Intelligent Service Robotics (2021) 14:535–548
https://doi.org/10.1007/s11370-021-00379-2

ORIGINAL RESEARCH PAPER

Genetic fuzzy-based method for training two independent robots to perform a cooperative task

Andrew Barth[1] · Yufeng Sun[1] · Lin Zhang[1] · Ou Ma[1]

Received: 9 September 2020 / Accepted: 28 July 2021 / Published online: 9 August 2021

Abstract
When two skilled human workers cooperate on a task, such as moving a sofa through a tight doorway, they often infer what needs to be done without explicit communication because they have learned cooperation skills from their prior work or training. This paper extends that concept to a two-robot team. The robots are given the task to carry a large payload through a narrow doorway while avoiding obstacles within the room. System dynamics and sensor noise were included in the study. Each robot is independently controlled with the knowledge of the goal location, its own position, and the pose of the payload. The decentralized control uses a Genetic Fuzzy System for each robot to learn its own decision-making skill through a training process without a pre-planned motion trajectory. The introduction of a genetic algorithm adds efficiency to the process of determining the shape of the fuzzy logic membership functions by using an evolutionary search algorithm to tune each parameter in the fuzzy system simultaneously. The contribution of this paper is to illustrate how genetic training can tune a simple, decentralized Fuzzy Logic System based on a given scenario and then be used, unaltered, for a scenario beyond that for which it was trained. The extended scenarios introduce unknown obstacles, new sizes and mass properties for the robots and payload, and random initial positions. The effectiveness of this approach for a 2D case is determined by dynamic simulation with results starting at a 95% success rate for the baseline scenario and 84% for the scenario that was extended furthest from how it was originally trained.

Keywords Collaborative robotics · Cooperative robot team · Decentralized control · Genetic Fuzzy system · Intelligent robots · Multi-robot system

1 Introduction

Cooperative robotics (also known as collaborative robotics) is a rapidly growing field of research. Robots that can work with either humans or with other robots will greatly expand the field of tasks that can be automated. With cooperation comes increased complexity because the robots will have to interact with human or robot teammates. Intelligent control methods will aid in handling this new complexity. This paper investigates the use of intelligent, decentralized control to perform a two-robot payload carrying task.

A Fuzzy Logic System (FLS) is an intelligent control technique that mimics human reasoning by replacing simple binary classification with fuzzy classification which assigns degrees of membership to each class. Lafta and Hassan [10], Omrane et al. [17], and Singh and Thongam [22] have implemented an FLS that was used in simulation to guide a mobile robot around obstacles and to a destination. Each of these implementations relied on manual tuning of the FLS parameters. As the complexity of the task increases, so will the difficulty in selecting the FLS parameters that define the system. A large collection of techniques has been developed over the years to tune the parameters of traditional Proportional Integral Derivative (PID) and sliding mode controllers, and now, the same knowledge base must be developed for tuning intelligent control designs. Faisal et al. [4] have compared four methods of defining an FLS for mobile robot control using naive manual tuning, Genetic Algorithms (GA),

✉ Andrew Barth
barthal@mail.uc.edu

Yufeng Sun
sunyf@mail.uc.edu

Lin Zhang
linzhank@gmail.com

Ou Ma
maou@ucmail.uc.edu

[1] University of Cincinnati, Cincinnati, USA

Particle Swarm Optimization (PSO), and expert manual tuning. The GA, and PSO techniques out-performed the naive manual tuning method; however, the expert manual tuning performed best out of the group. This indicates that while computational techniques based on naturally occurring processes have potential for success in defining FLS control parameters, additional research is needed.

The inspiration for these nature-based techniques has many sources. Hossain and Ferdous [7] used a Bacterial Foraging Optimization technique to find the optimal path for a mobile robot. Pandey and Parhi [18] used a Wind Driven Optimization algorithm which attempts to mimic the movement of air particles as the wind blows to equalize pressure throughout the atmosphere. Ant Colony Optimization (ACO) algorithms attempt to mimic the collective behavior of ants. Huang and Juang [8] have used ACO to develop an algorithm capable of guiding a two-robot team in a wall-following pattern. Castillo et al. [2] performed a comparison of ACO and PSO to tune the FLS parameters and Juang et al. [9] have implemented a hybrid ACO and PSO algorithm that seeks to use the advantages of each to guide a two-robot team in an object carrying task. Another method was developed by Machado et al. [14] who used an attractor dynamics approach to behavior generation to control a two robot team in a similar object carrying task that included travel around unexpected obstacles. Reinforcement learning is a computational technique that uses a reward incentive to train a system to produce the desired outcome. This has been used by Lin et al. [12] to train an algorithm based on bee colony behavior to perform a two-robot, object-carrying task. Genetic algorithms utilize random processes and evolutionary selection to produce more successful results after each generation. A GA similar in concept to that employed in this work has been used by Martinez-Soto et al. [16] to tune an FLS used for control of a single mobile robot.

In the previously cited works performing an object-carrying task [8,9,12,14], the control architecture was based on a leader-follower paradigm, where the leader makes the path planning decisions for the robot team and the follower(s) rely on the leader for direction. In a decentralized architecture, each agent in the robot team is independent and determines its own actions without the need for explicit communication. Cooperation using decentralized control is more challenging because the agents have the potential to take conflicting actions; however, it has the advantage of being modular and scalable [20]. A combination of the leader-follower strategy and decentralized control was used by Bechlioulis and Kyriakopoulos [1] for an object carrying task where the leader was given knowledge of the goal and location of the obstacles and the follower relied only on implicit communication, i.e. sensing the motion of the leader. Wang and Schwager [28] used a similar strategy for a box pushing task where the leader could be represented by either an autonomous robot or a human-controlled robot. The use of decentralized control for multiple mobile robots has been explored by Lee and Chwa [11], Luviano-Cruz et al. [13], and Tsai et al. [26] where the focus was on maintaining formation and obstacle avoidance. In recent years there has been significant research effort into decentralized control of multiple robots performing object manipulation tasks. Petitti et al. [19] used decentralized force control techniques to manipulate the twist of an object being carried, but do not attempt simultaneous position control of the robots. Marino [15] developed a decentralized adaptive control method for transporting an object using multiple manipulators that addresses both the kinematic control and the interaction control required for handling rigid objects. Culbertson and Schwager [3] have developed an adaptive control strategy that requires no communication or a priori knowledge of the payload but requires measurement at the center of mass location in order for each robot to track a reference profile. Franchi et al. [6] have a method to estimate all parameters required to transport an unknown load, but their method requires a communication network between the robots. Sathyan and Ma [21] developed a Genetic Fuzzy System (GFS) which uses a genetic algorithm to train the FLS of multiple independent robots to successfully complete an object manipulation task using completely decentralized control. As a final example of decentralized control, Farivarnejad et al. [5] have used sliding mode techniques to implement decentralized control for a team of robots to perform an object-carrying task.

This paper is an extension of previous work by the authors using decentralized control and a genetic fuzzy architecture to cooperatively transport an object out of an empty room [24]. The contribution of this paper is to introduce unknown obstacles into the environment and illustrate how the FLS can be trained on a given scenario and then be used, unaltered, for a scenario beyond that for which it was trained. The FLS combines obstacle avoidance and goal seeking behaviors into one system and does not rely on switching logic to control the behaviors as was used in previously cited works [9,12]. The purpose is to explore the feasibility of performing cooperative action using a decentralized control architecture with no explicit communication between the two robots. If communication protocols have been previously established, it would likely enhance performance; however, we are interested in exploring the more general case where intra-robot communication is not present which may occur due to hardware failure, environmental conditions, or incompatible protocols.

The paper is organized as follows. Section 2 defines the basics of the test scenario and gives a system description. Section 3 details the control architecture and genetic training methods. Section 4 presents the simulation results. Section 5 contains conclusions and future work.

