



RESEARCH ARTICLE

Cooperative collision avoidance in multirobot systems using fuzzy rules and velocity obstacles

Wenbing Tang¹ , Yuan Zhou^{2,*} , Tianwei Zhang², Yang Liu², Jing Liu¹ and Zuohua Ding³

¹Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China, ²School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore, and ³School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China

*Corresponding author. E-mail: y.zhou@ntu.edu.sg

Received: 13 June 2022; **Revised:** 22 September 2022; **Accepted:** 26 September 2022;

First published online: 28 October 2022

Keywords: collision avoidance, fuzzy rules, multirobot systems, velocity obstacles

Abstract

Collision avoidance is critical in multirobot systems. Most of the current methods for collision avoidance either require high computation costs (e.g., velocity obstacles and mathematical optimization) or cannot always provide safety guarantees (e.g., learning-based methods). Moreover, they cannot deal with uncertain sensing data and linguistic requirements (e.g., the speed of a robot should not be large when it is near to other robots). Hence, to guarantee real-time collision avoidance and deal with linguistic requirements, a distributed and hybrid motion planning method, named Fuzzy-VO, is proposed for multirobot systems. It contains two basic components: fuzzy rules, which can deal with linguistic requirements and compute motion efficiently, and velocity obstacles (VOs), which can generate collision-free motion effectively. The Fuzzy-VO applies an intruder selection method to mitigate the exponential increase of the number of fuzzy rules. In detail, at any time instant, a robot checks the robots that it may collide with and retrieves the most dangerous robot in each sector based on the predicted collision time; then, the robot generates its velocity in real-time via fuzzy inference and VO-based fine-tuning. At each time instant, a robot only needs to retrieve its neighbors' current positions and velocities, so the method is fully distributed. Extensive simulations with a different number of robots are carried out to compare the performance of Fuzzy-VO with the conventional fuzzy rule method and the VO-based method from different aspects. The results show that: Compared with the conventional fuzzy rule method, the average success rate of the proposed method can be increased by 306.5%; compared with the VO-based method, the average one-step decision time is reduced by 740.9%.

1. Introduction

A multirobot system is a system containing multiple robots, such as unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs), that are moving around in a given environment to accomplish tasks cooperatively. Compared with their single-robot counterparts, multirobot systems can increase functionalities, improve efficiency, enhance adaptability, and provide robustness [1–3]. Multirobot systems have been applied to deal with labor-consuming or dangerous missions, such as assembly, disaster rescue, environmental protection, traffic monitoring, military reconnaissance, cargo delivery, and many other fields [4–6].

Coordinated motion is one of the most important requirements in a multirobot system. However, due to the complexity of the environment and the simultaneous motion of robots, collisions are common in coordinated motion. Many methods have been proposed to avoid collisions during robot motion. They can be mainly classified into two categories: model-driven methods and data-driven methods. Model-driven methods, such as formal methods [7, 8], discrete event system methods [9–11], potential field methods [12], velocity obstacles (VOs) [13, 14], model predictive control [15], and mathematical optimization methods [16], rely on the models of robots and/or environments. Specifically, formal

Table I. Summary of different collision avoidance methods.

	Methods	Description	Advantages	Disadvantages
Model-driven methods	Formal methods	Describe motion requirements using LTL and/or CTL	Safety guarantee	Limited to structured environments
	Potential field methods	Build attractive and repulsive potential functions	Unknown environment	Local minima, high complexity
	Velocity obstacles	Construct a velocity obstacle in the velocity space	Moving obstacles, safety guarantee	High computation cost, oscillatory motion
	Model predictive control	Build a predictive model of the control system	Flexibility	High computation cost
	Mathematical optimization methods	Construct a proper optimization problem and solve it	Modeling multiple constraints	High computation cost
Data-driven methods	Fuzzy rules	Build a fuzzy rule base and select a proper inference mechanism	Uncertainty, real-time inference	Oscillating paths, poor generalization ability
	Swarm intelligence algorithms	Define a proper optimization goal and strategy	Fast generation of acceptable solutions	Local optimum, unexpected solutions
	Deep reinforcement learning	Learn to maximize the expected cumulative reward	Unstructured data, dynamic environment	High training cost, low sampling efficiency

methods apply the technologies such as formal verification and model checking to control robots' motion [7, 8]. Discrete event system methods apply supervisory control theory to avoid collisions and deadlocks for multirobot systems [9–11]. Potential field methods define proper attractive potential functions and repulsive potential functions to lead a robot to its target while avoiding obstacles [12]. VO-based methods compute a collision-free velocity from the union velocity space of all obstacles at each time instant [13]. Model predictive control applies an explicit model to describe the control system and obtains a sequence of control inputs by solving an optimization problem based on the model [15]. Mathematical optimization methods generate control actions by modeling the collision avoidance problem as an optimization problem [16]. Data-driven methods are learning-based methods using sample data to learn proper controllers, such as fuzzy rules [15], swarm intelligence algorithms [17], and deep reinforcement learning (DRL) [18, 19]. Detailedly, fuzzy rules generate an action via fuzzy inference of the rules extracted from the collected data [15]. Swarm intelligence algorithms iteratively search for actions in the region defined by the previous optimal movements of the robot and its neighbors [17]. DRL formalizes the collision avoidance problem as a Markov decision process, which is solved by learning a decision policy mapping from the state space to the action space [18, 19]. Table I gives a brief summary of each

method and its advantages and disadvantages. Even though collision avoidance has been widely studied, there are some challenging problems that are not adequately addressed. First, most of the model-driven methods can guarantee safety but require high computation costs. Second, data-driven methods leverage offline learning to improve online computation efficiency but cannot provide collision-free guarantees.

In this paper, a new real-time collision avoidance approach, named Fuzzy-VO, is proposed to guarantee safety and computation efficiency for multirobot systems with sensing uncertainty and linguistic data. It combines fuzzy rules and VOs. On the one hand, considering uncertainties in sensing data, linguistic requirements may exist for a robot's motion, for example, moving slowly and turning right slightly. Fuzzy rules are a well-established tool to (1) deal with not only crisp data but also uncertain and linguistic data and (2) express the behavior of a system in an interpretable way. However, most of the current fuzzy rule-based methods are for a single robot [20, 21]. The design of fuzzy rules for collision avoidance in a multirobot system is still challenging since: (1) the form of a fuzzy rule is dependent on the number of robots; (2) the number of fuzzy rules increases exponentially with the number of robots [22]; and (3) a robot's motion may oscillate because of the large number of rules (the claim will be empirically validated in experiments). On the other hand, VO-based methods, such as optimal reciprocal collision avoidance (ORCA) [14], are well defined and generally applicable techniques for reactive obstacle avoidance with the existence of dynamic obstacles [13]. However, computing an optimal velocity for a robot in a multirobot system is still challenging since: (1) the scale of the problem (e.g., the number of constraints) increases with the number of robots, and so does the computation cost and (2) the robot may have few velocity candidates if numerous obstacles are around.

To mitigate the above drawbacks, Fuzzy-VO first uses a unified intruder selection method to generate fuzzy rules with an arbitrary number of robots. Specifically, given a robot, Fuzzy-VO divides its sensing region into a fixed number of sectors. In each sector, the robot applies the VO technology to evaluate and select the most dangerous robot to perform collision avoidance. Hence, the maximal number of robots to be avoided by a robot is constant. In this way, Fuzzy-VO can determine the form and the number of fuzzy rules. Then, sample data are collected via ORCA to learn a fuzzy rule base. Thus, the robot can compute the candidate motion in realtime via fuzzy inference on the rule base. Second, to guarantee that the final motion is collision-free, Fuzzy-VO applies the VO technology to check and fine-tune, if necessary, the candidate motion. Since each robot only needs to retrieve the current states of its neighbors, which can be obtained immediately, Fuzzy-VO is fully distributed. The same right-of-way, such as the turn-right rule, is applied to guarantee mutual exclusion during distributed decision making. A set of simulations are carried out with multiple UAVs. The results demonstrate the effectiveness of Fuzzy-VO in addressing potential collisions. Extensive comparison results show that Fuzzy-VO can reduce the number of rules and generate smoother paths compared with the conventional fuzzy rule approach and improve computation efficiency compared with the ORCA method.

The main contributions of this paper are threefold:

1. Based on sensing region partition and intruder selection, a practical strategy is proposed to build a fuzzy rule base with an arbitrary number of robots.
2. For each robot, a strategy is developed to generate collision-free velocities based on fuzzy rule inference and VO-based fine-tuning. It leverages the computation efficiency of data-driven methods and the safety guarantee of model-driven methods.
3. Based on the above two strategies, a fully distributed and real-time collision avoidance method, Fuzzy-VO, is proposed for multirobot systems with an arbitrary number of robots.

The rest of this paper is organized as follows. Section 2 summarizes the related work. Section 3 states the problem solved in this paper. Section 4 gives an overview of Fuzzy-VO. Sections 5 and 6 present the procedures for intruder selection and collision-free velocity generation, respectively. Section 7 provides the detailed algorithms, as well as the complexity analysis. Simulations are conducted in Section 8 to demonstrate the effectiveness and efficiency of Fuzzy-VO. Conclusion and future work are finally provided in Section 9.

2. Related work

This paper is related to the topic of collision avoidance, which is a key and popular topic in robotics. Many methods have been proposed in this area. For example, Wang *et al.* [23] proposed a three-dimensional navigation strategy for nonholonomic robots with moving obstacles, where the robot's motion direction maintains a constant angle with the obstacles' boundary tangents to avoid collisions. Lindqvist *et al.* [15] formalized the collision avoidance problem as an optimization problem in the framework of model predictive control and resolved it using OpEn solver. In [24], a potential field-based collision avoidance approach was proposed for nonholonomic UAVs, which took the velocity direction of an obstacle into consideration during the design of potential field functions.

Among the existing approaches, fuzzy rules are a promising tool to deal with data uncertainties and linguistic requirements for multirobot systems. Llorca *et al.* [25] proposed a fuzzy control-based autonomous collision avoidance system. In this system, the lateral displacement and the actual speed of the vehicle are used as fuzzy inputs, and the output of the fuzzy steering controller is the steering-wheel position. Vadakkepat *et al.* [26] proposed a fuzzy behavior-based architecture for the control of mobile robots in a multiagent environment. In ref. [20], a fuzzy obstacle avoidance controller is proposed for an autonomous vehicle using both negative fuzzy rules and traditional positive rules. The proposed architecture can be decomposed into four robot roles, 12 robot behaviors, and 14 robot actions, where obstacle avoidance is fulfilled by independent behaviors. However, the major drawback of these works is the generalization ability, that is, they are unsuitable for scenarios with a variable number of robots. Wen *et al.* [21] divided the sensing regions of the ultrasonic sensors on a UAV into three groups: front, left, and right, and then took each group's minimum obstacle distance as the inputs of its fuzzy controller. Chang *et al.* [27] proposed a two-layer fuzzy logic controller for multirobot coordination, which divides the scanning area into seven sectors and selects the shortest distance to a detected obstacle in each sector as input. But the selection may miss obstacles that are threatening and emergent in each sector.

