Turkish Journal of Science & Technology
19(1), 265-277, 2024

Research Paper
https://doi.org/10.55525/tjst.1423794

Realization of Fuzzy-PI Controller-Based Path Planning of Differential Drive Mobile Robot

Ahmet TOP[1*], Muammer GÖKBULUT[2]
[1,2] Department of Electrical and Electronic Engineering, Faculty of Technology, Fırat University, Elazığ, Turkey
[*1] atop@firat.edu.tr, [2] mgokbulut@firat.edu.tr

(Geliş/Received: 22/01/2024; Kabul/Accepted: 26/03/2024)

Abstract: This paper uses a cascade-connected fuzzy-PI controller to control the position and speed of a differential drive and four-wheel drive of an autonomous mobile robot for optimal path planning. The angular speed information obtained from the encoder of each motor and the instantaneous position and angle information of the robot were calculated. The angle and position error between the reference points and these values is applied to the fuzzy logic controller as an input signal. The robot angular and linear speed data obtained from the fuzzy logic output were converted into reference speed values with kinematic equations to be applied to the motors. The speed controls of the motors were carried out with a PI controller based on these reference values. The study was performed both as a simulation in the MATLAB program and experimentally in the laboratory environment for one and more reference coordinates. In the experimental study, reference values were sent to the robot via Bluetooth with the Android application designed. At the same time, the instant data of the robot was also collected on the Android device through the same application. These data collected in Excel format were transferred to the computer via e-mail and the graphics were drawn in the MATLAB program. When the results were examined, it was seen that both speed and position control were successfully implemented with the fuzzy-PI controller for optimum path planning of the robot.

Keywords: Fuzzy logic, PI, mobile robot, Android application, path planning.

Diferansiyel Sürüşlü Mobil Robotun Bulanık PI Denetleyici Tabanlı Yol Planlamasının Gerçekleştirilmesi

Öz: Bu çalışmada, diferansiyel sürüşlü ve dört tekerden tahrikli otonom mobil robotun, optimum yol planlaması için, konum ve hız kontrolü kaskad bağlantılı fuzzy-PI kontrolör ile gerçekleştirilmiştir. Her bir motorun enkoderinden alınan açısal hız bilgileri ile robotun anlık konum ve açı bilgilerini hesaplanmıştır. Referans noktalar ile bu değerler arasındaki açı ve konum hatası bulanık mantık denetleyiciye giriş sinyali olarak uygulanmıştır. Bulanık mantık çıkışından alınan robot açısal ve lineer hız verileri ise kinematik denklemler ile motorlara uygulanacak olan referans hız değerlerine dönüştürülmüştür. Motorların hız kontrolleri bu referans değerler baz alınarak PI kontrolör ile gerçekleştirilmiştir. Bir ve birden fazla referans koordinatlar için gerçekleştirilen çalışma hem MATLAB programında simulasyonda hem de laboratuvar ortamında deneysel olarak gerçekleştirilmiştir. Deneysel olarak yapılan çalışmada, tasarımı gerçekleştirilen Android uygulama ile referans değerler robota bluetooth aracılığıyla gönderilmiştir. Aynı zamanda robotun anlık verileri de yine aynı uygulama üzerinden android cihazda toplanmıştır. Excel formatında toplanan bu veriler mail yolu ile bilgisayara aktarılarak MATLAB programında grafikleri çizdirilmiştir. Alınan sonuçlar incelendiğinde robotun fuzzy-PI kontrolör ile başarılı bir şekilde hem hız hem de konum kontrolünün gerçekleştirildiği görülmüştür.

Anahtar kelimeler: Bulanık mantık, PI, mobil robot, Android uygulama, yol planlama.

1. Introduction

Robotics is a sector that is expanding quickly along with technological advancements, and robots are increasingly playing a significant role in daily life for humans. In addition to classical areas such as industrial, medical, and rehabilitation, human-robot interaction increases its impact in areas such as exploration, urban search, and rescue, due to reasons such as saving manpower and time, being more economical, and working with fewer errors. These robots can be controlled manually by humans, depending on their area of use and purpose, or they can perform the tasks assigned to them autonomously [1]. An autonomous mobile robot (AMR) is a system that operates in an unpredictable and partially unknown environment. There is little or no human intervention in autonomous mobile robot movement [2].

Mobile robot (MR) systems are separated into three main modules: information sensing, path planning, and control of motion. Path planning is the connection between information perception and motion control, and it is a crucial aspect of a mobile robotic system. The continuous development of path-planning technology brings innovations to every part of life. For example, the sweeping robot can replace people doing housework and make people's work easier; As long as the destination location is entered in driverless vehicles, it can provide an optimum route and provide safe and accurate transportation; In emergency rescue and disaster assistance, AMRs can locate

[*] Corresponding author: atop@firat.edu.tr. ORCID Number of authors: [1] 0000-0001-6672-2119, [2] 0000-0003-1870-1772

targets quickly, accurately, and safely in hazardous conditions. In scientific research, it can replace humans by entering harsh environments to help humans recognize unknown planets and complete the task of acquiring knowledge [3].

To assist an MR in choosing the best path, numerous algorithms have been presented by researchers since the 1950s. These algorithms can be examined in general under three headings: classical, bionic, and artificial intelligence algorithms. Cell decomposition method (CD), sampling-based method (SBM), and graph search algorithm (GSA) can be given as examples of classical algorithms that have the advantage of easily observing the calculation results. Each algorithm can also be classified among themselves, and studies related to each are available in the literature. The CD was analyzed in three parts: regular (RCD) [4-6], approximate (ACD) [7,8], and exact decomposition (ECD) [9,10]. RRT (rapidly exploring random tree) [11] and PRM (probabilistic roadmap method) [12,13] algorithms were studied under the heading of SBM. Dijkstra algorithm [14,15] and A* algorithm [16] are in the GSA group. The bionic algorithm is a random algorithm inspired by the biological herd intelligence phenomenon in nature. It can be grouped according to different path-searching mechanisms. Genetic algorithm (GA) [17,18], ant colony optimization (ACO) [19,20], and particle swarm optimization (PSO) [21,22] are some of the studied bionic algorithms related to path planning. Artificial intelligence (AI), which has basic needs such as algorithms, data, and computing capability, simulates human behaviors such as learning, reasoning, thinking, and planning. The most used AI algorithms for path planning in MRs; are bioinspired neural network algorithms [23,24] and fuzzy logic control algorithms.

Fuzzy logic (FL), presented by Zadeh in 1965 [25], is an artificial intelligence algorithm that processes real-time information from the sensors as input data and generates the output values required for the path planning of the MR as a result of its analysis. Because it is less influenced by outside forces, it is appropriate for path planning in unknown environments [26]. However, it has the disadvantage that the optimum path to be followed depends on the rule base created by experts. Many researchers have used different algorithms together with the FL algorithm to improve the performance and accuracy of FL [27]. For example, Zagradjanin et al. used the D*lite algorithm with FL [28], while Ntakolia and Lyridis obtained high-quality solutions by integrating the swarm intelligence graph-based pathfinding algorithm and the FL algorithm [29]. Gharajeh & Jond presented the adaptive neuro-fuzzy inference system (ANFIS) method and shortened the planned path by 30 percent compared to other algorithms [30]. Besides, the most common integrated use is FL and PID (proportional-integral-derivative) algorithms [31-35].

