Multivehicle Motion Planning With Posture Constraints in Real World

Gang Xu, Yansong Chen, Junjie Cao, Deye Zhu, Weiwei Liu, and Yong Liu

***Abstract*—This article addresses the posture constraints problem in multivehicle motion planning for specific applications such as ground exploration tasks. Unlike most of the related work in motion planning, this article investigates more practical applications in the real world for nonholonomic unmanned ground vehicles (UGVs). In this case, a strategy of diversion is designed to optimize the smoothness of motion. Considering the problem of the posture constraints, a postured collision avoidance algorithm is proposed for the motion planning of the multiple nonholonomic UGVs. Two simulation experiments were conducted to verify the effectiveness and analyze the quantitative performance of the proposed method. Then, the practicability of the proposed algorithm was verified with an experiment in a natural environment.**

***Index Terms*—Collision avoidance, constrained motion planning for multivehicle.**

I. Introduction

IN RECENT years, the multivehicle system has witnessed prominent progress, which is mainly due to its practical potential in various applications [1], [2], ranging from military operations to formation transportation, collaborative search, rescue, etc. However, multivehicle motion planning, navigating the vehicles to specified targets without collisions with the obstacles and the other vehicles, remains a top priority for the multiple unmanned vehicle systems. As a fundamental challenge, motion planning considering physical constraints on multiple unmanned vehicles has attracted considerable attention from researchers in robotics and control communities. To take a concrete example, multiple unmanned ground vehicles (UGVs) motion planning has been in the spotlight as it is indispensable for ground transportation, exploration, patrol, etc.

Manuscript received January 20, 2022; revised March 22, 2022; accepted April 29, 2022. This work was supported by the basic Science Center of the National Natural Science Foundation of China under Grant 62088101. Recommended by Technical Editor Hugh Liu and Senior Editor Xiang Chen. *(Corresponding authors: Yong Liu; Junjie Cao.)*

Gang Xu is with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China, and also with the Polytechnic Institute, Zhejiang University, Hangzhou 310015, China (e-mail: wuuya@zju.edu.cn).

Yansong Chen, Junjie Cao, Deye Zhu, Weiwei Liu, and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: 22132009@zju.edu.cn; cjunjie@zju.edu.cn; 1328301164@qq.com; 11932061@zju.edu.cn; yongliu@iipc.zju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMECH.2022.3173130.

Digital Object Identifier 10.1109/TMECH.2022.3173130

Conventionally, multivehicle motion planning is mainly divided into centralized and decentralized methods. The former is mostly global and offline, while the latter is generally local and online. Most centralized planning methods are extended from the A* [3] approach, the rapidly exploring random trees* (RRT*) method [4], and the probabilistic roadmaps algorithm [5]. The first algorithm is a unitary heuristic search for maps, while the last two are sampling based. For example, both the conflict-based search [6] and the safe interval path planning (SIPP) [7] are developed from the A* method. Recently, Okumura *et al.* [8] solved the deadlock problem of the SIPP algorithm caused by fixed priorities. There are also many improvements in sampling-based methods, such as coordinated sampling-based expansion proposed by Le *et al.* [9], which improves the computational speed of centralized motion planning algorithms. However, centralized methods are logically simple but require hardware resources of high quality and holonomic information of all kinds with high computational cost and limitation of the number of agents, resulting in that it cannot be applied to real scenarios well.

Different from centralized motion planning methods, distributed ones can be practical in complex dynamic environments. Many of them do not require holonomic information and can address noise in sensors well. The methods mainly include artificial potential fields [10], deep reinforcement learning methods [11], sampling-based methods [12], geometric collision avoidance such as velocity obstacles (VO) [13], etc. Artificial potential fields are robust but difficult to integrate constraints on vehicles and environments. Deep reinforcement learning based methods have drawn a lot of attention in recent years, but most work may be only applied to scenarios where training data is provided [14]–[16]. Sampling-based motion planning methods are generally used by being combined with other distributed methods. The VO algorithms have been in the spotlight recently for their excellent performance. Among them, the VO series of algorithms are most related to the proposed algorithm in this article.

Geometrically, the VO algorithm can obtain a set of obstacle velocities. If not included in the set, the velocity chosen by the algorithm at the next moment will be regarded as a safety solution, among which the one closest to the ideal velocity is the optimal solution. However, the distribution of responsibility of avoiding collision leads to the oscillation of vehicles' trajectories, which cannot be resolved by the early improved versions, such as [17]. Reciprocal velocity obstacle (RVO) [18] was proposed to deal with the problem by assigning the responsibility equally for any

two vehicles with potential collision, thus, achieving smooth trajectories. Afterward, Snape *et al.* [19] proposed the hybrid reciprocal velocity obstacle to solve the problem that RVO cannot achieve collision avoidance on the same side in some particular scenarios. To implement dynamic constraints on vehicles, Van Den Berg *et al.* [20] proposed acceleration-velocity obstacles.

In the VO series of algorithms, the optimal reciprocal collision avoidance (ORCA) [21] is the most popular one except for RVO. The RVO algorithm takes collision avoidance between any two vehicles into account by selecting the intersection of two safety velocity sets. However, the ORCA considers two or more vehicles and is superior in computational speed, collision avoidance, and smoothness of trajectories, leading to its application as expert data in machine learning algorithms. Moreover, Guo *et al.* [22] improved the smoothness of trajectories and collision avoidance of ORCA by adjusting the distribution of responsibility of avoiding the collision. Gopalakrishnan *et al.* [23] put forward a way of distributing the responsibility according to certain probability distribution and considered the uncertainty related to state evaluation and robot driving as well. Arul *et al.* [24] integrated RVO and Voronoi diagrams, and then, achieved better performance than ORCA.

The VO series of algorithms can add constraints flexibly, such as kinematic and dynamic constraints. For example, Mao *et al.* [25] integrated linear model predictive control and ORCA with constraints from wheeled robots. Fuad *et al.* [26] added several common constraints into the ORCA algorithm, providing a reference idea for the motion planning problem with constraints. Yuan *et al.* [27] utilized the VO algorithm to control ships with dynamic and uncertain constraints considered to reduce collision. Liu *et al.* [28] proposed the Dubins-RVO algorithm to fit limited space and constraints from fixed wings in 2-D space by combining the Dubins curve and the RVO algorithm.

