frontiers in Artificial Intelligence

METHODS
published: 07 August 2020
doi: 10.3389/frai.2020.00050

Intelligent Multirobot Navigation and Arrival-Time Control Using a Scalable PSO-Optimized Hierarchical Controller

Yu-Cheng Chang[1]*, *Anna Dostovalova*[2], *Chin-Teng Lin*[1] and *Jijoong Kim*[2]

[1] Computational Intelligence and Brain Computer Interface (CIBCI) Lab, Centre for Artificial Intelligence (CAI), University of Technology, Sydney, NSW, Australia, [2] Defence Science & Technology Group, Adelaide, SA, Australia

We present a hierarchical fuzzy logic system for precision coordination of multiple mobile agents such that they achieve simultaneous arrival at their destination positions in a cluttered urban environment. We assume that each agent is equipped with a 2D scanning Lidar to make movement decisions based on local distance and bearing information. Two solution approaches are considered and compared. Both of them are structured around a hierarchical arrangement of control modules to enable synchronization of the agents' arrival times while avoiding collision with obstacles. The proposed control module controls both moving speeds and directions of the robots to achieve the simultaneous target-reaching task. The control system consists of two levels: the lower-level individual navigation control for obstacle avoidance and the higher-level coordination control to ensure the same time of arrival for all robots at their target. The first approach is based on cascading fuzzy logic controllers, and the second approach considers the use of a Long Short-Term Memory recurrent neural network module alongside fuzzy logic controllers. The parameters of all the controllers are optimized using the particle swarm optimization algorithm. To increase the scalability of the proposed control modules, an interpolation method is introduced to determine the velocity scaling factors and the searching directions of the robots. A physics-based simulator, Webots, is used as a training and testing environment for the two learning models to facilitate the deployment of codes to hardware, which will be conducted in the next phase of our research.

Keywords: hierarchical fuzzy system, fuzzy logic control, multi-agent control, navigation, arrival-time control

OPEN ACCESS

Edited by:
Cesar Isaza, Polytechnic University of Querétaro, Mexico

Reviewed by:
Vicente García, Universidad Autónoma de Ciudad Juárez, Mexico
Jonny Paul Zavala De Paz, Polytechnic University of Querétaro, Mexico

***Correspondence:**
Yu-Cheng Chang
Yu-Cheng.Chang@student.uts.edu.au

Specialty section:
This article was submitted to Fuzzy Systems, a section of the journal Frontiers in Artificial Intelligence

Received: *04 October 2019*
Accepted: *15 June 2020*
Published: *07 August 2020*

Citation:
Chang Y-C, Dostovalova A, Lin C-T and Kim J (2020) Intelligent Multirobot Navigation and Arrival-Time Control Using a Scalable PSO-Optimized Hierarchical Controller. Front. Artif. Intell. 3:50. doi: 10.3389/frai.2020.00050

INTRODUCTION

Mobile robot control has been widely used in automated navigation system. The aim of the automated navigation is to guide the robot or vehicle moving between obstacles to reach the target from the start point with collision-free performance (Kashyap and Pandey, 2018; Patle et al., 2019). During the navigation control, observations are made by the sensors and actuators equipped on a real robot or vehicle; the input signal is noisy and uncertain. Therefore, fuzzy logic system (FLS) (Zhu and Yang, 2007; Juang and Chang, 2011; Pothal and Parhi, 2015; Lai et al., 2016; Din et al., 2018; Jhang et al., 2019; Mohanta and Keshari, 2019; Pradhan et al., 2019) has been used in an automated navigation takes in order to enhance the robot control quality. Fuzzy logic systems (FLSs) provides a robust solution with anti-noise ability to defeat the uncertainty. However, the performance of fuzzy

logic system depends on the design of membership function and efficient rules, which often takes a considerable amount of time to analyze the experimental input and output data. Machine learning technology, therefore, has been used for fuzzy system design. Zhu and Yang (2007) and Pothal and Parhi (2015), respectively, exploit supervised learning to train neuro-fuzzy model for single and multiple robots to perform navigation task. The precise input–output training data should be collected in advance for supervised learning. To reduce the training effort, evolutionary algorithms have been used to design FLS. Two popular optimization algorithms are genetic algorithms (GAs) (Chia-Feng, 2005; Mansoori et al., 2008; Nantogma et al., 2019; Pradhan et al., 2019) and particle swarm optimization (PSO) (Juang and Lo, 2008; Juang and Chang, 2011; Ding et al., 2019). These two methods can be easily applied to the design of FLS since it can be formulated as an optimization problem by defining a metric for solution performance evaluation.

Apart from single-agent navigation, the last few years have seen an increase in research topic (Pothal and Parhi, 2015; Misra et al., 2018; Babel, 2019; Chandrasekhar Rao and Kabat, 2019; Yao and Qi, 2019) for multirobot coordination across multiple disciplines. The multirobot coordination entails time synchronization among individuals to accomplish a given task. This study considers time-arrival control during the multiple robots performing navigation task; the robots not only need to move toward their targets with collision-free motion in a timely manner. Yao and Qi (2019) propose a novel dynamical model to adjust the path length and voyage speed of each autonomous underwater vehicle (AUV) to achieve the simultaneous arrival at destination between multi-AUVs. Misra et al. (2018) combine cooperative localization technique with proportional navigation (PN) guidance law to manipulate multiple unmanned vehicles to simultaneously reach a moving target in GPS-denied environment. Babel (Lin and Lee, 1996; Chia-Feng and Chin-Teng, 1998) develops a multi-agent path planning algorithm considering shortest paths between all pairs of air vehicles and targets, target allocation, and concatenating feasible and suitable short path, which guarantees that all UAVs arrive at the targets in a timely manner and without the risk of mutual collision.

This paper developed a fuzzy-based control system that has two levels: the lower-level individual navigation control for obstacle avoidance and the higher-level coordination to ensure the same time of arrival for all robots at their destination points. An FLS methodology (Lin and Lee, 1996; Chia-Feng and Chin-Teng, 1998; Juang and Chang, 2011), combining the Takagi-Sugeno-Kang (TSK) type of a fuzzy inference system with a derivative-free global optimization technique, is used to design the fuzzy "IF–THEN" rules and tune the parameters of membership functions. The controllers are trained in a cascading manner. In the first phase of training, we employ the particle swarm optimization (PSO) algorithm (Shi and Eberhart, 1998) to optimize the fuzzy rules that comprise individual navigation control. The role of this controller is to generate the motion direction command that steers the robot away from obstacles but toward the target location based on the robot's sensory inputs (each robot is equipped with a laser ranger) and information about target location. In the second phase of training, the same technique is used for multiple robots to learn how to coordinate with each other to reach their targets at the same time. The coordination controller controls both moving speeds and moving direction of each robot to achieve the simultaneous target reaching task. We developed a fuzzy-logic-based coordinator and a recurrent-based coordinator. The fuzzy-logic-based coordinator is implemented by a PSO-optimized FLS. The recurrent-based coordinator includes a PSO-trained long short-term memory (LSTM) block (Greff et al., 2017). Webots software (Webots: Robot simulator) has been used as a physics-based robot simulation environment for the training and testing of the proposed solutions.

THE PROPOSED MODELS

This section describes the proposed models for multirobot navigation and arrival-time control. **Figure 1** shows the configuration. There are two hierarchical levels of control module. The lower-level controllers are the robot navigation controllers (RNCs). They enable each robot to perform collision-free navigation. The higher-level controller is the multiple robot coordinator (MRC), which coordinates the robots' speed and direction so that they reach their targets at the same time.

In the proposed model, each RNC controls a robot, and all RNCs share an identical structure and a set of parameters. The RNC receives the adjustment angle, i.e., $\theta_{adj}(t)$, from the output of MRC, the distances between the robot and nearby obstacles, i.e., $\vec{L}(t)$ from a 2D Lidar and the direction angle to the target from the robot, i.e., $\theta_{goal}(t)$. Specifically, a 2D Lidar rangefinder on the front of the robot scans from 0 to 180° and outputs $\vec{L}(t) = (L_1, \ldots, L_8)$, which are the minimum distances to any obstacle in each of eight sectors (**Figure 2**). The output of an RNC is the motion direction of the robot. To achieve collision-free navigation, a fuzzy logic controller (FLC) was added to the RNC. The robot avoids obstacles using boundary-following (BF) behavior. A behavior supervisor (BS) decides what the robot should do at each control time step.

