,q}^{(1)})^T\boldsymbol{p}_{i,s}[k+t_0|k]+\varepsilon_{i,s,q}^{(1)}\rho_{i,s,q}^{(1)} \ge \eta_{i,s,q}^{(1)}, \tag{28}$$

where $\boldsymbol{\omega}_{i,s,q}^{(1)} = \boldsymbol{Q}^{-2}(\boldsymbol{p}_{i,s}[k+t_0]-\boldsymbol{p}_{i,s,q}[k+t_0])$, $\rho_{i,s,q}^{(1)} = \|\boldsymbol{Q}^{-1}(\boldsymbol{p}_{i,s}[k+t_0]-\boldsymbol{p}_{i,s,q}[k+t_0])\|_2$ and $\eta_{i,s,q}^{(1)} = 2r_u\rho_{i,s,q}^{(1)} - (\rho_{i,s,q}^{(1)})^2 + (\boldsymbol{\omega}_{i,s,q}^{(1)})^T\boldsymbol{p}_{i,s}[k+t_0]$.

(2) The UAV ne_i^s only collides with static obstacles at instant $k+t_0$. This occurs in three cases: 1) there are no other UAVs in the safe area of ne_i^s; 2) there are other UAVs in the safe area of ne_i^s, but there will be no collision in the $K-1$ future time points; 3) the collision time between ne_i^s and other UAVs is after $k+t_0$. Assuming that the set of obstacles which collide with ne_i^s at $k+t_0$ is $\mathcal{O}_{i,s}^{k+t_0} \subseteq \mathcal{O}_{i,s}^k$, then for any $O_{i,s}^m \in \mathcal{O}_{i,s}^{k+t_0}$, ne_i^s needs to satisfy the following constraint:

$$\|\boldsymbol{Q}^{-1}(\boldsymbol{p}_{i,s}[k+t_0|k]-\boldsymbol{p}_{i,s,m}[k+t_0])\|_2 > r_u-\varepsilon_{i,s,m}^{(2)}. \tag{29}$$

It is linearized by Taylor expansion

$$(\boldsymbol{\omega}_{i,s,m}^{(2)})^T\boldsymbol{p}_{i,s}[k+t_0|k]+\varepsilon_{i,s,m}^{(2)}\rho_{i,s,m}^{(2)} \ge \eta_{i,s,m}^{(2)}, \tag{30}$$

where $\boldsymbol{\omega}_{i,s,m}^{(2)} = \boldsymbol{Q}^{-2}(\boldsymbol{p}_{i,s}[k+t_0]-\boldsymbol{p}_{i,s,m}[k+t_0])$, $\rho_{i,s,m}^{(2)} = \|\boldsymbol{Q}^{-1}(\boldsymbol{p}_{i,s}[k+t_0]-\boldsymbol{p}_{i,s,m}[k+t_0])\|_2$, and $\eta_{i,s,m}^{(2)} = r_u\rho_{i,s,m}^{(2)} - (\rho_{i,s,m}^{(2)})^2 + (\boldsymbol{\omega}_{i,s,m}^{(2)})^T\boldsymbol{p}_{i,s}[k+t_0]$.

(3) The UAV ne_i^s collides with both other UAVs and static obstacles at instant $k+t_0$. In this case, the above two constraints (27) and (29) ne_i^s need to be satisfied.

We unify the forms of collision avoidance constraints in the above three cases. $\varepsilon_{i,s,*}$, $\boldsymbol{\omega}_{i,s,*}$, $\rho_{i,s,*}$, $\eta_{i,s,*}$ are defined as general references to the corresponding symbols in (28) and (30). According to the actual situation, choose one type of specific symbol (corresponding to the above case (1) or (2)), or the combination of the two types of symbols (corresponding to the case (3)). (28) or (30) can be transformed into the following form by (10):

$$\boldsymbol{\sigma}_{i,s,*}^T\boldsymbol{\Phi}\boldsymbol{U}_{i,s}+\rho_{i,s,*}\varepsilon_{i,s,*} \ge \eta_{i,s,*}-\boldsymbol{\sigma}_{i,s,*}^T\boldsymbol{A}_0\boldsymbol{S}_{i,s}^0, \tag{31}$$

where $\boldsymbol{\sigma}_{i,s,*} = \left[\boldsymbol{0}_{3(t_0-1)\times 1}^T\ \boldsymbol{\omega}_{i,s,*}^T\ \boldsymbol{0}_{3(K-t_0)\times 1}^T\right]^T$.