VO-based methods are another kind of promising methods for collision avoidance. VO is first proposed by Fiorini and Shiller [28]. It is a velocity-based approach to avoiding collisions with moving obstacles. In VO, the velocity space of a robot is divided into collision and collision-free velocities, and an appropriate collision-free velocity is computed at any time instant. However, VO suffers from some weaknesses such as undesirable oscillatory motion and reciprocal dances [29, 30]. Some improved variations, such as reciprocal VO [13] and ORCA [14], have been proposed. van den Berg *et al.* [14] proposed the sufficient condition for multiple robots to avoid collisions and guarantee collision-free motion. By solving a linear program, each robot selects its optimal velocity from the intersection of all possible half-planes in the velocity space. Jenie *et al.* [31] proposed a cooperative autonomous collision avoidance algorithm named selective velocity obstacle (SVO), which is also an extension of the original VO. Especially, when SVO needs to avoid a possible collision according to the detection, the right-of-way rules for manned flight are taken into account in the decision-making process. Recently, Han *et al.* [19] combined VO with DRL to deal with the reciprocal collision avoidance problem under limited information scenarios. In ref. [32], VO is applied to deduce the collision conditions in connected and automated vehicles. However, these VO-based methods require high computation costs in crowded environments [33].

Compared with the aforementioned methods in the literature, the method proposed in this paper aims to achieve real-time computation efficiency and safety guarantees simultaneously for multirobot systems.

3. Problem statement

The scope of this paper is the cooperative motion of a set of robots moving in a 2D space, for example, a set of UGVs moving on the ground or multiple UAVs moving at the same height. Assume that there are N holonomic robots moving in the same environment. For nonholonomic robots, it is recommended to read refs. [23, 24] for more details. Note that to perceive the environment, each robot is equipped with different sensors, for example, cameras and LiDARs. However, due to measurement errors, uncertainties

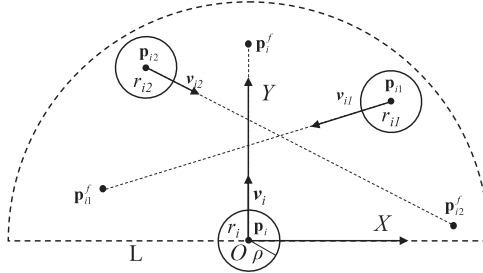


Figure 1. Robot r_i with two intruders r_{i1} and r_{i2} (i.e., the three circles) in the body frame C_i . The semicircle area is the collision region of the robot r_i .

may exist in the sensing data. The motion task for a robot r_i is to move from its initial position $\mathbf{p}_i^0 \in \mathbb{R}^2$ to the target position $\mathbf{p}_i^f \in \mathbb{R}^2$ within a given duration τ , where \mathbb{R}^2 is the 2D Euclidian space. Each robot r_i has a reference speed v_i^{ref} , where $v_i^{ref} \leq v_{max}$, and v_{max} is the upper bound of the robot speed. Moreover, it is assumed that each robot cannot move backward. Since multiple robots are moving in the same environment, different robots need to avoid collisions with each other. In this work, each robot is modeled as a sphere, and its position is identified by its center. Each robot regards other robots as dynamic obstacles. For simplicity, other dynamic obstacles are not considered.

Before giving the problem statement, some symbols and definitions are defined. Given a robot r_i , its position and velocity at time t are denoted as $\mathbf{p}_i(t)$ and $\mathbf{v}_i(t)$, respectively. Clearly, $\forall t > 0, \|\mathbf{v}_i(t)\| \leq v_{max}$. Since the motion is assumed in a 2D space, $\mathbf{p}(t) \in \mathbb{R}^2$ and $\mathbf{v}(t) \in \mathbb{R}^2$. The state of a robot, denoted as s , is a vector containing the robot's position and velocity, that is, $s = (\mathbf{p}, \mathbf{v})$. The set of all possible states of a robot r_i is denoted as \mathcal{S}_i . The trajectory of a robot r_i , denoted as Tr_i , is a time-parameterized function mapping from \mathbb{R}^+ to \mathcal{S}_i , that is, $Tr_i(t) = s_i(t) = (\mathbf{p}_i(t), \mathbf{v}_i(t)) \in \mathcal{S}_i$. By discretizing the time into discrete time instants with the same time step, that is, $0 = t_0, t_1, \dots, t_K = \tau$, the motion of r_i can be formalized as

$$\begin{aligned} \mathbf{p}_i(t) &= \mathbf{p}_i(t_k) + \mathbf{v}_i(t_k)(t - t_k), \quad t \in [t_k, t_{k+1}) \\ \mathbf{p}_i(t_0) &= \mathbf{p}_i^0, \quad \mathbf{p}_i(t_K) = \mathbf{p}_i^f, \\ \mathbf{v}_i(t_k) &= [v_i(t_k) \cos \theta_i(t_k), v_i(t_k) \sin \theta_i(t_k)]. \end{aligned} \tag{1}$$

where $v_i(t_k) \in \mathbb{R}$ and $\theta_i(t_k) \in \mathbb{R}$ are the speed and motion direction of robot r_i at t_k , respectively.

At any time instant, a robot needs to monitor a proper region, denoted as *collision region*, with respect to its current position. The *body frame* of a robot r_i is denoted as C_i . It is a Cartesian coordinate system whose origin is the center of r_i , the y-axis is the same as \mathbf{v}_i , and the x-axis is perpendicular to the y-axis. As shown in Fig. 1, at the current instant, r_i is at O , and its velocity is \mathbf{v}_i . Then the related C_i is XOY , where the Y-axis is \mathbf{v}_i . Since each robot cannot move backward, the possible motion area is $\{(x, y) | y \geq 0\}$. Hence, the collision region of a robot r_i at time t can be defined as $CR_i(t) = \{(x, y) \in C_i | 0 \leq x \leq L \cos(\theta), 0 \leq y \leq L \sin(\theta), 0 \leq \theta \leq \pi\}$, provided that the sensing range is L . To guarantee safety, each robot needs to avoid collisions with other robots in its collision region. At any time t , two robots r_i and r_j are in a collision if $\|\mathbf{p}_i(t) - \mathbf{p}_j(t)\|_2 < 2\rho$, where $\mathbf{p}_i(t) \in \mathcal{S}_i, \mathbf{p}_j(t) \in \mathcal{S}_j$, and ρ is the safe radius for each robot. Hence, there is the following definition.

Definition 1 (Intruder). A robot r_j is called an intruder of r_i at time t if $\mathbf{p}_j(t) \in CR_i(t)$ and $\exists t' \in (t, \tau)$ such that $\|\mathbf{p}_i(t') - \mathbf{p}_j(t')\|_2 < 2\rho$, where $\mathbf{p}_i(t') = \mathbf{p}_i(t) + \mathbf{v}_i(t)(t' - t)$ and $\mathbf{p}_j(t') = \mathbf{p}_j(t) + \mathbf{v}_j(t)(t' - t)$.

For example, as shown in Fig. 1, the three robots, r_i, r_{i1} , and r_{i2} , are moving to $\mathbf{p}_i^f, \mathbf{p}_{i1}^f$, and \mathbf{p}_{i2}^f , respectively. At the current time, r_i detects that r_{i1} and r_{i2} will collide with it if all of them keep their current velocities, so r_{i1} and r_{i2} are intruders of r_i .

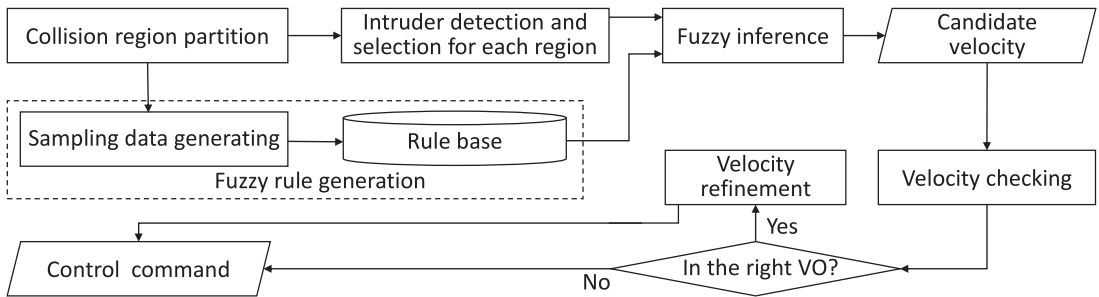


Figure 2. Framework of Fuzzy-VO.

Based on the above descriptions, to generate a collision-free trajectory for a robot, the robot needs to determine its velocity at the discrete time instants. Hence, the problem studied in this paper can be described as follows:

Problem 1. Given a set of robots, each of which may contain uncertain sensing data during its motion, decide the motion of each robot, that is, its velocity vectors, such that the robot can always detect and avoid potential collisions with its intruders.

4. Overview of the proposed method

To deal with uncertainties in sensing data and guarantee real-time efficiency, a VO-aided fuzzy inference method is proposed to generate collision-free motion for each robot. This section gives the framework of the proposed Fuzzy-VO, while the details are given in the following sections.

Figure 2 shows the high-level workflow of Fuzzy-VO. The main idea is to restrict the number of fuzzy rules by selecting a proper and fixed number of intruders for collision avoidance rather than considering all intruders. In this way, Fuzzy-VO can be adopted to a different number of robots and guarantee flexibility and scalability. It mainly contains three processes, that is, intruder selection, fuzzy rule generation and inference, and velocity generation.

Intruder selection. For each robot, the first step is to partition its collision region into a set of disjoint sectors. A sector may contain several intruders, and different sectors may contain a different number of intruders. Hence, to generate a universal method for a different number of robots, a proper number of intruders should be selected in each sector. In this paper, only the most dangerous intruder is selected for each sector based on the technology of VOs.

Fuzzy rule generation and inference. Fuzzy rule-based collision avoidance technology is used to generate a candidate motion command. First, to build the fuzzy base, the conventional motion planning algorithms are applied to generate corresponding sampling data. Then, fuzzy rules can be extracted from the collected data. When the rule base is constructed, at any instant, the robot can perform fuzzy inference based on the current selected intruders and generate a candidate velocity.

Velocity generation. Since the fuzzy rules are generated based on sampling data rather than exact system models, the reasoning results cannot always guarantee collision avoidance. Hence, validation of the candidate velocity is required to generate the actual velocity. In this paper, it is assumed that each robot moves to its right side to avoid collisions. Hence, the robot only needs to check whether the generated velocity is in the VOs of the right intruders and fine-tunes the candidate velocity if needed.

5. Intruder determination and selection

This section describes the method to select a proper number of intruders for collision avoidance by the robot. Specifically, the partition of the collision region is first introduced, followed by the VO-based intruder selection method to retrieve the most dangerous intruder in each partition.

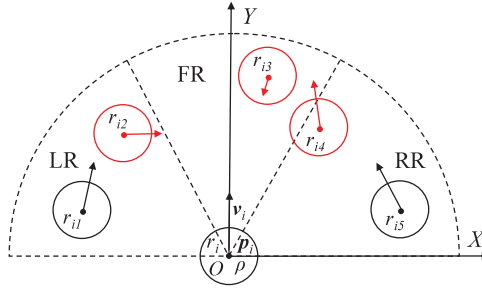


Figure 3. Partition the collision region into three sectors and screen the most dangerous intruder in each sector.