The MRC is a centralized controller to determine the speed and direction of each robot for the next control time step with five inputs: $D_{goal}^{rank_1}(t-1)$, $D_{goal}^{rank_n}(t-1)$, $v_r^{rank_1}(t-1)$, $v_r^{rank_n}(t-1)$, and $\Delta D(t-1)$. For each control loop, the robots are ranked in ascending distance from the target. Accordingly, $robot_{rank_1}$ is the closest robot to the target, and $robot_{rank_n}$ is the robot farthest away. $D_{goal}^{rank_1}$ and $D_{goal}^{rank_n}$ are, respectively, the distances from $robot_{rank_1}$ and $robot_{rank_n}$to the target. $v_r^{rank_1}$ and $v_r^{rank_n}$ are, respectively, the speeds of $robot_{rank_1}$ and $robot_{rank_n}$. Finally, $\Delta D(t-1) = D_{goal}^{rank_n}(t-1) - D_{goal}^{rank_1}(t-1)$. The outputs of MRC to the RNCs are the speed $v_r(t)$ and heading angle $\theta_{adj}(t)$ for robot, $rank_1, \ldots, rank_n$. We developed two types of MRC, one with a fuzzy-logic-based model and the other with a recurrent-based model. The fuzzy-logic-based MRC is implemented by an FLC, while the recurrent-based model uses long short-term memory (LSTM). Details of the proposed models are introduced in the following sections.

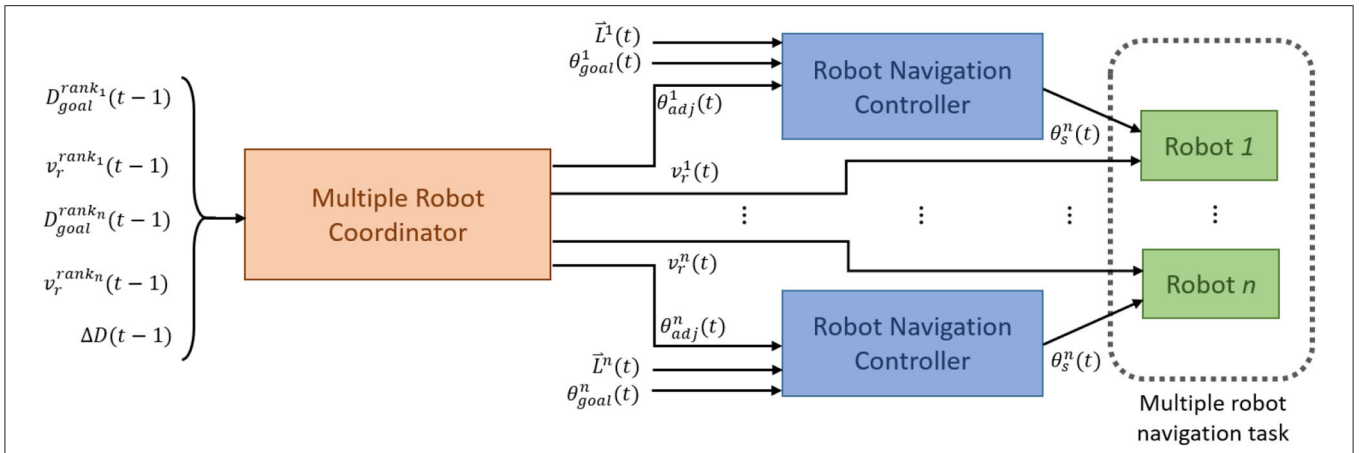

FIGURE 1 | The block diagram of the control configuration for multirobot navigation and arrival-time control.

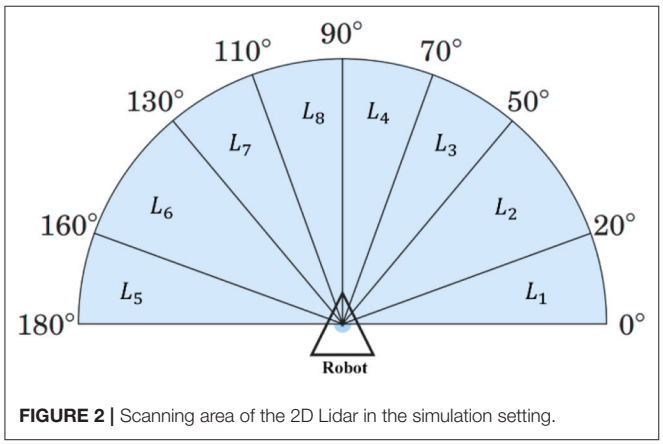

FIGURE 2 | Scanning area of the 2D Lidar in the simulation setting.

Fuzzy-Logic-Based Multiple Robot Coordinator

The proposed fuzzy-logic-based MRC is responsible for speed regulation and heading angle adjustment. The robot speed is changed at each control time step. If a robot is much closer to the target than the other robots, heading angle adjustment is made so that it moves away from the target; otherwise it would need to stop and wait for the others.

Speed Regulation

For each control loop, n robots are ranked based on their distance to the target in ascending order. An FLC called FLC_{SR} is used for robot speed regulation, which directly controls $robot_{rank_1}$ and $robot_{rank_n}$. The outputs of FLC_{SR} are speed factors α_1 and α_n for these two robots, which are used to increase or decrease their speeds. For the remaining robots ($robot_{rank_2}, \ldots, robot_{rank_{(n-1)}}$), the speed scale factors $\alpha_2, \ldots, \alpha_{(n-1)}$ are generated by the following interpolation process:

$$\alpha_i = \frac{i-1}{n-1} \cdot (\alpha_n - \alpha_1) + \alpha_1, \tag{1}$$

where $i = 2, \ldots, n-1$, and $\alpha_i \in [0.5, 1.5]$. The speeds have upper and lower bounds, which define a safe operating region.

The robots are not allowed to stop. The speeds for the robots at control time step t are given by

$$v_r^i(t) = \alpha_i v_r^i(t-1),\ i = 1, \ldots, n. \tag{2}$$

FLC_{SR} uses zero-order Takagi–Sugeno–Kang (TSK) fuzzy IF–THEN rules with the form

$$R_i^{SR}: \text{If } x_1 \text{ is } A_{i1} \text{ And } \ldots \text{ And } x_5 \text{ is } A_{i5} \text{ Then } y \text{ is } \vec{a}_i \tag{3}$$

where $x_1, \ldots, x_5$ correspond to the input variables; $D_{goal}^{rank_1}(t-1)$, $D_{goal}^{rank_n}(t-1)$, $v_r^{rank_1}(t-1)$, $v_r^{rank_n}(t-1)$, and $D(t-1)$; $A_{i1}, \ldots, A_{i5}$ are fuzzy sets; and $\vec{a}_i = (a_{i1}, a_{i2})$ is a real vector. Here we use a Gaussian membership function. Thus, A_{ij} is given by

$$\mu_{ij}(x_i) = exp\left\{-\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right\} \tag{4}$$

where m_{ij} and σ_{ij} represent the center and the width of the fuzzy set A_{ij}, respectively. The firing strength of rule R_i^{SR} is obtained by implementing the following algebraic product:

$$\Phi_i = \prod_{j=1}^{M} \mu_{ij}(x_i), \tag{5}$$

where M is the dimension of the input variable, e.g., $M = 5$.

