Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

# Realization of Consensus with Collision and Obstacle Avoidance in an Unknown Environment for Multiple Robots

Simon Wasiela, Nitin Kasshyap, Ya-Jun Pan, *Senior Member, IEEE* and Joshua Awe

*Abstract*— **In this paper, we present a fuzzy logic-based control approach for the collision avoidance of a multi-robot system. A simple and continuous-time consensus law is implemented to achieve the rendezvous of the multiple mobile robot agents. A fuzzy logic-based controller is designed over a potential field method, as fuzzy algorithms are often robust and are not very sensitive to changing environments. Fuzzy logic is also useful for unknown and semi-unstructured environments. The algorithms are applicable to a general N-robot system and they are applied to three robots which are available in the host lab. Detailed information on the real-time implementation of the algorithm on three Pioneer mobile robots are presented. We have included simulations and experiments carried out to verify the effectiveness of our approach in this paper.**

## I. INTRODUCTION

In the last decade, researchers have taken a strong interest in cooperative control of distributed multi-agent systems (MASs) [1]. MASs is a group of several intelligent agents connected and communicating with each other. As one of the examples on the consensus of MASs as in [2], the consensus control for quadcopters is addressed. In this paper we focus on the consensus on a specific MASs for a multi-mobile-robot system with collision and obstacle avoidance.

In fact, nowadays in industry as in research, the tasks to be performed by a robot can be complex or long where they are simpler when they were divided into smaller tasks between several robots. The interest of multi-robot system (MRS) appears: a system consisted of multiple robots which can cooperate and communicate with each other to accomplish certain tasks. However, real-time implementation is still the main challenge for cooperative multi-robot systems.

The communication and coordination between agents when performing a task remains complicated in real time due to latency or the control architecture (centralization and distribution)[3]. Since multi-agent robots need to coordinate and communicate, the question of finding a consensus caught the attention of researchers. Consensus refers to the degree of agreement between multi-agents to achieve certain quantities of interest. The main problem of consensus in MASs is to design a distributed protocol using local information that can guarantee agreements between robots. Some researchers conducted research on consensus building in the field of broadcasting. As in [4], Azuma et al studied the problem of consensus with a limited communication range and an unlimited diffusion range and introduced a concept of groups of connected agents.

This work was supported by NSERC Canada.

Authors are with the Department of Mechanical Engineering, Dalhousie University, Halifax, Canada, B3H 4R2 (Email:yajun.pan@dal.ca).

In this paper, we propose an application of a consensus algorithm in an unknown environment to three Pioneer robots available in the host lab. This algorithm is based on a consensus calculation phase and then an environment detection and displacement phase. In order to make this detection and avoidance possible, we have built new fuzzy logic rules for multiple robots, in our case applied to three Pioneer robots, extendable to N robots within the software limit.

## II. SYSTEM DESCRIPTION

In this section, the system used for testing is described.

### A. Pioneer Robots

One of the robots used for testing is the Pioneer 3-DX, which is a mobile robot platform intended for robotics research. The Pioneer 3-DX (P3-DX) uses a differential drive dynamic model which consists of left and right drive wheels and consists of a castor wheel centrally located at the back of the robot. The dynamic model of the robot means that it can only move linearly in one axis which is perpendicular to the axis that goes through the center of the drive wheels. The maximum linear speed of the robot is 1.2m/s and the maximum angular speed is 300 degrees/s; the maximum payload is 17kg [$www.generationrobots.com/en/402395-robot-mobile-pioneer-3-dx.html$]. For perception, the robot consists of front and rear sonar rings.

The other robot used for testing is the versatile Pioneer 3-AT (P3-AT), which is also a mobile robot platform intended for indoor and outdoor robotics research. The robot uses a skid-steering dynamic model which consist of four wheels. The robot has similar kinematic constraints as P3-DX. The maximum linear speed is 0.7m/s and the maximum angular speed is 140 degrees/s; the maximum payload on a level surface is 12kg [$www.generationrobots.com/en/402397-robot-mobile-pioneer-3-at.html$]. For perception purposes, the Pioneer P3-AT only consists of a front sonar sensor ring. Both type of robots used in the testing for this paper are also compatible with the Robotic Operating System (ROS) as explain in II-B.

