Article

Holonomic Mobile Robot Navigation System Based on Collaboration Fuzzy Controller and Particle Swarm Optimization

Indra Adji Sulistijono [*] , Andy Yuniawan , Ali Ridho Barakbah , Zainal Arief , Naoyuki Kubota [*]

Posted Date: 26 August 2024

doi: 10.20944/preprints202408.1815.v1

Keywords: holonomic mobile robot; fuzzy controller; position driver; obstacle avoidance; particle swarm optimization; path planning

Article

Holonomic Mobile Robot Navigation System Based on Collaboration Fuzzy Controller and Particle Swarm Optimization

Indra Adji Sulistijono [1],*, Andy Yuniawan [1], Aliridho Barakbah [1], Zainal Arief [1] and Naoyuki Kubota [2],*

[1] Graduate School of Engineering Technology, Politeknik Elektronika Negeri Surabaya, Kampus PENS, Jalan Raya ITS Sukolilo, Surabaya 60111, Indonesia
[2] Graduate School of Systems Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan
* Correspondence: indra@pens.ac.id; Tel: +62-31-594 7280 Ext. 7310; kubota@tmu.ac.jp

Abstract: Currently, many health workers have died of being infected by dangerous infectious diseases. One solution is to prevent the spread of transmissions by minimizing contact between patients and staff or others using a mobile robot for delivering logistics services. We propose an approach by applies a navigation system using a position driver and obstacle avoidance using a fuzzy controller and particle swarm optimization for path planning in an unknown environment for a holonomic mobile robot. A Fuzzy Controller was used for obstacle avoidance so that the mobile robot could avoid obstacles using existing proximity sensors as the perception of the robot. In the path planning section, Particle Swarm Optimization is used to find intermediate points that must be reached by the mobile robot to reach their target. First, we performed simulation under the condition that the simulation resolution was similar to that of real conditions. It was found that the Trigonometric Function Adaptive PSO (TFAPSO) was better for finding the optimal solution based on the safest, shortest path, and lowest time. Finally, the holonomic mobile robot is tested directly in an unknown environment with the trajectory path generated by TFAPSO in real condition.

Keywords: holonomic mobile robot; fuzzy controller; position driver; obstacle avoidance; particle swarm optimization; path planning

1. Introduction

The World Health Organization (WHO) declared Covid-19 as a global pandemic in March 11, 2020. [1] Respiratory droplets from coughing, sneezing, and talking; indirect contact by touching contaminated surfaces with hands leading to contact with oral and nasal cavities, eyes, and mucous membranes; fecal-oral transmission and aerosol transmission have been implicated as the causes of the spread of Covid-19. [2]. Therefore, the hospital is used as a place for dangerous infectious diseases patients to isolate themselves and avoid contact with non-infected patients. However, every patient requires food, medicine, or other items that will be delivered by hospital workers. This pandemic gives us many lessons that we can learn and improve in our future lives. One solution is to prevent the spread of dangerous infectious disease transmissions by minimizing contact between patients and staff or others.

With current technological developments, robotics continues to enter areas such as factories, homes, offices, military and even the medical field. Semi- or fully autonomous mobile robots that assist humans, service equipment, and perform other autonomous functions have applications in almost every industry [3]. Based on the annual report from the International Federation of Robotics in 2020, the use of mobile robots for general use continued to increase from 2010 to 2018. Only in 2019 did it decrease because of the global economic downturn. [4] The development of kinematic path-tracking control algorithm for differential wheeled mobile robot system using back-stepping technique have already done. It demonstrated by overcoming the unmodeled kinematic disturbances, minimizing the tracking error and also ptimal path planning of the wheeled mobile robot with collision avoidance by using an algorithm called grey wolf optimization (GWO) as a method for finding the shortest and safe. [5,6]

With the current function of the hospital due to dangerous infectious diseases transmission, which is crucial for the process of prevention and healing of infectious disease patients, it is necessary to have service support from various sides, including automation and robotics, to maximize the role of the main hospital service, namely patient care, one of which is logistics service to patients with notes by minimizing contact between patients and non-patients. Although the pandemic has passed, it has provided many cases for researchers to develop logistics delivery robots in hospitals to avoid direct contact between healthcare workers and patients with acute infectious diseases. Mobile robots are considered solutions to overcome these problems [7]. However, there are several challenges when implementing mobile robots for logistics services in hospitals, with the main objective being to ensure the safety of people or other objects when the mobile robot needs to navigate in an environment with many obstacles and humans. This study discusses an approach that might be an effective solution to the problem of hospital logistics services.

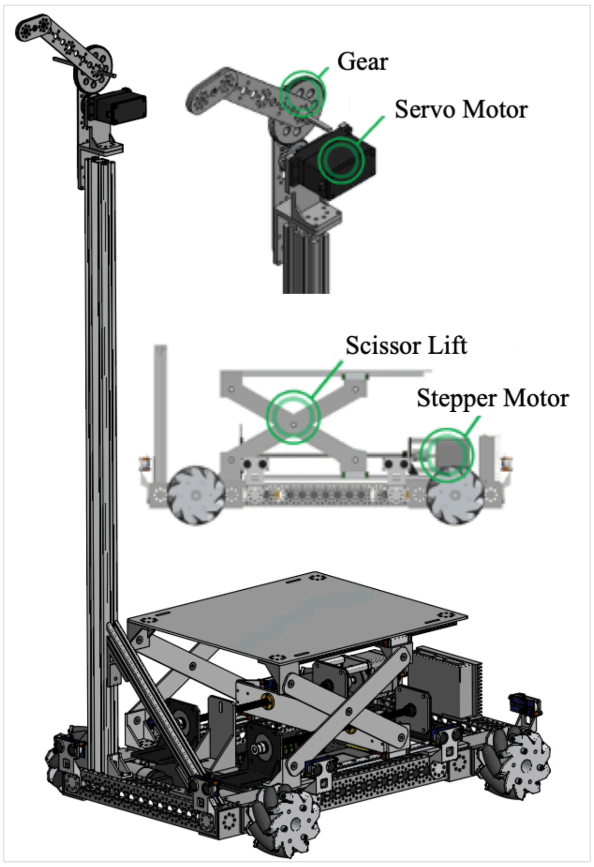
We propose an approach that applies a navigation system using a position driver and obstacle avoidance using a fuzzy controller and particle swarm optimization for path planning in an unknown environment for a holonomic mobile robot. A holonomic mobile robot was used because the controllable degrees of freedom are equal to the total degrees of freedom and can move freely in any direction. A Fuzzy Controller (FC) is used for obstacle avoidance so that the mobile robot can avoid obstacles using existing proximity sensors with a low computational cost and linguistic variables used so that the algorithm is easy to understand. In the path planning section, Particle Swarm Optimization (PSO) is used to find intermediate points that must be reached by the mobile robot to reach their target. PSO is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality [8,9]. Similar to other evolutionary algorithms, PSO searches for optimal spatial solutions through individual collaboration and competition. However, compared to evolutionary algorithms, the PSO concept is simple and easy to implement [10,11]. The proposed navigation algorithm for the holonomic mobile robot can optimize the motion to navigate based on the safest, shortest path, and lowest time.