Suppose that FLC_{SR} has r rules. An output $\vec{y} = (y_1, y_2) = (\alpha_1, \alpha_n)$ can be obtained using the weighted average defuzzification method:

$$\vec{y} = \frac{\sum_{i=1}^{r} \Phi_i \vec{a}_i}{\sum_{i=1}^{r} \Phi_i}. \tag{6}$$

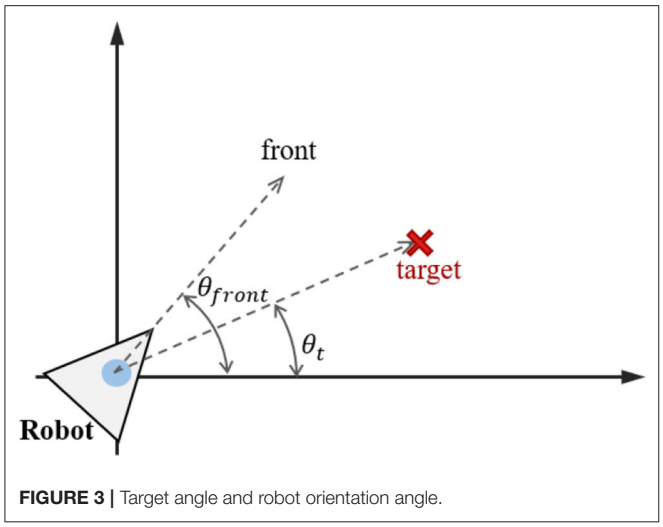

FIGURE 3 | Target angle and robot orientation angle.

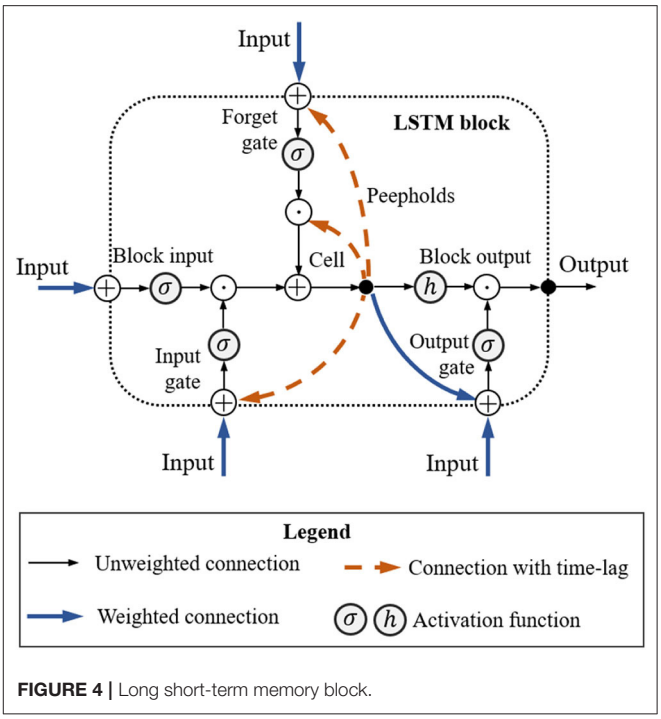

FIGURE 4 | Long short-term memory block.

Heading Angle Adjustment

To make the length of the robot paths roughly equal, the MRC adjusts each robot's heading angle as part of its arrival-time control. An adjusted heading angle for robot i at control time step t is calculated by

$$\theta^i_{adj}(t) = \theta^i_{goal}(t) + \beta_i \cdot \theta_{\max\_adj}, \tag{7}$$

where β_i is a scale factor that determines the strength of heading angle adjustment, θ^i_{goal} is the search angle for robot i, and $\theta_{\max\_adj}$ is the maximum angle in changing the direction of robot i, e.g., $\theta_{\max\_adj} = 90$. $\theta^i_{goal} = \theta_t - \theta^i_{front}$ is the deviation between the target angle θ_t and the robot orientation angle θ^i_{front}, as shown in **Figure 3**. β_i is calculated from $D^{rank_n}_{goal}$, D^i_{goal} and α_i, as follows:

$$\beta_i = \sqrt{\frac{1}{\alpha_i}\left(1 - \frac{D^i_{goal}(t-1)}{D^{rank_n}_{goal}(t-1)}\right)} \tag{8}$$

As β_i increases, robot i will be guided away from the target; on the other hand, as β_i decreases, robot i will be guided toward the target. This adjustment keeps changing the search behavior of each robot, except for the furthest $robot_{rank_n}$, until either $(D^{rank_n}_{goal} - D^{rank_1}_{goal}) < 0.1\ m$ or each robot is within 10 m of the target. Algorithm 1 is an overview of the fuzzy-logic-based MRC for the multirobot navigation and arrival-time control.

Recurrent-Based Multiple Robot Coordinator

In addition to fuzzy-logic-based MRC, we also consider an LSTM-based model because these perform well for problems involving sequential data with long time dependencies. Its memory mechanism allows the use of historical data, which could be useful for optimizing trajectory-related problems. The vanilla version of LSTM is used because it is simple to implement, and its performance is close to that of other variants. **Figure 4** shows the architecture of the LSTM block. In the recurrent-based MRC configuration, we use two LSTM blocks, with input (W_z, W_i, W_f, W_o), recurrent (R_z, R_i, R_f, R_o), peephole (p_i, p_f, p_o), and bias (b_z, b_i, b_i, b_o) weights. The input/output interface of the LSTM controller matches that of the fuzzy-logic-based MRC. Given input $x^k = (D^{rank_1}_{goal}(t-1),\ D^{rank_n}_{goal}(t-1)\ v^{rank_1}_r(t-1),\ v^{rank_n}_r(t-1),\ \Delta D(t-1))$, then the LSTM block forward pass is

$$\text{Block input}: z^k = h\left(W_z \mathrm{x}^k + R_z \mathrm{y}^{k-1} + b_z\right), \tag{9}$$

$$\text{Input gate}: i^k = \sigma\left(W_i \mathrm{x}^k + R_i \mathrm{y}^{k-1} + p_i \odot c^{k-1} + b_i\right) \tag{10}$$

$$\text{Forget gate}: f^k = \sigma\left(W_f \mathrm{x}^k + R_f \mathrm{y}^{k-1} + p_f \odot c^{k-1} + b_i\right), \tag{11}$$

$$\text{Cell}: c^k = z^k \odot i^k + c^{k-1} \odot f^k, \tag{12}$$

$$\text{Output gate}: o^k = \sigma\left(W_o \mathrm{x}^k + R_o \mathrm{y}^{k-1} + p_o \odot c^k + b_o\right), \tag{13}$$

$$\text{Block output}: y^k = h\left(c^k\right) \odot o^k, \tag{14}$$

where σ is the logistic sigmoid function used for gate activation, and h is the hyperbolic tangent function for the block input/output activation. During the training process, all bias weights are set to 0.5.

Robot Behavior Controller

The robot behavior controller (RBC) controls a robot to avoid obstacles by performing left BF behavior or right BF behavior. It consists of a behavior supervisor (BS) to determine behavior according to its current position, target position, and real-time outputs from the 2D Lidar sensor.

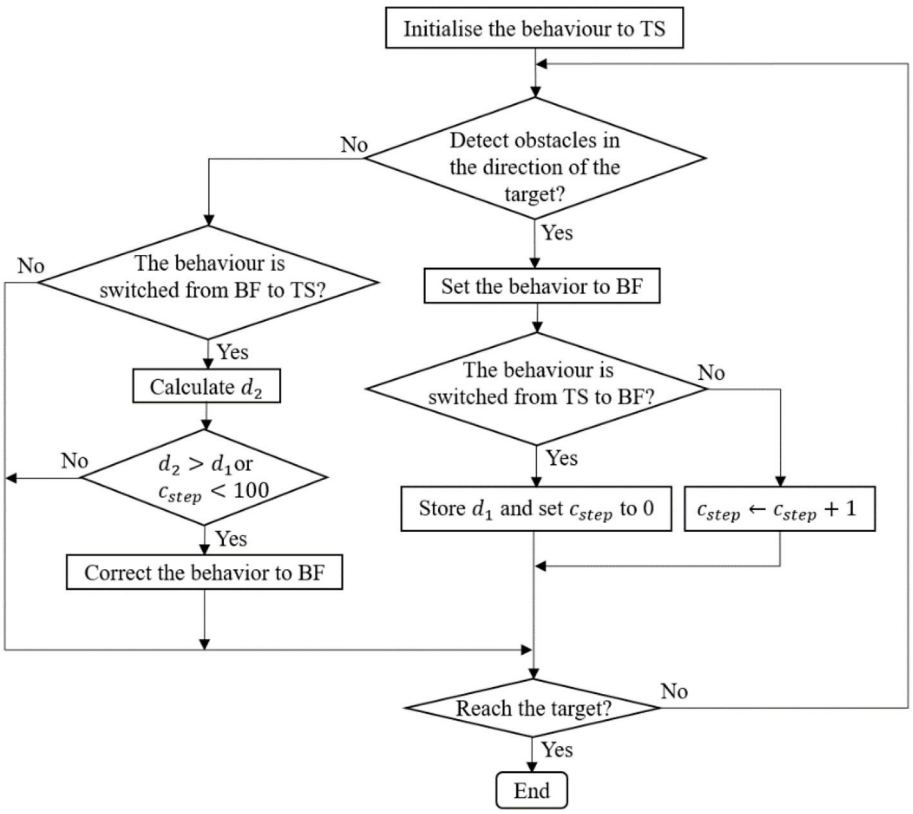

FIGURE 5 | The block diagram of the navigation control.