### B. ROS Environment

*1) Usefull Packages:* Despite its name, ROS is not actually an operating system but rather a free and open source framework to develop software for robots. ROS runs on Unix-based operating systems specifically Ubuntu Linux. Services such as hardware-abstraction, low level device control, implementation of commonly used functionality, message-passing between processes, and package management are what make ROS an effective system for robotics research [5][$wiki.ros.org/ROS/Introduction$].

ROSARIA is a node on ROS used to control P3-DX and P3-AT. The node is an interface between ROS and the open source ARIA library. ROSARIA includes many topics, services and parameters which can be used to receive and send data. For the tests presented in this paper, three topics have been used. To send velocity commands to the robots, the $geometry\_msgs/Twist$ messages are published to the $cmd\_vel$ topic. To obtain information about the position and orientation about the robot, $nav\_msgs/Odometry$ are subscribed to and from the pose topic. To obtain the range data from sonar sensors, $sensor\_msgs/PointCloud$ messages are subscribed to from the sonar topic [$wiki.ros.org/ROSARIA$].

*2) Robots Set up:* As for the connection of the robots, we connect them via USB to a computer that we load on the robot itself. In the case of connecting a single robot to use an algorithm, the ROSARIA package is sufficient. On the other hand, when an algorithm is intended for several robots, each robot is then equipped with a computer. One of them is defined as the master of the ROS nodes, executing the main algorithm. It communicates new commands to others and receives information from other robots such as positions or sonar information via wifi. In order to allow these connections, each robot must be associated with a node with a different name and connected to a single port. Thus, we have created launch files associating a port and a node name to each robot when they are connected.

## III. Leaderless Consensus Algorithm and Application

### A. Graph Topology

A multi-agent system is said to reach consensus when all agents agree on the value of a variable of interest  the information state  using an appropriate consensus algorithm. The communication topology of a multi-agent system consisting of N agents can be represented by a directed graph $\mathcal{G}_N \triangleq (\mathcal{V}_N, \mathcal{E}_N)$ where $\mathcal{V}_N = \{1, ..., N\}$ denotes the set of nodes and $\mathcal{E}_N \subseteq \mathcal{V}_N \times \mathcal{V}_N$ is the set of edges. Let $\mathcal{A}_N \in \mathbb{R}^{N \times N}$ be the adjacency matrix associated with $\mathcal{G}_N$.

The common continuous-time consensus algorithm is [1]:

$$\dot{x}_i = -\sum_{j=1}^{N} a_{ij}(t)[x_i(t) - x_j(t)], i = 1, ..., N, \quad (1)$$

where $a_{ij}$ is the $(i,j)^{th}$ entry of $\mathcal{A}_N$ and $x_i(t)$ is the information state of the $i^{th}$ vehicle at time $t$. Consensus is reached by the system if, for all $x_i(0)$ and all $i, j = 1, ..., N$, $\mid x_i(t) - x_j(t) \mid \to 0$ as $t \to \infty$.

The continuous-time consensus algorithm can be extended to make the difference in information states converge to desired values:

$$\dot{x}_i = -\sum_{j=1}^{N} a_{ij}(t)[x_i(t) - x_j(t) - \Delta_{ij}], i = 1, ..., N, \quad (2)$$

where $\Delta_{ij} \triangleq \delta_i - \delta_j, \forall i \neq j$ denotes the desired deviation or distance between the information states [6]. The system will reach consensus if zero is a simple eigenvalue of the Laplacian matrix $\mathcal{L}_N$ associated with $\mathcal{G}_N$. If the directed graph $\mathcal{G}_N$ is strongly connected, then zero is a simple eigenvalue of $\mathcal{L}_N$.

Three mobile robots used in our experiments are connected in the network topology shown in Fig.1. As the topology of the agents is strongly connected, the system will reach consensus. This can be generally applied to N robots as well while in this experiment, three robots are considered and available in the lab.