It can be seen from the above that the soft constraint is equivalent to adding a relaxation variable $\varepsilon_{i,s,*}$ to each collision avoidance constraint of ne_i^s, which is a new control variable. Let num_{is} denote the number of $\varepsilon_{i,s,*}$. Then when they are other UAVs that collide with ne_i^s at $k+t_0$, $num_{is} = dim(\Theta_{i,s}^{k+t_0})$; when they are static obstacles, $num_{is} = dim(\mathcal{O}_{i,s}^{k+t_0})$; If at $k+t_0$, other UAVs and obstacles collide with ne_i^s at the same time, then $num_{is} = dim(\Theta_{i,s}^{k+t_0}) + dim(\mathcal{O}_{i,s}^{k+t_0})$. Let $\boldsymbol{E}_{i,s} = [\varepsilon_{i,s,1}, \varepsilon_{i,s,2}, \ldots, \varepsilon_{i,s,num_{is}}]^T$ denotes the set of all relaxed variables in the collision avoidance constraints of ne_i^s, then $\boldsymbol{Z}_{i,s} = [\boldsymbol{U}_{i,s}^T, \boldsymbol{E}_{i,s}^T]^T$ can be defined, which denotes the set of all control variables of ne_i^s. According to (10), (13),

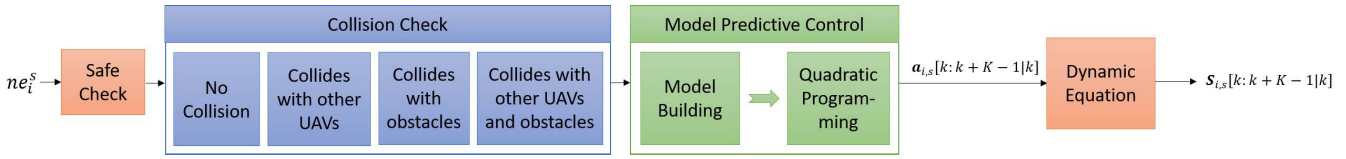

FIGURE 3. The proposed neighbors trajectory prediction process. For a neighbor ne_i^s of UAV n_i, after obtaining the states of other UAVs and obstacles in its safe area, a collision check is carried out to determine whether it will collide and the type of collision in the future. The future control sequence of ne_i^s is obtained by solving an MPC. According to its dynamic equation, the state sequence of ne_i^s at K times in the future is obtained.

(18)-(20), (31), all the constraints of ne_i^s can be written as follows:

$$\boldsymbol{U}_{min} \leq \boldsymbol{U}_{i,s} \leq \boldsymbol{U}_{max}, \tag{32}$$

$$\boldsymbol{P}_{min} - \boldsymbol{A}_0\boldsymbol{S}_{i,s}^0 \leq \boldsymbol{\Phi}\boldsymbol{U}_{i,s} \leq \boldsymbol{P}_{max} - \boldsymbol{A}_0\boldsymbol{S}_{i,s}^0, \tag{33}$$

$$\boldsymbol{V}_{min} - \boldsymbol{A}_0'\boldsymbol{S}_{i,s}^0 \leq \boldsymbol{\Phi}'\boldsymbol{U}_{i,s} \leq \boldsymbol{V}_{max} - \boldsymbol{A}_0'\boldsymbol{S}_{i,s}^0 \tag{34}$$

$$\boldsymbol{\Sigma}_{i,s}\boldsymbol{U}_{i,s} + \boldsymbol{P}_{i,s}\boldsymbol{E}_{i,s} \leq \boldsymbol{\Omega}_{i,s}. \tag{35}$$