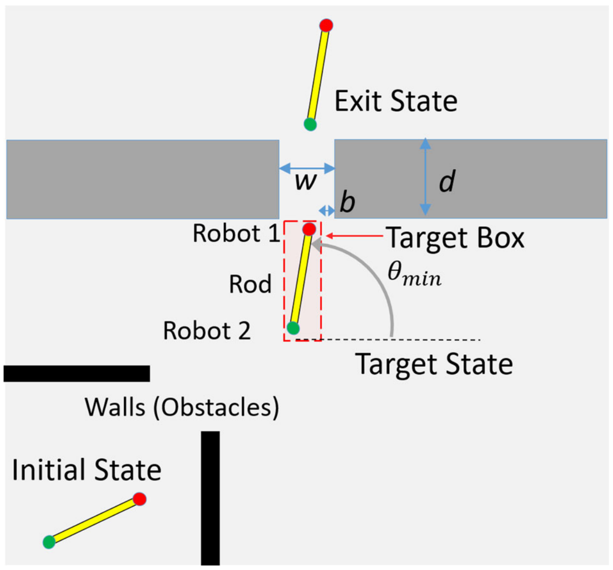

Fig. 1 The two-robot team moving a rod outside of the room through an opening

2 Problem definition

Consider a room with an opening of width w and depth d, shown in Fig. 1. Two robots, with each at one end of the rod, intend to move the rod from its initial position inside the room to a final position outside the room through the opening. The objective of this problem is to train the two robots so that they can cooperatively carry the rod through the opening to the outside with a minimal combined traveling distance while avoiding contact with obstacles. In this scenario, the obstacles are represented by the walls of the room as well as any interior walls that may be present. We have imposed a requirement for cooperation that excludes communication between the two robots, namely, each robot does not know how the other robot will react. The only information available to the robot is its own position within the environment (room), the orientation of the rod, and the location of the opening. By choosing to limit the amount of information available, we can establish a control architecture that is able to perform in a more restrictive environment.

Exiting the room is a two-step process, with the first step being to move the rod to a location directly in front of the opening. This is accomplished by defining a rectangular target box; illustrated by the dashed lines in Fig. 1. The target box serves as the initial goal location for the two robots which will attempt to maneuver themselves and the payload into this area. The target box is a virtual creation within each robot's software and is not part of the physical environment. The dimensions of the target box are determined at the start of the task and are based on the size of the robots and payload along with the minimum angle required for the robots and rod to pass through the opening. The definition of this minimum angle ($\theta_{\min}$) is given in Eq. 1.

$$\theta_{\min} = \cos^{-1}\left(\frac{w-2b}{L}\right) \tag{1}$$

where w is the opening width; L is the length of the rod; and b is a buffer added to each side of the opening which is used to account for the physical size of the robot and prevent collision with an obstacle in the presence of sensor/actuator errors and control overshoot.

Once a robot has entered the target box and the rod angle meets or exceeds $\theta_{\min}$, it will determine a new goal location on the other side of the opening and compute a new command for it to travel straight through the opening toward the exit state while maintaining the existing rod angle. Since the robots are not communicating with each other, each robot does not know when the other has initiated its exit. In order to prevent a premature exit command, the robot must rely on estimating the position of the opposite end of the rod based on the rod angle (θ) and the known length of the rod. It can then evaluate whether the other robot is also in the target box.

2.1 System dynamics

The overall system (consisting of the two robots and the rod) is represented as a single rigid body constrained to a 2D space and thus, each robot has three degrees of freedom, two translational along the x and y axes and one rotational along the z axis perpendicular to the x–y plane. Further, the following assumptions are used in the system dynamics formulation:

- Robots are rigidly attached to the rod
- No friction acts on robot motion
- The rod is rigid with no deformation

The robots are modeled as uniform flat disks located at each end of a uniform thin rod. The robot-rod system is integrated in the dynamics as a single rigid body, resulting in a dumbbell configuration. Due to the assumption of a rigid system, the mass properties relative to a point fixed in the body are static and can be computed using the relations given in Eqs. 2– 5. Note that the system inertia is first computed about the geometric center of the rod and then expressed about the system center of mass location using the parallel axis theorem.

$$m_s = m_1 + m_2 + m_r \tag{2}$$

$$\mathbf{d_s} = \frac{m_1\mathbf{d_1} + m_2\mathbf{d_2} + m_r\mathbf{d_r}}{m_s} \tag{3}$$

$$I_{rc} = \frac{1}{12}m_r L^2 + \frac{1}{2}m_1 r_1^2 + \frac{1}{2}m_2 r_2^2 + m_1\left(\|\mathbf{d_1}\|\right)^2 + m_2\left(\|\mathbf{d_2}\|\right)^2 \tag{4}$$

$$I_s = I_{rc} + m_s\|\mathbf{d_s}\|^2 \tag{5}$$

where m_1, m_2 are the masses of each robot; m_r is the mass of the rod; m_s is the system mass; $\mathbf{d_1}, \mathbf{d_2}$ are the vectors from each robot to the center of the rod; $\mathbf{d_r}$ is the vector from the mass center of the rod to the geometric center (0 for a uniform rod); and $\mathbf{d_s}$ is the location of the system center of mass relative to the center of the rod. In the inertia calculation, L is the length of the rod; r_1 and r_2 represent the radius of each robot; I_{rc} is the system moment of inertia about the center of the rod; and I_s is the system moment of inertia with respect to its mass center.

The system dynamics are integrated at the system center of mass. A global reference frame is established with the origin located at the lower left corner of the room and the x-axis horizontal, y-axis vertical, and z-axis pointing up out of plane. A body reference frame is also established with the origin located at the system center of mass and the x-axis pointing along the rod toward robot 1, the y-axis in plane and perpendicular to the rod, and the z direction pointing up out of plane.

The state vector consists of: the position vector of the system center of mass in the global reference frame, the angle of the body frame relative to the global frame, the velocity vector of the system center of mass in the global reference frame, and the angular rate of the body relative to the global frame $[\mathbf{r_s}, \theta, \mathbf{v_s}, \dot{\theta}]$. The Newton-Euler equations shown in Eq. 6 and Eq. 7 are used to integrate the state.

$$\ddot{\mathbf{r}}_{\mathbf{s}} = m_s^{-1}\mathbf{f_N} \tag{6}$$

$$\ddot{\theta} = I_s^{-1}\tau_N \tag{7}$$

Here $\mathbf{f}_{\mathrm{N}}$ and τ_{N} are the net force vector and scalar moment acting on the system. Each robot produces a force based on the output of its own control system. The force acts at the center of each robot and will therefore produce a moment about the center of mass of the system. The net force and moment acting on the system are computed in Eqs. 8 and 9.

$$\mathbf{f}_{\mathrm{N}} = \mathbf{f}_1 + \mathbf{f_2} \tag{8}$$

$$\tau_{\mathrm{N}} = (\mathbf{r_1} - \mathbf{r_s}) \times \mathbf{f_1} + (\mathbf{r_2} - \mathbf{r_s}) \times \mathbf{f_2} \tag{9}$$

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the force vectors applied by each robot and $\boldsymbol{\tau}_{\mathrm{N}}$ will only contain nonzero terms about the z-axis and therefore can be treated as a scalar.

Once the new state of the system center of mass has been computed, the position and velocity state of the rod and each robot can be derived based on the angle θ, angular rate $\dot{\theta}$, and the fixed geometry of the rod. Since the rod geometry is known in the body reference frame, it must be converted to the global reference frame using a rotation about θ $({}^G\mathcal{R}_B)$.