Behavior Supervisor

In the simulation for robot navigation, the mobile robot is equipped with a 2D Lidar sensor that scans the area in front of the robot from right (0±) to left (180±). The coverage area is divided into eight sectors $L_1,\ldots,L_8$, as shown in **Figure 2**. The behavior supervisor uses a simple logic proposed in Juang and Chang (2011) to switch between the target-searching (TS) behavior and left and right BF behavior. If there are no obstacles detected within the sensing range of the robot's Lidar, then the robot starts moving directly toward the target. **Figure 5** shows the logic of the behavior selection based on the robot-target distance and time-step counter. When the robot switches behavior from target searching (TS) to BF, the distance d_1 between the robot and the target is recorded, and the step counter c_{step} is set to zero. At the location where the robot decides to switch the behavior from BF to TS, the distance d_2 between the robot and the target is calculated. If $d_1 > d_2$, or if the step counter $c_{step} > 100$, the robot keeps the original BF behavior; otherwise, the robot switches from BF to TS. This time-step constraint prevents the robot from immediately switching between the TS and BF behaviors.

The control of a robot performing BF in the navigation task is implemented by two fuzzy controllers—a left BF controller and a right BF controller. The left BF controller controls when the robot is closing an obstacle at the left-hand-side region, whereas the right BF controller is for the right-hand-side region. The number of rules in the right BF controller is identical to that in the left BF controller. The rules for the right BF behavior share the same antecedent part with those for the left BF behavior except that the left sensor inputs $L_5,\ldots,L_8$ are changed to the right sensor inputs $L_1,\ldots,L_4$. For the rule consequent part, the steering angle in each rule for the right BF behavior is simply a reverse of that for the left BF behavior. For example, suppose that the ith rule in the left BF controller is represented as follows:

$$R_i^{left}: \text{If } L_5 \textit{ is } B_{i1} \text{ and } L_6 \textit{ is } B_{i2} \ \ldots \text{ and } L_8 \textit{ is } B_{i4} \text{ Then } \theta_{BF} \text{ is } \theta_i. \tag{15}$$

Then, the corresponding rule for the right BF controller is

$$R_i^{right}: \text{If } L_1 \textit{ is } B_{i1} \text{ and } L_2 \textit{ is } B_{i2} \ \ldots \text{ and } L_4 \textit{ is } B_{i4} \text{ Then } \theta_{BF} \text{ is } -\theta_i, \tag{16}$$

where $B_{i1},\ldots,B_{i4}$ are a fuzzy sets defined by a Gaussian membership function and given as Equation (4). The output of the left BF controller is computed as Equation (6) with a singleton consequent value $a_i = \theta_i$; and similarly, the output of the right BF controller is with a consequent value $a_i = -\theta_i$. During a navigation task, the robot should decide to carry out either the left or right BF behavior at each control time step.

Algorithm 1 | Pseudocode for the multirobot navigation and arrival-time control.

Initialize the robots
for all robots i **do**
  Get initial position
  Set the position of the target
  Set initial moving speed to 0.5 m/s
end for
Main control loop
while stop conditions have not been met **do**
for all robots i **do**
  $L_1^i, \ldots, L_8^i \leftarrow$ Lidar output
  Get distance to the target D_{goal}^i
end for
Rank all robots according to D_{goal}^i
$\Delta D \leftarrow D_{goal}^{rank_n} - D_{goal}^{rank_1}$
$\alpha_1,\ \alpha_n \leftarrow FLC_{SR}(v_{r1}, v_{r2}, D_{g1}, D_{gn}, \Delta D)$
for robot i **from** $robot_{rank_2}$ to $robot_{rank_{(n-1)}}$
  $\alpha_i \leftarrow$ equation (1)
end for
for all robots i **do**
  $v_r^i \leftarrow$ equation (2)
  if (robot i is performing TS behavior **and**
  $(\Delta D > 0.1\ m$ **or** $D_{goal}^{rank_n} > 10\ m))$
    $\beta_i \leftarrow$ equation (8)
    $\theta_{adj}^i \leftarrow$ equation (7)
  else
  $\theta_{adj}^i \leftarrow \theta_{goal}^i$
  end if
end for
for all robots i **do**
  Steering angle $\theta_r^i \leftarrow RBC(L_1^i, \ldots, L_8^i, D_{goal}^i, \theta_{adj}^i)$
end for
end while

TRAINING STRATEGY AND SIMULATION CONFIGURATION

In this study, both fuzzy-logic-based MRC and recurrent-based MRC are trained in a cascading manner. First, we train the BF controller in RBC to perform collision-free navigation toward the target. In the second phase of training, fuzzy-logic-based MRC and recurrent-based MRC learn to coordinate a group of RBC-equipped robots to arrive at a target at the same time. The particle swarm optimization (PSO) algorithm (Shi and Eberhart, 1998) is used to optimize the tunable parameters of all controllers.

Particle Swarm Optimization

PSO is a swarm intelligence optimization approach in which each solution is represented as a particle [3]. Each particle has a position, represented by vector s_i. The swarm in PSO is initialized with a population of random solutions. A swarm of particles moves through the solution space, and the velocity of each particle is represented by vector v_i. The performance of a particle is measured by a fitness function f, which is evaluated using s_i. Each particle keeps track of its own best position p_i, which is associated with the best fitness that the particle has achieved. Also, it is guided toward the best position found by any member of the swarm (the global best position g). For particle i at iteration t, each element k of the new velocity can be calculated as

$$v_i^{(t)}(k) = wv_i^{(t-1)}(k) + c_1 r_1 \left(p_i(k) - s_i^{(t-1)}(k)\right) + c_2 r_2 \left(g(k) - s_i^{(t-1)}(k)\right), \quad (17)$$

where w is the inertia weight, c_1 and c_2 are positive acceleration coefficients, and r_1 and r_2 are uniformly distributed random numbers in the interval [0, 1]. All components of v_i have lower and upper bounds defined by the geometry of the search space. The new position of each particle is calculated with

$$s_i^{(t)}(k) = s_i^{(t-1)}(k) + v_i^{(t)}(k). \quad (18)$$

With a careful choice of parameters w, c_1, and c_2, Equations (17) and (18) ensure that the particle population clusters around the best solution.

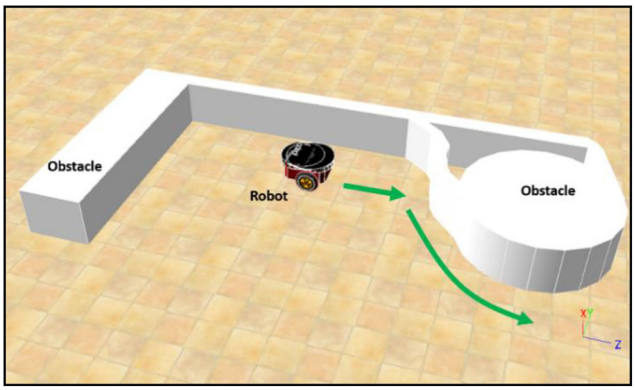

FIGURE 6 | The environment for training phase 1.