5.1. Collision region partition

At any time instant, r_i 's collision region $CR_i(t)$ is partitioned into l equal sectors, that is, $l_i = \{(x, y) | 0 \leq x \leq L \cos(\theta), 0 \leq y \leq L \sin(\theta), (i - 1)\pi/l \leq \theta \leq i\pi/l\}$, $i = 1, 2, \dots, l$. Consequently, an appropriate value of l should be selected. Considering that many real-world mobile robots are equipped with three groups of sensors in the front, such as [34, 35], in this paper, the collision region is equally partitioned into three sectors: the left region (LR), the front region (FR), and the right region (RR). The robots in LR are left intruders, those in FR are front intruders, and in RR are right intruders. For example, Fig. 3 shows the three sectors of r_i , where each sector has a central angle of $\pi/3$. r_{i1} and r_{i2} are in LR, r_{i3} is in FR, and r_{i4} and r_{i5} are in RR. They are the intruders of r_i at the current instant.

5.2. VO-based intruder selection

To avoid the explosion of the number of fuzzy rules with the number of intruders, in this subsection, a VO-based method is proposed to select a proper intruder in each sector such that the generated rules with a finite number, independent of the number of robots.

The main idea of VO is to select a velocity of a robot outside the VO, which is the set of velocity that may cause collisions with other robots or obstacles. As shown in Fig. 4, suppose robot r_i currently is at \mathbf{p}_i . It detects an intruder r_{i1} in its collision region CR_i , whose position and velocity are \mathbf{p}_{i1} and \mathbf{v}_{i1} , respectively. So r_i should select a velocity \mathbf{v}_i to avoid collision with r_{i1} . At the current time instant, r_i 's collision region with respect to r_{i1} can be described as $C_{i|i1} = \{\mathbf{p}(i) \in \mathbb{R}^2 | \|\mathbf{p}(i) - \mathbf{p}_{i1}\|_2 < 2\rho\}$, that is, the region within the dashed circle in Fig. 4. The relative velocity of r_i with respect to r_{i1} can be described as $\mathbf{v}_{i|i1} = \mathbf{v}_i - \mathbf{v}_{i1}$. The relative motion can be defined as $\lambda(\mathbf{p}_i, \mathbf{v}_{i|i1}) = \{\mathbf{p}_i + t\mathbf{v}_{i|i1} | t > 0\}$, that is, the blue ray in Fig. 4. Clearly, r_i and r_{i1} will collide in the future if $\lambda(\mathbf{p}_i, \mathbf{v}_{i|i1}) \cap C_{i|i1} \neq \emptyset$. Hence, the VO of r_i related to r_{i1} is defined as $VO_{i|i1} = \{\mathbf{v}_i | \lambda(\mathbf{p}_i, \mathbf{v}_{i|i1}) \cap C_{i|i1} = \emptyset\}$, that is, the gray cone in Fig. 4.

Based on VO, a method is proposed to select the most dangerous intruder in each section according to the collision risk, which is defined as the potential collision time. Furthermore, the collision time criterion is defined to evaluate the collision risk.

Definition 2 (Potential Collision Time). The potential collision time of r_i with respect to r_j , denoted as $\Delta T_i^c(j)$, is the estimated shortest time duration from the current time instant to the occurrence of a potential collision between r_i and r_j with their current velocities.

In detail, the computation of potential collision time is as follows. Consider the relative motion of r_i with respect to r_j . As shown in Fig. 5, for the relative motion, r_j is with zero velocity, and r_i has a relative velocity $\mathbf{v}_i(t) - \mathbf{v}_j(t)$. Clearly, the minimum distance is reached at the time when the relative

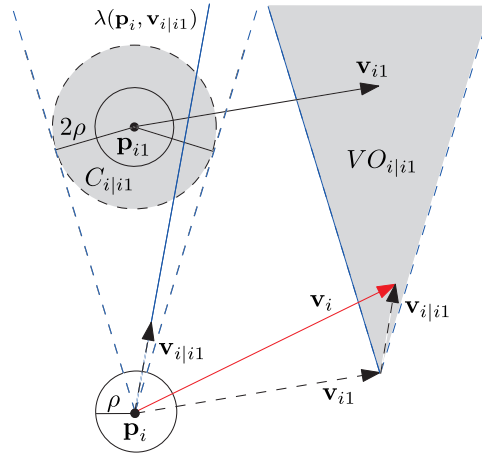


Figure 4. Illustration of VO. $VO_{i|i1}$ is the velocity obstacle of r_i . Each velocity in $VO_{i|i1}$ will cause a collision with r_{i1} in some time instant of the future.

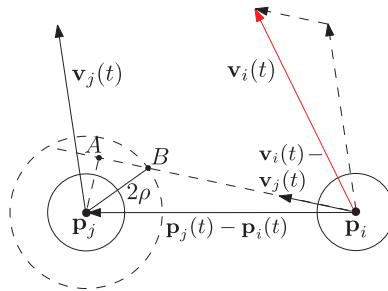


Figure 5. The relative motion of r_i with respect to r_j .

motion reaches position A. The distance from p_i to A is

$$d(p_i, A) = \frac{(p_j(t) - p_i(t))^T (v_i(t) - v_j(t))}{\|v_i(t) - v_j(t)\|_2} \tag{2}$$

Hence, the estimated time from p_i to A is $\frac{d(p_i, A)}{\|v_i(t) - v_j(t)\|_2}$, that is,

$$\Delta T_i(j) = \frac{(p_j(t) - p_i(t))^T (v_i(t) - v_j(t))}{\|v_i(t) - v_j(t)\|_2^2}. \tag{3}$$

Based on the procedure of VO, if $v_i \in VO_{ij}$, $d_i(j) < 2\rho$, and vice versa. In this case, a collision between r_i and r_j happens when the relative motion arrives at position B, as shown in Fig. 5. Hence, $\Delta T_i^c(j)$ can be computed as follows:

$$\begin{aligned} \Delta T_i^c(j) &= \Delta T_i(j) - t(B, A) \\ &= \Delta T_i(j) - \frac{\sqrt{4\rho^2 - d_i^2(j)}}{\|v_i(t) - v_j(t)\|_2}. \end{aligned} \tag{4}$$

Clearly, the smaller $\Delta T_i^c(j)$ is, the more dangerous r_j is, and the higher priority it has for collision avoidance. According to the estimated collision time, the most dangerous intruder is selected in each sector, that is, the intruder with the smallest $\Delta T_i^c(j)$ in each sector. For example, as shown in Fig. 3, the selected intruder in LR is r_{i2} as it has smaller collision time than r_{i1} .

6. Collision avoidance using fuzzy rules and velocity obstacles

In this section, following the collision region partition, the process to build a fuzzy rule base is first presented, and then the procedure for velocity generation is provided.

6.1. Introduction of fuzzy rules

This subsection first gives a brief introduction of fuzzy rules, and the details can be found in ref. [36]. Let U be the domain of discourse and $u \in U$. A fuzzy set ϕ in U is characterized by a real-value function $\mu_\phi : U \rightarrow [0, 1]$, which assigns each element in U with a real number in the interval $[0, 1]$.

A single fuzzy IF-THEN rule (or simply fuzzy rule) is defined on linguistic variables with the form:

IF x is A (premise) **THEN** y is B (consequence)

where x and y are two fuzzy/linguistic variables. A and B are two fuzzy sets. A more general type of fuzzy rules in practice can be described as:

IF x_1 is $A_1 \wedge \dots \wedge x_p$ is A_p **THEN** y_1 is $B_1 \wedge \dots \wedge y_q$ is B_q ,

where $p \geq 1$ and $q \geq 1$ are integers.

6.2. Fuzzy rule generation

On one hand, according to the selection of intruders, the input of a fuzzy rule is the collision time ΔT^c of the selected intruders. On the other hand, since the kinematic model of each robot considered in this paper is unicycle kinematics, the output variables of fuzzy rules are set as the *speed ratio* α and the *orientation change* $\Delta\theta$. The new speed is $v'_c = \alpha v_c$, which adjusts the current speed v_c according to a proper ratio α . Note that, to guarantee mutual exclusion during distributed decision making and avoid collisions, each robot is expected to turn right with a proper direction, so $\Delta\theta \in [0, \pi/2]$, and the new orientation is $\theta_i - \Delta\theta$, where θ_i is the current orientation of r_i . Then, the selected velocity is $\mathbf{v}'_i = (v'_c \cos(\theta_i - \Delta\theta), v'_c \sin(\theta_i - \Delta\theta))$. Note that the outputs of fuzzy rules are determined by the robot kinematics described in (1), rather than robot dynamics, such as inertia.

In the sequel, the generation of fuzzy rules is described, that is, fuzzification, data sampling, and rule determination.

6.2.1. Fuzzification

The main task of fuzzification is to translate crisp variables into the corresponding linguistic ones. Hence, for each crisp variable x , the set of fuzzy terms and their corresponding membership functions should be determined. Users can select any membership function as long as it can map the crisp data into desired degree of memberships. In this paper, the triangular membership function is applied for each fuzzy set as (1) it has been proven to have good quality results and computational efficiency in many practical applications (including robot motion control) [37, 38] and (2) it shows good performance in our simulation experiments.

First, consider the fuzzification process of the input variable ΔT^c . Usually, a robot needs a time duration, say response time, to perform collision avoidance, including collision prediction and decision execution. Based on the configurations of a robot and its history motion records, the minimal and maximal response time of the robots can be determined, denoted as t_1 and t_2 , respectively. If ΔT^c is less than t_1 , then the situation is very emergent, and the robot needs to perform some special actions, for example, stop immediately, to avoid collisions. Otherwise, if ΔT^c is small, meaning that the remaining time to take collision avoidance actions is short, then this situation is dangerous, and the robot needs to do its best to avoid collisions; while if ΔT^c is large, meaning that the robot has enough time to avoid collisions, then the current motion is safe. Hence, three fuzzy sets, that is, E (emergent), D (dangerous),

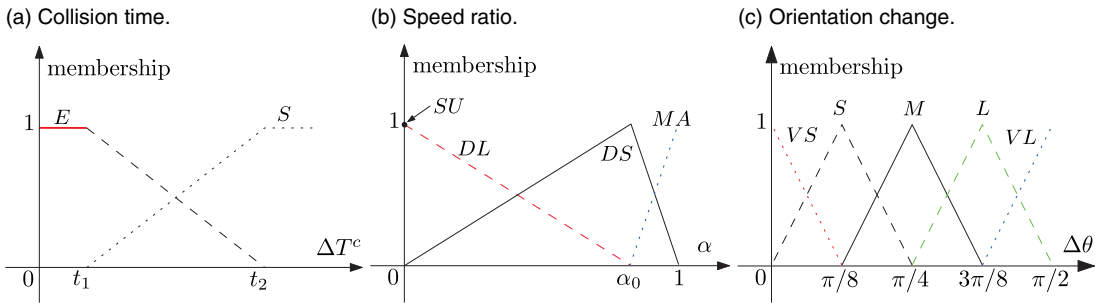


Figure 6. Membership functions for all variables. (a) Membership function of ΔT^c ; (b) membership function of α ; (c) membership function of $\Delta\theta$.