This paper presents our approach for holonomic mobile robots to help hospital logistics services minimize the transmission of dangerous infectious diseases. The paper is structured as follows: Section 2 presents the design of the hardware and features of our mobile robot system. Section 3 presents the first algorithm for the navigation system for obstacle avoidance and the second algorithm for the navigation system for path planning and navigation systems using a position driver and obstacle avoidance using a fuzzy controller and particle swarm optimization. Section 4 discusses the experimental results of obstacle avoidance and path planning under simulated and real condition. Section 5 presents the conclusions and future perspectives of this study.

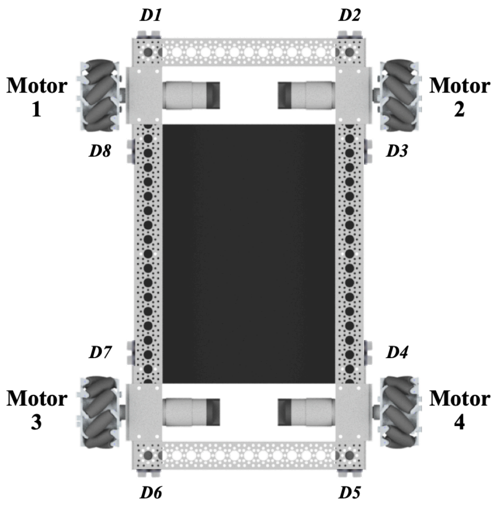
2. Hardware and Features

2.1. Mechanical Design

A mobile robot can move from one place to another autonomously, that is, without assistance from external human operators. Unlike most industrial robots that can move only in a specific workspace, mobile robot have the special feature of freely moving around within a predefined workspace to achieve their desired goals. A holonomic mobile robot was used because the controllable degrees of freedom are equal to the total degrees of freedom and it can move in any direction freely. The design of the holonomic mobile robot needs to be as strong as possible and has an optimal dimension for carrying the logistic stuff. The design concept of the mobile robot and its object and mobile management system is illustrated in Figure 1.



(a) Object management system



(b) Mobile management system

Figure 1. Design concept of the mobile robot

An object management system is used to manage objects; namely there are two elements: the scissor lift, which is used to lift the rack containing the logistics staff that must be delivered to the patient, and the door opener, which is used to open the door so that the mobile robot can enter the room. Figure 1 shows the scissor lift design [5].

A mobile management system is used to manage movement and as illustrated inFigure **??**. The motion of the mobile robot was regulated by the wheel velocity, which was actuated by a DC motor. There are four wheels that actuate four motors ($M1$, $M2$, $M3$, and $M4$), namely the mecanum wheel, which can make the mobile robot move holonomically. The perception system of the mobile robot was perceived by eight ultrasonic sensors $(D1, D2, D3, ..., D8)$.

2.2. Mobile Robot Inverse Kinematics

The main objective of this research is to design a holonomic mobile robot and generate optimal motion by designing a navigation algorithm such that it can navigate in an unknown environment with the challenge of obstacles that represent the environment of a hospital when a mobile robot performs a logistics delivery task.

The inverse kinematics are conducted to control the mobile robot motion and to convert the mobile robot velocity (V_o) component and the angular velocity of the wheel (V_w), [12]. The inverse kinematics equation of the system can be established using the kinematic analysis in Equation (1).

$$V_w = J(\alpha) \cdot V_o \tag{1}$$

where $V_w = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 & \omega_4 \end{bmatrix}^T$ denotes velocity of the wheel. $V_o = \begin{bmatrix} v_x & v_y & \omega_o \end{bmatrix}^T$ denotes the velocity of the mobile robot. $J(\alpha)$ is Jacobian matrix of the inverse equation of motion, which can be calculated using Equation (2).

$$J(\alpha) = \frac{1}{r}\begin{bmatrix} 1 & \frac{1}{\tan\alpha} & -\frac{L_1\tan\alpha + L_2}{\tan\alpha} \\ 1 & -\frac{1}{\tan\alpha} & \frac{L_1\tan\alpha + L_2}{\tan\alpha} \\ 1 & -\frac{1}{\tan\alpha} & -\frac{L_1\tan\alpha + L_2}{\tan\alpha} \\ 1 & \frac{1}{\tan\alpha} & \frac{L_1\tan\alpha + L_2}{\tan\alpha} \end{bmatrix} \tag{2}$$

In this model, because the mobile robot uses the mecanum wheel, α=45^o, the inverse kinematics equation of the system is obtained by Equation (3).

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & 1 & -L_1 + L_2 \\ 1 & -1 & L_1 + L_2 \\ 1 & -1 & -L_1 + L_2 \\ 1 & 1 & L_1 + L_2 \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ \omega_0 \end{bmatrix} \tag{3}$$

where ω_i is the angular velocity of the ith wheel, L_1 is the wheel separation width, L_2 is the wheel separation length, and r is wheel radius.

The velocity of the actual wheel can be measured using a rotary encoder on each motor of the actuator for the wheel and is expressed using Equation (4).

$$\omega_{act} = \frac{N}{PPR \times GR \times d(t)} \tag{4}$$

where ω_{act} denotes the actual wheel velocity at dimension i, N denotes the encoder pulse number, PPR denotes the pulse per rotation, GR denotes the gear ratio, and $d(t)$ denotes the time interval.

2.3. Odometry System

The odometry system is based on the position of the mobile robot related to the Cartesian coordinates (x, y, θ). To obtain the odometry data, first, calculate the robot velocity (v_x, v_y, ω_o) and using the forward kinematics from the wheel velocity ω_i as shown in Equation (5) [13].

$$\begin{bmatrix} v_x \\ v_y \\ \omega_0 \end{bmatrix} = \frac{r}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -\frac{1}{L_1+L_2} & \frac{1}{L_1+L_2} & -\frac{1}{L_1+L_2} & \frac{1}{L_1+L_2} \end{bmatrix}\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{5}$$

where ω_i is the angular velocity of the ith wheel, L_1 is the wheel separation width, L_2 is the wheel separation length, and r is the wheel radius.