Training Phase 1: BF Behavior

Figure 6 illustrates the environment for training phase 1. The main goal of this phase is to control the robot in BF behavior at a constant speed using the PSO-based fuzzy controller. Without loss of generality, this value is set as 0.4 m/s in this paper. The BF behavior enables collision-free movement of the robot during navigation. Since only the left BF controller is trained, the distances detected by sectors L_5, L_6, L_7, and L_8 are used and fed as the inputs to the left BF controller. The right BF behavior is directly available by a slight modification of the learned consequents for the left BF behavior. The left BF controller output is the steering angle of the robot boundary-following behavior θ_{BF}, where $\theta_{BF} \in [-3.14,\ 3.14]$ in radians. A positive value of θ_{BF} means a clockwise rotation.

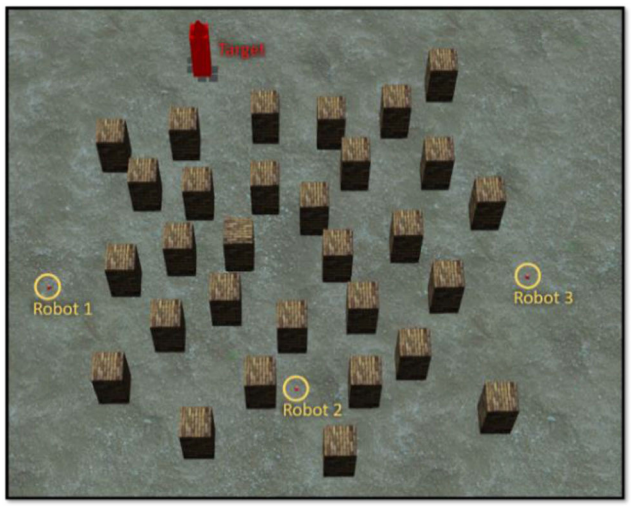

FIGURE 7 | Environment for training phase 2.

The constraints for successful left BF behavior at each time step during the learning process are

$$\min(L_5, L_6, L_7, L_8) > D_{min}, \text{and } L_5 \leq D_{max} \tag{19}$$

In this simulation, D_{min} and D_{max} are set to 0.5 and 1.5, respectively. The first constraint prevents a collision with the object, and the second constraint prevents the robot from moving too far from the object. In the PSO-optimized training phase 1, a particle represents a whole fuzzy controller for left BF behavior. The performance of left BF behavior is evaluated as follows. The robot moves along the side of an object and stops when one of the constraints in (19) is violated, which indicates that the controller has failed. If the robot stops, the total number of control time steps is recorded as $T_{control}$. The fitness function f_{phase_1} for training phase 1 is

$$f_{phase\_1} = \frac{1}{T_{control}}. \tag{20}$$

A low f_{phase_1} indicates good left BF behavior. The control process from when movement starts to when it stops is called a trial. If left BF behavior fails, the robot moves back to its initial position for the next trial, and a new fuzzy controller is constructed and evaluated. The learning process is repeated until a successful fuzzy controller is found or the maximum number of iterations is met. A left BF behavior is deemed successful if it successfully controls the robot for a total of T_{suc} time steps. In the training phase 1, T_{suc} is set to 4,000 so that the robot moves along the object boundary for over two cycles. The maximum number of iterations is set to 200 for a trial.

Training Phase 2: Multi-Robot Navigation and Arrival-Time Coordination Learning

In training phase 2, there are three robots moving in a complex environment, as shown in **Figure 7**. Each robot is controlled by the BRC whose BF controller was optimized in training phase 1. During training, both fuzzy-logic-based MRC and recurrent-based MRC are applied in the navigation of three robots so that they reach the target simultaneously. The robots start from different positions and head toward the same target. The performance of MRC is evaluated using a fitness function f_{phase_2}:

$$f_{phase\_2} = 10 \cdot f_1 + 0.1 \cdot f_2. \tag{21}$$

The first term of (21), f_1, is used to optimize the difference in the arrival times of $robot_{rank_1}$ and $robot_{rank_n}$:

$$f_1 = \left| T_{rank_1} - T_{rank_n} \right|, \tag{22}$$

where T_{rank_1} is the time that $robot_{rank_1}$ takes to reach the target, and T_{rank_n} is the time for $robot_{rank_n}$.

The second term of (21), f_2, is to get the robot to move as fast as possible:

$$f_2 = \frac{\left| T_{rank_1} - T_{rank_n} \right|}{2}. \tag{23}$$

SIMULATION RESULTS

Simulation 1 (BF Behavior Learning)

This example shows the simulation results of training phase 1: left BF learning result using the PSO-optimized FLC. The number of fuzzy rules is set to 10. The simulation environment is as shown in **Figure 6**. The environment is built using Webots 8.5.3 on a platform equipped with Intel i5-4200H 3.40 GHz CPU, NVIDIA GT 745M 2GB graphics card, and 8G 1600 MHz RAM. The learning objective is to find a successful FLC for left BF behavior satisfying the constraints in (19) for a total of 4000 time steps. The control loop stops when the robot violates the constraint of the left-BF FLC. For this optimization problem, the objective is to design a successful FLC using as minimum number of iterations as possible. **Figure 8** shows the left-BF behavior-learning results for all 50 runs. The PSO fails to find a successful left-BF FLC for one of the 50 runs. The average number of iterations of the PSO to find a successful FLC is 13.987.

Simulation 2 (Multirobot Navigation and Arrival-Time Coordination Learning)

The objective of this simulation is to optimize MRC for three robots to navigate and simultaneously reach the target in the clutter environment as shown in **Figure 7**. We set the centre point of the map as the origin of coordinate $(x, z) = (0, 0)$, and the target is located at $(x, z) = (-11, -23)$. The initial distance between $robot_1$ is 30.4 m, $robot_2$ is 39.5 m, and $robot_3$ is 43.8 m. The performance of both fuzzy-logic-based MRC and recurrent-based MRC are evaluated by Equation (21). A smaller solution to Equation (21) means that the MRC can control the three robots moving toward to the target as fast as possible and coordinate their arrival time as precise as well. For each evaluation process, the MRC controls all robots until they all reach the target. Once all robots have reached the target, the positions of the robots will be set to their initial positions, which are fixed during the whole training process. The number of learning iterations is set to 50.

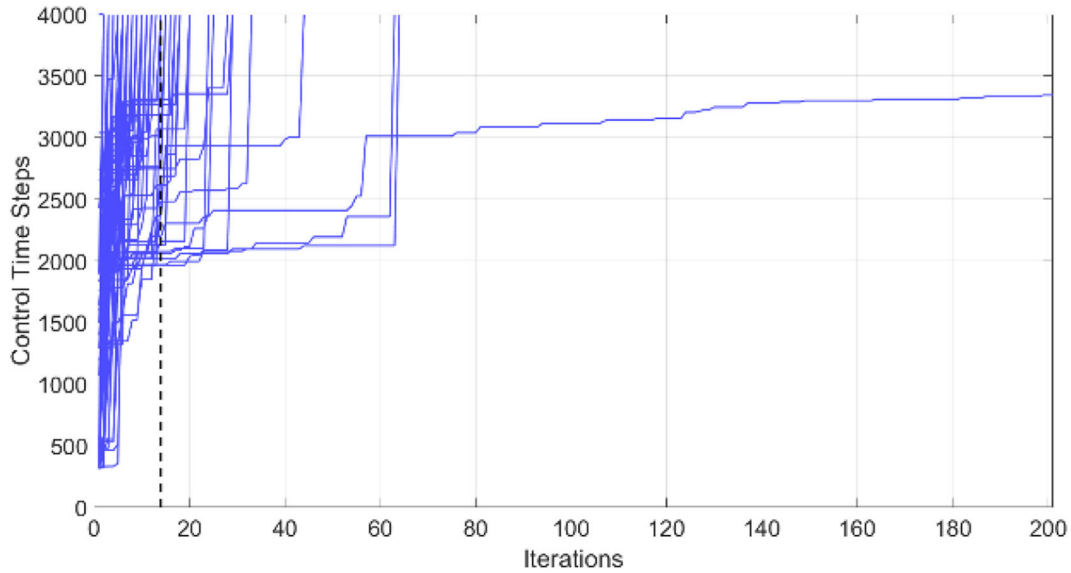

FIGURE 8 | Robot left-BF behavior-learning results for all 50 runs.