When the MR path planning studies with Fuzzy-PID in the literature are examined, it is seen that the fuzzy logic controller is generally used to regulate and adjust the proportional, integral, and derivative coefficients of the PID algorithm. In this study, unlike the studies in the literature, the reference value, not the parameters of the PI algorithm, was determined with the fuzzy logic controller. The reference target points were sent to the robot with the designed Android application and these values were compared with the instant data of the robot and the position and angle error values were calculated. These values are applied as input signals to the fuzzy logic controller. Linear and angular velocities of the robot are obtained as output signals. By using these speed values in the kinematic equations, the angular speeds that each motor must reach instantaneously are calculated. The instantaneous speed of the motors was controlled by applying the error between the instantaneous speed values obtained from the encoders and the reference motor angular speed values to the PI controller. The robot, which reached the target within the approach distance given for the reference positions, performed its next movement with the reference speed values obtained from the fuzzy logic. The instant speed, angle, and position values of the robot were transferred via Bluetooth with the same Android application and collected on the Android device in Excel format. This file was then sent to the computer by e-mail and the robot's graphics were drawn with the MATLAB program. When the graphics are examined, it is seen that the robot reaches the desired positions within the given approach distance and by taking the optimum path.

2. Materials and Methods

2.1 Mobile robot

For this study, the design and prototype of a four-wheel drive, **S**ymmetric **M**odul**a**r **R**obo**t** (SMaRt), which can be used in many fields such as education, research, military, health, etc. have been realized. The design of the robot, which consists of the components in Figure 2 and whose final assembly is given from different angles in Figure 1; consists of 3 main parts: mechanical, electronic, and software tools. It has the advantages of weighing less than 10 kg, adding attachments such as a robot arm, and the ability to remove and replace the desired parts thanks to its modular structure.

Figure 1. Symmetric modular robot-SMaRt

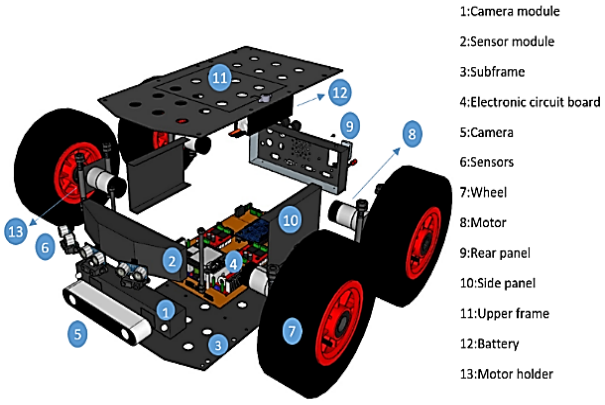


Figure 2. Components of the MR

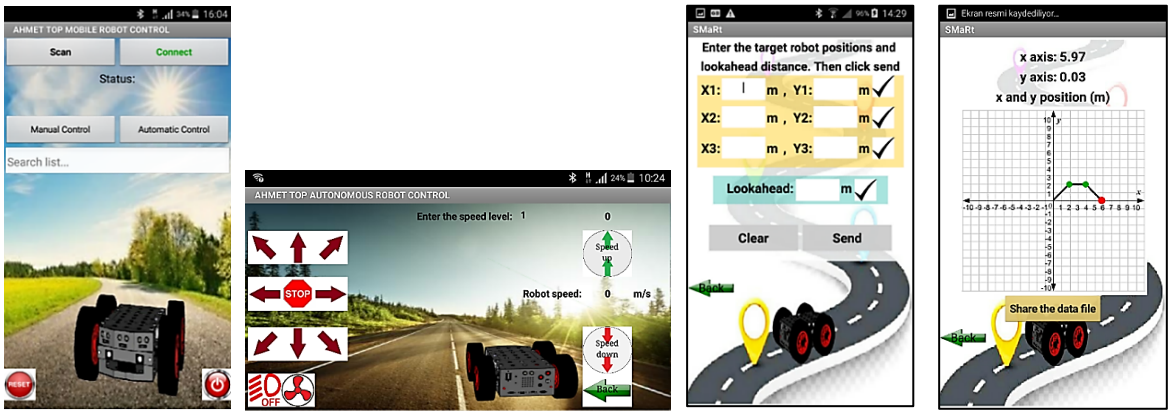
The DC motor (No Load Speed: 76 rpm, Stall torque: 45 kg-cm, Max. Impact torque: 6 kg-cm) was selected based on the motor torque value calculated for each wheel, taking into account the robot's weight, the wheel's dimensions, and any potential tilt and acceleration. The encoder has a resolution of 64 CPR and is attached to the motor shaft from behind. The microprocessor was able to determine the motor speed using the square wave data it received from the encoder, and the motor speed control was made possible. The XOR-HWT (exclusive OR-half-wave time) method was used to detect velocity with these signals [36].

The MR's four DC motors are driven by the Sparkfun Monster motor driver modules with dual outputs shown. With this driver module, it is possible to control motors with a maximum of 16 V, a continuous current of 14A, and a maximum PWM frequency of 20kHz. [38]. There are wired and wireless connections to enable the circuits and processors in the robot to communicate with each other or with the outside world. The HM-10 Bluetooth module was used to provide the connection between Arduino and Android devices. This module is a Bluetooth 4.0 module with low power consumption (when active, the current is 9mA) [39]. It is connected to Arduino with UART communication protocol. Arduino Due is an Atmel SAM3X8E 32-bit ARM Cortex-M3 CPU-based microcontroller board. It has a total of 54 digital input/output pins, 12 of which can be used for PWM, 12 analog inputs, 4 UARTs, 84 MHz oscillator frequency, and 2 digital to analog converter pins [40]. Arduino software is made with Arduino IDE (Integrated Development Environment), an open-source development platform.

2.2 Android application

A computer or a special electronic control circuit is usually used to send target coordinates to the robot. Likewise, robot information is collected on an additional memory card or sent to the computer. In this study, an Android-based application, whose screenshots are given in Figure 3, has been developed for smart devices (mobile phones, tablets) that are widely used today, which can do both operations [41]. The MIT App Inventor, which was initially made available by the Google firm and is currently maintained by the Massachusetts Institute of Technology (MIT), was used to create and implement the app. MIT App Inventor has been transformed over the past ten years into automation systems [43], quizzes and games [44], smart home control [45–48], education [49], and more. It has been utilized for Android applications in many different projects, including. In robotic applications, there are designs only for the manual use of the robot, but in this study, robot data was collected by using it for autonomous control.

Thanks to the buttons on the main screen in Figure 3 (a), it is possible to switch to manual and autonomous control screens, as well a Bluetooth connection is provided from here. With the manual control screen in Figure 3 (b), the movement direction of the robot, its speed, and the status of the LEDs and headlights on it can be changed and the robot can be brought to the desired starting point for autonomous control. When the screen in Figure 3 (c) is reached by pressing the automatic control button on the main screen, three different coordinate points where the robot desires to go and the approach distances to these points are entered and sent to the robot. After the reference points are sent and the robot starts to move, the coordinate plane in Figure 3 (d) comes to the Android device screen, the reference points are automatically marked and the instant position of the robot is drawn on this plane. In addition, the instantaneous speed and angle information of the robot, along with the location information, can be collected on the Android device in Excel file format and shared on a desired platform. The Excel file obtained in this study was transferred to the computer by e-mail and the graphs of the data of the robot were drawn in the MATLAB program.