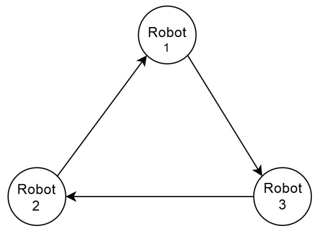


Fig. 1: Communication topology for three mobile robots

### B. Multi-level Control

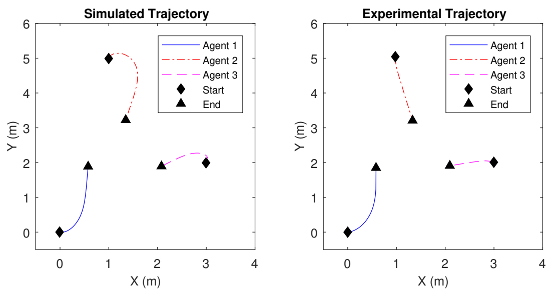


Fig. 2: Rendezvous trajectory of the three agents generated by their position controller

A high-level consensus controller first computes the continuous-time consensus of the position of the agents using (1) and determines the desired final position. Then a low-level position controller is used to navigate the individual agent from their initial to the desired final positions using a proportional controller:

$$\begin{aligned} \Delta\theta_i &= \tan^{-1}\left(\frac{r^d_{yi} - r_{yi}}{r^d_{xi} - r_{xi}}\right) - \theta_i \\ \Delta r_i &= \sqrt{(r^d_{xi} - r_{xi})^2 + (r^d_{yi} - r_{yi})^2} \\ v_i &= k_{vi}\Delta r_i \cos(\Delta\theta_i) \\ \omega_i &= k_{\omega i}\Delta\theta_i, \end{aligned}$$

where $[r_{xi}, r_{yi}, \theta_i]$ and $[r^d_{xi}, r^d_{yi}]$ are the initial pose and the final desired position of the $i^{th}$ agent with reference to the global coordinate frame.

Three agents with initial positions $r_1 = (0,0)^T$, $r_2 = (1,5)^T$, $r_3 = (3,2)^T$are used to implement the leaderless consensus experiment. The desired relative separation between the agents are chosen as $\delta_1 = (0,0)^T$, $\delta_2 = (0.75, 1.299)^T$, $\delta_3 = (1.5, 0)^T$. The trajectory followed by the agents navigating to achieve consensus is shown in Fig.2. The actual separation between the robots at the end of consensus experiment, is observed to deviate from the simulation due to tracking error caused by inaccurate odometry of the available Pioneer robots. Though the experimental trajectory seems to deviate from the simulated trajectory, the consensus reached by the system is not affected by this deviation. The system reaches the calculated final positions during the experiment, albeit in a different path.

### C. Hand Position Control

Since a differential-drive robot is a non-holonomic system, it cannot be stabilized with continuous static state feedback as suggested in [7]. The hand position is defined as the point $h \triangleq [h_x, h_y]^T$ that lies at a distance $L$ along the normal line intersecting the wheel axis at the centre point $r \triangleq [r_x, r_y]^T$ as shown in Fig.3. The experiment considers the problem of coordinating the hand positions of the robots instead of their center positions as this simplifies the control problem since the kinematics of the hand position are holonomic for $L \neq 0$.

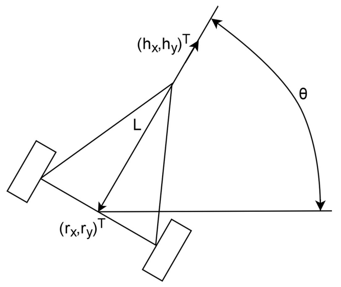


Fig. 3: Hand position, h, of the differential drive robot

Let $(r_{xi}, r_{yi})$, $\theta_i$, and $(v_i,\ \omega_i)$ denote the position, orientation, linear and angular speeds of the $i^{th}$ robot respectively. The hand position is given by

$$\begin{pmatrix} h_{xi} \\ h_{yi} \end{pmatrix} = \begin{pmatrix} r_{xi} \\ r_{yi} \end{pmatrix} + L_i \begin{pmatrix} \cos\theta_i \\ \sin\theta_i \end{pmatrix}. \tag{3}$$

On differentiating (3) w.r.t time,

$$\begin{pmatrix} \dot{h}_{xi} \\ \dot{h}_{yi} \end{pmatrix} = \begin{pmatrix} \cos\theta_i & -L_i \sin\theta_i \\ \sin\theta_i & L_i \cos\theta_i \end{pmatrix} \begin{pmatrix} v_i \\ \omega_i \end{pmatrix}. \tag{4}$$