The position and velocity of the center of the rod in the global reference frame can be computed as shown in Eq. 10 and Eq. 11 where $\mathbf{d_s}$ is the position of the center of mass relative to the center of the rod.

$$\mathbf{r}_{\mathrm{r}} = \mathbf{r_s} - {}^G\mathcal{R}_B\ \mathbf{d_s} \tag{10}$$

$$\mathbf{v}_{\mathrm{r}} = \mathbf{v_s} - \dot{\theta}\hat{z} \times {}^G\mathcal{R}_B\ \mathbf{d_s} \tag{11}$$

Next, the position and velocity of each robot is computed in the global reference frame using the previously computed values at the center of the rod. The equations are shown in Eq. 12 through Eq. 15.

$$\mathbf{x}_1 = \mathbf{x_r} + {}^G\mathcal{R}_B\ \mathbf{d_1} \tag{12}$$

$$\mathbf{v}_1 = \mathbf{v_r} + \dot{\theta}\hat{z} \times {}^G\mathcal{R}_B\ \mathbf{d_1} \tag{13}$$

$$\mathbf{x}_2 = \mathbf{x_r} + {}^G\mathcal{R}_B\ \mathbf{d_2} \tag{14}$$

$$\mathbf{v}_2 = \mathbf{v_r} + \dot{\theta}\hat{z} \times {}^G\mathcal{R}_B\ \mathbf{d_2} \tag{15}$$

The rotation states θ and $\dot{\theta}$ of the rod and robots are identical to that of the system due to the rigid body assumption. We now have the position, velocity, and rotational states of each element in the system expressed in the global reference frame.

3 Control design

3.1 Fuzzy logic system

An FLS is a system where unlike Boolean logic, an outcome is not simply true or false, but rather can have a continuous level of value from a minimum to a maximum value. In terms of the FLS that is used for the two-robot team, the control output is not simply "goal seeking" or "obstacle avoidance" but a weighted combination of each. The FLS used for control in this experiment is a two-input, two-output system.

An FLS functions by performing fuzzification of the input values. This process entails taking the input and assigning a value for each membership function that has been defined for a particular input. A general example is shown in Fig. 2 where an input value may not be crisply categorized as low, medium, or high, but rather will be a fuzzy blend of two or more categories. The next step in the process is the application of the fuzzy rulebase to the membership function values. The rules map the inputs to the outputs and take the general form of:

IF [Antecedent 1] AND [Antecedent 2] THEN [Consequent 1] AND [Consequent 2]

Each output is assigned a numerical value and the final step is to defuzzify this result by applying the output membership function and computing a single result for each output using a technique such as centroid calculation or mean-of-max.

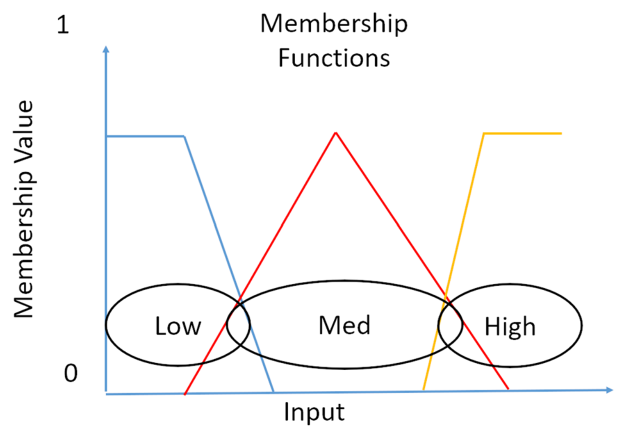

Fig. 2 General membership function example showing overlapping regions

3.1.1 Fuzzy logic inputs

The two inputs to the FLS are the wall distance (D_w) and the goal distance (D_g). The wall distance input is computed by establishing the scalar distance from the robot to each of the walls in the room (D_i). In this implementation, the computations are performed analytically; however, in practical cases the distances can be obtained from range sensor readings. The input to the fuzzy system is taken as the minimum value of this set of distances as shown in Eq. 16 where i ranges from 1 to the number of walls.

$$D_w = \min([D_1, D_2, \cdots, D_i]) \tag{16}$$

The goal distance input to the fuzzy controller is simply the scalar distance from the robot to the edge of the target box.

3.1.2 Fuzzy logic outputs

The two outputs of the FLS are Speed (S) and Correction Angle (CA). The speed output is used to prioritize the motion of the robot closest to a wall. When the robot is near a wall, it will be commanded to move at a higher speed, which will in turn produce a greater contribution to the net force acting on the system and "overpower" the force of the other robot which may be moving toward the goal location unconcerned with the wall.

Nominally, the robot will move in a straight line direction toward the target. The correction angle output is designed to adjust this path when near an obstacle. The correction angle will take a value from 0 to + 90 degrees with the convention that a 0 value indicates moving directly toward the goal and a 90 degree value indicates moving directly away from an obstacle.

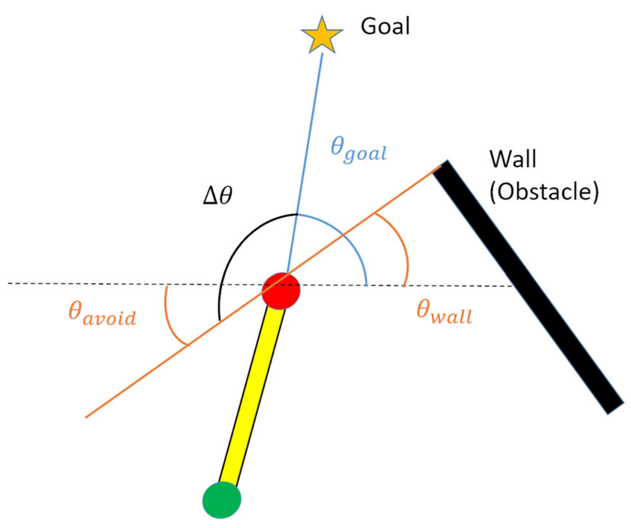

Fig. 3 Geometry of parameters used in the commanded angle computation

3.1.3 Determining the commanded angle

The correction angle that is output from the FLS is a scalar measure based on how close the robot or rod is to a wall. This control output is combined with data from the sensors to form an angle command that can be issued to the actuator using the following method:

First, compute the angle that would move the robot directly away from the wall (θ_{avoid}) as shown in Eq. 17 and illustrated in Fig. 3. Here, θ_{wall} is the angle to the nearest wall that has been determined by the navigation subsystem which will be discussed in Sect. 3.3.

$$\theta_{\text{avoid}} = \theta_{\text{wall}} - \pi \tag{17}$$

Next, compute the difference between the wall avoidance angle and the goal angle (θ_{goal}) which has also been computed by the navigation subsystem.

$$\Delta\theta = \theta_{\text{avoid}} - \theta_{\text{goal}} \tag{18}$$

Finally, the commanded angle (θ_c) is obtained by taking correction angle that was output from the FLS and computing the ratio to the maximum value of 90 degrees. This ratio is then combined with the $\Delta\theta$ from Eq. 18 and the goal angle using the relation shown in Eq. 19. The commanded angle is then constrained between $[-\pi, \pi]$.

$$\theta_c = \theta_{\text{goal}} + \Delta\theta \left| \frac{\text{CA}}{\pi/2} \right| \tag{19}$$

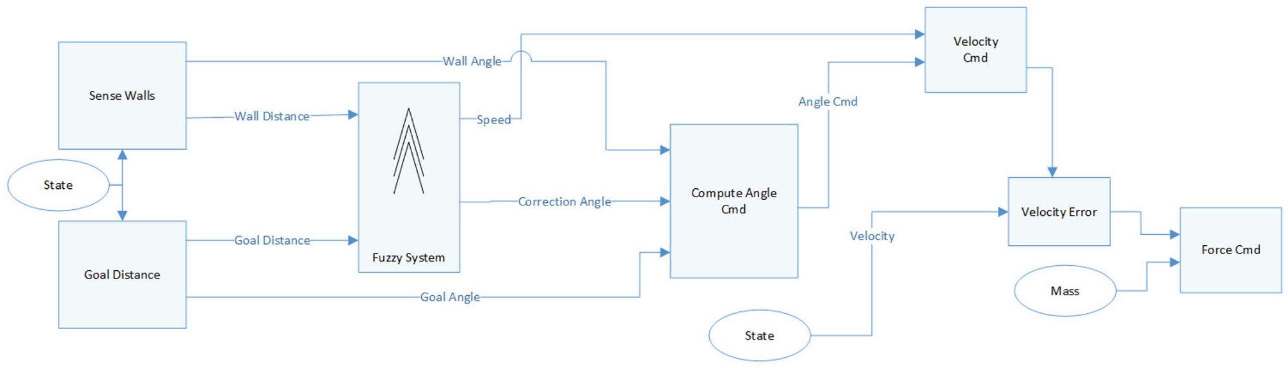

Fig. 4 Intelligent control block diagram

3.1.4 Control force

The output of the control system must be represented as a force vector prior to being passed to the robot. The commanded heading angle is combined with the speed to produce a commanded velocity vector ($\mathbf{v}_c$). A velocity error ($\Delta\mathbf{v}_i$) is then computed by using the current velocity of the robot ($\mathbf{v}_i$). The robot mass (m_i) is used to compute a delta momentum vector, which is equal to the desired force ($\mathbf{f}_c$).