(a) (b) (c) (d)

Figure 3. Android application; (a) Main screen, (b) Manual control screen, (c) Automatic control screen, (d) Instant motion graph of the robot

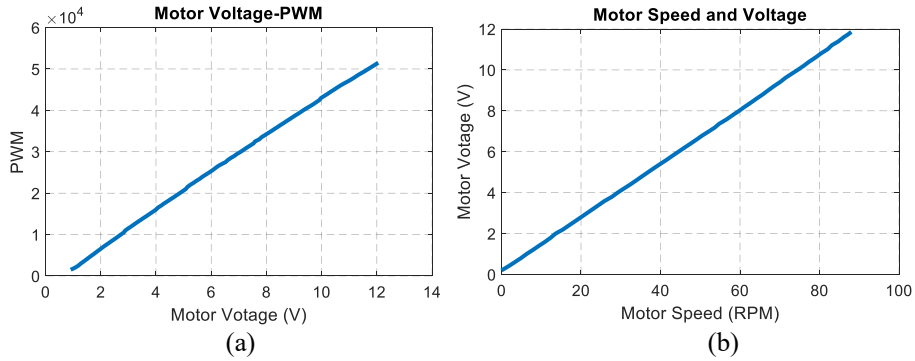
2.3 PID controller

PID controllers, also known as proportional, integral, and derivative (PID) controllers, are frequently used to regulate the speed of DC motors. The PID controller is regarded as the method most frequently employed in nonlinear control systems [50]. In essence, a PID controller uses a straightforward trinomial controller to increase stability and decrease steady-state error [51]. It offers the most efficient and simplest solution for many control problems, covering both transient and steady-state responses. The transfer function is usually written with the "gain notation" given by equation 1 or the "time constant notation" given by equation 2.

$$T(s) = K_p + K_i\frac{1}{s} + K_d s \tag{1}$$

$$T(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{2}$$

where Kp is the proportional gain, Ki is the integral gain, Kd is the derivative gain, Ti is the integral time constant and Td is the derivative time constant. It's crucial to regulate these motors properly because the robot's position is dictated by the speed information the DC motor encoder provides. Due to this, each motor has been separately PI controlled to closely and quickly follow the reference speed values. In Figure 5, the general block diagram of the motor controls is given. In this study, the PI controller was used by setting Kd = 0.



(a) (b)

Figure 4. (a) Motor voltage-PWM graph, (b) Motor speed and voltage graph

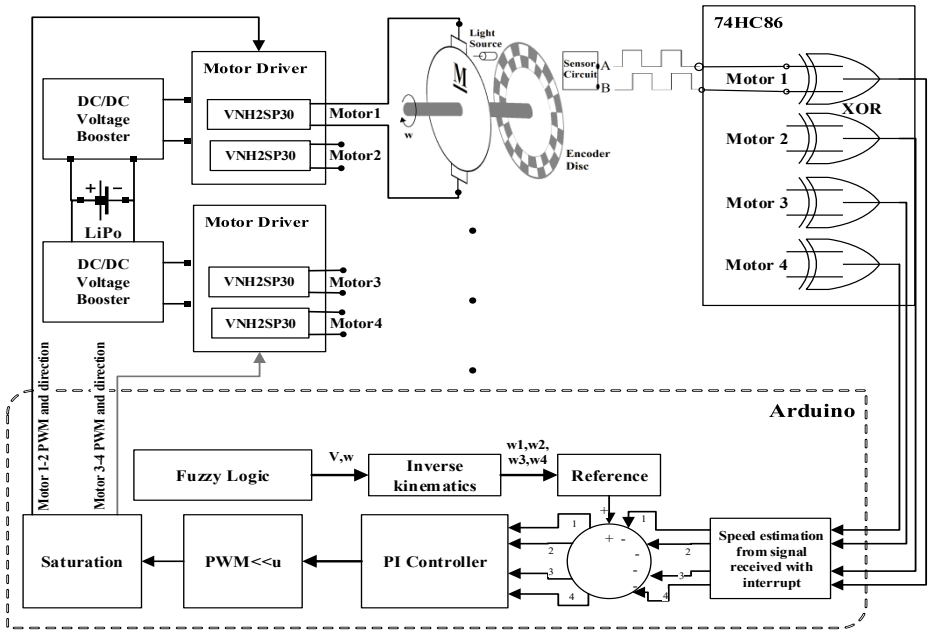


Figure 5. Motor control general block diagram with PI

According to this diagram, the output of the fuzzy logic controller used for position control of the robot is the reference linear and angular velocity of the robot. The reference angular velocities that each wheel must reach are rad./s by using these velocity values as inputs to the kinematics equations. These reference values were obtained from the output of the fuzzy controller and the instantaneous motor speeds taken from the XOR logic gate and the instantaneous and reference voltage values of the motors were obtained and the error was found by taking the difference of these values. This error has been converted to PWM values since the output value obtained when passed through the PI controller cannot be sent directly to the drivers. Experimental results revealed the association between motor speed and motor voltage as well as the relationship between the motor voltage and applied PWM. The graph of these values was drawn as in Figure 4 and the equational expression between them was created. The signals received from the encoder connected to the motor shaft are applied to the XOR logic gate input and each XOR output is connected to a different pin of the Arduino [36]. The motor and encoder disc shown in the block diagram are visually shown for only one motor. The same steps must be followed for other motors.

2.4 Fuzzy logic (FL)

The FL controller is a control algorithm that processes the input data taken from the physical environment and whose value is certain and produces outputs that can be used in the physical environment as output. In this study, Mamdani-type FL was used to perform position control and path planning in line with the target coordinates given to the robot. The controller consists of two input and two output variables. The distance error and angle error to the coordinate point where the robot should go are given as input variables. The linear and angular velocities that the robot needs to reach instant are taken as the output variable. Position error value is meter, angle error value is radian, linear velocity value is m/s and angular velocity value is radian/s evaluated in units. As seen in Figure 6, ten triangular membership functions (MFs) are used for the input membership functions created for the position error. Ten were used to gradually decrease its velocity every ten centimeters, one meter from the reference position.

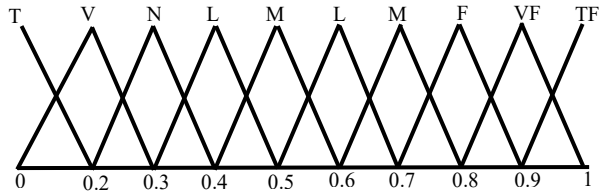


Figure 6. Position error MFs

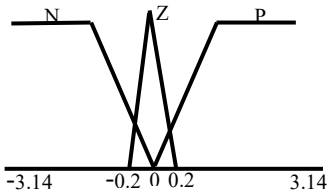


Figure 7. Angle error MFs

where, TN: Too near, VN: Very near, N: Near, LN: Little near, MN: Middle near, LF: Little far, MF: Middle far, F: Far, VF: Very far, TF: Too far. If the difference between the target and the position of the robot is more

than one meter, it is considered too far. When this error value falls below one meter, membership degrees change at each level until it reaches zero. The MFs created for the angle error are given in Figure 7. The limits of the angle error memberships consisting of two trapezoidal and one triangle membership functions are in the range of ±3.14 radians. In angle error membership, N: Negative, Z: Zero, P: Positive. The breaking points of the positive and negative angle MFs were determined as ±1.57 radians. There are two outputs in the system, angular velocity and linear velocity. Ten triangular MFs make up linear velocity, while three do the same for angular velocity. When the functions given in Figure 8 are examined, values are given such that the highest linear speed is 0.275 m/s and the lowest linear speed is 0.11 m/s. The reason for giving these values can be listed as the difficulties in the rotation movement of the motors at low speeds, while excessive energy consumption, microcontroller processing speed, and slips in sudden starts at high speeds.