In addition, aimed at the problem of physical constraints, Desaraju *et al.* [29] proposed the decentralized multiagent rapidly exploring random tree with constraints on moving directions. Wang *et al.* [30] proposed an optimization framework specific to multicopters with geometrical spatial and user-defined constraints.

To date, most of the existing related work focuses on simulation environments. At the same time, most of UGVs are subject to limited nonholonomic cases, i.e., they cannot move in any direction in the 2-D workspace, especially in loaded conditions. For this above challenge, a solution with curvature constraint is more qualified for multiple UGVs motion planning due to its strengths in nonholonomic. On the other hand, it is also of great significance that extensive benchmarks are conducted for validating the effectiveness of proposed methods in the real world. Motivated by the problems discussed above, the problem of multiUGVs motion planning with posture constraints is studied in this article.

The contributions can be summarized as follows:

1) The strategy of diversion is proposed in symmetric scenarios to choose the optimal velocity on the right of the current velocity direction for collision avoidance when there are two optimal solutions.
2) The postured collision avoidance (PCA) algorithm is presented based on the RVO algorithm expanded with the Dubins method and the strategy of diversion.
3) Specific exploration task is considered by imposing simulation experiment and real-world experiment to verify the proposed algorithm.

The rest of the article is organized as follows. In Section II, we formulate the problem investigated in this article. In Section III, we introduce our proposed multivehicle motion planning algorithm with posture constraints. In Section IV, simulation and real-world experiments are conducted to show the validity of our method in simulated and natural environments. Finally, Section V concludes the article.

II. Problem Formulation

In this section, we summarize the formulation of the motion planning problem in multivehicle system, the kinematic model of UGV, and the RVO algorithm. Note that in the rest of this article, scalars and vectors are denoted by ordinary and boldface letters, respectively.

A. UGV Kinematic Model

The UGV is modeled as a nonholonomic system in the shape of a circle with kinematic constraints needed to be considered. Specifically, the Ackermann-steering model is used to describe a UGV. Considering the origin of the rigid body the frame is placed at the center of its rear axle, the vector position and velocity of the UGV is denoted as $\mathbf{p} = [x, y]^T$ and $\mathbf{v} = \dot{\mathbf{p}} = [\dot{x}, \dot{y}]^T$, where x and y are its coordinates in a global frame. The angle between the x-axis of the body frame and the global frame is denoted as θ, which is also the angle of $\mathbf{v}$ in the global Cartesian coordinate system of *XY* plane. Note that the y-axis of the body frame points to the left side of the UGV. Additionally, the vector velocity can also be expressed as $\mathbf{v} = [v\cos\theta, v\sin\theta]^T$, where v represents the UGV's scalar speed. The steering angle of the front wheels is denoted as φ. Then, the radius of trajectory along which the UGV moves is expressed as $R = L/\tan\varphi$ with the steering angle fixed at φ. The distance covered along the trajectory within time dt is given by $dw = R \cdot d\theta$. Thus, the mathematical relation between φ and $\dot{\theta}$ can be written as

$$\dot{\theta} = \frac{v}{L}\tan\varphi. \tag{1}$$

Then, we discretize (1) above with sample time T_s and obtain the expression of the state of UGV at time index t as

$$\mathbf{p}^t = \mathbf{p}^{t-1} + T_s\mathbf{v}^{t-1}, \tag{2}$$

$$\theta^t = \theta^{t-1} + T_s\frac{v^{t-1}}{L}\tan\varphi. \tag{3}$$

The bound of UGV's velocity is given by $v \in [v_{b\max}, v_{f\max}]$, where negative $v_{b\max}$ and positive $v_{f\max}$ represents the maximum backward and forward speed, respectively. The steering angle satisfies the inequality $\varphi \le \varphi_{\max}$, implying the turning radius of an Ackermann-steering UGV has a minimum value.

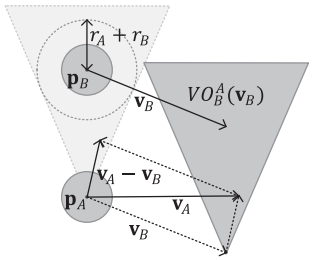


Fig. 1. VO geometric representation of obstacle velocity set $VO_B^A(\mathbf{v}_B)$ of UGV A for UGV B.

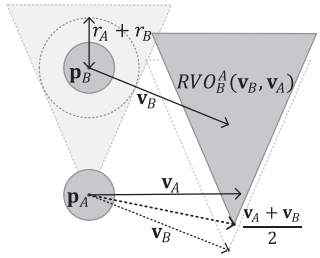


Fig. 2. RVO geometric representation of obstacle velocity set $RVO_B^A(\mathbf{v}_B, \mathbf{v}_A)$ of UGV A for UGV B.

B. Summary of the RVO Algorithm

Let UGV A and B be any two UGVs of $1, 2, \cdots, K$ in the workspace. We denote the position of UGV A as $\mathbf{p}_A$, and its radius as r_A, moving at $\mathbf{v}_A$ in the 2-D plane toward its goal position $\mathbf{p}_A^{\text{goal}}$ and pose θ_A^{goal}. Similarly, UGV B's reference position is $\mathbf{p}_B$ and radius is r_B, moving at $\mathbf{v}_B$ toward its goal position $\mathbf{p}_B^{\text{goal}}$ and pose θ_B^{goal}.

We assume that two UGVs reach the position relationship shown in Fig. 1 at t moment. In this condition, we first consider the obstacle avoidance of the UGV A, in which the UGV B is viewed as a dynamic obstacle of A. In the VO algorithm, the cone-shaped boundary was defined by the Minkowski sum of A as well as B and the position $\mathbf{p}_A$, in which the Minkowski sum is expressed as $D(\mathbf{p}_B, r_A + r_B)$ shown as the light-gray dotted circle whose center is $\mathbf{p}_B$ and radius is $r_A + r_B$ in Fig. 1. Then, the set $VO_B^A(\mathbf{v}_B)$ of A against B at $t+1$ time is expressed geometrically as cone-shaped zone translated at $\mathbf{p}_A + \mathbf{v}_B$ shown in Fig. 1 with dark-gray zone. Geometrically, the VO algorithm holds that A and B will certainly collide at some moment in the future if the rays of $\mathbf{v}_A - \mathbf{v}_B$ intersect with the light-gray zone in Fig. 1. Otherwise, the two UGVs will be collision free.