$$\mathbf{v}_c = S \begin{bmatrix} \cos\theta_c \\ \sin\theta_c \end{bmatrix} \tag{20}$$

$$\mathbf{v}_i = \mathbf{v}_c - \mathbf{v}_i \tag{21}$$

$$\mathbf{f}_c = m_i * \boldsymbol{\Delta}\mathbf{v}_i \tag{22}$$

3.1.5 Control summary

A block diagram of the intelligent control design is shown in Fig. 4. The blocks on the left side of the diagram: “Sense Walls” and “Goal Distance”, are concerned with establishing the current vehicle state. The distance and angle to both the goal location and the closest wall are computed, thus answering the question of where the robot is and where it needs to go. The blocks in the center of the diagram: “Fuzzy Systems” and “Compute Angle Cmd” were detailed in the previous discussion and represent the heart of the algorithm and this research. These algorithms are tasked with making the decision of what action to take given the current conditions. Specifically, how fast and in what direction the robot should move in order to balance the objectives of reaching the goal while avoiding the obstacles. Achieving this balance between competing objectives is where the fundamental principles of fuzzy logic are well suited. Finally, the blocks on the right side of the diagram: “Velocity Cmd”, “Velocity Error”, and “Force Cmd”, are concerned with taking the outputs from the FLS and establishing how the robot is going to achieve the desired motion by computing a force command that can be passed on to the robot’s actuators.

3.2 Genetic training

The control system for each robot was developed using a GFS which is a system that uses a genetic algorithm to determine the configuration of the FLS. Due to the compuational burdens of running a genetic algorithm, it must be executed a priori using a set of training scenarios. Once the FLS configuration has been determined through genetic training, it is then transferred to the operational environment for evaluation.

The genetic training operates by attempting to minimize a cost function through the use of random mutations and gene crossovers to produce offspring that result in improved performance from generation to generation. The genetic algorithm is arranged in a hierarchical structure. At the lowest level, each tunable parameter in the fuzzy system represents one gene in the system. All genes needed to fully define the system are grouped into a chromosome, also referred to as an individual. In the context of simulating a team of robots carrying a rod, an individual can be used to perform one execution of the simulation. Once the simulation is complete, a cost function is evaluated to score the performance of the individual. The next level consists of a collection of individuals, called a population. A population of n individuals represents n FLS configurations and n executions of the simulation. The cost function for each member of the population is determined and the genetic algorithm parameters for mutation, crossover, and elitism are used to determine which members of the population are used to create the next generation, where a generation is the highest level of the hierarchical structure. A genetic algorithm typically is executed until a fixed number of generations is reached, the desired performance is achieved, or no significant changes are seen from generation to generation. The genetic training algorithm used in this study was chosen due to its simplicity when compared to other optimization algorithms such as neuroevolution. By anchoring the algorithm in the well-studied concept of passing beneficial genes to future generations, it matches well with the philosophy of explainability and sim-

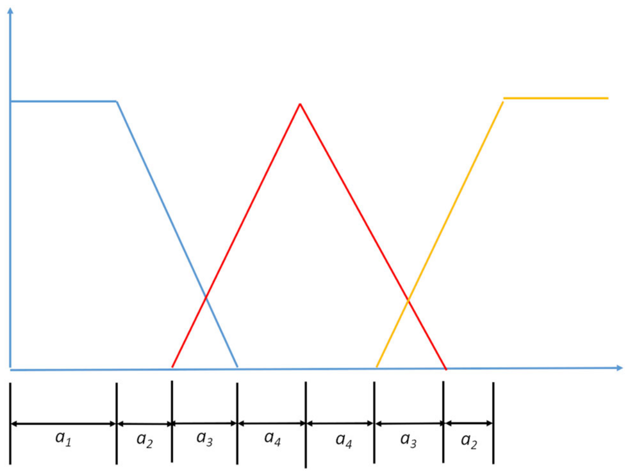

Fig. 5 Generalized construction of membership functions showing the four parameters required to define the system

plicity embraced by fuzzy logic [29].More information on the genetic algorithm used can be found in [25].

3.2.1 Membership function setup

The gains of a traditional control system are tuned for acceptable performance of the system using a set of general guidelines and a trial and error process. For the intelligent control system presented in this paper, a genetic algorithm is used to set the final parameters for the FLS membership functions. A set of three increasing parameters is required to define a triangular membership function. The genetic algorithm will randomly assign values using mutation and crossover techniques. This can very easily violate the strictly increasing rules to define a membership function. To prevent this occurrence, a strategy must be developed to design the FLS membership functions such that they can be assigned by the result of a genetic algorithm. An example of how the FLS membership functions are tuned is shown in Fig. 5.

With this setup, the entire set of three membership functions can be configured through linear combinations of only four parameters (a_1, a_2, a_3, a_4). Since these parameters are bounded within the genetic algorithm to positive values, this will ensure that each set of points for an individual membership function will be strictly increasing.

3.2.2 Rule base setup

The rule base was not included as part of the genetic algorithm training because the application of the rules for this scenario is rather straightforward and there would be little benefit in allowing these to vary. The general setup of the rules is that when a robot is near an obstacle, it will move at a fast speed with a high correction angle. When at a medium distance from an obstacle, it will use a medium speed and medium correction angle. When far from an obstacle, it will move slowly and at a small correction angle. The speed of a robot is set higher near an obstacle so it will result in a larger commanded force to move away from the obstacle and will contribute more to the net force than the other robot which is not in danger of a collision.

Table 1 Variables (genes) in genetic algorithm

Item	Number of variables
Wall distance input MF	4
Goal distance input MF	4
Speed output MF	4
Correction angle output MF	4
Speed scale factor	1
Total:	17

3.2.3 Genetic algorithm configuration

The MATLAB "ga" function was used to perform the genetic algorithm optimization. The variable list consisted of 17 items shown in Table 1.

Once the variables have been set for the given trial, the FLS for each robot is generated and loaded into the simulation. The final parameter, speed scale factor, is not part of the FLS and is used to control how quickly the robots are permitted to travel.

The population size for the genetic algorithm was set to 85. Given the 17 variables, this population size is in the suggested range between 5 and 10 times the number of variables from Storn and Price [23]. The number of generations was set to 25. This was arbitrarily selected initially and it was confirmed to be a reasonable value once the test results were obtained.

3.2.4 Training scenarios

A set of seven scenarios was defined to train each individual. The initial conditions for each scenario were chosen for spatial diversity within the room and cases where it is difficult to maneuver the rod without impacting an obstacle. The locations of these Initial Conditions (IC) are shown in Fig. 6.

The set of seven scenarios adds an extra dimension to the number of cases that must be executed in the training set. With 7 cases per individual, a population size of 85, and 25 generations, a total of 14,875 executions of the simulation occurs during training. Using a basic desktop computer, the training was completed in less than 12 hours.