Let

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} \cos\theta_i & \sin\theta_i \\ -\frac{1}{L_i}\sin\theta_i & \frac{1}{L_i}\cos\theta_i \end{pmatrix} \begin{pmatrix} u_{xi} \\ u_{yi} \end{pmatrix}. \tag{5}$$

Hence

$$\begin{pmatrix} \dot{h}_{xi} \\ \dot{h}_{yi} \end{pmatrix} = \begin{pmatrix} u_{xi} \\ u_{yi} \end{pmatrix}. \tag{6}$$

As Eq.(6) takes the form of a single integrator, Eq.(1) is directly applied to achieve a consensus of the hand position of the robots.

A hand position with $L = 0.1\ m$ is chosen for the experiment. The initial positions of the robots are $r_1 = (0,0)^T$, $r_2 = (1,5)^T$, $r_3 = (3,2)^T$ and the desired relative separation of the agents are again chosen as $\delta_1 = (0,0)^T$, $\delta_2 = (0.75, 1.299)^T$, $\delta_3 = (1.5, 0)^T$. The trajectory followed by the robots is shown in Fig.4. The separation between the hand positions at the end of consensus experiment is observed to deviate marginally from the simulation. The inaccurate odometry of the available Pioneer robots again causes this tracking error.

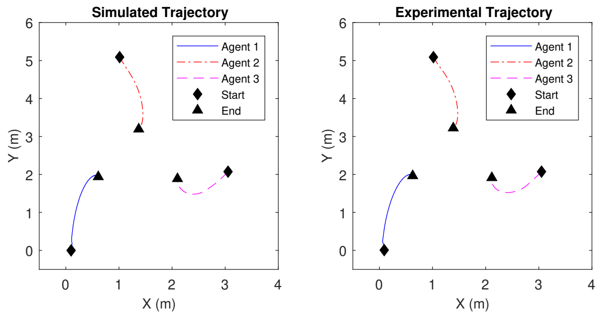


Fig. 4: Rendezvous trajectory of the hand position of the three agents

## IV. Real-Time Algorithm For Consensus with Avoidance

An algorithm was proposed to manage both obstacle and collision avoidance in an unknown environment. In this section, we will present the controller we have designed for each robot. This algorithm is based on the use of sonar for simplicity and minimal sensors.

### A. Sonar Sensors

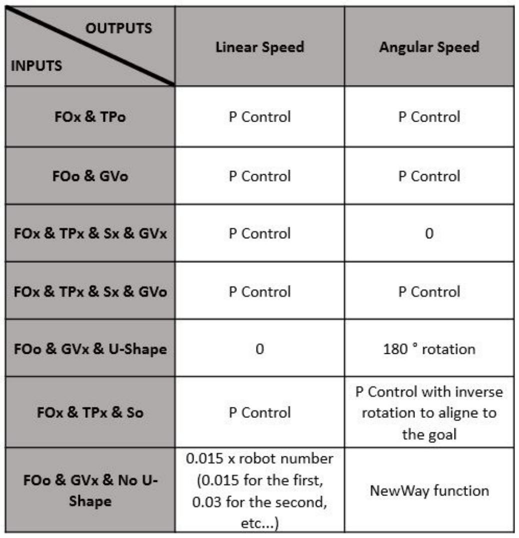
In order to allow the robot avoiding obstacles, a sonar-based detection was used. In fact, both P3-DX and P3-AT robots are equipped with sonar sensors positioned as described in [8]. P3-DX models are equipped with 16 sonars (front and rear) while the P3-AT model is only equipped with 8 front sonars. These sonars have a range of 10 centimetres to about 5 metres and are arranged so that they cover an arc of 180 degrees. However, since these sonars are fixed beams, they do not cover all possible angles at once.

It should be noted that in the real implementation, the sonars can read incorrect distances due to the reflection of the rays. Indeed, some obstacles that are too curved are not even detected from certain angles, while in the simulator the sonars have a perfect detection. This may explain differences in behaviour between the experiment testing and the simulation.