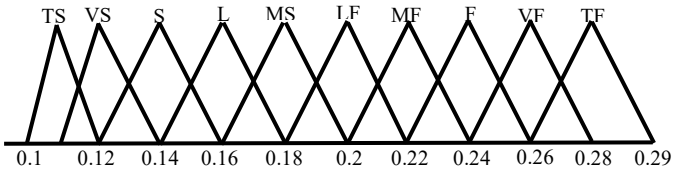


Figure 8. Linear speed MFs

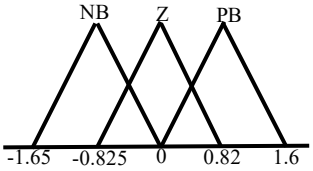


Figure 9. Angular speed MFs

where, TS: Too slow, VS: Very slow, S: Slow, LS: Little slow, MS: Middle slow, LF: Little fast, MF: Middle fast, F: Fast, VF: Very fast, TF: Too fast. The three membership functions used for the angular velocity of the robot are named NB: Negative big, Z: Zero, and PB: Positive big. The median values of the functions whose graphs are given in Figure 9 are given as -0.825, 0, and +0.825 radians, respectively. Since the mean of weights method is used in defuzzification, the value at the midpoint of each of the membership functions is used in the calculations.

Table 1. Rule base for linear speed

Position error		TN	VN	N	LN	MN	LF	MF	F	VF	TF
Angle error	**N**	S	VF	VF	F	MF	LF	TS	LF	VS	TS
	Z	S	S	S	LS	MS	LF	MF	F	VF	TF
	P	S	VF	VF	F	MF	LF	TS	LF	VS	TS

Table 2. Rule base for angular speed

Position error		TN	VN	N	LN	MN	LF	MF	F	VF	TF
Angle error	**N**	NB	NB	NB	NB	NB	NB	NB	NB	NB	NB
	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
	P	PB	PB	PB	PB	PB	PB	PB	PB	PB	PB

After the membership degrees and the degrees of precision found by using the algebraic product method are passed through the rule bases in Tables 1 and 2, the addition process does not take place. It is subjected to a direct defuzzification process. As a result, the robot linear and angular velocity values obtained from the fuzzy controller output are applied as input to the inverse kinematics equation in the control system whose flow chart is given in Figure 10, and reference angular velocity values for the motors on both sides are calculated.

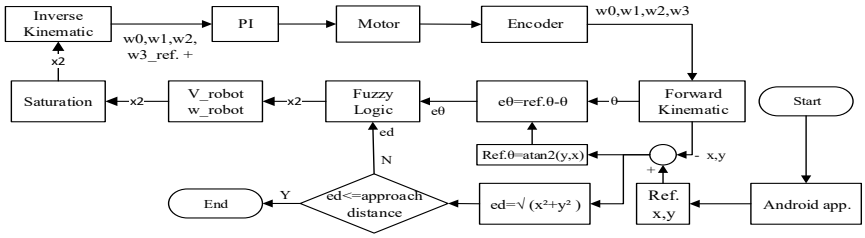


Figure 10. Position control flow chart with fuzzy logic controller

3. Simulation

Before examining the robot's performance with the experimental study, a simulation was prepared in the MATLAB/Simulink program to examine both the accuracy of the equations and the accuracy of the written algorithms and coefficient values. With different reference values for robot movements, firstly the results were obtained in the simulation, and then experimental studies were carried out in the laboratory environment.

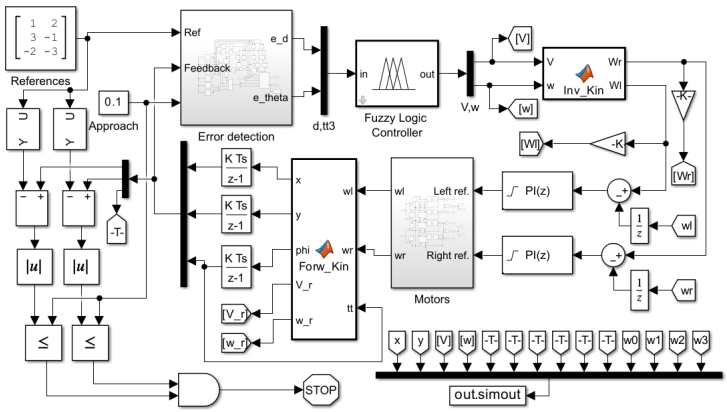


Figure 11. Robot position control simulink model

There are two input variables in the MATLAB/Simulink model in Figure 11. These are reference values and approach distance. For the robot to go to three different position points, x_1,y_1,x_2,y_2, and x_3,y_3 positions are given to the reference block in matrix format. If the robot is desired to go to only one reference, zero values are entered into the points other than the x_3 and y_3 values. Similarly, if it is desired to go to two references, zero values are entered in x_1 and y_1. The approach distance is the variable that determines how much error the robot can approach the given target coordinate. For example, if this value is entered as 0.1 m as in Figure 11, it means that the robot can stop after 0.1 meters approach to the reference point. The robot's approach can be in front of, right, left, or behind the target. Reference value, instantaneous feedback position values, and approach distance are given as input to the error detection block in Figure 12. In this block, the first reference point and instantaneous position and angle values from feedback are applied to the error function block. Here, the difference of the reference position and the instantaneous measured positions is taken and the position and angle error are calculated from these values. These values form the output of the error detection block. If the first reference is within the approach distance, the same operations are repeated by switching to the second reference with the multi-position switch block. If the second reference is reached, calculations for the third reference point begin. If the first two references are zero, since they will both be within the approach distance, the movement starts directly toward the third reference.

Position and angle error obtained from the error detection block is applied as input to the fuzzy logic block, and the reference linear and angular velocity of the robot is calculated from the block output. The reference angular velocity values of the left motor and right motors calculated by applying inverse kinematics according to Equations 3 and 4 are passed through the PI and applied to the motor models in the motors block in Figure 13.

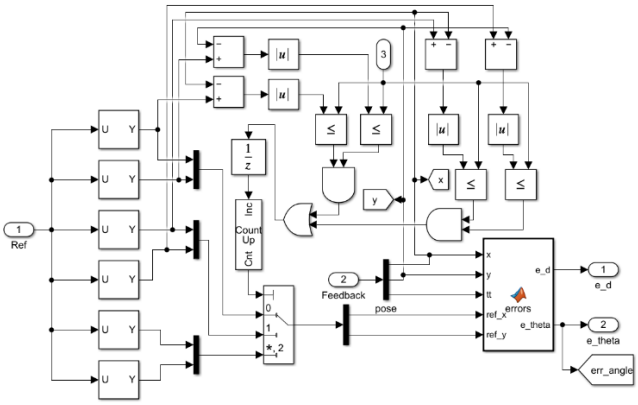


Figure 12. The internal structure of error detection block

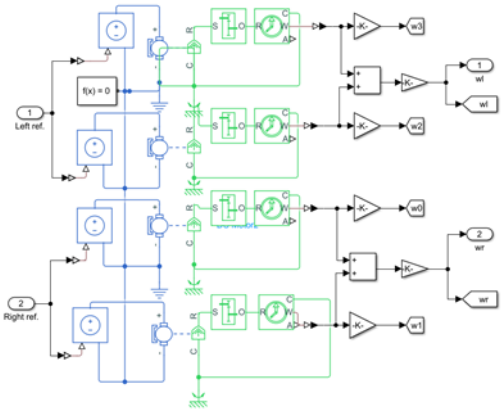


Figure 13. The internal structure of the motor block

$$w_L = \frac{1}{R}.(V - W\frac{L}{2}) \quad (3)$$

$$w_R = \frac{1}{R}.(V + W\frac{L}{2}) \quad (4)$$

where; w_L, is the angular velocity of the left motors, w_R is the angular velocity of the right motors, V is the robot linear velocity, W is the robot angular velocity, R is the radius of the wheels and L is the axis distance of the robot. The motor block contains four equal motors. The top two of them represent the motors on the left of the robot and the others on the right of the robot. The instantaneous speeds of the motors to which the reference speed is applied constitute the output of the block. These values, taken as left and right motor speeds, are applied as inputs to the forward kinematics block and the speed in the x direction (V_x), speed in the y direction (V_y), and angular velocity of the robot are calculated according to Equation 5-9.