Then, the mobile robot position related to the global coordinate system is calculated using Equation (6).

$$P_{t+1} = P_t + \begin{bmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} v_x \\ v_y \\ \omega_0 \end{bmatrix} d(t) \tag{6}$$

where P_t is the mobile robot position in x and y coordinates at time step t and θ_t is the mobile robot orientation at time step t. Mobile robot orientation is calculated by integrating the $\omega_o = \dot{\theta}$ which can be expressed by Equation (7).

$$\theta_{t+1} = \theta_t + \dot{\theta} d(t) \tag{7}$$

3. Proposed Methods

3.1. Fuzzy Controller and Particle Swarm Optimization

A Fuzzy Controller (FC) was proposed by Lotfi A. Zadeh in 1965 of the University of California at Berkeley [14], which contains the basic idea of a fuzzy controller including inclusion, union, intersection, complement, relationship, and convexity. This idea continues to be matured by Zadeh in several papers. The pioneer of the application of fuzzy controllers in the control field, which was the first and main application of fuzzy controller, is Prof. Ebrahim Mamdani and friends from Queen Mary College London [15]. Applications include processes in mixing tanks [16] and steam engines [17], on a laboratory scale. The application of fuzzy controllers in real time in the industry was pioneered by experts from Japan, such as Prof. Sugeno and colleagues from the Tokyo Institute of Technology [18,19].

And Particle Swarm Optimization (PSO) was originally invented by Kennedy, Eberhart and Shi [20,21] and was first intended for simulating social behavior as a stylized representation of the movement of organisms in a bird flock or fish school [22]. Optimization was performed and the algorithm was simplified as described by Kennedy and Eberhart. It describes many philosophical aspects of PSO and swarm intelligence[23].

We propose the application of navigation system using position drivers and obstacle avoidance using a fuzzy controller and particle swarm optimization for path planning in an unknown environment for a holonomic mobile robot. A Fuzzy Controller (FC) is used for obstacle avoidance so that the mobile robot can avoid obstacles using existing proximity sensors with a low computational cost and linguistic variables used so that the algorithm is easy to understand. In the path planning section, Particle Swarm Optimization (PSO) is used to find intermediate points that must be reached by the mobile robot to reach their target.

Figure 2 shows the system design from this research, which consists of a hardware system in the form of a holonomic mobile robot equipped with a perception system namely, the odometry system and distance sensor acquisition; the mobile robot can perceive the environment in which the mobile robot works, a low level control to control systems at a low level of the mobile robot that consists of inverse kinematics which equipped with wheel velocity control using a PID controller, control systems at a high level of the mobile robot that consists of a position driver, obstacle avoidance using a fuzzy controller (FC), and path planning using particle swarm optimization (PSO).

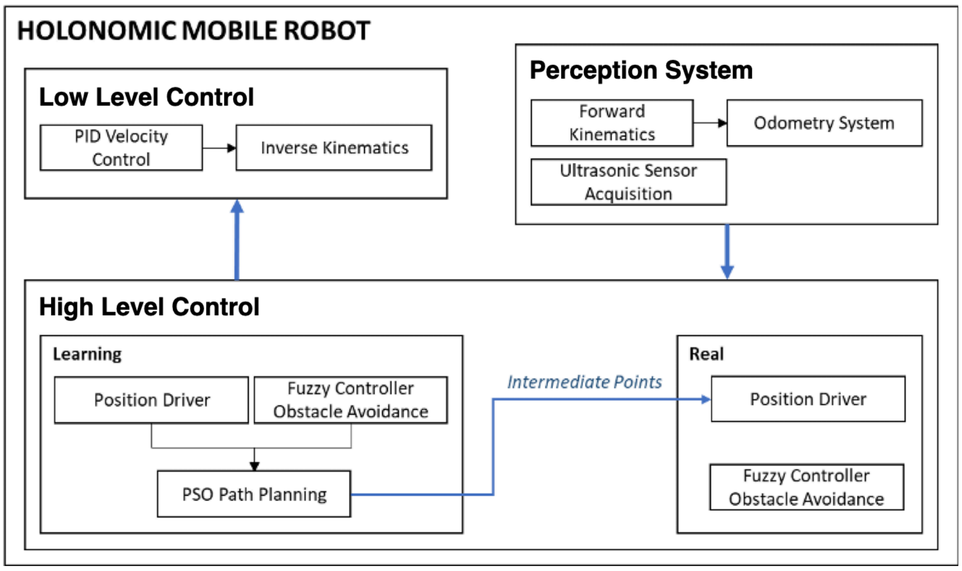


Figure 2. The design of navigation system of the holonomic mobile robot, which consist of a hardware system, perception system, low level control, and high level control.

The mobile robot data are operated by inputting the target position value in the form of room number from the hospital, where each room number has coordinates on the x and y axes. Then, path planning is executed using the particle swarm optimization algorithm (PSO).When conducting the fitness function evaluation process, the mobile robot is equipped with a fuzzy controller for obstacle avoidance, and the path planning process generates intermediate points that must be achieved by the mobile robot to arrive at the target position. Then, the mobile robot goes to each intermediate point using a position driver and obstacle avoidance using a fuzzy controller to avoid obstacles when heading to the intended targets.

3.2. Position Driver and Obstacle Avoidance

The mobile robot was required to move to the target point (position driver) and to avoid obstacles in the environment. This requires information regarding the distance and its direction toward the target point as the input variable for the position driver and the distance between the robot and the obstacles in each sensing direction as input for obstacle avoidance. There are several position driver methods that can be used, some use a camera mounted above the work area of the mobile robot, and some use odometry from the encoder [24]. We divide the control rules into two behaviors: position driver and obstacle avoidance, similar to a subsumption architecture or behavior-based control system [25]. A diagram of the control rules is shown in Figure 3.

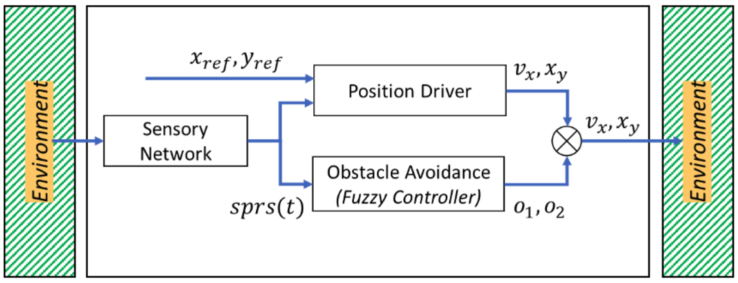


Figure 3. Diagram of the control rules: position driver and obstacle avoidance.