### B. Fuzzy Logic Based Controller

A fuzzy logic controller instead of a potential field algorithm was proposed. In fact, the advantages of the fuzzy logic approach are intuitive linguistic terms with a smaller computational load than conventional potential field algorithms. Fuzzy logic is also useful for unknown and semi-unstructured environments. Moreover fuzzy reasoning process are often simple, compared to computationally precise systems, so computing power is saved. This is a very interesting feature, especially for real time applications [9].

So our algorithm is based on a fuzzy logic controller where the inputs are the presence of front, side obstacles and the visibility of the goal. As for the outputs, they are the linear velocity and angular velocity of the robot. The speeds are defined by a P controller [10] when the robot can directly reach its goal, otherwise the linear speed is fixed and the angular speed is calculated according to the current position of the robot with respect to obstacles. Fig.5 shows the outputs/inputs rules of a controller.



| OUTPUTS / INPUTS | Linear Speed | Angular Speed |
|---|---|---|
| FOx & TPo | P Control | P Control |
| FOo & GVo | P Control | P Control |
| FOx & TPx & Sx & GVx | P Control | 0 |
| FOx & TPx & Sx & GVo | P Control | P Control |
| FOo & GVx & U-Shape | 0 | 180 ° rotation |
| FOx & TPx & So | P Control | P Control with inverse rotation to aligne to the goal |
| FOo & GVx & No U-Shape | 0.015 x robot number (0.015 for the first, 0.03 for the second, etc...) | NewWay function |

Fig. 5: Fuzzy rules table. **FOo**: Front Obstacle detected, **FOx**: No Front Obstacle detected, **TPo**: Turn Possible, **TPx**: Turn Impossible, **GVo**: Goal Visible, **GVx**: Goal Invisible, **So**: It's the start, **Sx**: It's not the start.

To determine the presence of side or front obstacles we used the bubble rebound algorithm described in [11]. We used the same obstacle detection method based on the creation of a safety region. However, we have defined a static safe region and have also separated the detection of front and side obstacles into two different functions. Thus, we can create two detection regions with different sensitivities, as shown in Fig.6, use a different number of sonars and can use the information received separately.

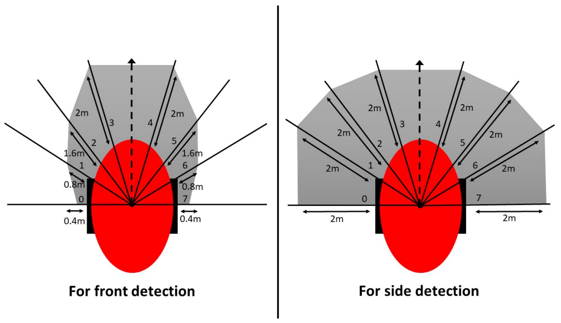
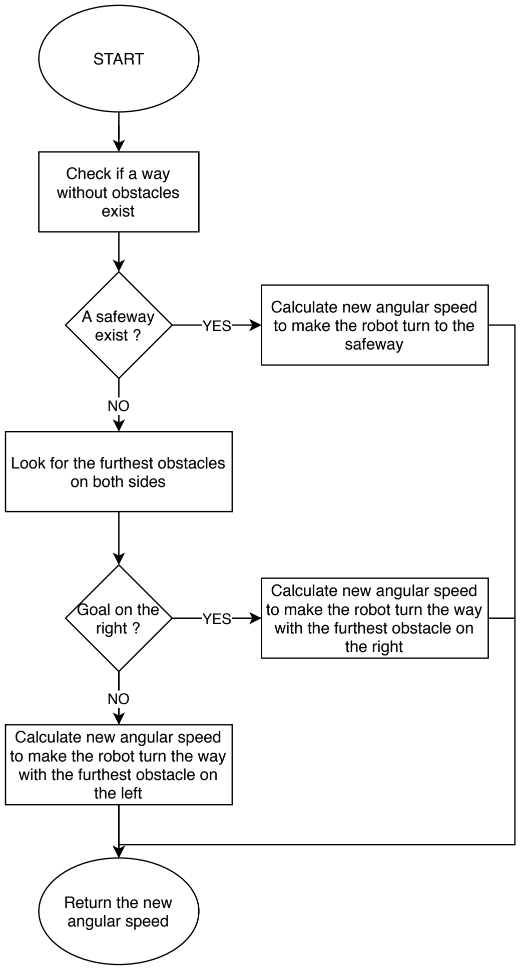