$$V = \frac{R}{2}.(w_R + w_L) \tag{5}$$

$$W = \frac{R}{L}.(w_R - w_L) \tag{6}$$

$$\text{Robot}=[V \quad 0 \quad W]' \tag{7}$$

$$\text{Rz}=[\cos(tt) \quad -\sin(tt) \quad 0; \quad \sin(tt) \quad \cos(tt) \quad 0; \quad 0 \quad 0 \quad 1] \tag{8}$$

$$[V_x, V_y, V_{tt}]^I=\text{Rz}*\text{Robot} \tag{9}$$

where; tt instantaneous angle of the robot. By integrating these obtained values concerning time, x position, y position, and angle information are calculated.

4. Simulation and Experimental Results

During the period from the start of the simulation to the end, all the desired data were collected with the help of the 'out.simout' block and graphed in MATLAB/m-file. The performance of the robot with different reference values (RVs) and different approach distances (ADs), both in the simulation and experimentally, is given below.

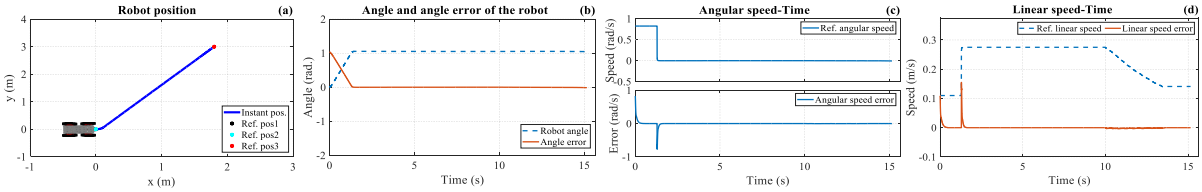


Figure 14. Simulation results for x_1:0, y_1:0, x_2:0, y_2:0, x_3:1.8, y_3:3 m reference and 0.05 m AD. Robot: (a) position, (b) Angle and angle error, (c) Angular speed and error, (d) Linear speed and error

In the MATLAB/Simulink program, when the x_1:0, y_1:0, x_2:0, y_2:0, x_3:1.8, y_3:3 m coordinates and 0.05 m approach distance are sent to the robot as a reference, the robot performances in figure 14 and figure 15 motor speed and errors were obtained. Since the reference value is x_3:1.8 m and y_3:3 m, it is seen in Figure 14 (b) that the angle to be rotated is calculated as 1.03 radians. When the angular velocity and linear velocity values in Figure 14 (c) and (d) are examined, it is observed that it first takes a turn with speeds of 0.825 radians/s and 0.11 m/s to go to the reference, and then increases to a linear velocity of 0.275 m/s over time, and its angular velocity decreases to zero.

Experimental work was carried out for the same reference values and the results in figure 16 and figure 17 were obtained. Looking at the robot position graph in Figure 16 (a), since the approach distance is given as 0.05 m, it is seen that the robot completes its movement at x:1.78, y:2.96 m. In cases where there is more than one reference change during the movement, a temporary state and a permanent state occur at each reference change. Therefore, the error amounts between the reference value and the instantaneous measured values in all graphs are completely taken into account without including the first transient in the evaluation. While Equation 10 gives the average absolute velocity errors of the graphs for this case, the instantaneous highest velocity error that occurs in other transients except for the initial transient case is given as the maximum velocity error.

$$\text{Performance criteria} = \frac{1}{N}\sum_{k=1}^{N}|e(k)| \tag{10}$$

where; e() is the error value and N is the number of samples. In this case, when looking at the graphs obtained for the reference value (1.8,3) m, the maximum linear velocity error between the two signals is 0.038 m/s and the average linear velocity error is 0.0021 m/s, while the angular velocity reference is 0.179 rad./s maximum. and an average of 0.0057 rad./s. followed by an error. When the motor speed graphs in Figure 17 (a) and (b) are examined, the maximum and average absolute error values of the motors for the same situation are given in Table 3.

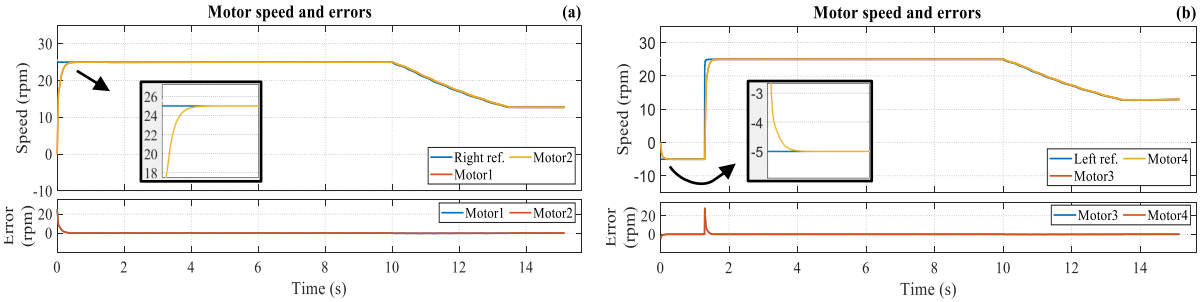


Figure 15. Simulation results for x_1:0, y_1:0, x_2:0, y_2:0, x_3:1.8, y_3:3 m RFs and 0.05 m AD. Robot: (a) Right motor speeds and errors, (b) Left motor speeds and errors

Table 3. Speed errors of motors for x_1:0, y_1:0, x_2:0, y_2:0, x_3:1.8, y_3:3 m RVs and 0.05 m AD

	Motor1	**Motor2**	**Motor3**	**Motor4**
Average absolute speed error (rpm)	0.194	0.134	0.229	0.249
|Maximum speed error (rpm)|	1.113	0.637	5.921	7.495

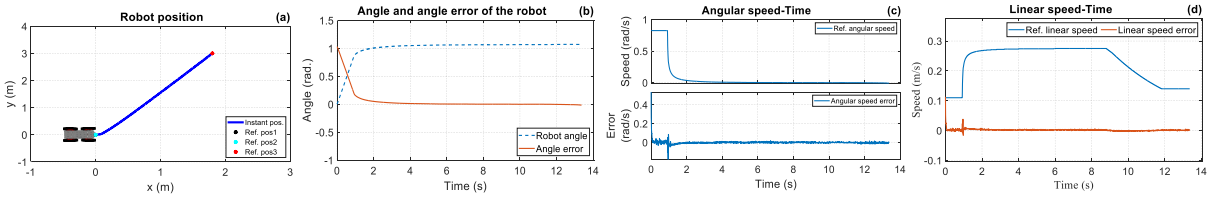


Figure 16. Experimental results for x_1:0, y_1:0, x_2:0, y_2:0, x_3:1.8, y_3:3 m reference and 0.05 m AD. Robot: (a) position, (b) Angle and angle error, (c) Angular speed and error, (d) Linear speed and error