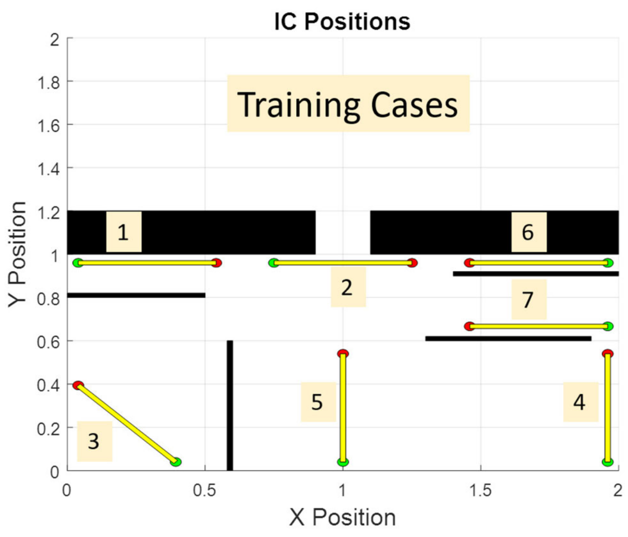

Fig. 6 Initial position of the rod for the seven scenarios used in genetic training

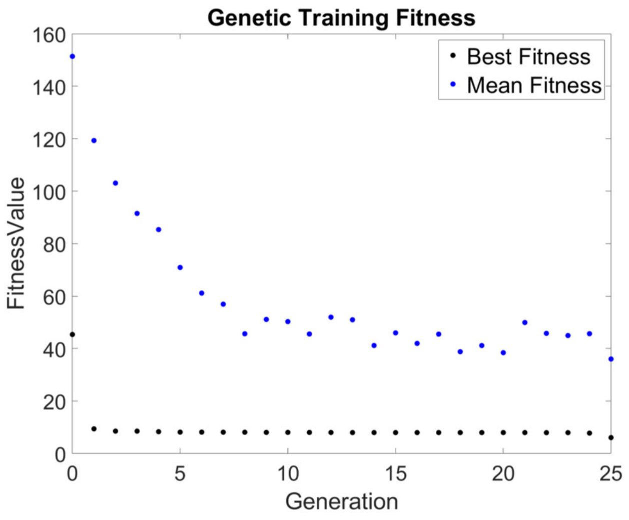

Fig. 7 Fitness values for each generation during genetic algorithm training

3.2.5 Cost function

The cost function value is defined as the combined distance traveled by both robots (tD), summed across all seven training scenarios. A steep penalty is added to the cost function for each scenario that collides with an obstacle (cP) or fails to exit the goal with a pre-defined maximum simulation time (tP). The cost function is shown in Eq. 23 through Eq. 25.

$$\text{Cost} = \sum_{n=1}^{7} (tD + cP + tP)_n \tag{23}$$

$$cP = \begin{cases} 0 & \text{if: no collision} \\ 40 & \text{else: collision occured} \end{cases} \tag{24}$$

$$tP = \begin{cases} 0 & \text{if: goal achieved} \\ 40 & \text{else: max time exceeded} \end{cases} \tag{25}$$

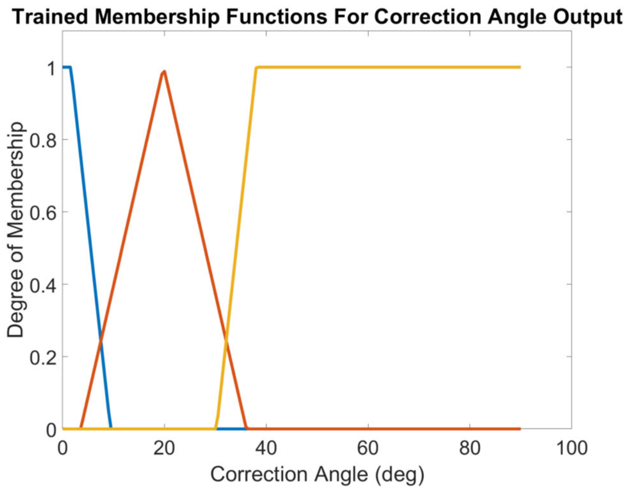

Fig. 8 Membership functions after genetic training completed. Example variable shown: correction angle output

3.2.6 Training results

The mean and best value of the cost function for each generation is shown in Fig. 7. The specific numerical values are not important to the solution as they can vary based on the room size or the chosen penalty values. The important feature of Fig. 7 is that the mean and best values converge to a steady state value indicating that further training will not significantly improve the result. This occurs after approximately 10 generations.

Inspection of the trained FLS membership functions indicated that the changes to the default values were not drastic and the general layout remained similar to the symmetrical layout shown previously in Fig. 5. The value with the most change was the correction angle output shown in Fig. 8. The “Center” membership function, where the robot moves directly toward the goal, has been compressed to the left and now only covers a range from 0 to 9.5 degrees.

For the training scenario, the two robots were configured with identical size and mass properties; therefore, the FLS that was developed was applied identically to each robot; however, the controller architecture can support the use of a unique FLS if required. The settings for the FLS based on the genetic training results were stored and used in all results documented herein.

3.3 Simulation

Simulation of the dual-robot rod-carrying task is performed using a combination of Simulink models and Integrated Matlab functions. The block diagram for the on-board software for each robot shown in Fig. 9. It consists of sensor, executive, navigation, and control routines. The software structure is the same for each robot; however, the implementation for each is instantiated separately and can be adjusted independently through parameter settings.

Modeling of sensor hardware was not included in this work; however in its place, the Sensor block in Fig. 9 applies white noise to the true robot position vector and angle. The white noise was configured with a zero mean (no bias) and a variance of 10^{-6}, resulting in an additive noise value with amplitude on the order of 3 mm and 0.17 degrees. It then uses this noisy data to compute the distance and angle from the robot to each wall. The Navigation routine is responsible for determining the closest wall, computing the distance to the goal, and forming the inputs for the FLS. The Executive block is responsible for determining when the robot reaches the target box (position and angle) and once this occurs, it updates the goal location to allow the rod to pass through the doorway. Finally, the Control block executes the controller discussed in Sect. 3.1.

The algorithms are data-driven so each robot can have unique control gains. The integration routine is a fixed-step solver. The step size for the simulation is set to 0.01 s and all blocks within the model are executed at this rate.

4 Results

Verification testing was performed with six different test scenarios to ensure that the control algorithms performed correctly under a variety of conditions.

Test Scenarios

1. Execute training cases and evaluate performance.
2. Randomize initial conditions using the same room layout as training scenarios.
3. Simulate an empty room with no interior obstacles.
4. Alter the room layout with obstacle placement and a goal location not seen in the training scenarios.
5. Alter the size and mass of one of the robots and change the length of the rod.
6. Alter the shape and mass of the rod making it significantly larger than the robots carrying it.

Each scenario builds upon the previous test and is intended to evaluate the ability of the FLS control system to perform in situations of accumulated separation from the scenario on which it was trained.

Table 2 Numerical results from the seven training cases

Case	Total distance (m)	Total time (s)
1	3.7	68
2	2.9	213
3	5.1	136
4	4.0	54
5	2.4	28
6	3.9	67
7	3.8	95

4.1 Training cases

Each of the seven training cases was executed using the final FLS parameters in order to evaluate the performance of the system. As expected, all seven cases were successful. The results are presented in Table 2 where the total distance is the sum of both robots' traveled distances.

The room size for these scenarios is 2 × 2 m. If the room were empty without obstacles and the robots followed an optimal path, the maximum distance needed to exit the room would be on the order of 3.4 m. Five of the seven cases exceeded the maximum distance indicating that the robots often have to take a non-optimal route to avoid the obstacles. The time needed to exit the room was not part of the cost function and therefore was not optimized, but it was tracked in order to find outliers where the robots became "stuck" trying to avoid the obstacles. This situation occurred in case 2 which needed 213 s to exit the goal. Figure 10 shows an overlap of dark lines where the rod was nearly stationary for 120 s. The robot/rod system was straddling the target box with robot 1 attempting to move left and robot 2 attempting to move right, while each was equidistant from an obstacle. This example illustrates a limitation of robots acting as independent agents without a central path planning algorithm. One common practice in machine learning is to add an element of randomness to the algorithm. This scenario introduces randomness through the simulated sensor noise in the position measurements. The slight variation in distance and direction to the walls allows the robots to eventually work free of this stalemate (Fig. 10). Example trajectories for several of the training cases are shown in Fig. 11 through Fig. 12.