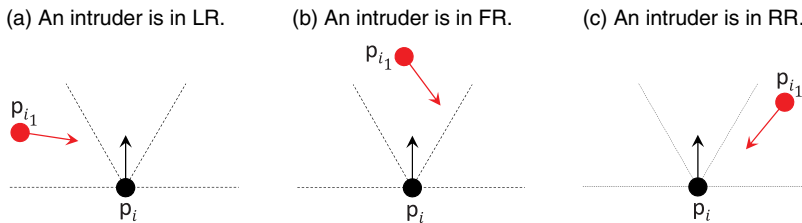


Figure 7. Scenarios that there is only one thread in a sector.

and S (safe), are defined to describe ΔT^c . Their membership functions are shown in Fig. 6(a). Moreover, the formal definitions of all membership functions are given in the appendix.

Second, consider the fuzzification of the output variables. (1) The speed variable is partitioned into four fuzzy sets: MA (maintain), DS (decelerate slightly), DL (decelerate largely), and SU (stop urgently). Since the level of speed deceleration is related to the current speed, the ratio of the new speed (v_c) to the current speed (v), denoted as α , is the inputs of the three membership functions. Note that $0 \leq \alpha \leq 1$. The graphical representation of these membership functions is given in Fig. 6(b), where α_0 is the threshold of a maintaining action and can be determined by an expert or based on users' requirements. (2) For the change of orientation, five fuzzy sets are applied to describe its values: VS (very small), SM (small), M (medium), L (large), and VL (very large). Their membership functions are shown in Fig. 6(c). Note that one can define more fuzzy sets for speed and orientation change if needed.

6.2.2. Building of fuzzy rules

As described before, to avoid an exponential increase in the number of fuzzy rules, a robot selects at most one intruder in each sector to perform collision avoidance. Hence, the premise of a rule contains at most three fuzzy propositions, each of which has four possible forms, including the empty situation. So there are $4^3 - 1 = 63$ possible premises. The consequence of a rule contains at most two independent fuzzy propositions, and they have four and five possible forms, respectively. Hence, the number of candidate rules is $63 * (4 + 5) = 567$. However, proper rules should be selected from the candidates. For each premise, the next step is to determine the consequences related to the two output linguistic variables, respectively. The rule selection is based on supporting degrees described in ref. [39]. For each premise, the supporting degrees of all candidate rules are computed, and only the rule with the maximal one is selected. Note that rules whose supporting degrees are zero are also filtered.

Based on the partition of the collision region and intruder selection, there are three kinds of scenarios during the generation of fuzzy rules. The first one is that there is only one sector existing a selected intruder, as shown in Fig. 7. For each one shown in Fig. 7, the sampling data are generated from multiple simulation runs by setting different status of the intruder. For each run, the ORCA algorithm is performed

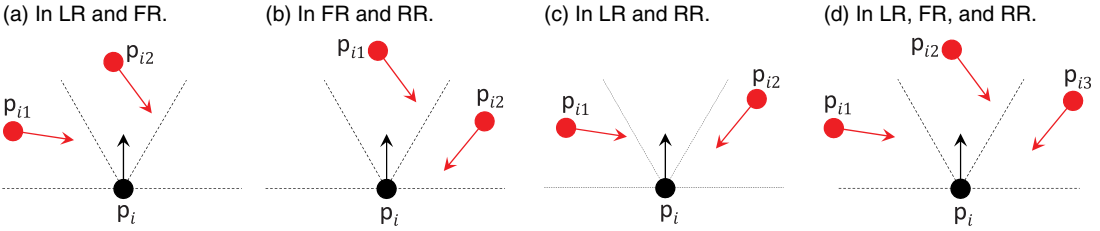


Figure 8. Scenarios with multiple intruders.

to generate the cooperative motion data between r_i and the intruder. For example, suppose at time instant t , the robot r_i , with the state $(\mathbf{p}_i(t), \mathbf{v}_i(t))$, detects an intruder r_{i1} , whose state is $(\mathbf{p}_{i1}(t), \mathbf{v}_{i1}(t))$, in LR, and the ORCA method generates a new velocity $\mathbf{v}_{i,ORCA}(t)$ for r_i . Let $\theta_{\mathbf{v}_i}(t)$ and $\theta_{\mathbf{v}_{i,ORCA}}(t)$ denote the *orientation angle* of $\mathbf{v}_i(t)$ and $\mathbf{v}_{i,ORCA}(t)$ in the *inertial frame XOY*. When $\theta_{\mathbf{v}_{i,ORCA}}(t) \leq \theta_{\mathbf{v}_i}(t)$ and $\|\mathbf{v}_{i,ORCA}(t)\|_2 \leq \|\mathbf{v}_i(t)\|_2$, a sample $(L_{\Delta T^c}, \Delta T_i(i1), \alpha_i(t), \Delta\theta_i(t))$ is collected, where $\Delta T_i(i1)$ is computed based on (4), while $\alpha_i(t)$ and $\Delta\theta_i(t)$ are computed based on the following equations.

$$\alpha_i(t) = \frac{\|\mathbf{v}_{i,ORCA}(t)\|_2}{\|\mathbf{v}_i(t)\|_2}, \tag{5}$$

$$\Delta\theta_i(t) = \theta_{\mathbf{v}_i}(t) - \theta_{\mathbf{v}_{i,ORCA}}(t) \tag{6}$$

In this way, a total of 1664 valid records are generated from the first kind of scenarios. With all the records, six rules are generated, whose details are given in the appendix and the website <https://fuzzyvo.github.io/>. Note that the original rules are in the form with single-input–single-output, such as “**IF** $L_{\Delta T^c}$ is D , **THEN** v_c is DS ”. However, the rules with the same premise can be combined, for example, for the rules “**IF** $L_{\Delta T^c}$ is D , **THEN** v_c is DS ” and “**IF** $L_{\Delta T^c}$ is D , **THEN** v_c is L ”, they can be combined as one “**IF** $L_{\Delta T^c}$ is D , **THEN** v_c is DS and $\Delta\theta$ is L ”.

The second kind of scenarios is that there exist two sectors such that either of them contains an intruder, as shown in Fig. 8(a)–(c). Similarly, 29,347 samples are collected in this kind of scenarios, and 12 fuzzy rules are generated, which are given in the appendix. The third one is that each sector contains a selected intruder, as shown in Fig. 8(d). To generate proper rules for this kind of scenarios, a total of 5299 records are collected to train rules.

When the collision time related to the intruder in a sector is emergent, the robot should stop urgently. Hence, three emergent rules are also introduced, which are also given in the appendix. Finally, a rule base containing 29 fuzzy rules is built. All generated fuzzy rules are listed in the appendix.

6.3. Velocity generation via fuzzy inference

When the rule base is built, the velocities can be generated directly via fuzzy inference. In this paper, the Mamdani (min) inference mechanism is applied, whose output is a fuzzy set [40]. The mechanism can be described as follows. Given an input $x^0 = (x_1^0, \dots, x_p^0)$ and the activated fuzzy rule “**rule**₁ : **IF** x_1 is $A_1 \wedge \dots \wedge x_p$ is A_p **THEN** y is B ,” the inferential fuzzy set for the consequence is computed based on (7).

$$\begin{aligned} \inf(x^0, rule_1) &= A_1(x_1^0) \wedge \dots \wedge A_p(x_p^0) \\ B(y, x^0, rule_1) &= \inf(rule_1) \wedge B(y) \end{aligned} \tag{7}$$

where $A_1(x_1^0) \wedge \dots \wedge A_p(x_p^0) = \min\{A_1(x_1^0), \dots, A_p(x_p^0)\}$. For the activation of multiple rules, the final inferential fuzzy set is determined based on (8).

$$\begin{aligned} B(y, x^0) &= B(y, x^0, rule_{i1}) \vee \dots \vee B(y, x^0, rule_{ij}) \\ &= \max\{B(y, x^0, rule_{i1}), \dots, B(y, x^0, rule_{ij})\} \end{aligned} \tag{8}$$

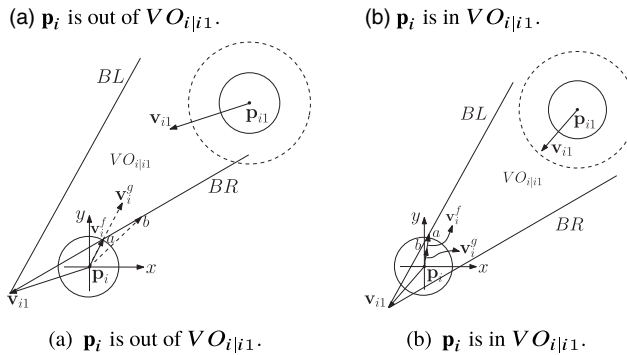


Figure 9. Illustration of velocity refinement. \mathbf{v}_i^g is obtained from fuzzy inference, \mathbf{v}_i^f is refined velocity. (a) \mathbf{p}_i is out of $VO_{i|j1}$. (b) \mathbf{p}_i is in $VO_{i|j1}$.

Since $B(y, x^0)$ is a fuzzy set, defuzzification is required to generate a crisp value of the velocity for the robot. One of the common defuzzification methods is the center of gravity, which determines a crisp value based on the center of gravity of the generated fuzzy set [41]. The computation of the center of gravity is given in (9).

$$y^* = \frac{\int yB(y, x^0)dy}{\int B(y, x^0)dy} \tag{9}$$

Based on the above procedure, given the current states of a robot and its intruders, the collision times related to different intruders are first computed and the most dangerous intruder in each sector is selected. Then, a new velocity is generated via the following fuzzy inference process: (1) decide all possible premises, that is, all combinations of the fuzzy sets of the selected intruders' collision time; (2) for each premise, activate the corresponding rule and return a fuzzy set based on (7); (3) compute the final fuzzy set based on (8); and (4) compute the speed ratio α^* and the orientation change $\Delta\theta^*$ based on (9). Hence, the new velocity can be described as

$$\mathbf{v}_i^g = [\alpha^*v_c \cos(\theta - \Delta\theta^*), \alpha^*v_c \sin(\theta - \Delta\theta^*)]^T \tag{10}$$

Using the above three steps, a candidate velocity can be generated for each robot for collision avoidance.

6.4. Velocity fine-tuning

Since fuzzy inference is a learning-based method depending on the quality of the sampling data, the generated velocity may still result in a collision. Hence, fine-tuning is required to get a collision-free velocity.