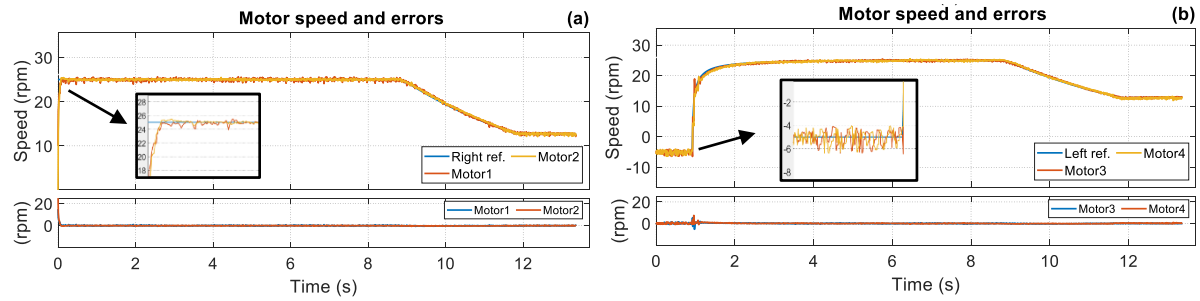


Figure 17. Experimental results for x_1:0, y_1:0, x_2:0, y_2:0 x_3:1.8, y_3:3 m RVs and 0.05 m AD. Robot: (a) Right motor speeds and errors, (b) Left motor speeds and errors

Performance graphics for x_1:1, y_1:2, x_2:3, y_2:-1, x_3:-2, y_3:-3 m reference values and 0.1 m approach distance given in different coordinate regions are given in figure 18 and figure 20. When the graphs of Figure 18 (a), and Figure 20 (a) are examined, the first target is in the first region of the coordinate plane, the second target is in the fourth region, and the last target is in the third region of the coordinate axis for both simulation and experimental work.

The angles calculated according to the point where the robot arrives and the point where it is desired to go and the angles realized as seen in figures 18(b)-20 (b) are given in Table 4.

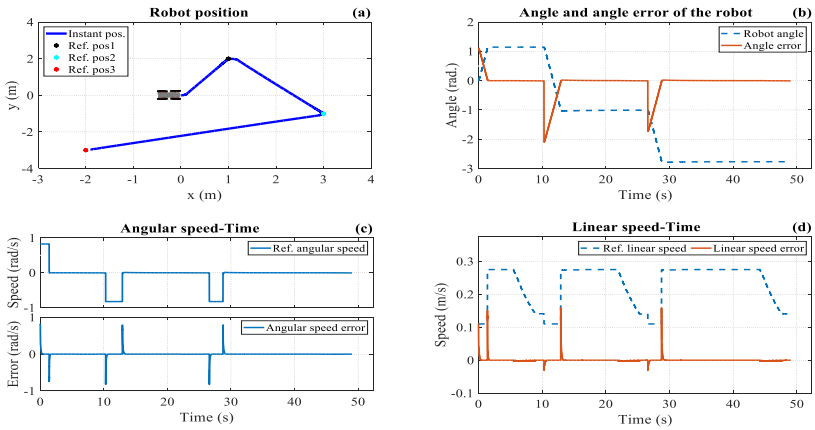


Figure 18. Simulation results for x_1:1, y_1:2, x_2:3, y_2:-1, x_3:-2, y_3:-3 m RVs and 0.1 m AD. Robot: (a) position, (b) Angle and angle error, (c). Angular speed and error, (d) Linear speed and error

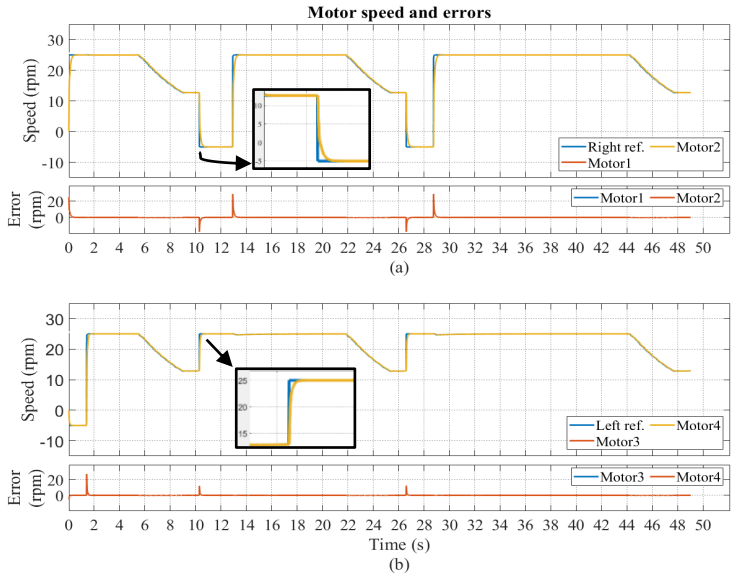


Figure 19. Simulation results for x_1:1, y_1:2, x_2:3, y_2:-1, x_3:-2, y_3:-3 m RVs and 0.1 m approach distance. Robot: (a) Right motor speeds and errors, (b) Left motor speeds and errors

Table 4. Calculated and actual angle values for x_1:1, y_1:2, x_2:3, y_2:-1, x_3:-2, y_3:-3 m RVs and 0.1 m AD

Theory			Experimental			Simulation		
Current position (m)	**Target point (m)**	**Calculated angle) (radian)**	**Current position (m)**	**Target point (m)**	**Realized angle (radian)**	**Current position (m)**	**Target point (m)**	**Realized angle (radian)**
(0,0)	(1,2)	1.107	(0,0)	(1,2)	1.107	(0,0)	(1,2)	1.107
(1.2)	(3 -1)	-0.98	(0.96, 1.91)	(3 -1)	-2.1	(0.95, 1.9)	(3,-1)	-2.096
(3, -1)	(-2,-3)	-2.76	(2.97, -0.95)	(-2,-3)	-1.72	(2.94, -0.9)	(-2,-3)	-1.71

According to Table 4, when the theoretical and experimental results are compared, the rotation angle reference is 1,107 radians, since the beginning point and the first target point are the same for both. -0.98 and -2.76 radians are calculated for the second and third turns, respectively. However, in the results obtained from the experimental and simulation studies, an angle reference of -2.1 radians for the second rotation and -1.72 radians for the third rotation was determined. This is because when the robot comes to the first reference point, its instantaneous angle is relative to the x-axis |1.117| is radians. Since the robot will turn 0.98 radians in the negative direction when the

instantaneous angle is added to this value, |1.117|+|0.98|=|2.1| radian angle reference. Likewise, since its angle at the second point is -1,036 radians and the angle calculated for the third position is -2.76 radians, the angle at which the robot must turn is -1.72 radians. When the motor speeds given in Figure 19 and Figure 21 are examined, the speed error information in Table 5 was obtained.