(35) represents the combination of num_{is} collision avoidance constraints, $\boldsymbol{\Sigma}_{i,s} \in \mathbb{R}^{num_{is}\times 3K}$, $\boldsymbol{P}_{i,s} \in \mathbb{R}^{num_{is}\times num_{is}}$ and $\boldsymbol{\Omega}_{i,s} \in \mathbb{R}^{num_{is}}$. We can vertically stack inequality constraints (32)-(35):

$$\boldsymbol{\Pi}_{i,s}'\boldsymbol{Z}_{i,s} \leq \boldsymbol{b}_{i,s}'. \tag{36}$$

where $\boldsymbol{\Pi}_{i,s}' \in \mathbb{R}^{(18K+num_{is})\times(3K+num_{is})}$ and $\boldsymbol{b}_{i,s}' \in \mathbb{R}^{(18K+num_{is})\times 1}$.

In order to solve the quadratic programming problem smoothly, instead of restricting the magnitude of each relaxation variable, we add a penalty term in the objective function (15) to punish the larger relaxation variables:

$$\begin{aligned} J_{i,s}' = {} & \boldsymbol{U}_{i,s}^T(\boldsymbol{R}_1 + \boldsymbol{\Gamma}^T\boldsymbol{R}_2\boldsymbol{\Gamma} + \boldsymbol{\Phi}^T\boldsymbol{R}_3\boldsymbol{\Phi})\boldsymbol{U}_{i,s} \\ & + 2(\boldsymbol{S}_{i,s}^{0T}\boldsymbol{A}_0^T\boldsymbol{R}_3\boldsymbol{\Phi} - \boldsymbol{P}_{fis}^T\boldsymbol{R}_3\boldsymbol{\Phi} - \boldsymbol{U}_{i,s}'^T\boldsymbol{R}_2\boldsymbol{\Gamma})\boldsymbol{U}_{i,s} \\ & + \boldsymbol{E}_{i,s}^T(\kappa\boldsymbol{I}_{num_{is}})\boldsymbol{E}_{i,s}, \end{aligned} \tag{37}$$

where $\kappa > 0$ is the penalty term of relaxation variables, $\boldsymbol{I}_{num_{is}} \in \mathbb{R}^{num_{is}\times num_{is}}$ is denoted as the identity matrix.

Finally, the optimization problem of ne_i^s with collision avoidance constraints can be expressed as

$$\begin{aligned} & \min_{\boldsymbol{Z}_{i,s}} J_{i,s}' = \frac{1}{2}\boldsymbol{Z}_{i,s}^T\boldsymbol{H}_{i,s}'\boldsymbol{Z}_{i,s} + \boldsymbol{f}_{i,s}'\boldsymbol{Z}_{i,s} \\ & s.t.\ \boldsymbol{\Pi}_{i,s}'\boldsymbol{Z}_{i,s} \leq \boldsymbol{b}_{i,s}'. \end{aligned} \tag{38}$$

By solving (38), n_i can get the predicted control sequence of ne_i^s at K future time points, and then the future state sequence of ne_i^s can be predicted through the dynamic equation of UAV. The whole process of the proposed neighbors trajectory prediction algorithm is shown in Fig 3. Similar to the above method, n_i can calculate its own collision avoidance constraints. Different from the case of ne_i^s, $p_i[k + t_0]$ can only be obtained by $p_i[(k-1) + (t_0 + 1)|k-1]$ in the predicted state sequence of n_i at the last time. Similarly, the optimization problem of n_i with collision avoidance constraints is as follows:

$$\begin{aligned} & \min_{\boldsymbol{Z}_i} J_i' = \frac{1}{2}\boldsymbol{Z}_i^T\boldsymbol{H}_i'\boldsymbol{Z}_i + \boldsymbol{f}_i'\boldsymbol{Z}_i \\ & s.t.\ \boldsymbol{\Pi}_i'\boldsymbol{Z}_i \leq \boldsymbol{b}_i'. \end{aligned} \tag{39}$$

By solving this problem for UAV n_i, its control quantity at the current instant k and predicted state sequence of K future time points can be obtained.