In case the generated velocity for r_i is still in the VO region of some selected intruders, the velocity should be adjusted such that it is out of the VO region. Indeed, any VO-based method can be applied, such as ORCA, to compute the current velocity. However, considering the computation cost, a heuristic fine-tuning method is proposed. As shown in Fig. 9, \mathbf{v}_i^g is the velocity obtained by fuzzy inference, BL and BR are the boundaries of the VO region. There are two situations based on the robot's current position. The first one is shown in Fig. 9(a), where the current position \mathbf{p}_i is out of the VO $VO_{i|j1}$. In this case, the solution is to decrease the generated speed $\|\mathbf{v}_i^g\|_2$ to the boundary of VO (i.e., point a in Fig. 9(a)) without changing the orientation of the velocity. The second one is shown in Fig. 9(b), where the current position \mathbf{p}_i is in the VO $VO_{i|j1}$. In this case, the main idea is to increase the speed and/or change the orientation of \mathbf{v}_i^g . However, since the built fuzzy rules prefer to decrease the speed to avoid collisions, the generated speed may be too small to guarantee collision avoidance by only changing

Algorithm 1: Update of collision-free velocity for r_i at time instant t .

```

Input: The current state of  $r_i$ :  $(\mathbf{p}_i, \mathbf{v}_i)$ ; the states of other robots  $(\mathbf{p}_{ij}, \mathbf{v}_{ij})_{j=1}^m$  within its sensing range;  $\rho$ : safe radius.
Output: Update the velocity of  $r_i$  to a new velocity  $\mathbf{v}_i^{new}$ 
1 Initialization:  $LR_i = \emptyset, FR_i = \emptyset, RR_i = \emptyset, LI = \emptyset, FI = \emptyset, and RI = \emptyset$ ;
2 Partition the collision-checking region into  $LR, FR,$  and  $RR$ ;
3 for  $k=1$  to  $m$  do
4     Compute the velocity obstacle  $VO_{i|ik}$ ;
5     if  $\mathbf{v}_i \in VO_{i|ik}$  then
6         if  $\mathbf{p}_{ik} \in LR$  then
7              $LR_i = LR_i \cup \{r_{ik}, \mathbf{p}_{ik}, \mathbf{v}_{ik}\}$ ;
8         else if  $\mathbf{p}_{ik} \in FR$  then
9              $FR_i = FR_i \cup \{r_{ik}, \mathbf{p}_{ik}, \mathbf{v}_{ik}\}$ ;
10        else
11             $RR_i = RR_i \cup \{r_{ik}, \mathbf{p}_{ik}, \mathbf{v}_{ik}\}$ ;
12  $\Delta T = +\infty$ ;
13 while  $LR_i \neq \emptyset$  do
14     for  $(r_j, \mathbf{p}_j, \mathbf{v}_j) \in LR_i$  do
15         Compute the minimal distance  $d_i(j)$  based on (2);
16         if  $d_i(j) < 2\rho$  then
17             Compute the collision time  $\Delta T_i^c(j)$  based on (4);
18             if  $\Delta T > \Delta T_i^c(j)$  then
19                  $\Delta T = \Delta T_i^c(j)$ ;
20                  $LI = (\mathbf{p}_j, \mathbf{v}_j, \Delta T)$ ;
21      $LR_i = LR_i \setminus \{(r_j, \mathbf{p}_j, \mathbf{v}_j)\}$ ;
22 Repeat Lines 12–22 for  $FR_i$  and  $RR_i$ , respectively;
23 Perform fuzzy inference and return  $\alpha^*$  and  $\Delta\theta^*$ ;
24 Compute the current speed  $v_c = \|\mathbf{v}_i\|_2$  and orientation  $\theta$ ;
25  $\mathbf{v}_i^g = [\alpha^* v_c \cos(\theta - \Delta\theta^*), \alpha^* v_c \sin(\theta - \Delta\theta^*)]^T$ ;
26 Check and refine the velocity  $\mathbf{v}_i^g$  to a collision-free velocity  $\mathbf{v}_i^{new}$ . Update the current velocity to  $\mathbf{v}_i^{new}$ ;
27 return  $\mathbf{v}_i^{new}$ .

```

the orientation. Hence, for this situation, the speed will be increased to the boundary (i.e., point a in Fig. 9(b)).

7. Distributed algorithm for cooperative collision avoidance

This section summarizes the distributed collision avoidance algorithm based on fuzzy rules, as well as the analysis of the computation complexity.

Algorithm 1 describes the velocity computation at each time instant for a robot. In this algorithm, Lines 3–11 classify the intruders into different sectors. Since there are N_r robots in the system, in the worst case, a robot needs to check all other robots. So the computation complexity is $O(N_r)$. Lines 12–22 select an intruder in each sector. Hence, the computation complexity is $O(N_r)$ in the worst case. Line 23 execute fuzzy inference. Based on the described inference procedure, a robot activates multiple rules with different combinations of fuzzy sets of the selected intruders each time. In a general case, suppose each robot partitions its collision region into q sectors, and each intruder is assigned with f fuzzy sets. Hence, each time a robot activates at most f^q rules, and the computation complexity is $O(f^q)$. Once

Algorithm 2: Distributed collision avoidance for a cooperative multi-robot system.

Input: The initial and target positions of all robots, the target position \mathbf{p}_i^0 and $\mathbf{p}_i^f, i = 1, 2, \dots, N_r$
Output: Trajectories leading all robots to move from their initial positions to the target positions without any collisions.

```

1 Determine the partition of collision region;
2 Generate a set of sample data based on the partition;
3 Building a proper rule base using the sample data;
4 for each robot  $r_i$  do
5     Initialization: set  $k = 0, \mathbf{p}_i(t) = \mathbf{p}_i^0$ , time step  $\Delta t$ ;
6     while  $\mathbf{p}_i(t_k) \neq \mathbf{p}_i^f$  do
7         if has intruders then
8              $\mathbf{v}_i(t_k) = \text{Algorithm 1}$ ;
9              $\mathbf{p}_i(t) = \mathbf{p}_i(t) + \mathbf{v}_i(t)(t - t_k), t \in [t_k, t_{k+1})$ ;
10             $t_{k+1} = t_k + \Delta t, k = k + 1$ ;
11        else
12            Move directly to its target with  $v_i^{ref}$ .

```

completing fuzzy inference, the robot computes the new velocity and updates its current velocity, the computation complexity is $O(1)$. Hence, the total complexity of Algorithm 1 is $O(N_r + f^q)$.

Algorithm 2 gives the distributed collision avoidance algorithm for a cooperative multirobot system. Sector partition and fuzzy rule generation are done offline before a robot starts to perform collision avoidance and move forward. The number of fuzzy rules depends on only the number of generated fuzzy sets and the number of divided sectors for each robot. With q divided sectors and f generated fuzzy sets, the maximal number of fuzzy rules is $(f + 1)^q - 1$. Each robot in a system is controlled with the same rule base. Lines 6–10 describe the motion control for each robot. Each robot executes this part in a distributed way by communicating with its neighbors to retrieve their current states. After initialization, at each time instant t_k , a robot updates its velocity based on Algorithm 1 (Line 8), then moves with the new velocity in the next time duration $[t_k, t_{k+1})$ (Line 9). When time elapses, the robot updates the current time instant and check whether it arrives at its target position. If there are no intruders in its sensing range, the robot will update its speed to v_i^{ref} and move directly to the target position (Line 12).

Note that since the rule generation depends on only the number of partitioned sectors of a robot’s collision region, Fuzzy-VO is suitable for systems with different numbers of robots, and even for a system changing the number of robots in real-time, such as deleting existing robots or adding new robots during the evolution of the system. However, Fuzzy-VO suffers from the challenge of Sim-to-Real transfer. The fundamental procedure in the proposed method is to generate a fuzzy rule base, which highly depends on the sampling data. Usually, the training data are collected from simulation since it is cost- and time-consuming to collect data from real-world execution. Due to the Sim-to-Real gap, a practical strategy is to fine-tune the rule base for real-world applications.

8. Simulation results

In this section, to demonstrate the efficiency of Fuzzy-VO, a set of simulations with different numbers of UAVs are conducted, equipped with generic odometry sensors, on *RotorS*, which is a micro air vehicle gazebo simulator with different multirotor models such as the AscTec Hummingbird, the AscTec Pelican, or the AscTec Firefly [42]. Figure 10 shows a snapshot of a simulation with 6 AscTec Firefly UAVs. In simulations, all robots are flying to their target positions from initial positions at the same

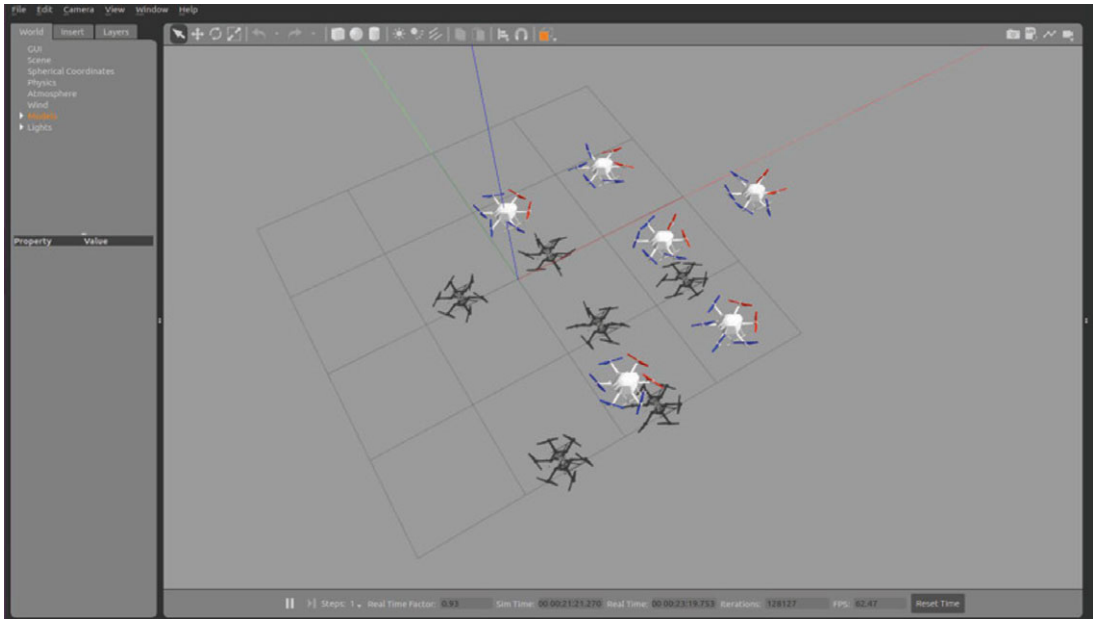


Figure 10. The simulation environment in Gazebo.

height. Each robot communicates with others through topic subscription and publication in ROS. In this way, the robot can subscribe to the states of others and publish its own state asynchronously in real-time. Even though assume that there are no other dynamic obstacles in the environment, each robot regards others as dynamic obstacles. Hence, the simulation results in all experiments can also extend to environments with various dynamic obstacles.

All simulation experiments are carried out on a Dell Precision Tower 5810 desktop running Ubuntu 16.04.6 LTS and ROS Kinetic, and equipped with Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60 GHz and 32.0 GB RAM. In experiments, the safe radius of each robot is $\rho = 0.55$ unit; t_1 and t_2 in Fig. 6(a) are equal to 1.2 and 8 s, respectively; the α_0 in Fig. 6(b) is 0.8; the initial speed of each robot is 2 unit/s; the sensing range $L = 8$ units; and the time step $\Delta = 0.01$ s, which satisfies the default publishing rate of topics in *RotorS*.

8.1. Multiple robots moving in the same environment

To evaluate the effectiveness of Fuzzy-VO, the first simulation experiment is conducted with 4 AscTec Firefly UAVs, denoted as $r_1 - r_4$. Their initial positions are $(0, 20)$, $(0, 0)$, $(20, 0)$, and $(20, 20)$, respectively. They need to move to $(20, 0)$, $(20, 20)$, $(0, 20)$, and $(0, 0)$, respectively.