Fig. 6: Safety regions



START
Check if a way without obstacles exist
A safeway exist ?
YES
Calculate new angular speed to make the robot turn to the safeway
NO
Look for the furthest obstacles on both sides
Goal on the right ?
YES
Calculate new angular speed to make the robot turn the way with the furthest obstacle on the right
NO
Calculate new angular speed to make the robot turn the way with the furthest obstacle on the left
Return the new angular speed

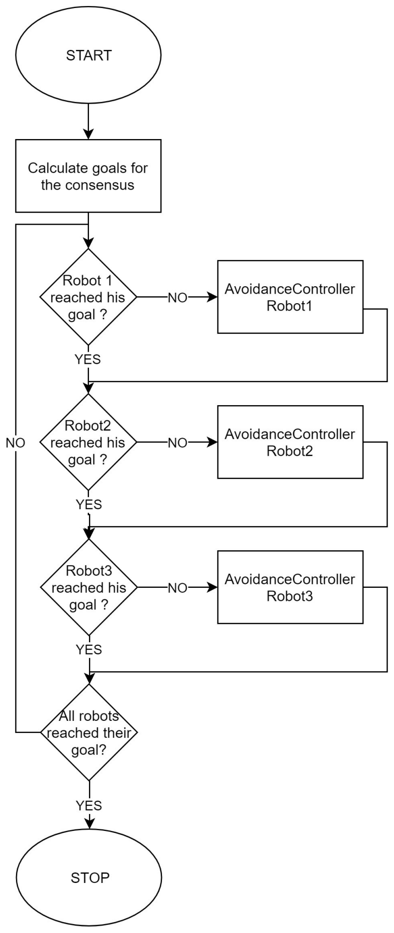
Fig. 7: Flowchart of the "NewWay" function

The purpose of our obstacle detection and avoidance functions is to ensure the most secure avoidance. Therefore, in calculating the new route, the robot will always try to pass through the side that has no obstacles if it exists. In the case that both sides present obstacles, the algorithm will make the robot passed by the way having the furthest obstacle in the side where the goal is located. The flowchart of a "NewWay" function is as shown in Fig.7.

In both cases, a new steering angle $\beta$ is calculated by looking both sides around the direction of the sonar who detected the obstacle and choosing the minimum angle to make this sonar free again as descirbed in [12].

### C. Cascaded Algorithm

The entire algorithm is then a cascading combination of the consensus calculation function (see III-B) and controllers of each robot in a C++ file. Indeed, we start by calculating the final positions of each robot based on the knowledge of their initial positions made possible by the ROS environment. Then, knowing their goals, we independently apply to each robot a version of the controller described in the previous part. Thus, at each iteration loop, each robot communicates its current position and sonar information and receives new speed instructions accordingly.



START
Calculate goals for the consensus
Robot 1 reached his goal ?
NO
AvoidanceController Robot1
YES
Robot2 reached his goal ?
NO
AvoidanceController Robot2
YES
Robot3 reached his goal ?
NO
AvoidanceController Robot3
YES
NO
All robots reached their goal?
YES
STOP

Fig. 8: Flowchart of the algorithm

As stated above, each robot has its own controller but executes identical functions as the other robots with different topic names as pose, sonar and velocity command. In addition, since we want robots to avoid static and dynamic obstacles, each controller of each robot has slightly different linear speed outputs in order to avoid that robots follow each other indefinitely while trying to avoid each other.
A multi-level algorithm is applied:

1) it calculates the consensus points for each robot;

2) it uses each controller of the robots that have not yet reached their final position;

3) each controller adjusts the robot speeds independently of the other controllers based on the information provided by its robot as shown in Fig.8.

## D. Simulation and Experimental Results

In the experiments, each robot is connected to its own computer connected by USB. The computer of Robot 1 is the master, executing the main algorithm and the other robots communicating their information via wifi to the master computer.