4.2 Randomized inputs

A set of 1000 monte-carlo cases was run with a randomized input location for each case. The rod (x,y) position and angle were randomized independently using a uniform distribution. Varying these parameters independently can produce configurations where one or both robots begin outside an exterior wall or on top of an obstacle. To avoid this, the configuration

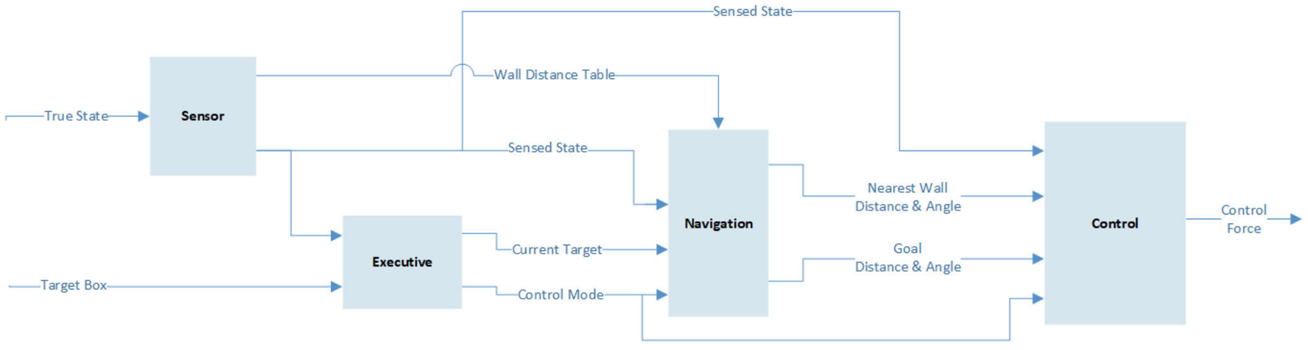

Fig. 9 Block diagram of Robot's on-board software

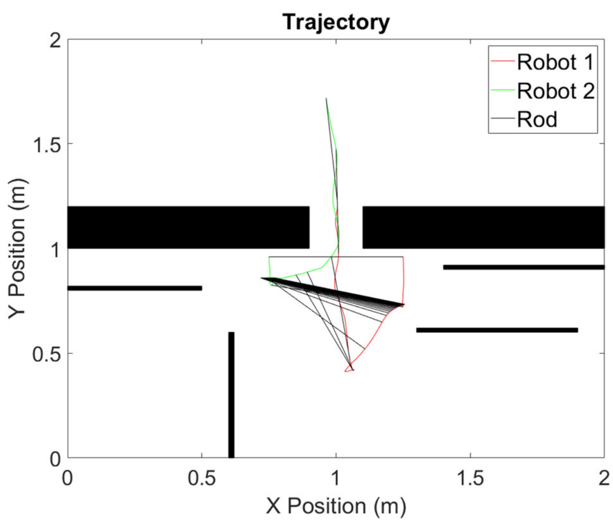

Fig. 10 Path travelled for training Case 2

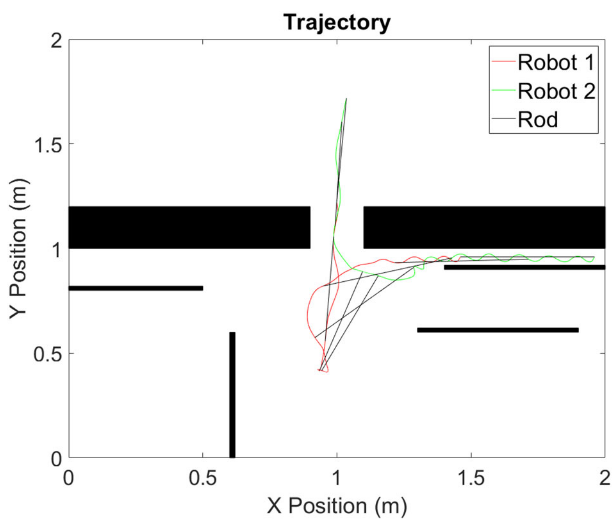

Fig. 12 Path travelled for training case 6

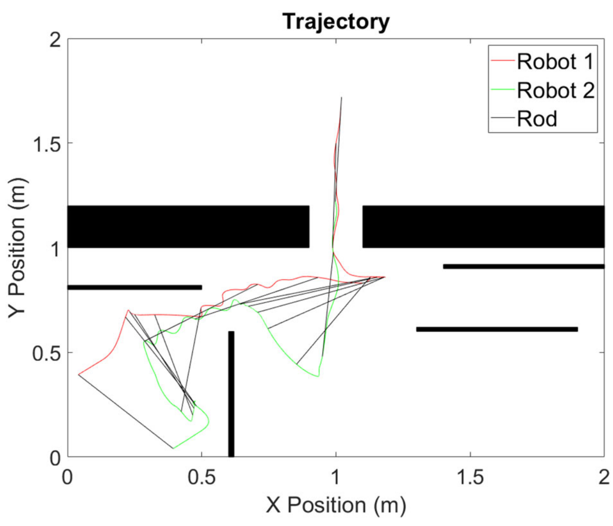

Fig. 11 Path travelled for training case 3

was first checked for validity and a new sample was drawn if it was found to be invalid. A seed value was used for the random number generator so the same set of conditions can be regenerated. The plot of all the 1000 starting positions is shown in Fig. 13. This plot indicates that the monte-carlo technique produced good coverage of the entire room with the exception of the upper left and upper right corners. These cases were covered in the training scenarios so we know that they can be successfully executed.

4.3 Empty room

A test was performed where all interior obstacles were removed. While this scenario differs from the training data, the lack of obstacles is expected to be easier to navigate and produce improved results. A set of 1000 randomized inputs was generated and executed.

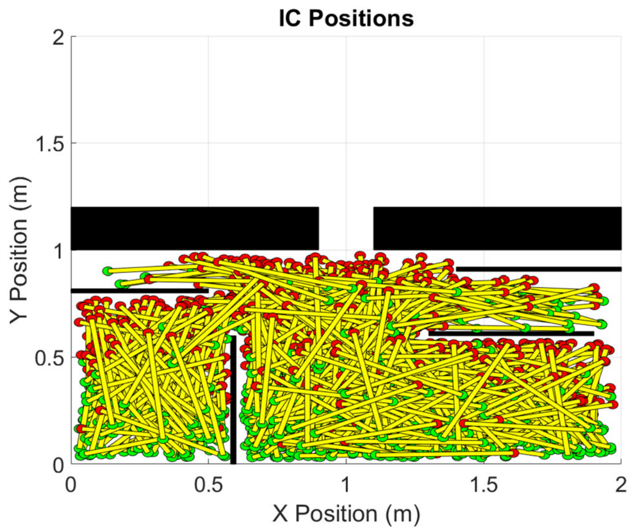

Fig. 13 Initial conditions for the 1000 case Monte-Carlo test set

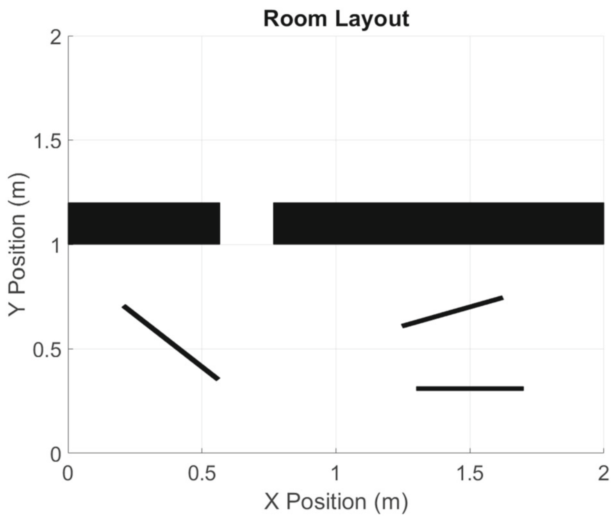

Fig. 14 New room layout with 3 added obstacles

4.4 New room layout

The next round of testing was conducted using a new room layout. The size of the room was kept the same; however, the goal location was moved and the interior obstacles were rearranged. The new room layout is shown in Fig. 14. In contrast to the training room, the new layout contains interior walls that are not perpendicular or parallel to the room and a goal location that is offset from center. The robot and rod configuration was identical to that of the training scenarios (Configuration A).