Figure 11 shows some snapshots of the motion of the four UAVs at different time instants. As shown in Fig. 11(a), from the start time to the time instant $k = 342$ (i.e., $t = 3.42$ s), each UAV detects that there are no intruders within its sensing range, so they move directly to their targets. When the time instant is $k = 428$, r_3 arrives at $(14.135, 5.923)$ with a speed of $v_3 = 2$ units/s and an orientation $\theta_1 = 135.121^\circ$. At this time, it detects that there are two intruders in its sensing range, namely, r_2 and r_4 , whose positions are $(6.218, 6.331)$ and $(13.808, 13.748)$, respectively. With intruder selection, r_3 finds that r_2 is in its RR, while r_4 is in its LR. Thus, r_3 activates rules 15 – 18 to generate the motion command. The motion command generated by Fuzzy-VO for r_3 at this time is: $\alpha = 0.285$, $\Delta\theta = 43.423^\circ$. Hence, as shown in Fig. 11(b) and (c), r_3 changes its motion to right from $k = 428$ to $k = 449$. Similarly, during the motion from $k = 449$ to $k = 474$, r_2 , r_1 , and r_4 detect intruders in their sensing ranges sequentially, so they turn a little bit to right to avoid collisions, as shown in Fig. 11(d). At $k = 474$ ($t = 4.74$ s), r_4 is

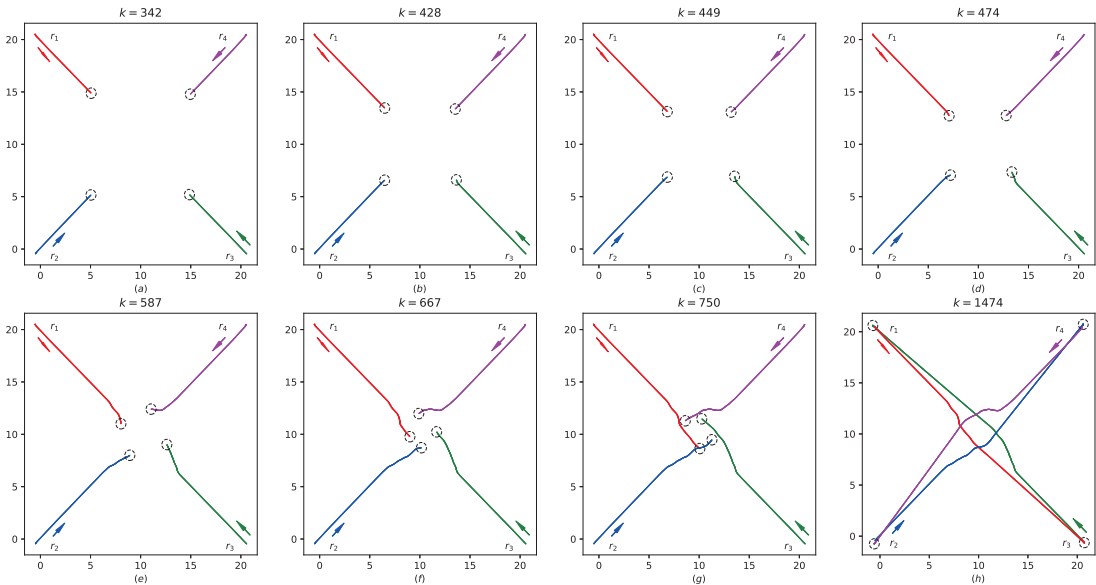


Figure 11. Detailed simulation process of collision avoidance for four UAVs.

at (12.787, 12.756) and detects r_1 , r_2 , and r_3 , whose positions are (7.141, 12.635), (7.201, 7.049), and (13.546, 6.924), respectively, are in its sensing range, so there are three intruders need to address for r_4 . Clearly, r_1 is in RR, r_2 is in FR, and r_3 is in LR. Since there is one intruder in each sector, rules 19 – 26 are activated. The generated command is: $\alpha = 0.267$, $\Delta\theta = 82.5^\circ$. Hence, r_4 should turn greatly to right and slow down its speed, as shown by the paths in Fig. 11(e). At $t = 5.87$ s, r_2 detects r_3 is in its FR, so rules 3 and 4 are activated to generate command ($\Delta\theta = 23.279^\circ$) for r_2 , which brings r_2 to turn right. After $t = 5.87$ s, r_4 does not detect any intruder in its sensing range, so r_4 only needs to move towards the target position. Such motion of them can be found in Fig. 11(f). When $k = 667$, that is, $t = 6.67$ s, the orientation of the four UAVs are -41.618° , 48.85° , 140.068° , and -129.561° , respectively. For r_3 and r_4 , the initial orientation is 135° and -135° , respectively. Based on the motion shown in Fig. 11(f) and (g), from $t = 6.67$ s to $t = 7.50$ s, each robot turns a little bit to right first to avoid collisions with other robots. When there are no intruders in its sensing range, each robot turns to left to move directly to its target.

As time elapses, the system reaches Fig. 11(g) at $t = 7.50$ s. From now on, even through there are others UAVs within their sensing ranges, the four UAVs detect that there are no collisions in the future, and all potential collisions are resolved. Hence, each UAV moves directly to its target again. As shown in Fig. 11(h), all robots reach their targets at $t = 14.74$ s. From their traversed trajectories, an observation is that in order to pass the intersection without any collision, each robot moves to its right side to avoid collisions. From their traversed trajectories, it is can be seen that in order to pass the intersection without any collision, each robot moves to its right side to avoid collisions. As in the ground traffic system, such motion of these robots is similar to the motion within a roundabout [16].

Furthermore, more simulations are also conducted with 3, 4, 6, and 8 UAVs, respectively. All videos of simulations can be found at <https://fuzzyvo.github.io/>. It can be seen that Fuzzy-VO can serve as an effective and universal collision avoidance method, for multirobot systems with different numbers of robots, by leveraging the advantages of fuzzy rules and VOs.

8.2. Comparison of Fuzzy-VO with other fuzzy rule-based methods

Since our method improves fuzzy rule-based methods, which can deal with uncertainties and linguistic variables, we compare Fuzzy-VO with the pure fuzzy rule method on the number of rules and the

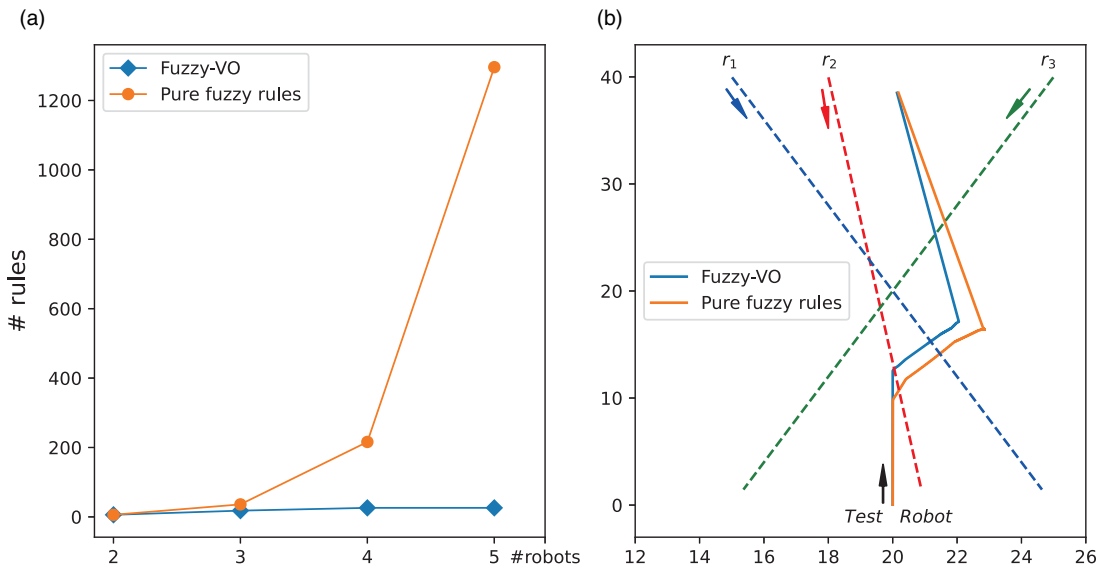


Figure 12. The number of rules and the generated paths of Fuzzy-VO and pure fuzzy rule method. The dotted lines indicate the paths of the dynamic obstacles.

smoothness of the generated paths. In addition, more experiments are also conducted to compare the performance of the proposed intruder selection method with other selection methods.

8.2.1. Comparison with pure fuzzy rule method

Let us first consider the number of rules. For the pure fuzzy control method, at any time instant, a robot evaluates the risks related to all other robots and then determines its motion. Hence, the rules are generated directly based on the number of robots in a system. Indeed, in this method, the number of rules increases exponentially with the number of robots in the system. Suppose the risk related to each robot is described by two fuzzy sets: dangerous and safe, and the orientation is described by three fuzzy sets: left, front, and right. If there are M robots in a system, the possible rules are 6^{M-1} . For example, as shown in Fig. 12(a), when there are three, four, and five robots, respectively, the required numbers of rules is 36, 261, and 1296, respectively. However, in Fuzzy-VO, by intruder selection, the number of rules is fixed and does not change when the number of robots is larger than 3. As shown in Fig. 12(a), when there are two robots in the system, there is one obstacle to be processed, so only six rules required for both Fuzzy-VO and pure fuzzy rule control. There are 18 rules when the system contains three robots. When $M \geq 4$, the number of rules in Fuzzy-VO is always 26 (without considering the emergent situation described before), which is independent of the number of robots. The result indicates that Fuzzy-VO decreases the number of fuzzy rules significantly by selecting a proper and fixed number of intruders.

Let us further compare the paths generated by the two methods. Consider the situation that there are three dynamic obstacles ($r_1 - r_3$) and a *Test Robot*. The *Test Robot* is placed to (20, 0) with an initial speed is 2 units/s, and its target position is (20, 40). The three obstacles' initial positions are (15, 40), (18, 40), and (25, 40), respectively; their missions are that $r_1 - r_3$ need to move directly to (25, 0), (21, 0), and (15, 0), respectively, with a fixed speed 2 units/s. Figure 12(b) shows the paths that the robots traversed. The solid lines represent the paths of the *Test Robot*. From the paths, an observation is that the path generated by Fuzzy-VO is smoother. Indeed, in pure fuzzy rule method, due to the excessive considering of the larger number of fuzzy rules, some unnecessary rules are activated to generate motion sometimes, which make the generated path oscillating. The result validates that the path of a robot may

be oscillating when there are a large number of rules. Since it generates a proper number of rules, Fuzzy-VO can produce smoother paths.