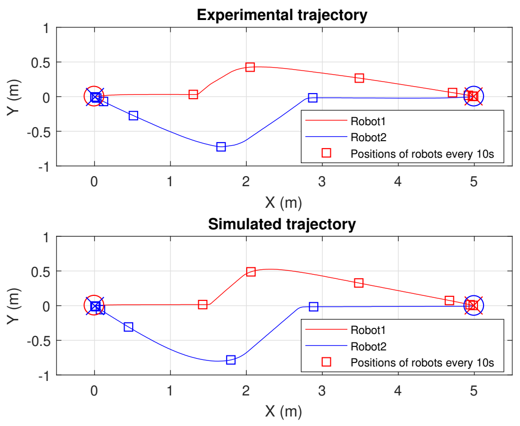


Fig. 9: Trajectory for 2 robots with collision avoidance, O: Starting pose, X: Desired goal

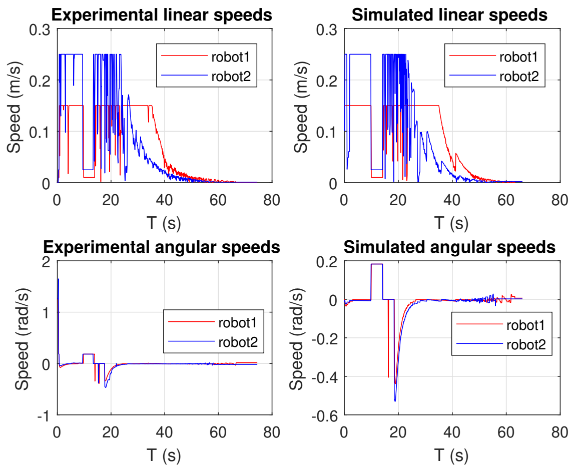
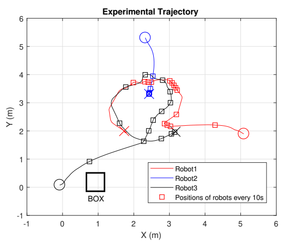


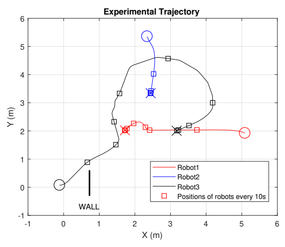
Fig. 10: Speeds for 2 robots with collision avoidance

**Case 1:** Collision Avoidance using 2 robots
First of all, we start with a collision avoidance experiment with 2 robots. The robots must exchange their starting position there, we can see that the robots avoid each other with the same trajectory (Fig.9) both in simulations and experimentations, and start to detect each other about 1.5 m away from each other.
Fig.10 shows the angular and linear velocities during the experiment and simulation. It can also be seen here that the speed curves between simulation and experiment are very similar except for the first points of the experimental curves. This is due to the latency in real conditions that slows down the arrival of the first speed control. On these curves, we can also distinguish linear speed limits and avoidance phases.
**Case 2:** Consensus with Obstacle and Collision Avoidance using 3 robots and 1 box
In this case, we perform an experiment with 3 robots to test the consensus with obstacle and collision avoidance as shown in Fig.11(a). The corresponding simulation of this experiment testing setup is as in Fig.11(b), where we can see that the trajectories are a bit different. Indeed, we notice

that Robot 1 is faster during the simulations and that Robot 3 is slower due to the shape of the obstacle which is different (impossibility to reproduce the box during the simulations). This implies that Robots 1 and 3 detect themselves closer to the goal of Robot 1, and on the left of Robot 2 leading to a different avoidance phase. Nevertheless, robots join their goal calculated by consensus without hitting obstacles or touching each other, which shows the effectiveness of the proposed avoidance algorithm.

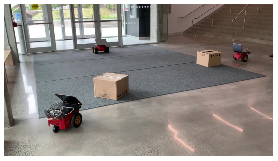


(a) Experimental trajectory



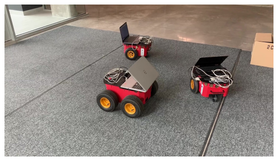
(b) Simulated trajectory

Fig. 11: Simulated and experimental trajectory for 3 robots in collisions avoidance, O: Starting pose, X: Desired goal