C. DETAILED DESCRIPTION OF THE ALGORITHM

Algorithm 1 shows the sequential implementation form of multi-UAV trajectory planning based on DMPC. The input is UAVs' initial positions ($\boldsymbol{p}_0$) and target positions ($\boldsymbol{p}_f$), the output is their flight trajectories. In the first line of Algorithm 1, $\boldsymbol{\Lambda}$ is used to store the trajectories of UAVs. UAVs' initial positions are assigned by $\boldsymbol{p}_0$, their initial velocities are set to 0. When UAVs do not all reach their target points and the flight time does not reach the specified time, for an UAV n_i, we need to judge whether it has reached its target point. If it has, stop updating its trajectory and start to consider the next UAV (lines 5-6). If not, it will conduct a safe check to obtain information about other individuals and static obstacles in its sensing area and safe area. Then, by solving an optimization problem modeled by Algorithm 2, n_i gets its own control quantity sequence in the future and the predicted future trajectories of neighbors. The future state sequence of n_i can be predicted. The position point of n_i at the next time is added to the trajectory sequence of n_i in $\boldsymbol{\Lambda}$, and the first quantity in n_i's future state sequence is selected as its state quantity at the next time, the first quantity in n_i's control quantity sequence is selected as the control quantity at the current time (lines 7-11, $t \in \{1, 2, \ldots, K\}$ in line 8, the same below). The distances between UAVs and their target points are calculated. When all UAVs arrive at their target points or the flight time reaches the specified time, the algorithm ends and returns to the trajectory of each UAV.

Algorithm 2 describes the establishment and solution of an optimization problem based on the proposed neighbors trajectory prediction algorithm for a certain UAV (assumed as n_i). Supposing the current time is k, the input includes the state and control sequence of n_i, as well as the information of neighbors and obstacles perceived by n_i at the current instant and the neighbors' information calculated by n_i at the last instant. The output is the control sequence of n_i in the future K instants, the neighbors' control quantities at instant k and state sequences in the future K instants predicted by n_i. If there are no other UAVs and static obstacles in the safe area of n_i, the control sequence of n_i in the future K instants can be obtained by solving a quadratic programming problem without collision avoidance constraints (lines 1-2). If there are other UAVs in n_i's safe area, for each neighbor ne_i^s, n_i

Algorithm 1 DMPC-Based Multi-UAV Trajectories Planning in the Environment Without Communication and With Static Obstacles

Input: Initial and final positions
Output: UAVs' trajectory
1: $[\boldsymbol{\Lambda}(0), \boldsymbol{x}(0)] \leftarrow$InitialAllTrajectory($\boldsymbol{p}_0$)
2: $k \leftarrow 0, \quad AtTarget \leftarrow 0$
3: **while** not $AtTarget$ and $k \leq T_{max}$
4: **for** each UAV $n_i (i = 1, 2, \ldots, N)$
5: **if** TargetDistCheck($\boldsymbol{p}_i(k)$)$< D_{max}$
6: continue
7: $[ne_i_UAV, ne_i_obs, sen_i_UAV, sen_i_obs] \leftarrow$SafeCheck($\boldsymbol{\Lambda}, d_u, d_o, d_s$)
8: $[\boldsymbol{a}_i[k+t-1|k], \boldsymbol{S}_{ne_i}[k+t|k], \boldsymbol{a}_{ne_i}[k|k]] \leftarrow$Algorithm2
9: $\boldsymbol{S}_i[k+t|k] \leftarrow$GetStates($[\boldsymbol{S}_i[k+t-1|k], \boldsymbol{a}_i[k+t-1|k]$)
10: $\boldsymbol{\Lambda}_i[k+1] \leftarrow \boldsymbol{p}_i[k+1|k]$
11: $\boldsymbol{S}_i[k+1], \boldsymbol{a}_i[k] \leftarrow \boldsymbol{S}_i[k+1|k], \boldsymbol{a}_i[k|k]$
12: $AtTarget \leftarrow$TargetDistCalcu($\boldsymbol{\Lambda}[k+1], \boldsymbol{p}_f$)
13: $k \leftarrow k+1$
14: **return** $\boldsymbol{\Lambda}$