8.2.2. Performance comparison of different selection methods

The common metrics to assess collision risks in a dynamic environment include: distance to collision (DTC), time to collision (TTC), and time to react (TTR) [43]. As mentioned before, this paper focuses on TTC, that is, *potential collision time* in Fuzzy-VO, to select the most dangerous intruder in each sector. In some scenarios, a robot can only determine the nearest obstacle considered in each region using its onboard sensor information directly, such as that of ultrasonic sensors and laser sensors. Thus, the collision risk is evaluated based on DTC, such as [21]. For a deeper exploration of Fuzzy-VO, the performance comparison between the two methods is studied. The systems are initialized with 3, 4, 5, 6, 8, and 10 robots, respectively, using Monte Carlo simulation. For each system, 10,000 experiments are performed. In each experiment, the initial and target positions for each robot are randomly assigned. Then, Fuzzy-VO and the DTC-based method (the configurations of fuzzy sets, membership functions, and fuzzy rules are the same as [21]) are run independently. The comparison metrics are as follows.

1. *Success rate.* It is defined as the ratio of the number of experiments in which each robot arrives at its target position successfully while maintaining a safe distance with other robots at any time instant.
2. *Minimum separation.* It is defined as the minimum distance between any two robots during their motion in each collision-free experiments.
3. *Trajectory length.* It is defined as the path length of a robot. Due to different initial and target positions, the trajectory length is normalized by the distance between each pair of initial and target positions. Hence, trajectory length can be computed as follows.

$$L_{tr}(i) = \frac{\sum_k \|\mathbf{p}_i(k+1) - \mathbf{p}_i(k)\|_2}{\|\mathbf{p}_i^o - \mathbf{p}_i^f\|_2} \quad (11)$$

4. *Motion time.* It is defined as the discrete steps performed from the initial position to the target position of a robot.

For each system, the mean of each metric and the comparison results are shown in Fig. 13. As shown in Fig. 13(a), with the increase of robots, DTC-based selection cannot be applied anymore since most of the experiments are in collisions, while Fuzzy-VO is still with a low collision rate. In addition, from the average statistics aspect, compared to the DTC-based selection, the average success rate of Fuzzy-VOs can be increased by 306.5%. Note that the reasons for the happening of collisions in Fuzzy-VO are that: (1) since the initial and target positions are generated randomly, they may be in collisions when they are spawned; (2) currently, there are only two fuzzy sets associated to the collision time in Fuzzy-VO, so a robot in some situations may not be able to respond to collisions timely; it can be addressed by refining the collision time with more fuzzy sets; however, the more fuzzy sets, the more fuzzy rules; further study on how to balance the number of fuzzy sets and motion performance is one of future directions. Figure 13(b) shows that Fuzzy-VO can deal with collisions more precisely since it generates lower minimum separation in the context of collision avoidance. Figure 13(c) and (d) show that Fuzzy-VO can generate lower trajectory length and shorter motion time to complete the motion tasks.

Additionally, the variances of minimum separation, trajectory length, and steps are shown as in Table II. It can be found that in all experiments, Fuzzy-VO can generate a smaller variance than fuzzy rules with DTC. Hence, it can be concluded that the proposed intruder selection method can assess the collision risk accurately and guarantee a higher success rate.

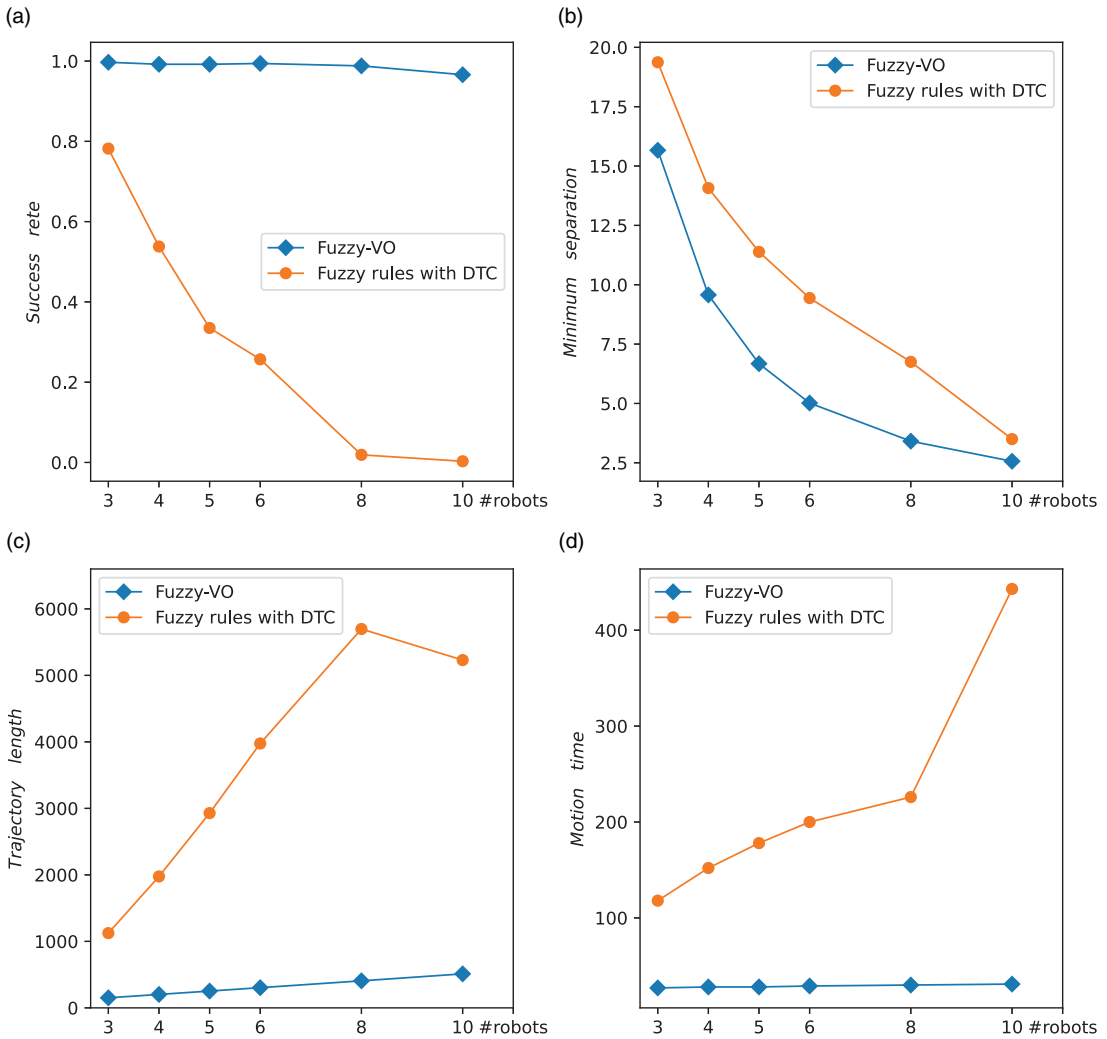


Figure 13. Comparison of motion performance between Fuzzy-VO and DTC-based method.

8.3. Comparison of Fuzzy-VO with ORCA

At last, the comparison between Fuzzy-VO and ORCA [14] is carried out. It aims to investigate the real-time decision efficiency of Fuzzy-VO. The experiments on six systems are conducted with different robots, that is, 3, 4, 5, 6, 8, and 10 robots, respectively. For each system, running Fuzzy-VO and ORCA 10,000 times, respectively. Figure 14(a) shows different methods' average success rates under the control of the two methods. It can be observed that even though the success rate decreases as the number of robots vary from 3 to 10, Fuzzy-VO can remain a high success rate. Note that in Fuzzy-VO, it is a trade-off between the number of selected intruders and the collision avoidance performance. If the number of selected intruders is too large, the computation cost will increase, and the smoothness of generated trajectory will decrease. If the selected intruders are too few, the generated velocities may cause collisions. Hence, sometimes, users need to determine an appropriate number of collision regions.

To further compare the real-time performance, the average one-step decision time (average time consumed for generating one candidate velocity) of the two methods is calculated. The results are shown in Fig. 14(b). From the results, it can be found that Fuzzy-VO has a shorter average decision time at each

Table II. Comparison of the variances of minimum separation, trajectory length, and steps.

# robots	Method	Minimum separation	Trajectory length	# steps
3	Fuzzy-VO	126.7	1.472×10^3	5.146×10^1
	Fuzzy rules with DTC	134.9	5.351×10^5	1.619×10^4
4	Fuzzy-VO	44.64	2.414×10^3	5.171×10^1
	Fuzzy rules with DTC	53.99	4.692×10^6	2.382×10^4
5	Fuzzy-VO	6.967	2.293×10^3	2.181×10^1
	Fuzzy rules with DTC	44.36	1.934×10^6	6.077×10^3
6	Fuzzy-VO	12.97	3.318×10^3	6.088×10^2
	Fuzzy rules with DTC	25.05	5.936×10^6	1.316×10^4
8	Fuzzy-VO	4.377	4.546×10^3	2.565×10^1
	Fuzzy rules with DTC	10.71	9.890×10^6	1.256×10^4
10	Fuzzy-VO	2.174	6.307×10^3	8.043×10^1
	Fuzzy rules with DTC	7.843	1.216×10^6	9.748×10^3

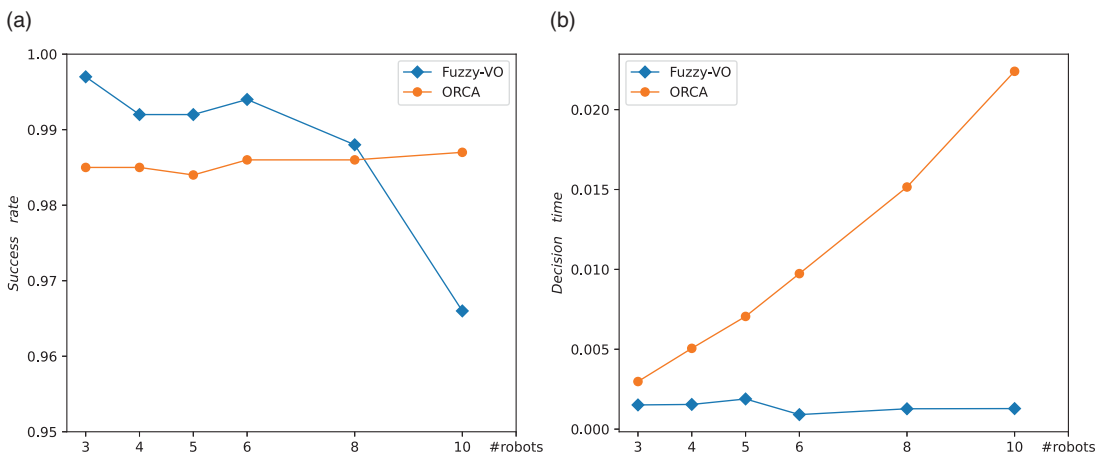


Figure 14. Comparison of avoidance performance between Fuzzy-VO and ORCA.

time step. With the increase of the number of robots, Fuzzy-VO can filter more nonurgent robots and has a less average one-step decision time for each robot. But the decision time of ORCA method increases with the number of robots in a system, as ORCA makes decision by resolving an optimization problem considering all other robots. The average one-step decision time of Fuzzy-VO is reduced by 740.9%, compared with the ORCA. This comparison results also further validate that Fuzzy-VO can improve the real-time decision efficiency without losing much success rate.