3.2.1. Position Driver

The velocity of the mobile robot (v_x and v_y) is also scaled by the result of the sensory network, which is the degree of the sparseness of obstacles $sprs(t)$ as shown in Equation (8) and (9).

$$v_x(t) = sprs(t) \cdot v_x(t) \tag{8}$$

$$v_y(t) = sprs(t) \cdot v_y(t) \tag{9}$$

When $sprs(t)$ are small, the velocity is reduced, and the mobile robot is capable of moving carefully at a high density of obstacles. Next, we consider the control rules for position-driver behavior. The control rules for both velocities (v_x and v_y) are expressed by Equations (10) and (11).

$$v_x(t) = K_p \cdot e\cos\alpha + K_i \int_0^t e\cos\alpha + K_d \cdot \frac{d}{dt} e\cos\alpha \tag{10}$$

$$v_y(t) = K_p \cdot e\sin\alpha + K_i \int_0^t e\sin\alpha + K_d \cdot \frac{d}{dt} e\sin\alpha \tag{11}$$

where K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain, e is the error distance between the actual position (x_{act},y_{act}) and the target reference positions (x_{ref},y_{ref}), and α are the heading errors. Then, e and α can be calculated using Equations (12) and (13), respectively.

$$e = \sqrt{(x_{ref} - x_{act})^2 + (y_{ref} - y_{act})^2} \tag{12}$$

$$\alpha = tan^{-1}\left(\frac{y_{ref} - y_{act}}{x_{ref} - x_{act}}\right) \tag{13}$$

3.2.2. Obstacle Avoidance

When dealing with obstacle avoidance, it means dealing with decision making. The algorithm used for obstacle avoidance in this study is a fuzzy controller. The fuzzy controller comprises of three main parts: linguistic variables, membership functions, and rules. In other words, linguistic variables were used to control the input and output variables of the system. When we create a linguistic variable to represent an input or output variable, we determine the number linguistic terms, or categories of the values of the linguistic variable. Membership functions (MFs) are numerical functions that correspond to the linguistic terms. A membership function represents the degree of membership of the linguistic variables within their linguistic terms. There are two membership functions: input and output linguistic variables. A triangular membership function was used to reduce the computation time and the parameter settings of the fuzzy controller. The triangular membership function is described by Equation (14).

$$\mu_{A_{i,j}}(x_j) = \begin{cases} 1 - \frac{|x_j - a_{i,j}|}{b_{i,j}}, & |x_j - a_{i,j}| \le b_{i,j} \\ 0, & |x_j - a_{i,j}| > b_{i,j} \end{cases} \tag{14}$$

where $A_{(i,j)}$ is a membership function for the jth input of the ith rule, $a_{(i,j)}$ and $b_{(i,j)}$ are the central value and width of the membership function, $A_{(i,j)}$, respectively.

The input linguistic variable membership function is the degree of danger and represents the degree of danger using a distance sensor. Two linguistic terms, dangerous and safe, are used to represent the degree of danger as a function of distance as the jth input x_j, where μ_{danger} corresponds to the degree of danger. The obstacle avoidance behavior of the mobile robot should involve actions that reduce the value of μ_{danger}. The total number of input linguistic variables was eight because eight distance sensors were used as inputs. Figure 4 shows the input variable membership function for the degree of danger.

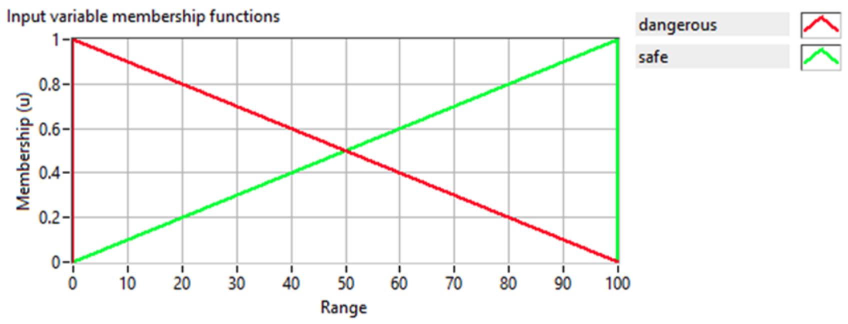


Figure 4. Input variable membership function for the degree of danger.

The output linguistic variable membership function is the robot velocity and represents the velocity of the mobile robot that will be controlled. The linguistic terms of the robot velocity must represent both the direction and magnitude of velocity changes. Therefore, we used negative large, negative small, zero, positive small, and positive large for this output variable. The total number of output linguistic variables is two, because there are two velocities that will be controlled: linear velocity x (v_x) and linear velocity y (v_y). Figure 5 shows the output variable membership function of the robot velocity.

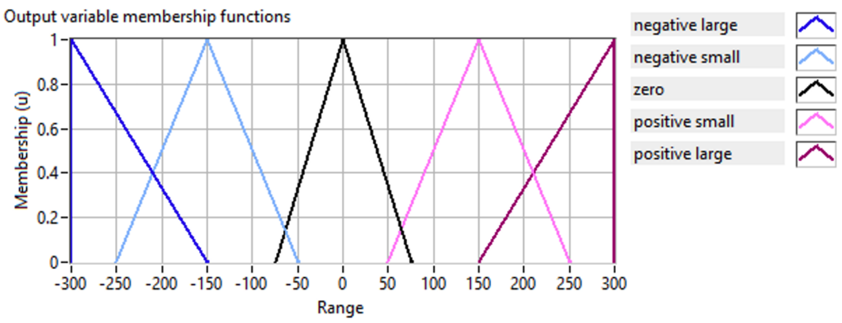


Figure 5. Output variable membership function of the robot velocity.

In this fuzzy controller, we apply a simplified fuzzy controller owing to its simple architecture and low computational complexity cost. A fuzzy if-then rule using the simplified fuzzy controller method is described as follows:

IF x_1 is $A_{(i,1)}$ and ... and x_j is $A_{(i,j)}$ and ... and x_n is $A_{(i,n)}$
THEN y_1 is $w_{(i,1)}$ ... and y_j is $w_{(i,j)}$ and ... and y_o is $w_{(i,o)}$

where $A_{(i,j)}$ is a membership function for the jth input of the ith rule, $w_{(i,j)}$ is a singleton for the jth output of the ith rule, and n, o and r are the numbers of inputs, outputs, and rules, respectively. The complete and detailed rules are shown in Figure 6.