A set of 1000 randomized inputs was generated and executed for this scenario. The initial conditions will not be exactly the same as those from Sect. 4.2 because some of those cases would be in contact with the new obstacle locations and some of the cases previously discarded will now be valid in the new room.

Table 3 Summary of 1000 case Monte-Carlo results

Case	Success	Dist. (m)		Time (s)	
	%	μ	σ	μ	σ
Training room	95	2.8	0.9	57	46
Empty room	100	2.9	0.5	56	17
New room	93	3.0	1.0	65	65
Config B new room	86	3.1	1.0	61	62
Config C new room	84	3.9	1.4	125	75

4.5 Configuration B, new room

The mass value and radius of one robot was increased by 100% from the training configuration. For all previous trials, the system center of mass was located at the geometric center of the rod. This new configuration will offset the system center of mass along the axis of the rod. An additional modification was made to increase the length of the rod by 20%. This test was run using the new room layout from Sect. 4.4 Again, a set of 1000 randomized initial conditions was generated and executed.

4.6 Configuration C, new room

This test scenario examines a different shape for the rod. Previous cases treated the rod width as negligible as compared with the robot size. For this case, the rod width was set to be 50% larger than the diameter of the robot. In addition, the rod length was increased by 20% and the rod mass was increased by 200% over the training scenario. The new scenario represents a situation where two small robots are required to carry a payload that is larger and more massive than their own size. This test was run using the new room layout from Sect. 4.4. Again, a set of 1000 randomized initial conditions was generated and executed.

4.7 Summary of results

A summary of the results for each of the monte-carlo sets is given in Table 3 where the mean (μ) and standard deviation (σ) are given for the total distance and time.

Using randomized initial conditions in the same room as the training scenario produced excellent results with a 95% success rate for exiting the room. This indicates that the genetic training was successful in producing a controller capable of exiting from nearly any point in the room. Analysis of the failure cases showed that most failures occurred from cases that were initialized in the lower left portion in the room and were required to navigate through the small

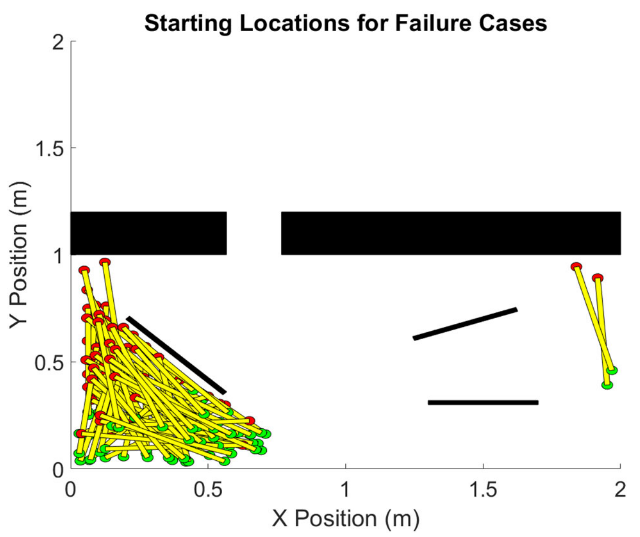

Fig. 15 Initial conditions for the cases that failed when using the new room layout

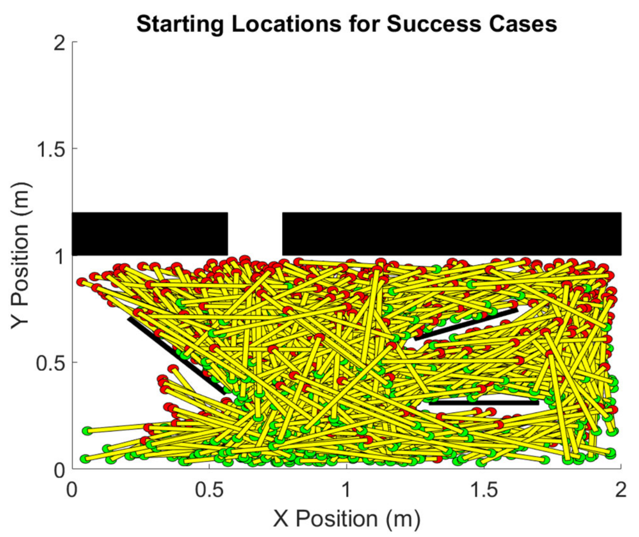

Fig. 16 Initial conditions for the cases that succeeded when using the new room layout

corner opening between the two obstacles on the left side of the room.

The results of the empty room scenario showed a 100% success rate, which is expected due to the simplicity of the task with no obstacles present. The standard deviations of the total distance and time were smaller for the empty room indicating that the robots were able to use a direct route to exit the room and did not have to spend distance and time avoiding obstacles.

The 93% success rate for the new room layout indicates that the genetic training of the system produced a result that can be applied to scenarios outside of the training set with only a small decrease in performance. Analysis of the failure cases for this scenario showed that the majority of failures occurred for initial conditions placing the robots in the lower left corner behind the angled obstacle. A plot of the failure cases is shown in Fig. 15 and a plot of the success cases is shown in Fig. 16. It is clear from the second plot that there is a distinct empty region where no cases were successful.

Further analysis of these cases shows that of the 74 failure cases, only 8 failed due to hitting an obstacle. Most of the failures were due the robots getting "stuck"as discussed previously for training case # 2. Cases with starting points in this region need to travel toward the right past the goal and then move up and back toward the left to reach the target box increasing the likelihood of encountering a situation where the robots are equidistant from one or more obstacles. This situation emphasizes one of the major challenges of using independently controlled agents, with each robot issuing conflicting commands. The simple FLS controller implemented here does not have the sophisticated path planning capability to escape this situation and here the randomness of the sensor noise was not always sufficient to end the stalemate within the allotted time as occurred in the second training scenario.

Changing the mass and size of one robot and increasing the length of the rod for configuration B lowered the success rate to 86% and changing the shape and mass of the rod for configuration C lowered the success rate to 84%. The time spent for configuration C also increased significantly over the previous scenarios, indicating that more cases are encountering the "stuck" conditions and reaching the time limit. Overall, these results are still quite good considering no additional training or controller modifications were made and both of these scenarios also utilized the new room layout. This result indicates that the GFS control strategy developed here has excellent potential to be applied to a variety of robot and payload types.

Examination of the failure cases has shown that the most common cause is when the robots encounter a condition where they are between obstacles located on opposite ends of the rod. Each robot attempts to move in opposing directions with equal force. As discussed with training case #2, they can sometimes work their way out of this condition due to sensor errors, but this is not always the case. If we were to allow for communication between the robots, they could enact a leader-follower relationship and likely overcome the obstacle; however, since we have prohibited this communication, the robots must independently recognize the "stuck" condition. An adaptive algorithm could be useful in this scenario. If a given time has elapsed with no significant progress toward the goal, the robot could temporarily reduce the safety buffer being used for obstacle avoidance. This may provide the clearance needed to escape, at the expense of higher risk of collision. Another, though more risky, option would be for one robot to temporarily cease its force command and hope that the other robot would push it clear without unknowingly pushing it into the obstacle.