Algorithm 2 The Build and Solution of Optimization Model Based on Neighbors Trajectory Prediction

Input: $\boldsymbol{S}_i[k], \boldsymbol{a}_i[k-1], ne_i\_UAV, ne_i\_obs, sen_i\_UAV, sen_i\_obs, \boldsymbol{S}_i[k-1+t|k-1], \boldsymbol{S}_{ne_i}[k-1+t|k-1], \boldsymbol{a}_{ne_i}[k-1]$
Output: $\boldsymbol{a}_i[k+t-1|k], \boldsymbol{S}_{ne_i}[k+t|k], \boldsymbol{a}_{ne_i}[k|k]$
1: **if** $ne_i\_UAV == \emptyset$ and $ne_i\_obs == \emptyset$
2: $\boldsymbol{a}_i[k+t-1|k] \leftarrow SolveQP(\boldsymbol{S}_i[k], \boldsymbol{a}_i[k-1])$
3: **elseif** $ne_i\_UAV \neq \emptyset$
4: **for** each neighbor s $= 1, 2, \ldots, 1$
5: $\boldsymbol{p}_{i,s}(k+t) \leftarrow$IsNeighborBefore($s, \boldsymbol{S}_{ne_i}[k-1+t|k-1]$)
6: $[ne_{i,s}_UAV, ne_{i,s}_obs] \leftarrow$SafeCheck($sen_i\_UAV, sen_i\_obs, \boldsymbol{p}_{i,s}[k]$)
7: $fircolltime1 \leftarrow$CollTCheck($ne_{i,s}\_UAV, ne_{i,s}\_obs, \boldsymbol{p}_{i,s}[k+t]$)
8: $\boldsymbol{a}_{i,s}[k+t-1|k] \leftarrow$SloveQP($\boldsymbol{S}_{i,s}[k], \boldsymbol{a}_{i,s}[k-1], fircolltime1, ne_{i,s}\_obs$)
9: $\boldsymbol{S}_{i,s}[k+t|k] \leftarrow$GetStates($\boldsymbol{S}_{i,s}[k+t-1|k], \boldsymbol{a}_{i,s}[k+t-1|k]$)
10: $\boldsymbol{a}_{i,s}[k] \leftarrow \boldsymbol{a}_{i,s}[k|k]$
11: $fircolltime2 \leftarrow$CollTCheck($\boldsymbol{S}_{i,s}[k+t|k], ne_i\_obs, \boldsymbol{S}_i[k-1+t|k-1]$)
12: $\boldsymbol{a}_i[k+t-1|k] \leftarrow$SolveQP($\boldsymbol{S}_i[k], \boldsymbol{a}_i[k-1], fircolltime2, ne_i\_obs$)
13: **return** $[\boldsymbol{a}_i[k+t-1|k], \boldsymbol{S}_{ne_i}[k+t|k], \boldsymbol{a}_{ne_i}[k|k]]$

first determines whether it is the neighbor at the last time, if so, n_i calls the future state sequence of ne_i^s predicted at the last time, otherwise ne_i^s is regarded as moving at a constant speed. Then, the ne_i^s is checked to get the information of other individuals or obstacles in its safe area. Determine the time of the first collision of ne_i^s, an optimization problem with collision avoidance constraints is solved to obtain its future control sequence (if there is no collision, an optimization problem without collision avoidance constraints is solved), and then its future K instants sate sequence is obtained. The first value of ne_i^s's control quantity sequence is taken as its control quantity at the current time (lines 3-10). After obtaining the states of all neighbors in K future instants, n_i judges the time of the first collision with neighbors or obstacles, and then solves an optimization problem to obtain its own control sequence in the future K times (lines 11-12). The output is returned and the algorithm ends.