The mathematical formulation of the VO algorithm is as follows. We denote $D(\mathbf{p}, r)$ in (4) as a circle-zone with center $\mathbf{p}$, radius r, and any point $\mathbf{q}$ in the workspace. In addition, the ray $\lambda(\mathbf{p}, \mathbf{v})$ is denoted by (5) where $\mathbf{p}$ is the starting point, the direction is $\mathbf{v}$, and t is the parameter of extension. Finally, the obstacles velocity set $VO_B^A(\mathbf{v}_B)$ is expressed in (6), where $\mathbf{v}_{AB}$ is $\mathbf{v}_A - \mathbf{v}_B$ and r_{AB} is $r_A + r_B$

$$D(\mathbf{p}, r) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{p}\| < r\}, \tag{4}$$

$$\lambda(\mathbf{p}, \mathbf{v}) = \{\mathbf{p} + c\mathbf{v} \mid c \geq 0\}, \tag{5}$$

$$VO_B^A(\mathbf{v}_B) = \{\mathbf{v}_A \mid \lambda(\mathbf{p}_A, \mathbf{v}_{AB}) \cap D(\mathbf{p}_B, r_{AB}) \neq \emptyset\}. \tag{6}$$

Similarly, the concept of RVO is illustrated geometrically in Fig. 2. The basic idea can be summarized as follows. When UGV A calculates the collision-avoidance solutions $\mathbf{v}'_A$ at the $t+1$ moment, it believes that UGV B will also take a certain effort for collision-avoidance. Hence, A no longer holds that the velocity of B at the next moment must be $\mathbf{v}_B$ (VO is), but another one considered the motion of A, such as $\frac{\mathbf{v}_A + \mathbf{v}_B}{2}$. Geometrically, the cone-shaped with the inscribed circle (the light-gray zone in Fig. 2) is translated at $\frac{\mathbf{v}_A + \mathbf{v}_B}{2}$ for expressing the reciprocal obstacle velocity set of A expressed as $RVO_B^A(\mathbf{v}_B, \mathbf{v}_A)$ shown in Fig. 2 with the dark-gray zone. Sequentially, when A calculates the velocity of collision-free, its relative one also changes to $\mathbf{v}'_A - \frac{\mathbf{v}_A + \mathbf{v}_B}{2}$. Finally, the formulation of RVO is shown as

$$RVO_B^A(\mathbf{v}_B, \mathbf{v}_A) = \{\mathbf{v}'_A \mid (2\mathbf{v}'_A - \mathbf{v}_A) \in VO_B^A(\mathbf{v}_B)\}. \tag{7}$$

The RVO concept follows the lemma of the same side that can guarantee that both UGVs automatically choose the same side to pass each other. This lemma was expressed as Formula (8), where $\mathbf{v}_A + \mathbf{u}$ and $\mathbf{v}_B - \mathbf{u}$ are the optimal velocity for UGV A and UGV B, respectively

$$\mathbf{v}_A + \mathbf{u} \notin RVO_B^A(\mathbf{v}_B, \mathbf{v}_A) \Leftrightarrow \mathbf{v}_B - \mathbf{u} \notin RVO_A^B(\mathbf{v}_A, \mathbf{v}_B). \tag{8}$$

Although, the RVO algorithm is more feasible to integrate the physical constraints of the UGV than other states of the art, the lemma of the same side will not be feasible when there are more than two UGVs in symmetric scenarios. In this case, the symmetric situations may generate two optimal solutions, resulting in the stuck or oscillation. On the other hand, the RVO method only takes into account the simple kinematic constraints, but posture constraints of the UGV. Therefore, the above two problems are investigated in this article.

III. Proposed Method

In this section, the method of motion planning with posture constraints is presented for the multi-UGVs applications. The proposed method is named as PCA algorithm, in which it overcomes the problems of two optimal solutions and especially the posture constraints in the 2-D workspace.

A. Strategy of Diversion

We first denote a universal set of velocity U.. Meanwhile, we can achieve two velocity sets through the RVO algorithm, named RVO representing the obstacles velocity set and AV expressed as the collision-free velocity set, respectively. As long as the UGV takes action in the AV set, there will be no collision. In the view of UGV A, its formulation can be defined by (9). Then, one of the optimal solutions is calculated through (10) where the $\mathbf{v}_A^{\text{pref}}$ is expressed as the desired velocity of A. Finally, the optimal solutions set, namely AV_{OPT}^A, are generated by (11). Actually, a threshold parameter Δv_{opt}, empirically 10^{-5}, needs

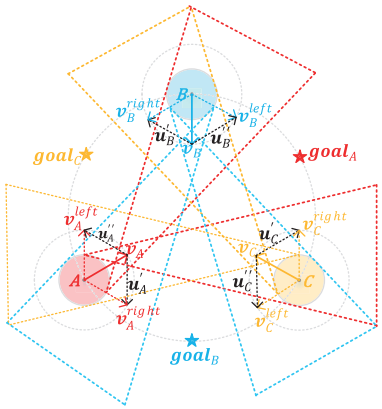


Fig. 3. Symmetric situation of three UGVs, where every UGV has two optimal velocities.

to be set slightly greater than zero

$$AV_B^A(\mathbf{v}_B, \mathbf{v}_A) = U_B^A(\mathbf{v}_B, \mathbf{v}_A) - RVO_B^A(\mathbf{v}_B, \mathbf{v}_A), \tag{9}$$

$$\mathbf{v}_A^{opt} = \arg\min_{\mathbf{v}_A \in AV_B^A} (\|\mathbf{v}_A - \mathbf{v}_A^{\text{pref}}\|), \tag{10}$$

$$AV_{\text{OPT}}^A = \{\mathbf{v}_A \mid \|\mathbf{v}_A - \mathbf{v}_A^{opt}\| < \Delta v_{\text{opt}}, \mathbf{v}_A \in AV_B^A\}. \tag{11}$$

Then, we implement the diversion strategy to pick the optimal solution from the set AV_{OPT}^A and compute their angle θ. Theoretically, the angle of safety velocity $\mathbf{v}_A$ must be larger than or equal to the θ_{right} of the optimal velocity on the right of $\mathbf{v}_A$ and must be smaller than or equal to the θ_{left} of another optimal one on the left of $\mathbf{v}_A$. In a word, the inequations $\theta_{\text{right}} \leq \theta(\mathbf{v}_A) \leq \theta_{\text{left}}$ and $\theta_{\text{right}} \leq \theta_{\text{left}}$ are always true. Finally, the formulation of the strategy of diversion is shown as

$$\mathbf{v}'_A = \arg\min_{\mathbf{v}_A^{opt} \in AV_{\text{OPT}}^A} (\theta(\mathbf{v}_A^{\text{opt}})). \tag{12}$$