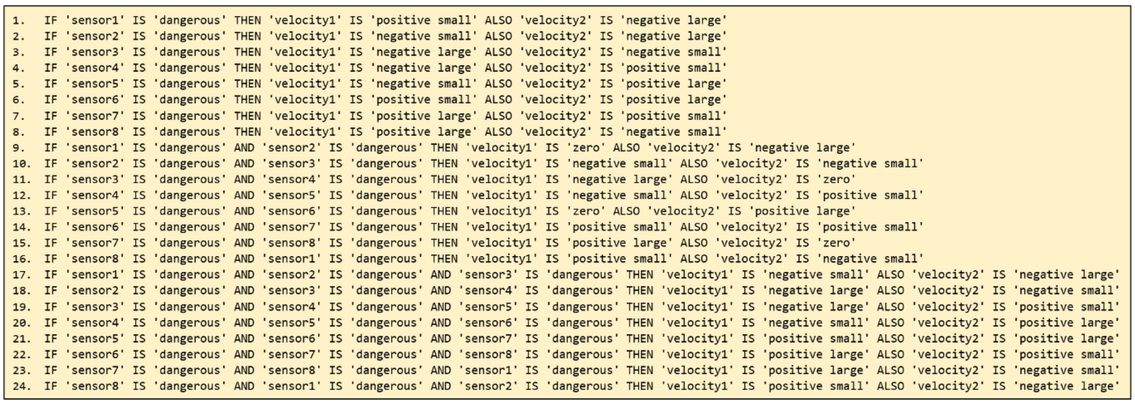

```
1.   IF 'sensor1' IS 'dangerous' THEN 'velocity1' IS 'positive small' ALSO 'velocity2' IS 'negative large'
2.   IF 'sensor2' IS 'dangerous' THEN 'velocity1' IS 'negative small' ALSO 'velocity2' IS 'negative large'
3.   IF 'sensor3' IS 'dangerous' THEN 'velocity1' IS 'negative large' ALSO 'velocity2' IS 'negative small'
4.   IF 'sensor4' IS 'dangerous' THEN 'velocity1' IS 'negative large' ALSO 'velocity2' IS 'positive small'
5.   IF 'sensor5' IS 'dangerous' THEN 'velocity1' IS 'negative small' ALSO 'velocity2' IS 'positive large'
6.   IF 'sensor6' IS 'dangerous' THEN 'velocity1' IS 'positive small' ALSO 'velocity2' IS 'positive large'
7.   IF 'sensor7' IS 'dangerous' THEN 'velocity1' IS 'positive large' ALSO 'velocity2' IS 'positive small'
8.   IF 'sensor8' IS 'dangerous' THEN 'velocity1' IS 'positive large' ALSO 'velocity2' IS 'negative small'
9.   IF 'sensor1' IS 'dangerous' AND 'sensor2' IS 'dangerous' THEN 'velocity1' IS 'zero' ALSO 'velocity2' IS 'negative large'
10.  IF 'sensor2' IS 'dangerous' AND 'sensor3' IS 'dangerous' THEN 'velocity1' IS 'negative small' ALSO 'velocity2' IS 'negative small'
11.  IF 'sensor3' IS 'dangerous' AND 'sensor4' IS 'dangerous' THEN 'velocity1' IS 'negative large' ALSO 'velocity2' IS 'zero'
12.  IF 'sensor4' IS 'dangerous' AND 'sensor5' IS 'dangerous' THEN 'velocity1' IS 'negative small' ALSO 'velocity2' IS 'positive small'
13.  IF 'sensor5' IS 'dangerous' AND 'sensor6' IS 'dangerous' THEN 'velocity1' IS 'zero' ALSO 'velocity2' IS 'positive large'
14.  IF 'sensor6' IS 'dangerous' AND 'sensor7' IS 'dangerous' THEN 'velocity1' IS 'positive small' ALSO 'velocity2' IS 'positive small'
15.  IF 'sensor7' IS 'dangerous' AND 'sensor8' IS 'dangerous' THEN 'velocity1' IS 'positive large' ALSO 'velocity2' IS 'zero'
16.  IF 'sensor8' IS 'dangerous' AND 'sensor1' IS 'dangerous' THEN 'velocity1' IS 'positive small' ALSO 'velocity2' IS 'negative small'
17.  IF 'sensor1' IS 'dangerous' AND 'sensor2' IS 'dangerous' AND 'sensor3' IS 'dangerous' THEN 'velocity1' IS 'negative small' ALSO 'velocity2' IS 'negative large'
18.  IF 'sensor2' IS 'dangerous' AND 'sensor3' IS 'dangerous' AND 'sensor4' IS 'dangerous' THEN 'velocity1' IS 'negative large' ALSO 'velocity2' IS 'negative small'
19.  IF 'sensor3' IS 'dangerous' AND 'sensor4' IS 'dangerous' AND 'sensor5' IS 'dangerous' THEN 'velocity1' IS 'negative large' ALSO 'velocity2' IS 'positive small'
20.  IF 'sensor4' IS 'dangerous' AND 'sensor5' IS 'dangerous' AND 'sensor6' IS 'dangerous' THEN 'velocity1' IS 'negative small' ALSO 'velocity2' IS 'positive large'
21.  IF 'sensor5' IS 'dangerous' AND 'sensor6' IS 'dangerous' AND 'sensor7' IS 'dangerous' THEN 'velocity1' IS 'positive small' ALSO 'velocity2' IS 'positive large'
22.  IF 'sensor6' IS 'dangerous' AND 'sensor7' IS 'dangerous' AND 'sensor8' IS 'dangerous' THEN 'velocity1' IS 'positive large' ALSO 'velocity2' IS 'positive small'
23.  IF 'sensor7' IS 'dangerous' AND 'sensor8' IS 'dangerous' AND 'sensor1' IS 'dangerous' THEN 'velocity1' IS 'positive large' ALSO 'velocity2' IS 'negative small'
24.  IF 'sensor8' IS 'dangerous' AND 'sensor1' IS 'dangerous' AND 'sensor2' IS 'dangerous' THEN 'velocity1' IS 'positive small' ALSO 'velocity2' IS 'negative large'
```

Figure 6. Fuzzy rules of obstacle avoidance.