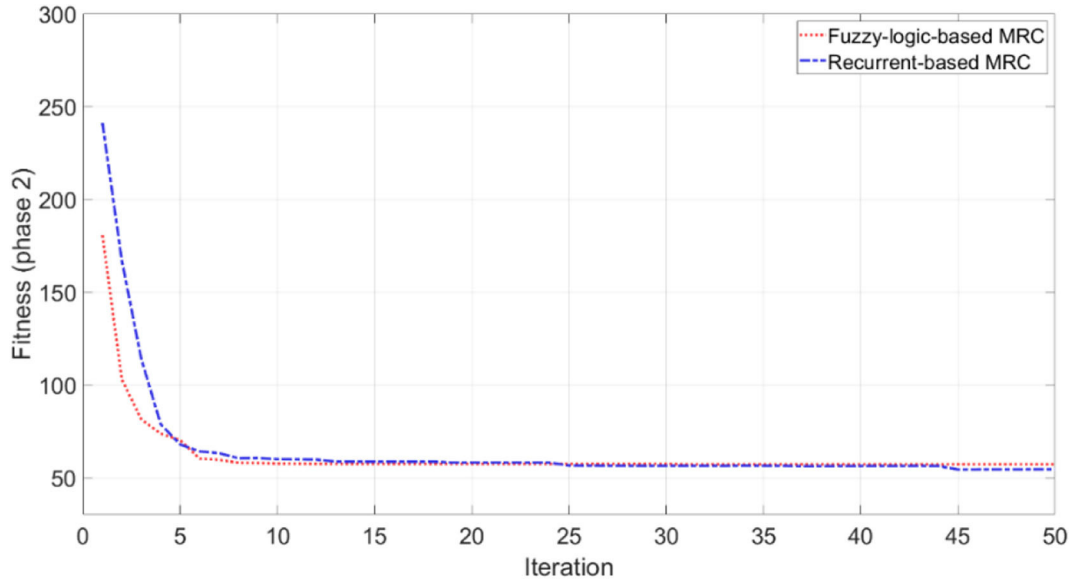

FIGURE 9 | Average best-so-far fitness value at each iteration for the MRCs during the training phase 2.

The PSO optimization process for training phase 2 conducts 50 runs for statistical evaluation. **Figure 9** demonstrates the average best-so-far fitness of the MRCs. **Table 1** presents the performance of fuzzy-logic-based MRC and recurrent-based MRC in training phase 2. The average best-so-far fitness of fuzzy-logic-based MRC converges at 57.2 (average value of f_1 is 532.0 time steps, average value of f_2 is 0.41 time steps), while the recurrent-based is 54.46 (average value of f_1 is 524.62 time steps, average value of f_2 is 0.20 time steps).

Simulation 3 (Multirobot Navigation and Arrival-Time Coordination)

In this simulation, the optimized MRCs and BF controller are applied to perform navigation and arrival-time task. We deployed three robots and six robots in a clutter environment with variant starting position to testify the scalability of the optimized MRCs. Both fuzzy-logic-based and recurrent-based MRCs are used in this simulation. To demonstrate the ability of time-arrival coordination, examples of robots controlled without MRC

are included for comparison with those examples applied with MRCs.

TABLE 1 | Performance of fuzzy-logic-based and recurrent-based MRC in the training phase 2.

		f_{phase_2}	f_1	f_2
Fuzzy-logic-based MRC	Average	57.73	532.0	0.41
	STD	2.97	29.85	0.068
Recurrent-based MRC	Average	54.46	524.62	0.20
	STD	3.31	32.42	0.053

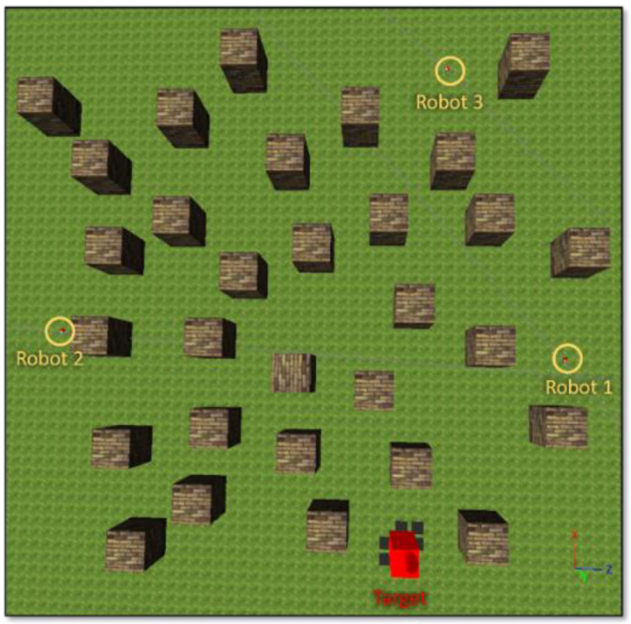

FIGURE 10 | An environment setting for three-robot navigation.

Three-Robot Navigation and Arrival Time Control

To validate the performance of the proposed control systems, we deployed three robots in a complex environment (see **Figure 10**), and the target building is set at $(x, z) = (-19.54, 8.78)$. The initial distance between the $robot_1$ is 19.34 m, $robot_2$ is 32.94 m, and $robot_3$ is 39.05 m. **Figure 11** illustrates the trajectories of the three robots, respectively, controlled by fuzzy-logic-based MRC, recurrent-based MRC, and in the absence of MRC during the navigation task. **Figure 12** shows the remaining distance between the target and the three robots. The performance of time arrival coordination with three-robot setting is shown in **Table 2**. The time difference is evaluated by measuring the difference of arrival time of the fastest robot and the slowest robot in the simulation. The best achieved time difference controlled by fuzzy-logic-based MRC is five time steps, while it takes 785 time steps for the slowest robot to complete the navigation task. The recurrent-based configuration showed a result with an eight-time-step difference between the first and last arriving robots, while it takes 778 time steps for the slowest robot to complete the navigation task. By comparing **Figures 12A,B**, the robots controlled by recurrent-based MRC have faster convergence speed, but fuzzy-logic-based has a better coordinating ability in this simulation. We further compare with the case without MRC control; see **Figures 11C**, **12C**. The three robots directly move toward to the target without changing their searching direction and the moving speed so that $robot_1$ arrives at the target much earlier than the other two robots.

Six-Robot Navigation and Arrival Time Control

The reason for us to use the simple interpolation method to decide the velocity scaling factors and searching direction of robots is to increase the scalability of the proposed methods such that they can deal with a different and changing number of robots without retraining the neural networks. To validate the scalability of the proposed methods, we deployed six robots in this simulation, as shown in **Figure 13**, and

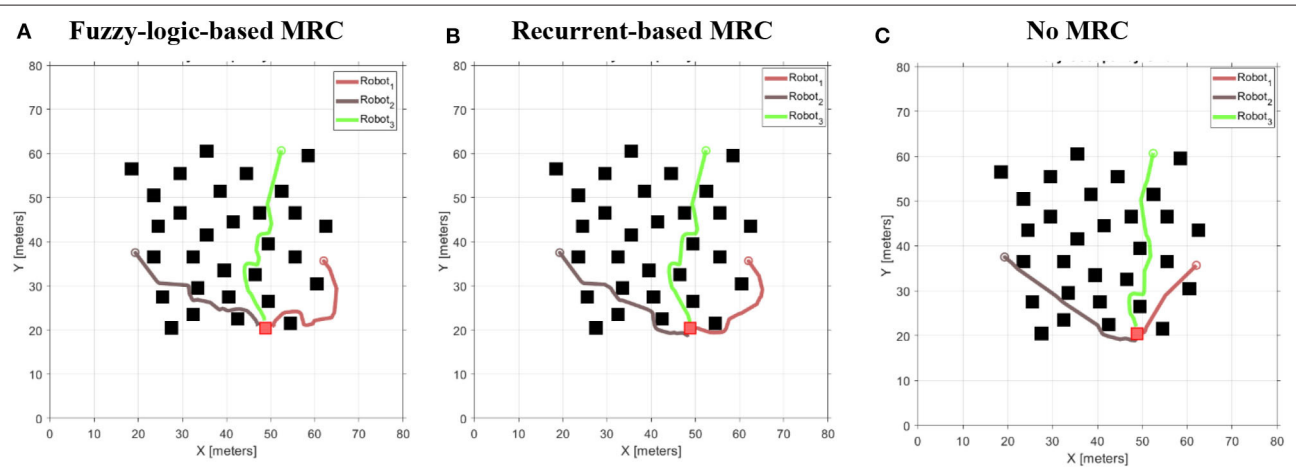

FIGURE 11 | The moving speeds of four-robot navigation in a complex environment setting. **(A)** Robots are controlled by the fuzzy-logic-based MRC. **(B)** Robots are controlled by the recurrent-based MRC. **(C)** Robots are controlled by only their own RBC.