The validity of the strategy of diversion can be proven in the following example. Imagine a symmetric situation that might occur in the real field and as shown in Fig. 3, three UGVs are traveling to their goal position from the symmetric start position where the current velocities of three UGVs are $\mathbf{v}_A, \mathbf{v}_B$, and $\mathbf{v}_C$, respectively. In this case, every UGV has two optimal velocities $\mathbf{v}_i^{\text{right}}(\mathbf{v}_i^{\text{right}} = \mathbf{v}_i + \mathbf{u}'_i)$ and $\mathbf{v}_i^{\text{left}}(\mathbf{v}_i^{\text{left}} = \mathbf{v}_i + \mathbf{u}''_i), i \in \{A, B, C\}$ in geometrically. With our strategy of diversion, the more collaborative optimal velocities achieved by (12) are $\mathbf{v}_A^{\text{right}}$, $\mathbf{v}_B^{\text{right}}$, and $\mathbf{v}_C^{\text{right}}$ for three UGVs, respectively.

For UGV A, $\mathbf{v}_A^{\text{right}}$ is the optimal velocity under the strategy of diversion. In this condition, the Formula (13) must be true. This illustrates that the optimal velocity $\mathbf{v}_C^{\text{left}}$ is not better than $\mathbf{v}_C^{\text{right}}$ for UGV C. In contrast, the Formula (14) might be true. To sum up both Formulas (13) and (14), the optimal velocity $\mathbf{v}_C^{\text{right}}$ is better than $\mathbf{v}_C^{\text{left}}$ for UGV C. Similarly, for UGV B and C, we can prove the optimal velocity $\mathbf{v}_A^{\text{right}}$ and $\mathbf{v}_B^{\text{right}}$ are better than $\mathbf{v}_A^{\text{left}}$ and $\mathbf{v}_B^{\text{left}}$, respectively. In a word, the strategy of diversion can guarantee all the UGVs passing on the right side, improving the capacity of cooperation among UGVs. The same holds for other symmetric situations

$$\mathbf{v}_A^{\text{right}} \in RVO_C^A\left(\mathbf{v}_C^{\text{left}}, \mathbf{v}_A^{\text{right}}\right) \wedge \mathbf{v}_A^{\text{left}} \in RVO_B^A\left(\mathbf{v}_B^{\text{right}}, \mathbf{v}_A^{\text{left}}\right), \tag{13}$$

$$\mathbf{v}_A^{\text{right}} \notin RVO_C^A\left(\mathbf{v}_C^{\text{right}}, \mathbf{v}_A^{\text{right}}\right) \wedge \mathbf{v}_A^{\text{right}} \notin RVO_B^A\left(\mathbf{v}_B^{\text{right}}, \mathbf{v}_A^{\text{right}}\right). \tag{14}$$

Algorithm 1: The Strategy of Diversion.

Input: $\mathbf{p}_A, \mathbf{v}_A, \mathbf{p}_B, \mathbf{v}_B, \mathbf{v}_A^{pref}$ – UGV A config.
Output: The UGV A's optimal velocity $\mathbf{v}'_A$

$RVO_B^A \leftarrow$ComputeRVO$(\mathbf{p}_A, \mathbf{v}_A, \mathbf{p}_B, \mathbf{v}_B)$;
$AV_B^A \leftarrow U_B^A - RVO_B^A$;
$\mathbf{v}_A^{opt} \leftarrow \arg\min_{\mathbf{v}_A \in AV_B^A}(\|\mathbf{v}_A - \mathbf{v}_A^{pref}\|)$;
for $\mathbf{v}_A$ **in** AV_B^A **do**
    $AV_{OPT}^A \leftarrow \{\mathbf{v}_A \mid \|\mathbf{v}_A - \mathbf{v}_A^{opt}\| < \Delta v_{opt}\}$;
end
$\mathbf{v}'_A \leftarrow \arg\min_{\mathbf{v}_A^{opt} \in AV_{OPT}^A}(\theta(\mathbf{v}_A^{opt}))$;
return $\mathbf{v}'_A$;

In summary, the implementation of the strategy of diversion has mainly the following three steps, and its details are shown in Algorithm III-A.

1) All the optimal $\mathbf{v}_A^{opt}$ of the UGV are solved by applying the RVO algorithm and put into the optimal velocity set AV_{OPT}^A.
2) The angles of both these optimal velocities and the $\mathbf{v}_A$ are computed.
3) The optimal solution $\mathbf{v}'_A$ will be achieved by (12).

B. Dubins Path With Collision Avoidance

In fact, most of UGVs are subject to limited nonholonomic, in which an allowed minimum turning radius is needed to consider. We view this challenge as the posture constraints problem. The Dubins method is mainly explored to solve this problem, where the shortest distance between the starting position and the end position will be the length of the Dubins curve.

We denote the length of the Dubins curve as $L(D) = \int_0^D d_t$, where d_t indicates the sampling distance of the curve. Then, we define the total length of the trajectory traveled by all UGVs as $\sum_{i=1}^{i=K} L_i$, where i represents the id of the vehicle and K represents the number of vehicles in the environment. Hence, the multivehicle shortest path problem using the Dubins method can be expressed as (15), where L_i and $L(D_i)$ are the actual and ideal length of path for the UGV i, respectively. Meanwhile, we set the preferred speed v_i^{pref} for all the UGVs. Then, the preferred velocity is achieved by $\mathbf{v}_i^{\text{pref}} = v_i^{\text{pref}} \cdot \mathbf{d}_D(t)$, where $\mathbf{d}_D(t)$ is $D(t+1) - D(t)$. Note that $D(t)$ indicates the node coordinates of the Dubins curve at the t moment. In this way, we get the optimal solution considered posture constraints of the UGV i. It can be expressed as (16)

$$P\left(\sum_{i=1}^{i=K} L_i\right) = \arg\min\left(\|\sum_{i=1}^{i=K} L_i - \sum_{i=1}^{i=K} L(D_i)\|\right), \tag{15}$$

Algorithm 2: The Dubins Path with Collision-Free.