The activation degree (fuzzification) of the ith rule (i=1,2,...,r) was calculated using Equation (15).

$$\mu_i = \prod_{j=1}^{n} \mu_{A_{i,j}}(x_j) \tag{15}$$

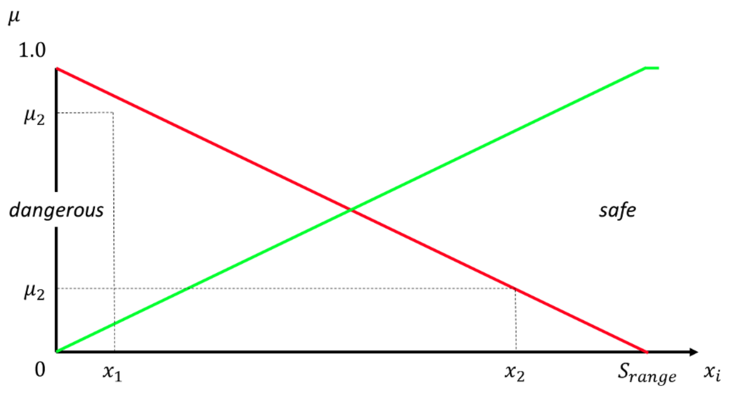
Next, we obtained the jth resulting output (j=1,2,...,o) by a weighted average (defuzzification) using Equation (16).

$$y_i = \frac{\sum_{i=1}^{r} \mu_i \cdot w_{i,j}}{\sum_{i=1}^{r} \mu_i} \tag{16}$$

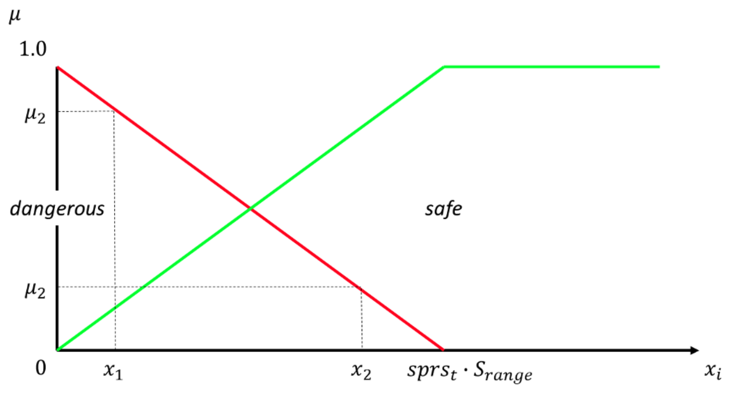
where μ_i is the activation degree of the ith rule of the fuzzy controller and y_j is the result of jth output of the fuzzy controller.

3.3. Sensory Network

A fuzzy controller for obstacle avoidance can be trained using teaching data suitable for a given environmental condition. However, if the fuzzy rules are not generalized, the fuzzy controller must be retrained when environmental conditions change. To construct compact and useful fuzzy rules, we used a sensory network with a callable attention range, which adjusts the shape of the membership functions [26,27]. Figure 7 shows the membership functions corresponding to the maximum and minimum attention ranges. Even if the input data and are the same, the resulting outputs and differ in scaled MFs.



(a) Based on the sensor range



(b) Based on the attention range

Figure 7. Membership functions corresponding.

The velocity and attention range of the mobile robot should be changed according to the density of obstacles. The attention range corresponds to $A_{(i,j)}$ of the membership function in fuzzy rules. The attention range $A_{range}(t)$ is changed as shown in Equation (17) and (18).

$$A_{range}(t) = sprs(t) \cdot S_{range} \tag{17}$$

$$f(x) = \begin{cases} \gamma^{-1} \cdot sprs(t), & \frac{\sum_{i=1}^{n} x_i}{n} \geq A_{range}(t) \\ \gamma \cdot sprs(t), & \frac{\sum_{i=1}^{n} x_i}{n} < A_{range}(t) \end{cases} \tag{18}$$

where $sprs(t)$ is the degree of sparseness of obstacles satisfying $0<spr_{min} \geq 1.0$ for the perception of the environment. S_{range} is the maximum sensing range and γ ($0<\gamma<1.0$) is the perception coefficient.

3.4. Path Planning

Mobile robots need intermediate points that they need to pass through to avoid getting trapped in the local area to reach their target. The mobile robot searches for the best position of intermediate points generated by evolutionary computation, namely, particle swarm optimization (PSO).

The mobile robot will be given a map of the environment where it performs its task, the unit of the map will be in pixels, and a pixel will represent a few centimeters in the real world. Then, the starting and target position are determined based on real-world case. The mobile robot will carry out the iterative learning process using PSO to be able to find the best intermediate point as solution that mobile robot needs to pass through to make the mobile robot be able to reach its target without getting trapped by local area that usually surrounded by the obstacles. The PSO method used was the gbest method, which allows every particle to obtain information from the best particle in the entire swarm. Figure 8 shows flowchart of the PSO for path planning.

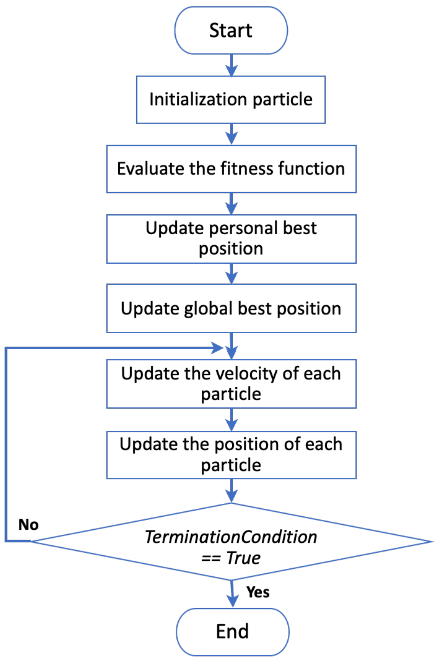

Start
Initialization particle
Evaluate the fitness function
Update personal best position
Update global best position
Update the velocity of each particle
Update the position of each particle
TerminationCondition == True
No
Yes
End

Figure 8. Flowchart of PSO for path planning.

The explanation of each step of particle swarm optimization (PSO) for path planning is

1. Initialization particle

   In this process the particles are initialized starting with the number, and then the velocity and position of each particle are initialized at random with the provisions of the minimum and maximum positions of the particles according to the given map. Initialize the number, random velocity, and random position of the particle. To avoid losing randomness and to avoid velocity and position values exceeding the specified minimum and maximum position conditions that can cause search stagnation, the random values are distributed evenly, as in Equation (19) and (20).

$$x_i(0) = x_{min} + r * (x_{max} - x_{min}) \tag{19}$$

$$v_i(0) = \beta * (x_{min} + r * (x_{max} - x_{min})) \tag{20}$$

   where x_{min} and x_{max} are the minimum and maximum values based on the workspace, respectively, $x_i(0)$ is the initial value of the position of the candidate solution, $v_i(0)$ is the initial value of the velocity of the candidate solution, β is the weight coefficient of the velocity, $r \sim U(0,1)$ is a random number.