Table 5. Speed errors of motors for x_1:1, y_1:2, x_2:3, y_2:-1, x_3:-2, y_3:-3 m RVs and 0.1 m AD

	Motor1	Motor2	Motor3	Motor4
Average absolute speed error (rpm)	0.283	0.237	0.231	0.238
|Maximum speed error (rpm)|	8.6660	9.1600	12.4590	12.4000

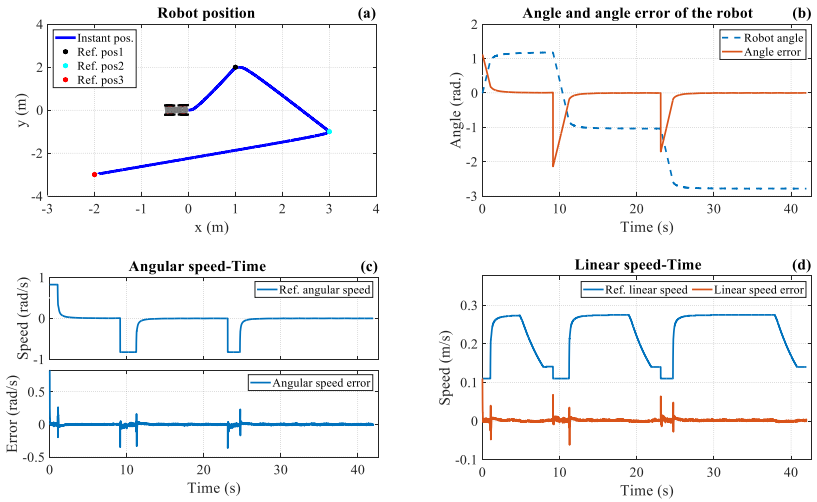


Figure 20. Experimental results for x_1:1, y_1:2, x_2:3, y_2:-1, x_3:-2, y_3:-3 m RVs and 0.1 m AD. Robot: (a) position, (b) Angle and angle error, (c) Angular speed and error, (d) Linear speed and error

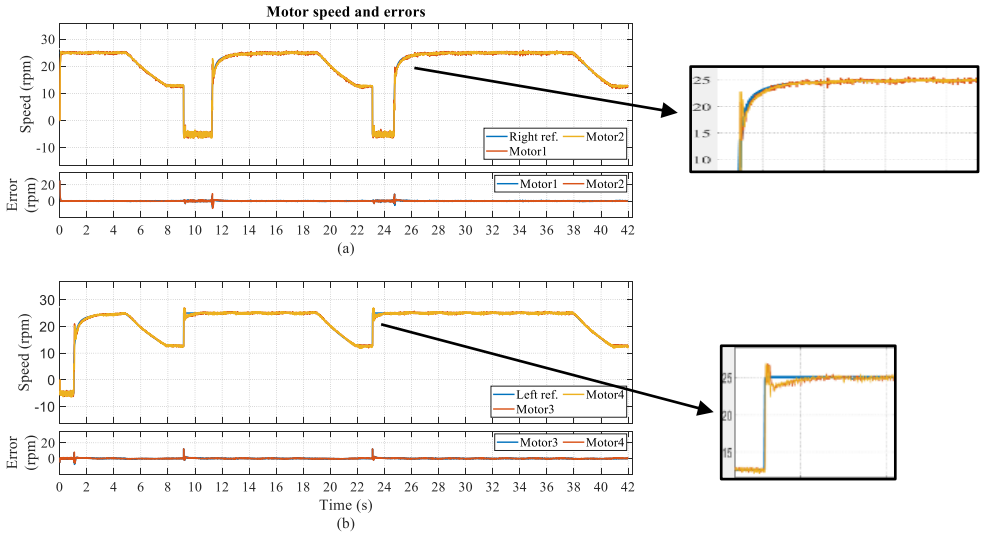


Figure 21. Experimental results for x_1:1, y_1:2, x_2:3, y_2:-1, x_3:-2, y_3:-3 m RVs and 0.1 m AD. Robot: (a) Right motor speeds and errors, (b) Left motor speeds and errors

5. Conclusion

In this study, a four-wheeled AMR's position control and path planning for single or multiple target coordinate points were carried out using a cascade connection of fuzzy logic and PI controllers. While the reference values of the robot were carried out remotely via Bluetooth connection with the Android application, the instant data of the robot was transferred to the Android device thanks to the same application. These data, which were collected in Excel file format, were then transferred to the computer via e-mail, and graphs were drawn in MATLAB. The position and angle information of the robot is obtained by passing the motor angular velocity information instantly from the encoders connected to the shaft of all four motors of the MR through the kinematic equations. Angle and position errors obtained by comparing reference values with instantaneous values are applied to the fuzzy logic controller as input signal. By applying the robot angular and linear velocity values obtained as the output value thanks to the created rule base to the inverse kinematic equations, the instantaneous reference angular velocity values that the motors should reach are calculated. The speed error was calculated by taking the difference between these values and the angular speed information received from the encoders and applied to the PI controller input. The motor speeds required for the position that the robot needs to reach are controlled by PI. When the graphics are examined, it was seen in both simulation and experimental studies that the MR reaches the position most shortly and without any errors in line with the given approach distance. In addition, the fact that the average absolute speed error in the motors is a maximum of 0.283 rpm shows the effectiveness of both PI and fuzzy logic controllers.

Funding: This study is supported by Fırat University Scientific Research Projects Unit with the project number TEKF.19.07.

References

[1] Shabalina K, Sagitov A, & Magid E. Comparative analysis of mobile robot wheels design. In 2018 11th International Conference on Developments in Systems Engineering; 2018; (pp. 175-179)
[2] Alatise MB, & Hancke GP. A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access 2020; 8, 39830-39846.
[3] Liu L, Wang X, Yang X, Liu H, Li J, & Wang P. Path Planning Techniques for Mobile Robots: Review and Prospect. Expert Systems with Applications 2023; 120254.
[4] Jiusheng B, Muye Z, & Shirong G. Underground driverless path planning of trackless rubber tyred vehicle based on improved A* and artificial potential field algorithm [J]. Journal of China Coal Society 2022; 47(03), 1347-1360.
[5] Zhou X, Yu X, & Peng X. UAV collision avoidance based on varying cells strategy. IEEE Transactions on Aerospace and Electronic Systems, 2018; 55(4), 1743-1755.
[6] Challita U, Saad W, & Bettstetter C. Deep reinforcement learning for interference-aware path planning of cellular-connected UAVs. In 2018 IEEE International Conference on Communications (ICC);2018; (pp. 1-7).
[7] Guruprasad KR, & Ranjitha TD. CPC algorithm: Exact area coverage by a mobile robot using approximate cellular decomposition. Robotica 2021; 39(7), 1141-1162.
[8] Samaniego F, Sanchis J, García-Nieto S, & Simarro R. Recursive rewarding modified adaptive cell decomposition (RR-MACD): a dynamic path planning algorithm for UAVs. Electronics 2019; 8(3), 306.
[9] Jung JW, So BC, Kang JG, Lim DW, & Son Y. Expanded Douglas–Peucker polygonal approximation and opposite angle-based exact cell decomposition for path planning with curvilinear obstacles. Applied Sciences 2019; 9(4), 638.
[10] Park J, Karumanchi S, & Iagnemma K. Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming. IEEE Transactions on Robotics 2015; 31(5), 1101-1115.
[11] Wang H, Li G, Hou J, Chen L, & Hu NA path planning method for underground intelligent vehicles based on an improved RRT* algorithm. Electronics 2022; 11(3), 294.
[12] Ravankar AA, Ravankar A, Emaru T, & Kobayashi Y. HPPRM: hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots. IEEE Access 2020; 8, 221743-221766.
[13] Esposito JM, & Wright JN. Matrix completion as a post-processing technique for probabilistic roadmaps. The International Journal of Robotics Research 2019; 38(2-3), 388-400.
[14] Fink W, Baker VR, Brooks AJW, Flammia M, Dohm JM, & Tarbell MA. Globally optimal rover traverse planning in 3D using Dijkstra's algorithm for multi-objective deployment scenarios. Planetary and Space Science 2019; 179, 104707.
[15] Balado J, Díaz-Vilariño L, Arias P, & Lorenzo H. Point clouds for direct pedestrian pathfinding in urban environments. ISPRS Journal of Photogrammetry and Remote Sensing 2019; 148, 184-196.
[16] Wang YF, Cao XH, & Guo X. Warehouse AGV path planning method based on improved A* algorithm and system short-term state prediction. Computer Integrated Manufacturing System 2021; 1-22.
[17] Lamini C, Benhlima S, & Elbekri A. Genetic algorithm-based approach for autonomous mobile robot path planning. Procedia Computer Science 2018; 127, 180-189.
[18] Shivgan R & Dong Z. Energy-efficient drone coverage path planning using genetic algorithm. In 2020 IEEE 21st International Conference on High Performance Switching and Routing 2020; (pp. 1-6).