Input: D, t, $\mathbf{v}_i$, $\mathbf{pose}_i^{current}$, $\mathbf{pose}_i^{goal}$, v_i^{pref}, ρ_{min} − UGV i config.
Output: The UGV i's preferred velocity $\mathbf{v}_i^{pref}$

$\mathbf{d}_D(t-1) \leftarrow D(t) - D(t-1)$;
$d_\theta \leftarrow |\theta(\mathbf{v}_i) - \theta(\mathbf{d}_D(t-1))|$;
$dist \leftarrow$ComputeDistance$(\mathbf{pose}_i^{current}, \mathbf{pose}_i^{goal})$;
if $d_\theta < \varepsilon_\theta$ **or** $dist < \kappa$ **then**
    $\mathbf{d}_D(t) \leftarrow D(t+1) - D(t)$;
    $\mathbf{v}_i^{pref} \leftarrow v_i^{pref} \cdot \mathbf{d}_D(t)$;
    return $\mathbf{v}_i^{pref}$;
end
else
    $D \leftarrow$ComputeDubins$(\mathbf{pose}_i^{current}, \mathbf{pose}_i^{goal})$;
    $t \leftarrow$update(t);
    $\mathbf{d}_D(t) \leftarrow D(t+1) - D(t)$;
    $\mathbf{v}_i^{pref} \leftarrow v_i^{pref} \cdot \mathbf{d}_D(t)$;
    return $\mathbf{v}_i^{pref}$;
end

$$\mathbf{v}_i^{\text{opt}} = \arg \min_{\mathbf{v}_i \in AV^i} \left(\|\mathbf{v}_i - \mathbf{v}_i^{\text{pref}}\| \right). \tag{16}$$

The Dubins curve with collision-avoidance can be summarized into the following three steps, and the algorithm is given in Algorithm III-B.

1) The path node at time t is set by $D(t)$, and $\mathbf{d}_D(t-1)$ is denoted as the direction from the node $D(t-1)$ to the node $D(t)$. Then, the absolute difference d_θ is got by computing the angle of the current velocity $\mathbf{v}_i$ and the angle of $\mathbf{d}_D(t-1)$.
2) Parameter ε_θ, generally 10^{-5}, is set as the parallel condition between vector $\mathbf{v}_i$ and vector $\mathbf{d}_D(t-1)$. Then, the UGV i continues to track the current Dubins trajectory if d_θ is less than or equal to ε_θ; otherwise, the Dubins curve is replanned based on the current posture and goal posture.
3) The minimum turning radius of the UGV i is denoted as $\rho_{\min}$, and the distance parameter κ is defined with $\kappa = 3 \cdot \rho_{\min}$. The Dubins curve will not be updated according to the physical size of the UGV i when the distance between the current position and end position is less than κ.

C. PCA Algorithm

From the above, the PCA method is proposed by integrating the strategy of diversion with the Dubins method. The process of the PCA algorithm is as follows. Firstly, the preferred velocity of a UGV A can be achieved by the Algorithm III-B, in which the Dubins method considers the posture constraints. Then, the actual optimal velocity with posture constraints for UGV A can be generated by introducing this preferred velocity into the Algorithm III-A such that the strategy of diversion can obtain the input of $\mathbf{v}_A^{\text{pref}}$ in Algorithm III-A. Note that the proposed algorithm is not simply the combination of the RVO algorithm and the Dubins method, which makes it different from the work of Liu *et al.* [28] in the optimization and the fields of application.

IV. Experimental Results

In this section, we present the experiments in simulated and real environments to validate the proposed method. Notably, all code was finished by using Python in this article. The parameters for the motion planning algorithm are set as follows. The preferred speed is set as $v^{\text{pref}} = 0.22$ m/s, maximum speed is set as $v^{\max} = 1.0$ m/s, neighborhood range is set as $d^{\max} = 5$ m, the maximum rotational speed is set as $\omega^{\max} = 1.1$ rad/s, and the maximum number of neighbor is set as $N^{\max} = 15$. In addition, the minimum turning radius of vehicles is set as $\rho_{\min} = 0.5$ m. For simplicity, we describe all the UGVs' physical space with a predefined circle with a radius of 0.2 m according to the physical size of vehicles. All the experiments were performed on an ASUS laptop equipped with Intel(R) Core(TM) i7-8700 CPU at 3.20 GHz with 16 GB memory.

In the simulations, we first show the special performance of the proposed method by executing an exploration task. Then, we compared the PCA algorithm with the RVO algorithm under the same conditions. Especially, we did not compare the proposed algorithm with the ORCA algorithm as it indeed does not satisfy the posture constraints.

In the first simulation, we verify the unique performance of the proposed method by executing an exploration task. 100 UGVs are deployed for a ground exploration task, where all UGVs have required a specific heading angle for smoothly backing to the initial area when reaching their goals. The configurations of the exploration task are as follows. The benchmark scenario is set as a circle whose radius is 40 m, in which a square obstacle with 16×16 m^2 is placed on the center of this circle as a forbidden area. The positions of both the initial and the target are set as antipodal positions, and the differences between the initial heading angle and the target heading angle are 180°. The regions of exploration are assumed as a square with 1×1 m^2. The maximum exploration range is assumed as 0.5 m.

The simulation results are presented in Fig. 4. In (a), (b), and (c) of Fig. 4, the process of traveling of all vehicles is presented by selecting representative states at several specific time instants, respectively. In particular, the start and exploration regions are highlighted for every vehicle. Fig. 4(d), (e), and (f) depicts the traveled trajectories under the proposed PCA algorithm for all the vehicles from the same specific time instants with the above, respectively. By inspection of these figures, it can be found that when all vehicles traveled close to the corresponding exploration region, every vehicle moved along a circular arc in the allowed minimum turning radius to the appointed heading angle for the preparation of returning to the start region. Moreover, Table I separately shows four UGVs' average computation time in every step under our PCA algorithm. By inspection of Table I, it is possible that the proposed method computes new action for the next-step execution before the vehicles update their parameters. It is also observed that these trajectories with posture constraints are feasible for the vehicles to follow in real environments.