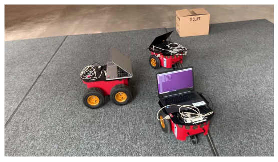


(a) Set up of the experiment

(b) Obstacle avoidance phase

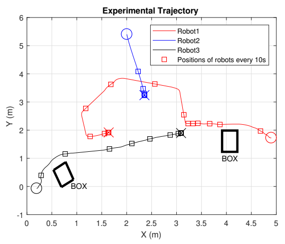


(c) Collision avoidance phase

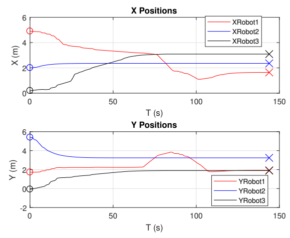


(d) End of the experiment

Fig. 12: Experimental setup



(a) Experimental trajectory of robots



(b) Experimental positions of robots

Fig. 13: Trajectory and positions of robots in experiments with 2 boxes and a collision avoidance, O: Starting pose, X: Desired goal

**Case 3:** Consensus with Obstacles and Collision Avoidance using 3 robots and 2 boxes

Finally, we perform another experiment by adding more obstacles as shown in Fig.12(a), the experimental trajectories are as shown in Fig.13(a). With the help of the robot positions Fig.13(b), the robots avoid each other with reasonable margins but also avoid the boxes. The robots reach their goals as shown in Fig.12(d).

## V. Conclusions

This paper presented the implementation of a real-time consensus algorithm with obstacle and collision avoidance. We decided to build our avoidance controller based on new fuzzy rules instead of using a potential field algorithm to provide a more robust control and a more suitable implementation for real time. Our algorithm has been successfully applied sequentially in many cases and environments with three pioneer robots but can be applied to any type and number of mobile robots equipped with sonars within software and hardware limits.

Using this configuration (sequential) with a connection via Wi-Fi and a laptop for each robot we found a latency of 200ms for each robot in our experiment. This suggests that in order to connect more robots, an application of the controllers in parallel would be more appropriate, as well as a connection on a local area network and a download of the program to the robot's embedded computer.

## References

[1] W. Ren and R. Beard, "Overview of consensus algorithms in cooperative control," in *Distributed consensus in multi-vehicle cooperative control: Theory and Applications*, pp. 3–22, Springer-Verlag London, 2008.

[2] Z. Huang, Y.-J. Pan, and R. Bauer, "Leaderfollower consensus control of multiple quadcopters under communication delays," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, pp. 1–10, 2019.

[3] Z. H. Ismail and N. Sariff, "A survey and analysis of cooperative multi-agent robot systems: Challenges and direction.," *Applications of Mobile Robots*, pp. 8–14, 2018.

[4] S. Azuma, Y. Tanaka, and T. Sugie, "Multi-agent consensus under a communication broadcast mixed environment.," *International Journal of Control*, vol. 87, no. 6, pp. 1103–1116, 2014.

[5] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: An open-source robot operating system.," *International Conference on Robotics and Automation*, 2009.

[6] W. Ren and R. Beard, "Consensus algorithms for single-integrator dynamics," in *Distributed consensus in multi-vehicle cooperative control: Theory and Applications*, pp. 25–53, Springer-Verlag London, 2008.

[7] W. Ren and R. Beard, "Distributed formation control of multiple wheeled mobile robots with a virtual leader," in *Distributed consensus in multi-vehicle cooperative control: Theory and Applications*, pp. 197–199, Springer-Verlag London, 2008.

[8] M. Inc., "Sonar," in *Pioneer 3 Operations Manual*, pp. 12–13, MobileRobots Inc., 2006.

[9] M. Hromatka, *A Fuzzy Logic Approach to Collision Avoidance in Smart UAVs.* PhD thesis, College of Saint Benedict/Saint John's University, 2013.

[10] L. Ssebazza and Y.-J. Pan, "dGPS-based localization and path following approach for outdoor wheeled mobile robots.," *International Journal of Robotics and Automation*, vol. 30, no. 1, 2015.

[11] I. Susnea, V. Minzu, and G. Vasiliu, "Simple, real-time obstacle avoidance algorithm for mobile robots.," *Recent Advances in Computational Intelligence, Man-Machine Systems and Cybernetics*, pp. 24–29, 2009.

[12] Y. Peng, D. Qu, Y. Zhong, S. Xie, and J. Luo, "The obstacle detection and obstacle avoidance algorithm based on 2-d lidar.," *International Conference on Information and Automation*, pp. 1648–1653, 2015.