IV. SIMULATIONS

In this section, two scenes are provided to verify the performance of the proposed method. One scene is that multi-UAV fly from one side of static obstacles to the other, and the other scene is that multi-UAV fly from opposite sides of static obstacles. In order to illustrate the feasibility and effectiveness of the proposed method, we compare it with two methods in terms of success probability and computation time. One is the DMPC algorithm that treats neighbors as constant velocity individuals (we call it CVN-DMPC), and the other is the DMPC algorithm with communication. In each scene, the experiment is repeated 30 times for different numbers of UAVs. We call the planning failure if the following three situations occur: (1) The distance between an UAV and an other individual or a obstacle is less than the set threshold; (2) the optimization problem becomes infeasible; (3) Some UAVs failed to reach their target within the specified time. Otherwise, it is called planning success. For each number of UAVs, we calculate the proportion of planning success times of each method in 30 experiments as the success probability of this method. Computation time of each method is the average time of its successful experiments. These simulations are implemented as a sequence form described in Algorithm 1.

The hardware environment is a PC with an Intel Core i7-10875H CPU with 8 cores and 16GB RAM, running at 2.3GHz. In these experiments, $r_u = 0.175m$, $\boldsymbol{Q} = diag(1, 1, 2)$, $p_{max} = -p_{min} = 400m$, $v_{max} = -v_{min} = 3.8m/s$, $u_{max} = -u_{min} = 2.5m/s^2$, $d_u = 2r_u = 0.35m$, $d_o = 5d_u = 1.75m$ and $d_s = 10m$. The discrete time step $d = 0.2$, the number of discrete time steps predicted each time is $K = 15$, and the planned flight time is $T_{max} = 100s$. We assume that when $\varepsilon^1_{i,s,*} > 0.07$, UAVs will collide with others; when $\varepsilon^2_{i,s,*} > 0.035$, UAVs will collide with obstacles. We keep the density of the flight space constant, and there are 12 static obstacles with a radius of 2m.

A. SCENE 1: MULTI-UAV FLYING IN THE SAME DIRECTION

In this scene, UAVs randomly take the initial points on one side of the static obstacles, and then fly through the static obstacles to the other side. Their target points are also randomly selected. It is required that UAVs do not communicate with each other during flight and reach their respective target points on the premise of avoiding collision between UAVs and static obstacles. This scene is suitable for applications such as UAVs performances and target tracking.

We test the success probability and computation time of the proposed method, and compare it with CVN-DMPC and DMPC with communication. The results are shown in Fig.4. It can be seen from Fig.4(a) that when the number of UAVs is less than 60, the success probability of the proposed method is above 90%, and it is not very different from the method with communication. With the increase of the number of UAVs, the success probability of both three methods decreased, but the gap between the proposed method

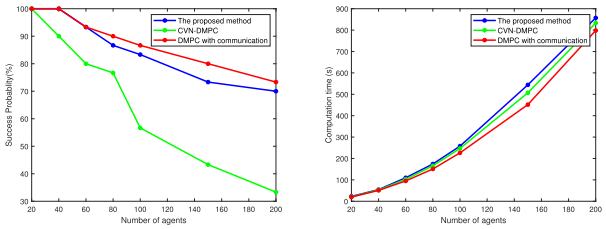

FIGURE 4. The comparison between the proposed method, CVN-DMPC and DMPC with communication under different numbers of UAVs in scene 1 ((a) the success probability, (b) the computation time). For each number of UAVs, the proportion of planning success times of each method in 30 experiments as the success probability of this method. Computation time is the average time of their respective successful experiments.