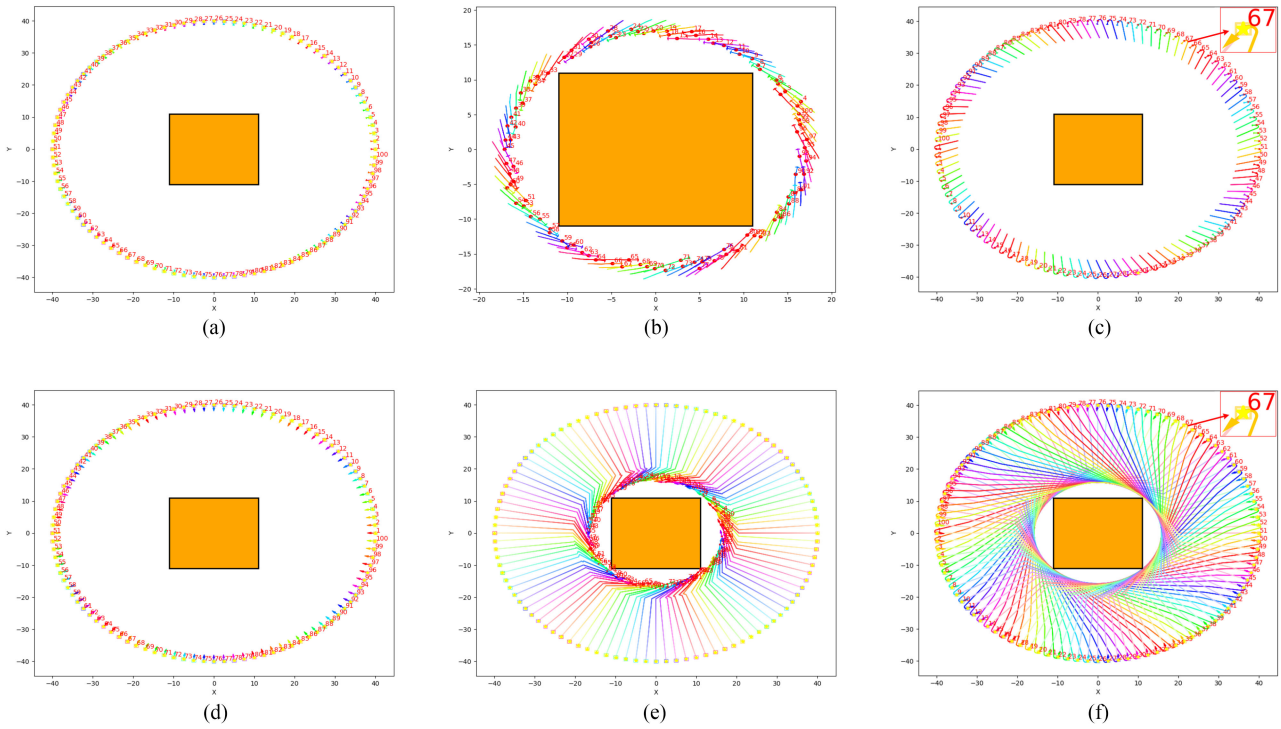


Fig. 4. 100 UGVs are deployed for a ground exploration task. The red points are the UGVs' current poses. The yellow points are the UGVs' goals. The squares are the exploration regions. The arrows indicate the UGVs' current heading. Three representative states of UGVs moving are presented in (a), (b), and (c). The trajectories of UGVs under the same state are shown in (d), (e), and (f).

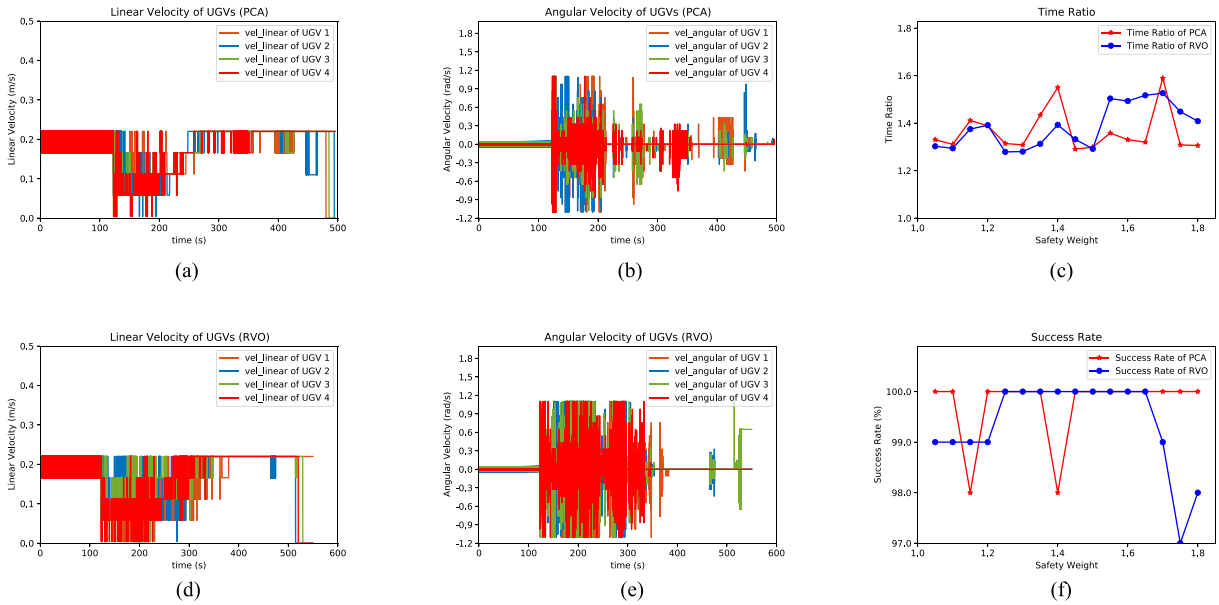


Fig. 5. Evaluation results, which the safety weight is 1.55 in (a), (b), (d), and (e). (a) The linear velocity of four UGVs under the PCA algorithm. (b) The angular velocity of four UGVs under the PCA algorithm. (c) The time ratio in the PCA and RVO algorithms. (d) The linear velocity of all UGVs under the RVO algorithm. (e) The angular velocity of all UGVs under the RVO algorithm. (f) The success rate in the PCA and RVO algorithms.