5 Conclusion and future work

5.1 Future work

One area for future work on this project is to add complexity to the task by adding additional robots and replacing the two dimensional rod with a three dimensional, asymmetric object as the payload. Second, a comparison should be performed with alternate control methods. A neural network-based control policy trained using a reinforcement learning technique would be a suitable approach since a reward function similar to that shown in Eq. 23 could be used. Finally, the most significant future work is to move from the simulated environment to the real world. A mechanism, or arm, to hold the rod in place and simulate the constraint forces present at the attach points will be required. For the system to be controllable and stable, the mechanism must have some degree of compliance where the rod attaches to the robot similar to how a human makes small corrective motions with their arms when cooperatively carrying an object. Tsiamis et al. [27] presents dynamic modeling for a compliant mechanism and Machado et al. [14] have developed a two degree of freedom payload support base that can sense the displacement of the payload relative to the robot center. Each of these works can be used as a model for future development. A navigation sensor, either camera or lidar based, will add the ability to acquire the locations of the obstacles in real-time. Testing with a real robot will identify the off-nominal performance of the steering and drive mechanisms. With the improved knowledge from hardware testing, additional factors affecting control performance can be investigated including effects of the true sensor noise, biases in the hardware output, and transmission or processing delays in delivering the data between the hardware and the control software. Also, the true performance limits of the robot in terms of minimum and maximum speed can be established and it may be beneficial to include the time required to exit the room directly as part of the cost function and retrain the GFS for a more efficient exit.

5.2 Conclusion

The task of carrying a rod through an opening of a room by two mobile robots using a GFS approach was investigated. System dynamics and sensor noise were included in the study. The two robots operated independently with no explicit communication and the environment was populated with arbitrarily set obstacles. The robots were able to exit the room with 95% success for the nominal scenario. The GFS design and training was shown to be generalizable to new scenarios not part of the training set with a 93% success rate for an altered room configuration, an 86% success rate for an altered robot and rod size in addition to the new room, and an 84% success rate for an altered rod shape and mass in addition to the new room.

The GFS control formulation presented in this work is readily extensible to a real-world scenario. All required inputs to the control system, i.e., goal location and distance to the nearest obstacle, can be obtained by sensor measurements in real time. No trajectory pre-planning, beyond declaration of a goal location, is necessary. The simplicity of the control algorithm and promising results for generalization to new scenarios will permit this GFS control strategy to be rapidly applied to future cooperative tasks among mobile robots.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Bechlioulis CP, Kyriakopoulos KJ (2018) Collaborative multi-robot transportation in obstacle-cluttered environments via implicit communication. Front Robot AI 5:90
2. Castillo O, Martinez-Marroquin R, Melin P, Valdez F, Soria J (2012) Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. Inf Sci 192:19–38
3. Culbertson P, Schwager M (2018) Decentralized adaptive control for collaborative manipulation. In: 2018 IEEE international conference on robotics and automation (ICRA), pp. 278–285. IEEE
4. Faisal M, Algabri M, Abdelkader BM, Dhahri H, Al Rahhal MM (2017) Human expertise in mobile robot navigation. IEEE Access 6:1694–1705
5. Farivarnejad H, Wilson S, Berman S (2016) Decentralized sliding mode control for autonomous collective transport by multi-robot systems. In: 2016 IEEE 55th conference on decision and control (CDC), pp. 1826–1833. IEEE
6. Franchi A, Petitti A, Rizzo A (2018) Distributed estimation of state and parameters in multiagent cooperative load manipulation. IEEE Trans Control Network Syst 6(2):690–701
7. Hossain MA, Ferdous I (2015) Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. Robot Auton Syst 64:137–141
8. Huang CA, Juang CF (2017) Evolutionary fuzzy control of two cooperative object-carrying wheeled robots for wall following through multiobjective continuous aco. In: 2017 Joint 17th world congress of international fuzzy systems association and 9th international conference on soft computing and intelligent systems (IFSA-SCIS), pp. 1–3. IEEE
9. Juang CF, Lai MG, Zeng WT (2014) Evolutionary fuzzy control and navigation for two wheeled robots cooperatively carrying an object in unknown environments. IEEE Trans Cybern 45(9):1731–1743
10. Lafta HA, Hassan ZF (2015) Mobile robot control using fuzzy logic. J Univ Babylon 23:524–532

11. Lee G, Chwa D (2018) Decentralized behavior-based formation control of multiple robots considering obstacle avoidance. Intell Serv Robot 11(1):127–138
12. Lin CH, Wang SH, Lin CJ (2018) Interval type-2 neural fuzzy controller-based navigation of cooperative load-carrying mobile robots in unknown environments. Sensors 18(12):4181
13. Luviano-Cruz D, Garcia-Luna F, Pérez-Domínguez L, Gadi SK (2018) Multi-agent reinforcement learning using linear fuzzy model applied to cooperative mobile robots. Symmetry 10(10):461
14. Machado T, Malheiro T, Monteiro S, Erlhagen W, Bicho E (2016) Multi-constrained joint transportation tasks by teams of autonomous mobile robots using a dynamical systems approach. In: 2016 IEEE international conference on robotics and automation (ICRA), pp. 3111–3117. IEEE
15. Marino A (2017) Distributed adaptive control of networked cooperative mobile manipulators. IEEE Trans Control Syst Technol 26(5):1646–1660
16. Martínez-Soto R, Castillo O, Castro JR (2014) Genetic algorithm optimization for type-2 non-singleton fuzzy logic controllers. Recent Adv Hybrid Approaches Des Intell Syst. https://doi.org/10.1007/978-3-319-05170-3_1
17. Omrane H, Masmoudi MS, Masmoudi M (2016) Fuzzy logic based control for autonomous mobile robot navigation. Comput Intell Neurosci. https://doi.org/10.1155/2016/9548482
18. Pandey A, Parhi DR (2017) Optimum path planning mobile robot in unknown static and dynamic environments using fuzzy-wind driven optimization algorithm. Def Technol 13:47–58
19. Petitti A, Franchi A, Di Paola D, Rizzo A (2016) Decentralized motion control for cooperative manipulation with a team of networked mobile manipulators. In: 2016 IEEE international conference on robotics and automation (ICRA), pp. 441–446. IEEE
20. Ren W, Cao Y (2010) Distributed coordination of multi-agent networks: emergent problems, models, and issues. Springer, Berlin
21. Sathyan A, Ma O (2019) Collaborative control of multiple robots using genetic fuzzy systems. Robotica 37(11):1922–1936
22. Singh NH, Thongam K (2017) Fuzzy logic-genetic algorithm-neural network for mobile robot navigation: a survey. Int Res J Eng Technol 4(8):24–45
23. Storn R, Price K (1997) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
24. Sun Y, Barth A, Ma O (2020) An intelligent approach for a two-robot team to perform a cooperative task. In: AIAA Scitech 2020 Forum. p. 1116
25. The Mathworks Inc. (R2019b) Matlab genetic algorithm and global optimaztion toolbox. https://www.mathworks.com/help/gads/ga.html
26. Tsai CC, Wu HL, Tai FC, Chen,YS (2016) Decentralized cooperative transportation with obstacle avoidance using fuzzy wavelet neural networks for uncertain networked omnidirectional multi-robots. In: 2016 12th IEEE international conference on control and automation (ICCA), pp. 978–983. IEEE
27. Tsiamis A, Bechlioulis CP, Karras GC, Kyriakopoulos KJ (2015) Decentralized object transportation by two nonholonomic mobile robots exploiting onlyh implicit communication. In: 2015 IEEE international conference on robotics and automation. pp. 171–176
28. Wang Z, Schwager M (2016) Kinematic multi-robot manipulation with no communication using force feedback. In: 2016 IEEE international conference on robotics and automation (ICRA), pp. 427–432. IEEE
29. Zadeh LA, Klir GJ, Yuan B (1996) Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers. World Scientific, Singapore