9. Conclusion

In this paper, a distributed and hybrid method, Fuzzy-VO, for collision avoidance in multirobot systems is proposed. The proposed method leverages the advantages of fuzzy rules and VOs simultaneously. It can deal with uncertain data and linguistic variables and also guarantee both safety and computation efficiency. It is suitable for multirobot systems with different numbers of robots. In the method, a fixed form and number of fuzzy rules are generated via collision region partition and intruder selection. To guarantee safety, Fuzzy-VO applies VO-based fine-tuning to further check and adjust the velocity generated by fuzzy inference. Experimental results with different scenarios show a 306.5% improvement in

success rate and a 740.9% reduction in decision time, demonstrating the effectiveness of the proposed method.

Future work includes implementing Fuzzy-VO on real robot platforms to further investigate the performance of the proposed method. Another interesting topic is to study the combinations of other model-driven methods and data-driven methods and compare their performance and scopes of application.

Supplementary material. To view supplementary material for this article, please visit <https://doi.org/10.1017/S0263574722001515>.

Author Contributions. All authors conceived and designed the study. Wenbing Tang and Yuan Zhou implemented the research and wrote the manuscript. Zuohua Ding and Jing Liu developed the fuzzy-logic controller. Tianwei Zhang and Yang Liu reviewed and edited the manuscript.

Financial Support. This work was supported by the Natural Science Foundation of China under Grant Nos. 61751210, 61210004 and 61170015, Academic Research Fund Tier 2 by Ministry of Education in Singapore under Grant No. MOE-T2EP20120-0004.

Conflicts of Interest. The authors declare that there is no conflict of interests.

Ethical Approval. None.

References

- [1] L. Jin, Y. Qi, X. Luo, S. Li and M. Shang, "Distributed competition of multi-robot coordination under variable and switching topologies," *IEEE Trans. Autom. Sci. Eng.* **19**(4), 3575–3586 (2022). doi: [10.1109/TASE.2021.3126385](https://doi.org/10.1109/TASE.2021.3126385).
- [2] Z. Zhou, J. Liu and J. Yu, "A survey of underwater multi-robot systems," *IEEE-CAA J. Autom. Sin.* **9**(1), 1–18 (2022).
- [3] N. Nfaileh, K. Alipour, B. Tarvirdizadeh and A. Hadi, "Formation control of multiple wheeled mobile robots based on model predictive control," *Robotica* **40**(9), 1–36 (2022).
- [4] L. Zhou and P. Tokekar, "Active target tracking with self-triggered communications in multi-robot teams," *IEEE Trans. Autom. Sci. Eng.* **16**(3), 1085–1096 (2019).
- [5] K. Brown, O. Peltzer, M. A. Sehr, M. Schwager and M. J. Kochenderfer, "Optimal Sequential Task Assignment and Path Finding for Multi-Agent Robotic Assembly Planning," **In: IEEE International Conference on Robotics and Automation, Pairs, France** (2020) pp. 441–447.
- [6] S. H. Jazi, M. Keshmiri, F. Sheikholeslam, M. G. Shahreza and M. Keshmiri, "Adaptive manipulation and slippage control of an object in a multi-robot cooperative system," *Robotica* **32**(5), 783–802 (2014).
- [7] P. Yu and D. V. Dimarogonas, "Distributed motion coordination for multirobot systems under LTL specifications," *IEEE Trans. Robot.* **38**(2), 1047–1062 (2022).
- [8] Y. Kantaros, M. Malencia, V. Kumar and G. J. Pappas, "Reactive Temporal Logic Planning for Multiple Robots in Unknown Environments," **In: IEEE International Conference on Robotics and Automation, Pairs, France** (2020) pp. 11479–11485.
- [9] Y. Zhou, H. Hu, Y. Liu and Z. Ding, "Collision and deadlock avoidance in multirobot systems: A distributed approach," *IEEE Trans. Syst. Man Cybern. Syst.* **47**(7), 1712–1726 (2017).
- [10] Y. Zhou, H. Hu, Y. Liu, S.-W. Lin and Z. Ding, "A distributed approach to robust control of multi-robot systems," *Automatica* **98**(6), 1–13 (2018).
- [11] Y. Zhou, H. Hu, Y. Liu, S.-W. Lin and Z. Ding, "A distributed method to avoid higher-order deadlocks in multi-robot systems," *Automatica* **112**, 108706:1–108706:13 (2020).
- [12] H. G. Tanner and A. Boddu, "Multiagent navigation functions revisited," *IEEE Trans. Robot.* **28**(6), 1346–1359 (2012).
- [13] J. Van den Berg, M. Lin and D. Manocha, "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation," **In: IEEE International Conference on Robotics and Automation, Pasadena, California, USA** (2008) pp. 1928–1935.
- [14] J. van den Berg, S. J. Guy, M. Lin and D. Manocha, "Reciprocal n-body collision avoidance," *Robot. Res.* **70**, 3–19 (2011).
- [15] B. Lindqvist, S. S. Mansouri, A.-A. Agha-Mohammadi and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robot. Autom. Lett.* **5**(4), 6001–6008 (2020).
- [16] Y. Zhou, H. Hu, Y. Liu, S.-W. Lin and Z. Ding, "A real-time and fully distributed approach to motion planning for multirobot systems," *IEEE Trans. Syst. Man Cybern. Syst.* **49**(12), 2636–2650 (2019).
- [17] B. Tang, K. Xiang, M. Pang and Z. Zhanxia, "Multi-robot path planning using an improved self-adaptive particle swarm optimization," *Int. J. Adv. Robot. Syst.* **17**(5), 1–19 (2020).
- [18] N. Thumiger and M. Deghat, "A multi-agent deep reinforcement learning approach for practical decentralized UAV collision avoidance," *IEEE Control Syst. Lett.* **6**, 22174–22179 (2022).
- [19] R. Han, S. Chen, S. Wang, Z. Zhang, R. Gao, Q. Hao and J. Pan, "Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards," *IEEE Robot. Autom. Lett.* **7**(3), 5896–5903 (2022).

- [20] J. H. Lilly, "Evolution of a negative-rule fuzzy obstacle avoidance controller for an autonomous vehicle," *IEEE Trans. Fuzzy Syst.* **15**(4), 718–728 (2007).
- [21] Z. M. Wen, S. D. Zhou and M. Wang, "Fuzzy control for the obstacle avoidance of a quadrotor UAV," *Appl. Mech. Mater.* **775**, 307–313 (2015).
- [22] T. Edward, "Mobile Robot Autonomy via Hierarchical Fuzzy Behavior Control," **In: International Symposium on Robotics and Manufacturing**, Montpellier, France (1996) pp. 837–842.
- [23] C. Wang, A. V. Savkin and M. Garratt, "A strategy for safe 3D navigation of non-holonomic robots among moving obstacles," *Robotica* **36**(2), 275–297 (2018).
- [24] C. Kownacki and L. Ambroziak, "A new multidimensional repulsive potential field to avoid obstacles by nonholonomic UAVs in dynamic environments," *Sensors* **21**(22), 7495 (2021).
- [25] D. F. Llorca, V. Milanés, I. P. Alonso, M. Gavilán, I. G. Daza, J. Pérez and M.Á. Sotelo, "Autonomous pedestrian collision avoidance using a fuzzy steering controller," *IEEE Trans. Intell. Transp. Syst.* **12**(2), 390–401 (2011).
- [26] P. Vadakkepat, O. C. Miin, X. Peng and T. H. Lee, "Fuzzy behavior-based control of mobile robots," *IEEE Trans. Fuzzy Syst.* **12**(4), 559–565 (2004).
- [27] Y.-C. Chang, Y. Shi, A. Dostovalova, Z. Cao, J. Kim, D. Gibbons and C.-T. Lin, "Interpretable fuzzy logic control for multirobot coordination in a cluttered environment," *IEEE Trans. Fuzzy Syst.* **29**(12), 3676–3685 (2021).
- [28] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments Using the Relative Velocity Paradigm," **In: IEEE International Conference on Robotics and Automation**, Atlanta, GA, USA (1993) pp. 560–565.
- [29] M. Kim and J.-H. Oh, "Study on optimal velocity selection using velocity obstacle (OVVO) in dynamic and crowded environment," *Auton. Robot.* **40**(8), 3676–3685 (2016).
- [30] K. Cai, C. Wang, J. Cheng, C. W. De Silva and M. Q. H. Meng, "Mobile robot path planning in dynamic environments: A survey," arXiv preprint arXiv: 2006.14195, (2020).
- [31] Y. I. Jenie, E.-J. Kampen, C. C. de Visser, J. Ellerbroek and J. M. Hoekstra, "Selective velocity obstacle method for deconflicting maneuvers applied to unmanned aerial vehicles," *J. Guid. Control Dyn.* **38**(6), 1140–1146 (2015).
- [32] S. Wang, Z. Li, B. Wang, J. Ma and J. Yu, "Velocity obstacle-based collision avoidance and motion planning framework for connected and automated vehicles," *Transp. Res. Rec.* **2676**(5), 748–766 (2022).
- [33] J. A. Douthwaite, S. Zhao and L. S. Mihaylova, "Velocity obstacle approaches for multi-agent collision avoidance," *Unmanned Syst.* **7**(1), 55–64 (2019).
- [34] M. Shen, Y. Wang, Y. Jiang, H. Ji, B. Wang and Z. Huang, "A new positioning method based on multiple ultrasonic sensors for autonomous mobile robot," *Sensors* **20**(1), 237–252 (2019).
- [35] G. Liu, M. Yao, L. Zhang and C. Zhang, "Fuzzy Controller for Obstacle Avoidance in Electric Wheelchair with Ultrasonic Sensors," **In: International Symposium on Computer Science and Society**, Kota Kinabalu, Malaysia (2011) pp. 71–74.
- [36] Z. Ding, Y. Zhou and M. Zhou, "Modeling self-adaptive software systems by fuzzy rules and Petri nets," *IEEE Trans. Fuzzy Syst.* **26**(2), 967–984 (2018).
- [37] V. Kreinovich, O. Kosheleva and S. N. Shahbazova, "Why triangular and trapezoid membership functions: A simple explanation," *Recent Dev. Fuzzy Logic Fuzzy Sets* **391**, 25–31 (2020).
- [38] S. Ping and Z. Yu, "Tracking control for a cushion robot based on fuzzy path planning with safe angular velocity," *IEEE-CAA J. Autom. Sin.* **4**(4), 610–619 (2017).
- [39] Z. Ding, Y. Zhou, G. Pu and M. Zhou, "Online failure prediction for railway transportation systems based on fuzzy rules and data analysis," *IEEE Trans. Reliab.* **67**(3), 1143–1158 (2018).
- [40] M. Figueiredo, F. Gomide, A. Rocha and R. Yager, "Comparison of Yager's level set method for fuzzy logic control with Mamdani's and Larsen's methods," *IEEE Trans. Fuzzy Syst.* **1**(2), 156–159 (1993).
- [41] T. Jiang and Y. Li, "Generalized defuzzification strategies and their parameter learning procedures," *IEEE Trans. Fuzzy Syst.* **4**(1), 64–71 (1996).
- [42] F. Furrer, M. Burri, M. Achtelik and R. Siegwart, "Rotors—A modular gazebo MAV simulator framework," *Robot Oper. Syst.* **625**, 595–625 (2016).
- [43] C. Katrakazas, M. Qudus, W.-H. Chen and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. Part C Emerg.* **60**, 416–442 (2015).