The same benchmark scenario is set as the first simulation in the second simulation. The positions of both the initial and the target are set as antipodal positions, and the differences between the initial heading angle and the target heading angle are 0°. 100 UGVs are deployed for comparing the PCA algorithm with the RVO algorithm under the same conditions, in which time ratio and success rate are considered as metrics to evaluate algorithm performance. The definitions of the two metrics are as follows. The time ratio is expressed as the ratio of the time of all UGVs reaching their targets to the time of all UGVs reaching their

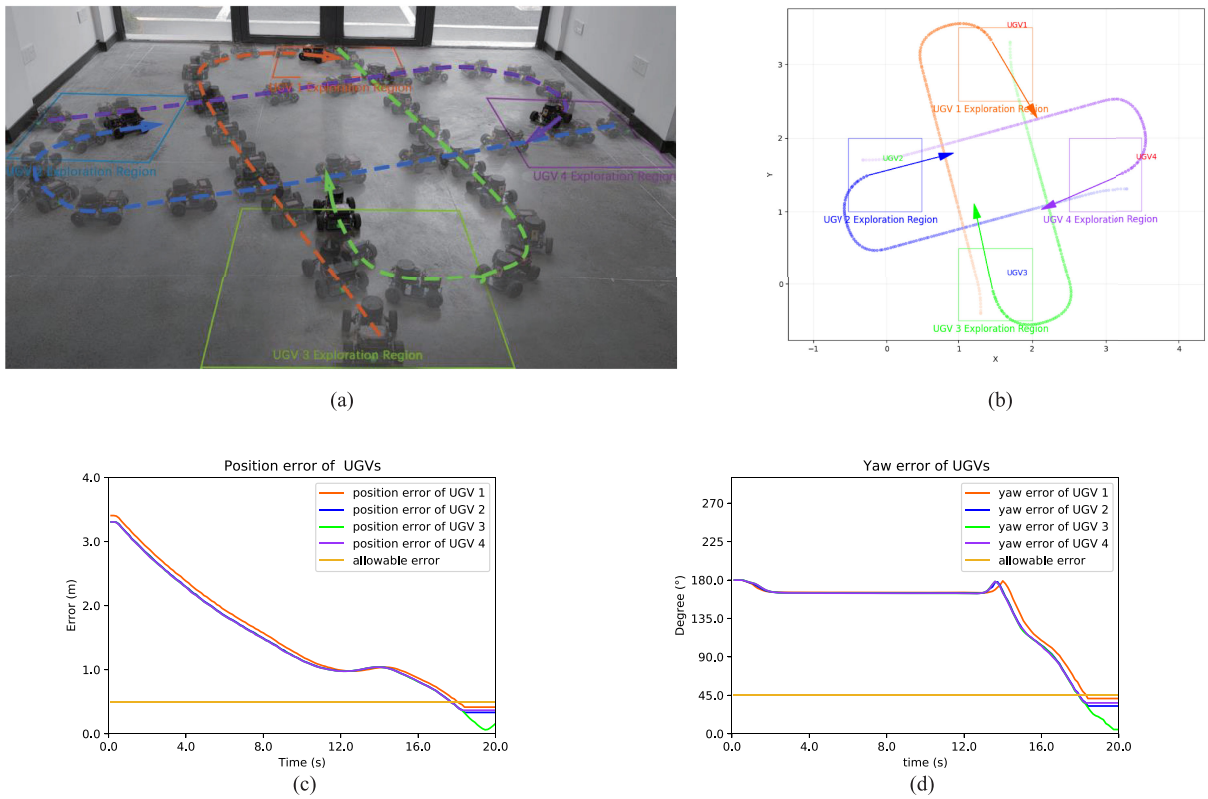


Fig. 6. Exploration task in the real world. (a) and (b) The traveled processes of four UGVs executing the exploration task. (c) The deviations between the ideal position and the actual position. (d) The deviations between the ideal heading angle and the heading angle.

TABLE I
AVERAGE EXECUTION TIME OF THE PCA ALGORITHM [MS]

UGV NO.	UGV 1	UGV 2	UGV 3	UGV 4
Simulation 1 ^{PCA}	35.39	38.62	35.87	40.89
Simulation 2 ^{PCA}	42.67	42.58	41.59	41.66
Simulation 2 ^{RVO}	37.16	39.09	40.43	40.66
Real Experiment ^{PCA}	38.69	39.55	38.51	38.95

targets at the desired velocity. The success rate is expressed as the rate of the number of UGVs reaching their targets to the number of all UGVs in the environment. Moreover, the safety weight is expressed as the ratio of the actual computation radius to the vehicle's physical radius. Finally, the experimental results are presented in Fig. 5.

Fig. 5(a) and (b) shows the linear and angular velocities of four vehicles under the PCA algorithm, respectively. Similarly, Fig. 5(d) and (e) shows the linear and angular velocities of the same four vehicles under the RVO algorithm, respectively. From these figures, it is observed that these velocities generated by the PCA algorithm are more feasible for the UGVs to follow as the smaller changes in both linear and angular velocity, result in a smoother trajectory. Here, chattering can be found in the velocity profiles due to the vehicle changing the heading for collision avoidance. However, the velocities can be followed by the vehicle as long as the range of chattering is within maximum linear and angular velocity. Otherwise, some optimization methods may be considered to reduce the chattering. Moreover, Fig. 5(c) and (f) are depicted in the above two metrics under the PCA method and the RVO method, in which it can be observed that the PCA algorithm has a higher success rate than the RVO algorithm, and has a comparable travel time. The average computation costs of four UGVs are presented in Table I, where the safety weight is 1.55, from which it can be found that the PCA algorithm requires more computation time than the RVO algorithm as the former increases the calculation of the Dubins curve.

The real-world experiment was performed on Ubuntu 16.04 with ROS Kinetic, and the rate of ROS is set as 10 Hz. We verify the practicability of the PCA algorithm by executing an exploration task. The workspace is set as a rectangle with around 3.5×3.5 m^2. Four UGVs are deployed for a ground exploration task, where all UGVs are required to adjust their posture so that they can smoothly return to the initial region once the return command is received. The communication of all vehicles adopts a WiFi module connected with a router. The configurations of the exploration task are as follows. The region of exploration is set as a rectangle with 1×1 m^2. The maximum exploration range of the vehicle is set as 0.5 m. The configurations of posture in start and target are shown in Table II. The experimental results are shown in Fig. 6, in which the processes of traveling of all UGVs are presented and the deviations between the appointed pose and the actual pose are also presented. Note that the start and exploration regions are highlighted for every vehicle. The average computation costs in every step of all UGVs are presented in Table I. From Fig. 6, it can be observed that all UGVs arrived at the exploration regions at the specific heading angle. Although

TABLE II
POSTURE OF INITIAL AND TARGET OF FOUR UGVS

Unit	Initial pose	Target pose
UGV 1	(3.0, 1.5, 180°)	(0.0, 1.5, 0°)
UGV 2	(1.5, 3.0, −90°)	(1.5, 0.0, 90°)
UGV 3	(0.0, 1.5, 0°)	(3.0, 1.5, 180°)
UGV 4	(1.5, 0.0, 90°)	(1.5, 3.0, −90°)

the vehicles did not totally follow the appointed goal position and heading angle, the exploration tasks are completed finally because all the deviations of position are within the maximum exploration range of 0.5 m. Meanwhile, all the deviations of heading angle are within 45°, which cannot impact the process of returning as the deviation is less than the maximum rotational angle. Overall, all experimental results indicate the effectiveness and practicability of the proposed algorithm.