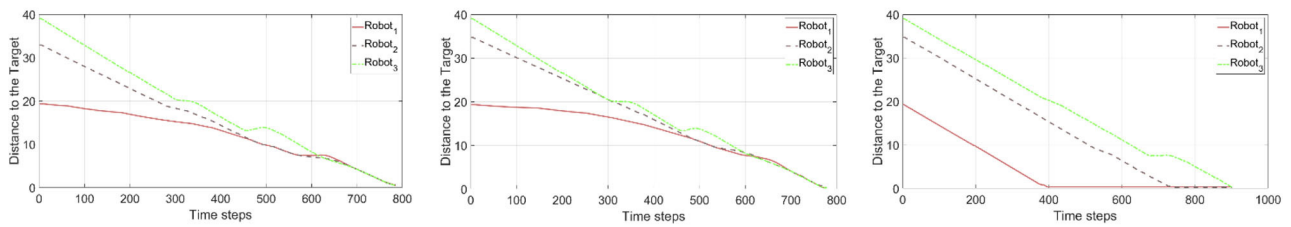

FIGURE 12 | The remaining distance between the target and the three robots during the three-robot navigation and arrival time control simulation. **(A)** Robots are controlled by the fuzzy-logic-based MRC. **(B)** Robots are controlled by the recurrent-based MRC. **(C)** Robots are controlled by only their own RBC.

TABLE 2 | The performance of time arrival coordination with three-robot setting.

Unit: Time step	Time difference	Time to complete task
Fuzzy-logic-based MRC	5	785
Recurrent-based MRC	8	778
No MRC	504	901

the target building is set at $(x, z) = (-17.71, -20.34)$. The initial distance between $robot_1$ is 50.11 m, $robot_2$ is 15.26 m, $robot_3$ is 49.44 m, $robot_4$ is 41.69 m, $robot_5$ is 24.08 m, and $robot_6$ is 34.27 m. Each proposed model uses identical parameters, which are used in the simulation of section IV-C.

The trajectories of the six-robot setting controlled by each proposed model are illustrated in **Figure 14**. **Figure 15** shows the remaining distance between the target and the six robots. **Table 3** presents performance of the arrival-time coordination in the six-robot setting. Fuzzy-logic-based MRC achieves 68-time-step time difference between the fastest robot and the slowest robot, while it takes 965 time steps for the slowest robot to complete the navigation task. The recurrent-based MRC has a better result with a 34-time-step difference between the first and last arriving robots, while it takes 907 time steps for the slowest robot to complete the navigation task. The robots controlled by recurrent-based MRC have faster convergence speed, as well as smaller time difference between the fastest robot and the slowest robot. In this simulation, recurrent-based MRC shows its better performance in arrival-time control than fuzzy-logic-based MRC does. The control results also demonstrated that the proposed models have the scalability and are both able to navigate different number of robots without retraining the model.

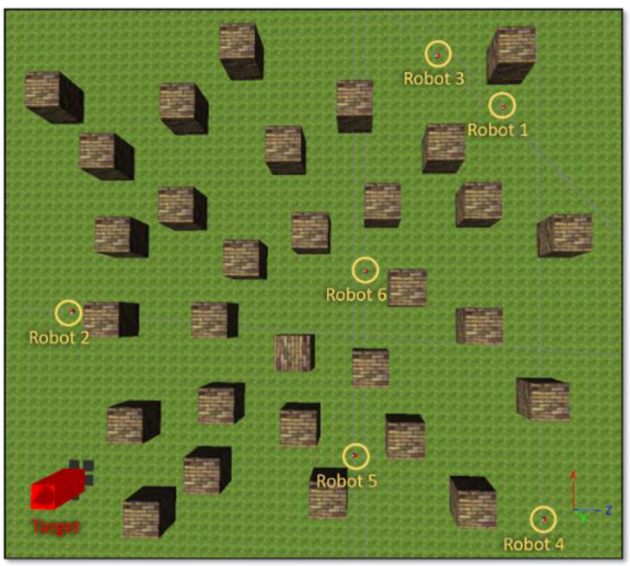

FIGURE 13 | An environment setting for six-robot navigation.

CONCLUSION AND FUTURE WORK

We developed a fuzzy-logic-based coordinator and a recurrent-based coordinator for safely navigating multiple robots in cluttered environments, where the controller regulates their speeds and adjusts their searching direction to enable simultaneous arrival time on targets. The environment for the test was designed in an imbalanced setting, in which each robot starts at the positions with totally different distances to the target. The simulation results demonstrate that the two proposed models successfully control a different number of robots to safely navigate and reach the target on time. In the future, we intend to develop a systematic method for optimizing the hyper-parameters of this cascaded model, including the number of rules in FLS. We are currently designing a technique for automatically tidying up the membership functions for improved interpretability. We will also consider scenarios where the robots lose communication when they are separated by building structures, losing a direct line of sight. We believe that intermittent sharing of controller policy will allow the agent to predict the future motions of the other agents when the communication is blocked and to make correct motion decisions.

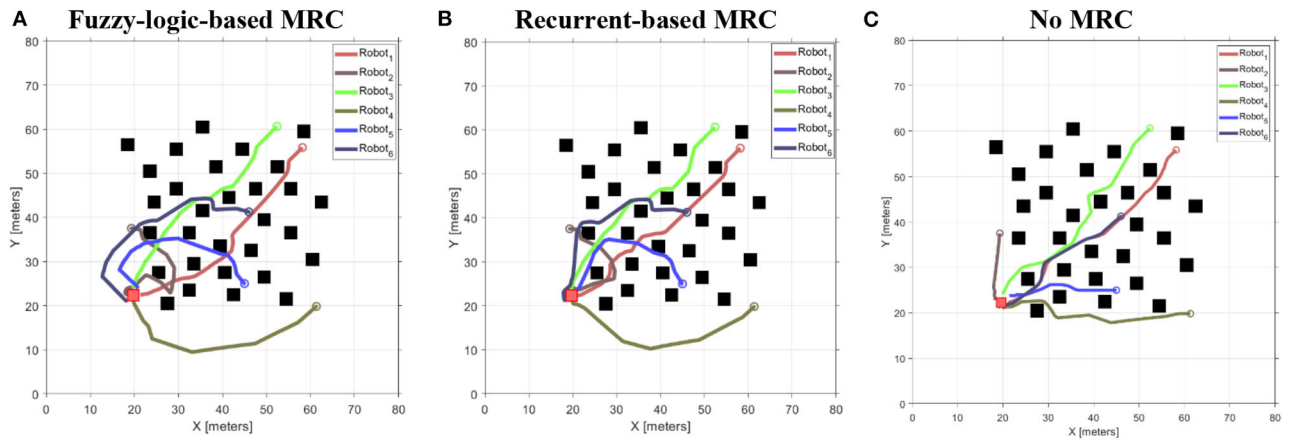

FIGURE 14 | The remaining distance between the target and the three robots during the three-robot navigation and arrival time control simulation. **(A)** Robots are controlled by the fuzzy-logic-based MRC. **(B)** Robots are controlled by the recurrent-based MRC. **(C)** Robots are controlled by only their own RBC.