2. Evaluate the fitness function

   Each intermediate point was evaluated based on the given fitness function. The mobile robot simulates the navigation process using the position driver and obstacle avoidance to reach intermediate points to the target point on the map, which has been given to generate a fitness value. The set of intermediate points is evaluated only when the mobile robot reaches the target point. The minimization on the evaluation function is shown in Equation (21).

$$f(x_i(t)) = W_1 \cdot ML + W_2 \cdot PL + W_3 \cdot MD + W_4 \cdot AD \tag{21}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 August 2024 doi:10.20944/preprints202408.1815.v1

where $f(x_i(t))$ is the fitness value of particle i at time t, ML is the moving length or total distance traveled by the mobile robot. PL is the path length from the starting, intermediate, and target points. MD is the actual minimum distance to the target point. AD is the average danger, and W_1, W_2, W_3, W_4 are the weight coefficients. The equations of the fitness function elements can be seen in Equations (22), (23), (24), and (25).

$$ML = \sum_{t=1}^{N} \sqrt{(v_x(t) \times v_x(t)) + (v_y(t) \times v_y(t))} \tag{22}$$

$$\begin{aligned} PL = \sqrt{(IPx_1 - X_{start})^2 + (IPy_1 - X_{start})^2} + \sum_{t=1}^{n} \sqrt{(IPx_{i+1} - IPx_1)^2 + IPy_{i+1} - IPy_1)^2} \\ + \sqrt{(x_{target} - IPx_n)^2 + (y_{target} - IPy_n)^2} \end{aligned} \tag{23}$$

$$MD = \sqrt{(x_{target} - x_{actual})^2) + (y_{target} - y_{actual})^2)} \tag{24}$$

$$AD = \frac{\sum_{t=1}^{N} \sum_{i=1}^{8} \frac{S_{range} - d_i}{Srange}}{N} \tag{25}$$

where $v_x(t)$ is the linear velocity x of the mobile robot at time t, $v_x(t)$ is the linear velocity y of the mobile robot at time t, IPx_i is the x coordinates of the intermediate point i, IPy_i is the y coordinates of the intermediate point i, n is the maximum number of intermediate points, S_{range} is the maximum sensing range, d_i is the actual value of distance sensor i.

3. Update personal best position

The search for the solution will be a minimization problem, so the personal best position, $y_i(t)$, at the next time step, $t + 1$, where $t \in [0, \ldots, N]$, is calculated as Equation (26).

$$y_i(t+1) = \begin{cases} y_i(t), & f(x_i(t+1)) \geq f(y_i(t+1) \\ (x_i(t+1), & f(x_i(t+1)) < f(y_i(t+1) \end{cases} \tag{26}$$

where f is the fitness function, $x_i(t)$ is position at time t.

4. Update global best position

The global best position $\hat{y}(t)$ at time step t is calculated using Equation (27).

$$\hat{y}(t) = \min\{f(x_o(t)), ..., f(x_{n_s}(t))\} \tag{27}$$

5. Update the velocity of each particle

PSO includes three parts: the current motion influence, individual particle influence, and particle swarm influence. The velocity of the particle was calculated using Equation (28).

$$v_{ij}(t+1) = \omega * v_{ij}(t) + c_1 * r_{ij}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 * r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)] \tag{28}$$

where $v_{ij}(t)$ is the velocity vector of particle i in dimension j at time t, $x_{ij}(t)$ is the position vector of particle i in dimension j at time t, $y_{ij}(t)$ is the personal best position of particle i in dimension j found from initialization through time t, $\hat{y}_j(t)$ is the global best position of particle i in dimension j found from initialization through time t, c_1 and c_2 are positive acceleration constants that are used to level the contribution of the cognitive and social components, respectively, $r_1j(t)$, $r_2j(t) \sim U(0,1)$ are two uniformly distributed random vectors (generated every iteration) at time t in where each component is in the range [0,1].

6. Update the position of each particle

$$x_{ij}(t+1) = x_{ij}(t)t + v_{ij}(t+1) \cdot \Delta t \tag{29}$$

3.4.1. The Improved PSO Using Learning Factors

The values of the learning factors are compared, namely ω as inertia weight and c_1 and c_2 as acceleration factors from standard PSO, Linear Adaptive PSO (LAPSO), and Trigonometric Function Adaptive PSO (TFAPSO). The values of ω, c_1 and c_2 were obtained using the following equation:

- Standard PSO
  In standard PSO, the inertia weight ω for the current velocity influence, the acceleration factors c_1 for the individual or cognitive influence and c_2 for the swarm or social influence are constant from the first iteration until the last iteration [21].
- Linear Adaptive PSO (LAPSO)
  LAPSO is another variant of standard PSO that was first introduced by Bonyadi and Michalewicz in 2013 [28]. With LAPSO, the learning factors (ω, c_1, and c_2) can change adaptively based on the search iteration time using Equations (30)–(32).

$$\omega(t) = \frac{t}{t_{max}}(\omega_{max} - \omega_{min}) \tag{30}$$

  where t_{max} is the number of final iterations, t is the number of iterations of the algorithm, and $\omega(t)$ is the inertia weight corresponding to iteration t. ω_{max} and ω_{min} are minimum and maximum values of ω.

$$c_1(t) = (1 - \frac{t}{t_{max}}(c_{max} - c_{min})) + a \tag{31}$$

$$c_2(t) = \frac{t}{t_{max}}(c_{max} - c_{min}) + a \tag{32}$$

- Trigonometric Function Adaptive PSO (TFAPSO)
  TFAPSO was introduced by Lian, Yu, and Xiao in 2020 as improved method for standard PSO [29]. With TFAPSO, the inertia weight factor and acceleration factors are adaptively adjusted using the trigonometric function at each stage of the algorithm operation using Equations (33)-(35).