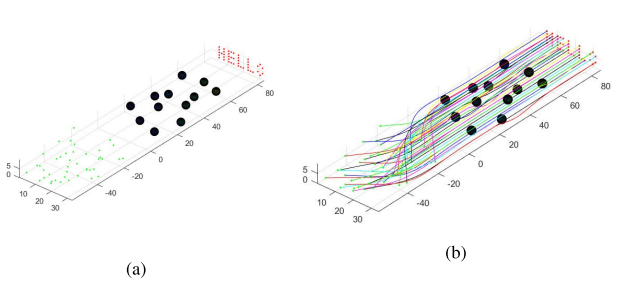

FIGURE 5. 42 UAVs pass through obstacles to form the word "HELLO" ((a) the initial positions and targets of multi-UAV, (b) the multi-UAV trajectories planned by the proposed method).

and DMPC with communication is not very large. However, the success probability of CVN-DMPC decreases rapidly. In the proposed method, UAVs need to use MPC to predict the future trajectories of neighbors, this process increases the computation time, which can be seen from Fig.4(b), especially when the number of UAVs is large. However, comparing with the other two methods, the time consumption of the proposed method increases little.

Fig.5 shows 42 UAVs passing through obstacles from random initial positions to their target points, which form the word ''hello''. Because the space between the letters of ''hello'' is small, it puts forward higher requirements for UAV collision avoidance, but our method can still successfully complete the planning task.

B. SCENE 2: MULTI-UAV FLYING IN THE OPPOSITE DIRECTION

In this scene, UAVs are divided into two groups. Each group randomly takes the initial points on one side of the obstacles, and then takes the initial points of the opposite UAVs as their target points. They also avoid collision and fly to the target points without communication. This scene is suitable for multiple UAVs flying in a cluttered environment, such as warehouses, workshops, etc.

Similar to scene 1, the success probability and computation time of the three methods are shown in Fig 6. The scene of opposite flight poses a great challenge to the trajectory planning of multi-UAV. This is mainly because when UAVs

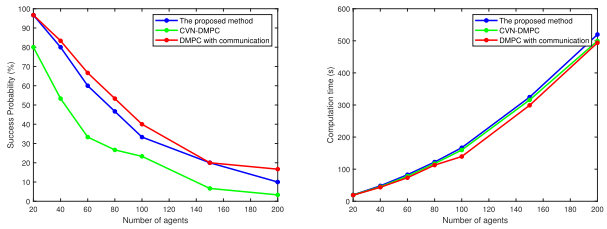

FIGURE 6. The comparison between the proposed method and DMPC method with communication under different numbers of UAVs in scene 2 ((a) the success probability, (b) the computation time). For each number of UAVs, the proportion of planning success times of each method in 30 experiments as the success probability of this method. Computation time is the average time of their respective successful experiments.

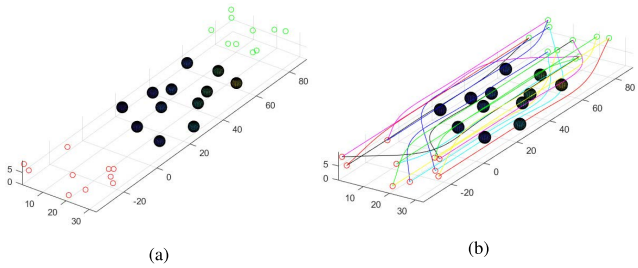

FIGURE 7. 20 UAVs are divided into two groups and fly through obstacles in opposite directions ((a) the initial positions and targets of multi-UAV, (b) the multi-UAV trajectories planned by the proposed method).

fly in the opposite direction in the environment with static obstacles, the time available for UAV response and adjustment becomes shorter. In the absence of communication, UAVs need to predict the future trajectories of their neighbors more accurately. As can be seen from Fig. 6(a), comparing with DMPC method with communication, the success probability of the proposed method is not much different. When the number of UAVs is less than 40, the proposed method can also maintain more than 80% success probability. However, with the increase of the number of UAVs, the success probability of CVN-DMPC quickly dropped below 50%. This shows that our method has better performance than CVN-DMPC in the case of more strict requirements for trajectory prediction accuracy. The computation time of the three methods is shown in Fig. 6(b). Fig. 7 shows 20 UAVs divided into two groups and flying in opposite directions. It can be seen that the UAVs can pass through obstacles and avoid inter-UAV collisions, then reach their respective target points smoothly.