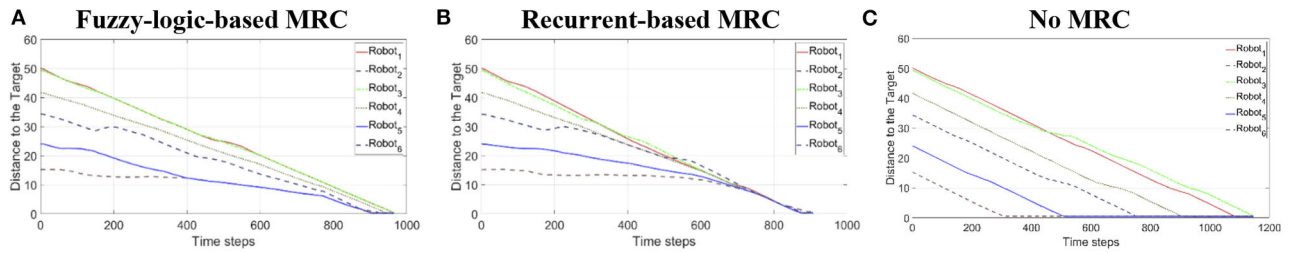

FIGURE 15 | The moving speeds of six-robot navigation in complex environment setting. **(A)** Robots are controlled by the speed coordination controller. **(B)** Robots are controlled by the improved controller. **(C)** Robots are controlled by only their own RBC.

TABLE 3 | The performance of time arrival coordination with six-robot setting.

Unit: Time step	Time difference	Time to complete task
Fuzzy-logic-based MRC	58	965
Recurrent-based MRC	34	907
No MRC	839	901

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR'S NOTE

We developed a fuzzy-logic-based coordinator and recurrent-based coordinator for safely navigating multiple robots in clutter environments, where the controller regulates their speeds and adjusts their searching direction to enable simultaneous arrival time on targets. The environment for the test was designed in an imbalanced setting, in which each robot starts at the positions with totally different distances between it and the targets. The simulation results demonstrate that the two proposed models successfully control a different number of robots to safely navigate and reach the target on time.

AUTHOR CONTRIBUTIONS

Y-CC developed the methodology, performed the experiments, and wrote the manuscript. AD proposed the experiments and partially contributed to the manuscript. The project was administered by C-TL and JK.

FUNDING

This research has been supported by the generous funding of Defence Science & Technology Group, Australia.

ACKNOWLEDGMENTS

We gratefully acknowledge our colleague Dr. Shi Ye for his assistance in simulations and for his contribution to an earlier version of this manuscript.

REFERENCES

Babel, L. (2019). Coordinated target assignment and UAV path planning with timing constraints. *J. Intelligent Robot. Syst.* 94, 857–869. doi: 10.1007/s10846-018-0910-9

Chandrasekhar Rao, D., and Kabat, M. R. (2019). "A study on cooperation and navigation planning for multi-robot using intelligent water drops algorithm," in *Emerging Research in Computing, Information, Communication and Applications,* eds N. R. Shetty, L. M. Patnaik, H. C. Nagaraj, P. N. Hamsavath, and N. Nalini (Singapore: Springer). 577–590.

Chia-Feng, J. (2005). Combination of online clustering and Q-value based GA for reinforcement fuzzy system design. *IEEE Trans. Fuzzy Syst.* 13, 289–302. doi: 10.1109/TFUZZ.2004.841726

Chia-Feng, J., and Chin-Teng, L. (1998). An online self-constructing neural fuzzy inference network and its applications. *IEEE Trans. Fuzzy Syst.* 6, 12–32. doi: 10.1109/91.660805

Din, A., Jabeen, M., Zia, K., Khalid, A., and Saini, D. K. (2018). Behavior-based swarm robotic search and rescue using fuzzy controller. *Comput. Elect. Eng.* 70, 53–65. doi: 10.1016/j.compeleceng.2018.06.003

Ding, W., Lin, C., and Cao, Z. (2019). Deep neuro-cognitive co-evolution for fuzzy attribute reduction by quantum leaping PSO with nearest-neighbor memeplexes. *IEEE Trans. Cybern.* 49, 2744–2757. doi: 10.1109/TCYB.2018.2834390

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). LSTM: a search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 2222–2232. doi: 10.1109/TNNLS.2016.2582924

Jhang, J.-Y., Lin, C.-J., and Young, K.-Y. (2019). Cooperative carrying control for multi-evolutionary mobile robots in unknown environments. *Electronics* 8:298. doi: 10.3390/electronics8030298

Juang, C., and Chang, Y. (2011). Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments. *IEEE Trans. Fuzzy Syst.* 19, 379–392. doi: 10.1109/TFUZZ.2011.2104364

Juang, C.-F., and Lo, C. (2008). Zero-order TSK-type fuzzy system learning using a two-phase swarm intelligence algorithm. *Fuzzy Sets Syst.* 159, 2910–2926. doi: 10.1016/j.fss.2008.02.003

Kashyap, A. K., and Pandey, A. (2018). Different nature-inspired techniques applied for motion planning of wheeled robot: a critical review. *Int. J. Adv. Robot. Autom.* 3, 1–10. doi: 10.15226/2473-3032/3/2/00136

Lai, M., Zeng, W., and Juang, C. (2016). "Navigation for two fuzzy controlled cooperative object-carrying robots in concave maps with the consideration of dead-cycle problem," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (Vancouver, BC: IEEE), 1905–1909. doi: 10.1109/FUZZ-IEEE.2016.7737923

Lin, C.-T., and Lee, C. S. G. (1996). *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems (Har/Dskt edition)*. Upper Saddle River, NJ: Prentice Hall.

Mansoori, E. G., Zolghadri, M. J., and Katebi, S. D. (2008). SGERD: a steady-state genetic algorithm for extracting fuzzy classification rules from data. *IEEE Trans. Fuzzy Syst.* 16, 1061–1071. doi: 10.1109/TFUZZ.2008.915790

Misra, S., Chakraborty, A., Sharma, R., and Brink, K. (2018). "Cooperative simultaneous arrival of unmanned vehicles onto a moving target in gps-denied environment," in *2018 IEEE Conference on Decision and Control (CDC)*, 5409–5414. doi: 10.1109/CDC.2018.8619652

Mohanta, J. C., and Keshari, A. (2019). A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation. *Appl. Soft Comput.* 79, 391–409. doi: 10.1016/j.asoc.2019.03.055

Nantogma, S., Ran, W., Yang, X., and Xiaoqin, H. (2019). "Behavior-based genetic fuzzy control system for multiple usvs cooperative target protection," in *2019 3rd International Symposium on Autonomous Systems (ISAS)*, 181–186. doi: 10.1109/ISASS.2019.8757732

Patle, B. K., Babu, L. G., Pandey, A., Parhi, D. R. K., and Jagadeesh, A. (2019). A review: on path planning strategies for navigation of mobile robot. *Defence Technol.* 15, 582–606. doi: 10.1016/j.dt.2019.04.011

Pothal, J. K., and Parhi, D. R. (2015). Navigation of multiple mobile robots in a highly clutter terrains using adaptive neuro-fuzzy inference system. *Robot. Auton. Syst.* 72, 48–58. doi: 10.1016/j.robot.2015.04.007

Pradhan, B., Roy, D. S., and Hui, N. B. (2019). "Multi-agent navigation and coordination using GA-fuzzy approach," in *Soft Computing for Problem Solving,* eds J. C. Bansal, K. N. Das, A. Nagar, K. Deep, & A. K. Ojha (Singapore: Springer). 793–805.

Shi, Y., and Eberhart, R. C. (1998). "A modified particle swarm optimizer". In *Presented at the 1998 IEEE International Conference on Evolutionary Computation Proceedings*. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK.

Webots: Robot simulator. Available online at: https://cyberbotics.com/ (accessed September 30, 2019).

Yao, P., and Qi, S. (2019). Obstacle-avoiding path planning for multiple autonomous underwater vehicles with simultaneous arrival. *Sci. China Technol. Sci.* 62, 121–132. doi: 10.1007/s11431-017-9198-6

Zhu, A., and Yang, S. X. (2007). Neurofuzzy-based approach to mobile robot navigation in unknown environments. *IEEE Trans. Syst. Man Cybern. Part C* 37, 610–621. doi: 10.1109/TSMCC.2007.897499

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Chang, Dostovalova, Lin and Kim. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.