[19] Miao C, Chen G, Yan C, & Wu Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. Computers & Industrial Engineering 2021; 156, 107230.
[20] Ji Y & Liu B. Research and Implementation of Robot Path Planning Based on Ant Colony Algorithm. In Journal of Physics: Conference Series 2022; (Vol. 2171, No. 1, p. 012074).
[21] Chai R, Tsourdos A, Savvaris A, Chai S & Xia Y. Solving constrained trajectory planning problems using biased particle swarm optimization. IEEE Transactions on Aerospace and Electronic Systems 2021;57(3), 1685-1701.
[22] Qiuyun T, Hongyan S, Hengwei G & Ping W. Improved particle swarm optimization algorithm for AGV path planning. Ieee Access 2021; 9, 33522-33531.
[23] Wang Z, Li H & Zhang X. Construction waste recycling robot for nails and screws: Computer vision technology and neural network approach. Automation in Construction 2019; 97, 220-228.
[24] Zhu D & Yang SX. Bio-inspired neural network-based optimal path planning for UUVs under the effect of ocean currents. IEEE Transactions on Intelligent Vehicles 2021; 7(2), 231-239.
[25] Zadeh LA. Fuzzy sets. Information and control 1965; 8(3), 338–353.
[26] Li M. Mobile robot path planning based on fuzzy control. Hebei University of Technology. 2015
[27] Xie YN. The research for the mobile robot path planning algorithm[D]. Xi ' a University of Architecture and Technology. 2016.
[28] Zagradjanin N, Rodic A, Pamucar D & Pavkovic B. Cloud-based multi-robot path planning in complex and crowded environment using fuzzy logic and online learning. Information Technology and Control 2021; 50(2), 357-374.
[29] Ntakolia C & Lyridis DV. A swarm intelligence graph-based pathfinding algorithm based on fuzzy logic (SIGPAF): A case study on unmanned surface vehicle multi-objective path planning. Journal of Marine Science and Engineering 2021; 9(11), 1243.
[30] Gharajeh MS, & Jond HB. An intelligent approach for autonomous mobile robots path planning based on adaptive neuro-fuzzy inference system. Ain Shams Engineering Journal 2022; 13(1), 101491.
[31] Jin X, Chen K, Zhao Y, Ji J, & Jing P. Simulation of hydraulic transplanting robot control system based on fuzzy PID controller. Measurement 2020; 164, 108023.
[32] Cao G, Zhao X, Ye C, Yu S, Li B, & Jiang C. Fuzzy adaptive PID control method for multi-mecanum-wheeled mobile robot. Journal of Mechanical Science and Technology 2022; 36(4), 2019-2029.
[33] Babunski D, Berisha J, Zaev E, & Bajrami X. Application of fuzzy logic and PID controller for mobile robot navigation. In 2020 9th Mediterranean Conference on Embedded Computing; 2020; (pp. 1-4).
[34] Cai C. Autonomous Mobile Robot Obstacle Avoidance Using Fuzzy-PID Controller in Robot's Varying Dynamics. In 2020 39th Chinese Control Conference; 2020; (pp. 2182-2186). IEEE.
[35] Lee K, Im DY, Kwak B, & Ryoo YJ. Design of fuzzy-PID controller for path tracking of mobile robot with differential drive. International Journal of Fuzzy Logic and Intelligent Systems 2018; 18(3), 220-228.
[36] Top A, & Gökbulut M. A novel period–based method for the measurement direct current motor velocity using low-resolver encoder. Transactions of the Institute of Measurement and Control 2023; 45(4), 711-722.
[37] Pololu 37D Metal Gearmotors Datasheet 18s., www.pololu.com/product/4756/specs, Access: 08.12.2023
[38] Sparkfun Monster Moto Shild and VNH2SP30 datasheet https: //www.sparkfun .com/ products / retired/ 10182, Access:20.12.2023
[39] HM-10 bluetooth modul**,** http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/#HM-10Services_and Character istics, Access: 08.12.2023
[40] Arduino Due, https://store.arduino.cc/products/arduino-due, Access: 08.12.2023
[41] Top A, & Gökbulut M. Android Application Design with MIT App Inventor for Bluetooth Based Mobile Robot Control. Wireless Personal Communications 2022; 126(2), 1403-1429.
[42] Hong S, & Hwang Y. design and implementation for iort-based remote control robot using block-based programming. Issues in Information Systems 2020; 21(4), 317-330.
[43] De Moura Oliveira PB. Teaching automation and control with App Inventor applications. In 2015 IEEE Global Engineering Education Conference; 2015; (pp. 879-884). IEEE.
[44] Asghar MZ, Sana I, Nasir K, Iqbal H, Kundi FM, & Ismail S. Quizzes: Quiz application development using Android-based MIT APP Inventor platform. International Journal of Advanced Computer Science and Aplications 2016; 7(5).
[45] Sullivan D, Chen W, & Pandya A. Design of remote control of home appliances via Bluetooth and Android smartphones. In 2017 IEEE International Conference on Consumer Electronics-Taiwan; 2017; (pp. 371-372).
[46] Prayogo SS, Saptariani T, & Salahuddin NS. Rancang Aplikasi Android Pengendali Mobil dan Kamera Menggunakan APP inventor, Seminar Nasional Aplikasi Teknologi Informasi 2015; (Vol. 1, No. 1).
[47] Kannapiran S, & Chakrapani A. A novel home automation system using Bluetooth and Arduino, international journal of advances in computer and electronics engineering 2017; 2(2), 41-44.
[48] Adiono T, Anindya SF, Fuada S, Afifah K, & Purwanda IG. Efficient android software development using mit app inventor 2 for bluetooth-based smart home. Wireless Personal Communications 2019;105(1), 233-256.
[49] Karakus M, Uludag S, Guler E, Turner SW, & Ugur A. Teaching computing and programming fundamentals via App Inventor for Android, 2012 International Conference on Information Technology Based Higher Education and Training; 2012;(pp. 1-8). IEEE.
[50] Kushwah M, & Patra A. Tuning PID controller for speed control of DC motor using soft computing techniques-A review. Advance in Electronic and Electric Engineering 2014; 4(2), 141-148.
[51] Ang KH, Chong G, & Li Y. PID control system analysis, design, and technology. IEEE transactions on control systems technology 2005; 13(4), 559-576.