V. CONCLUSION

In this paper, we proposed an MPC-based algorithm for each UAV to predict its neighbors' trajectories when it can't communicate with others. Then the proposed algorithm is integrated into DMPC framework to realize the trajectory planning of multi-UAV in an environment with static obstacles. In the simulation of two scenes, we showed that the proposed method is feasible and effective. Moreover, with only a small increase in computation time, it can achieve a

performance close to DMPC with communication and much better than the DMPC algorithm that treats neighbors as constant velocity individuals. In this paper, the proposed method is implemented in a sequential form. In the future work, we will try to implement it in a distributed form and carry out real machine experiments.

REFERENCES

[1] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search rescue: A potential field based approach," in *Autonomous Robots and Agents*. Berlin, Germany: Springer, 2007, pp. 9–16.

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

[3] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–10, Jul. 2017.

[4] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *Proc. Int. Conf. Intell. Environ.*, Jul. 2015, pp. 140–143.

[5] Y. Yongjie and Z. Yan, "Collision avoidance planning in multi-robot based on improved artificial potential field and rules," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Feb. 2009, pp. 1026–1031.

[6] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[7] J. Van Den Berg, "Reciprocal N-body collision avoidance," in *Robotics Research*. Berlin, Germany: Springer, 2011, pp. 3–19.

[8] S. H. Semnani, A. H. J. de Ruiter, and H. H. T. Liu, "Force-based algorithm for motion planning of large agent," *IEEE Trans. Cybern.*, vol. 52, no. 1, pp. 654–665, Jan. 2020.

[9] J. Tang, G. Liu, and Q. Pan, "A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 10, pp. 1627–1643, Oct. 2021.

[10] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2018, pp. 3052–3059.

[11] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2017, pp. 285–292.

[12] S. H. Semnani, H. Liu, M. Everett, A. de Ruiter, and J. P. How, "Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3221–3226, Apr. 2020.

[13] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Syst.*, vol. 22, no. 1, pp. 44–52, Feb. 2002.

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

[15] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 51, no. 5, pp. 4282–4286, 1995.

[16] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 5, pp. 1782–1797, Sep. 2018.

[17] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán, "Exploiting map information for driver intention estimation at road intersections," in *Proc. Intell. Veh. Symp. (IV)*, 2011, pp. 583–588.

[18] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 797–803.

[19] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 261–268.

[20] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 5921–5928.

[21] H. Zhu, F. M. Claramunt, B. Brito, and J. Alonso-Mora, "Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2256–2263, Apr. 2021.

[22] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2255–2264.

[23] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme, "Downwash-aware trajectory planning for large quadrotor teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Oct. 2017, pp. 250–257.

[24] J. Tang, S. Lao, and Y. Wan, "Systematic review of collision-avoidance approaches for unmanned aerial vehicles," *IEEE Syst. J.*, early access, Sep. 1, 2021, doi: 10.1109/JSYST.2021.3101283.

[25] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[26] Q. Wang, Z. Duan, Y. Lv, Q. Wang, and G. Chen, "Distributed model predictive control for linear–quadratic performance and consensus state optimization of multiagent systems," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 2905–2915, Jun. 2021.

[27] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for MAVs in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 776–783, Apr. 2019.

ZIJIA NIU received the B.S. degree in information and computing sciences and the M.S. degree in operational research and cybernetics from Yanshan University, Qinhuangdao, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree in applied mathematics with Beihang University, Beijing, China. His research interests include optimal and distributed control, multi-agent trajectory planning, and swarm intelligence.

XIAOHU JIA received the B.S. degree in applied statistics from the Nanjing University of Science and Technology, in 2020. He is currently pursuing the M.S. degree in applied mathematics with Beihang University, Beijing, China. His current research interests include optimal control, reinforcement learning, and game theory.

WANG YAO received the B.S. and Ph.D. degrees in mathematics from Beihang University, Beijing, China, in 2012 and 2017, respectively. He is currently a Lecturer with Beihang University. His research interests include multi-agent systems, optimization, and game theory.