V. CONCLUSION

In this article, we investigated the motion planning problem of the multiple nonholonomic UGVs. Considering the problem of the posture constraints, we proposed the novel PCA algorithm in the 2-D workspace. First, two simulation experiments were conducted in this article to verify the effectiveness and to analyze the quantitative performance of the proposed method, respectively. Then, the practicability of the PCA algorithm was verified by an experiment in a real environment. Here, both the first simulation experiments and the real-world experiment are conducted by executing exploration tasks. In addition, the velocity profiles may occur chattering, and the proposed algorithm simply regards the UGV as a circle. In the future, we will mainly overcome the limitations and expand the proposed method to a 3-D workspace for fixed-wing multiple unmanned aerial vehicles.

REFERENCES

[1] Y. Liu and R. Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica*, vol. 36, no. 7, pp. 1019–1047, 2018.

[2] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A. survey," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–31, 2019.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[4] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," Iowa State University, Tech. Rep. 98-11, Oct. 1998.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[6] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370214001386

[7] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 5628–5635.

[8] K. Okumura, Y. Tamura, and X. Defago, "Iterative refinement for real-time multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 9690–9697.

[9] D. Le and E. Plaku, "Multi-robot motion planning with dynamics via coordinated sampling-based expansion guided by multi-agent search," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1868–1875, Apr. 2019.

[10] C. W. Warren, "Multiple robot path coordination using artificial potential fields," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1990, pp. 500–505.

[11] H. Sun, W. Zhang, Y. Runxiang, and Y. Zhang, "Motion planning for mobile robots–focusing on deep reinforcement learning: A systematic review," *IEEE Access*, vol. 9, pp. 69061–69081, 2021.

[12] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 4956–4961.

[13] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[14] 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. Automat. Lett.*, vol. 5, no. 2, pp. 3221–3226, Apr. 2020.

[15] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1343–1350.

[16] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 856–892, 2020.

[17] F. Lamarche and S. Donikian, "Crowd of virtual humans: A new approach for real time navigation in complex and structured environments," in *Computer Graphics Forum*, vol. 23, no. 3. Oxford, UK and Boston, USA: Blackwell Publishing, Inc, 2004, pp. 509–518.

[18] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 1928–1935.

[19] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011.

[20] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3475–3482.

[21] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Berlin, Heidelberg: Springer, 2011, pp. 3–19.

[22] K. Guo, D. Wang, T. Fan, and J. Pan, "VR-ORCA: Variable responsibility optimal reciprocal collision avoidance," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4520–4527, Jul. 2021.

[23] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha, "PRVO: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1089–1096.

[24] S. H. Arul and D. Manocha, "V-rvo: Decentralized multi-agent collision avoidance using voronoi diagrams and reciprocal velocity obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 8097–8104.

[25] R. Mao, H. Gao, and L. Guo, "A novel collision-free navigation approach for multiple nonholonomic robots based on orca and linear mpc," *Math. Problems Eng.*, vol. 2020, pp. 1–16, 2020.

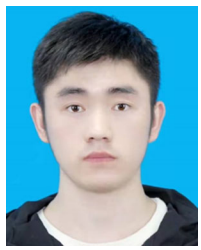
[26] M. Fuad, T. Agustinah, and D. Purwanto, "Collision avoidance of multi modal moving objects for mobile robot using hybrid velocity obstacles," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 3, pp. 407–421, 2020.

[27] X. Yuan, D. Zhang, J. Zhang, M. Zhang, and C. G. Soares, "A novel real-time collision risk awareness method based on velocity obstacle considering uncertainties in ship dynamics," *Ocean Eng.*, vol. 220, 2021, Art. no. 108436.

[28] J. Liu, W. Han, X. Wang, and J. Li, "Research on cooperative trajectory planning and tracking problem for multiple carrier aircraft on the deck," *IEEE Syst. J.*, vol. 14, no. 2, pp. 3027–3038, Jun. 2020.

[29] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Auton. Robots*, vol. 32, no. 4, pp. 385–403, 2012.

[30] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Trans. Robot.*, 2022.



Gang Xu received the B.S. degree in electrical engineering and automation from Hangzhou Dianzi University, Hangzhou, China, in 2019. Since 2020, he has been working toward the master's degree in electronics and information from the Polytechnic Institute, Zhejiang University, Hangzhou, China.

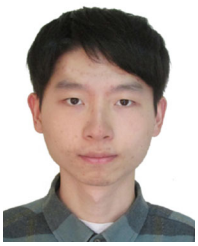
His current research interests include multi-robot motion planning and machine learning.

Yansong Chen received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2021. He is currently working toward the master's degree at the College of Control Science and Engineering, Zhejiang University.

His research interests include UAV formation and reinforcement learning.

Junjie Cao received the B.S. degree in mechanical engineering and automation from Nanjing Tech University, Nanjing, China, in 2014, and the M.S. degree in mechanical engineering (mechatronics), the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2017 and 2021, respectively.

His current research interests include machine learning, sequential decision making, and robotics.



Deye Zhu is majoring in control science and engineering at Zhejiang University. He will graduate with a bachelor's degree in 2022 and continue to study for a master's degree at the College of Control Science and Engineering, Zhejiang University.

Weiwei Liu was born in Hefei, China. Since 2019, he has been working toward the Ph.D. degree in electronics and information from the Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University, Hangzhou, China.. His research interests include artificial intelligence, decision-making, and control.

Yong Liu received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively.

He is currently a Professor with the Department of Control Science and Engineering, Institute of Cyber Systems and Control, Zhejiang University. He has authored or coauthored more than 90 research papers in machine learning, computer vision, information fusion, and robotics. His current research interests include machine learning, robotics vision, information processing, and granular computing.