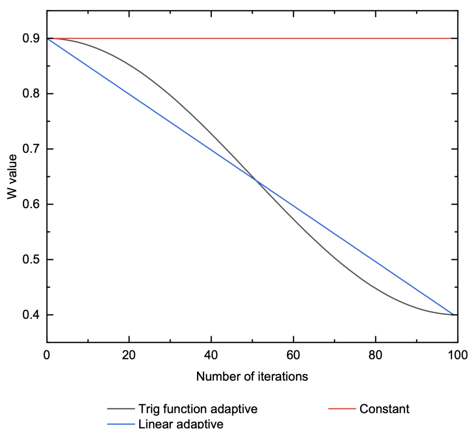
$$\omega(t) = \frac{\omega_{max} - \omega_{min}}{2} * \cos\left(\frac{\pi * t}{t_{max}}\right) + \frac{\omega_{max} + \omega_{min}}{2} \tag{33}$$

$$c_1(t) = \cos\left(\frac{\pi * t}{t_{max}}\right) + a \tag{34}$$

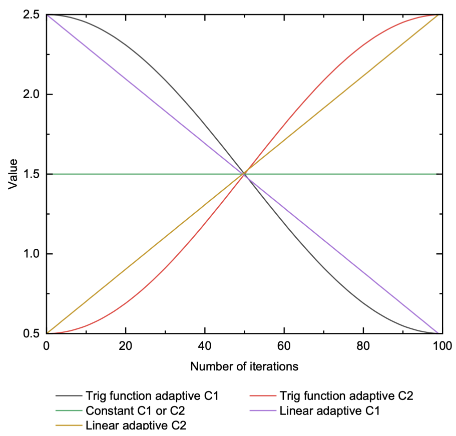
$$c_2(t) = -\cos\left(\frac{\pi * t}{t_{max}}\right) + a \tag{35}$$

The parameters ω, c_1 and c_2 in PSO are used to balance the search capabilities of the local and global search of particles in the swarm, and symbolizes self-cognition, symbolizes social influence. Figure 9 shows a comparison of ω, c_1 and c_2 between the PSO variants. We can see that in standard PSO the values of ω, c_1 and c_2 are constant from the first to the last iteration. However, in adaptive PSO, the values of ω, c_1 and c_2 change according to the iteration time. The values of ω, c_1 and c_2 are more likely to change based on the iteration time to be able to make the solution finding process, as explorative as possible at the beginning and as exploitative as possible at the end.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 August 2024 doi:10.20944/preprints202408.1815.v1



(a) ω as inertia weight



(b) c_1 and c_2 are positive acceleration constants

Figure 9. Learning factor comparison between PSO variant.

A larger ω is useful for jumping out of the local optimum, whereas a smaller ω is suitable for the algorithm to converge. While the optimal value of the individual particle is significant in the later stage, the optimal value of the particle swarm is crucial in the early stage of algorithm optimization.

4. Experimental Results And Analysis

This section describes the experiments and analyses conducted this study. The entire test data are explained starting from the realization of the mobile robot and the experimental results of the position driver, obstacle avoidance, and path planning.

4.1. Mobile Robot Realization

The hardware design of the mobile robot include a mobile management system and an object management system, as shown in Figure 10. The dimensions of the mobile robot were 61.35cm × 45.83cm × 113.68cm.

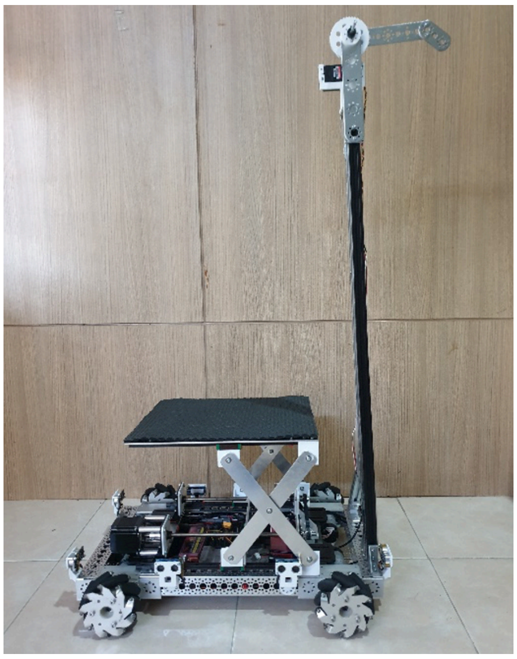


Figure 10. Mobile robot realization from side view.

Starting from the mobile management system, four DC motors equipped with an encoder with four mounted mecanum wheels on the main structural frame in the shape of a rectangle. Eight ultrasonic sensors were installed on each side of the two pieces, starting from the front, left, back, and left sides. In the object management system, the scissor lift, which is actuated by a stepper motor, assisted by two linear screws and two rail guides, has also been installed properly. There is an additional door-opening mechanism in the object management system that is actuated by a servo motor assisted by two gears to strengthen the torque of the servo motor which has also been installed properly. Thus, all the hardware required for this mobile robot has been installed and can be tested.

The procedure for operating the holonomic mobile robot is as follows: First, we input the target position in the x and y coordinates, and then the mobile robot runs the Particle Swarm Optimization (PSO) Path Planning with the Fuzzy Controller (FC) Obstacle Avoidance and Position Driver in the simulation to obtain the optimal intermediate point when needed. Then send it to the robot to execute the Position Driver and FC obstacle avoidance.

Figure 11 shows a snapshot of a mobile robot task process sequence. The mobile robot can carry out its functions starting from approaching the rack (1), entering the rack by moving backwards (2)(3), lifting the rack using a scissor lift (4), carrying the rack that is moving while the rack is still lifted (5), and delivering the rack to the desired target (6), namely at in front of the patient's door.

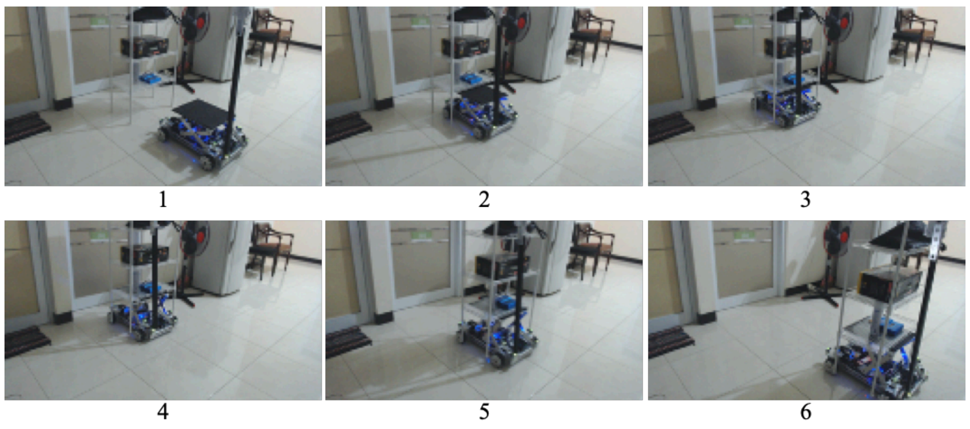
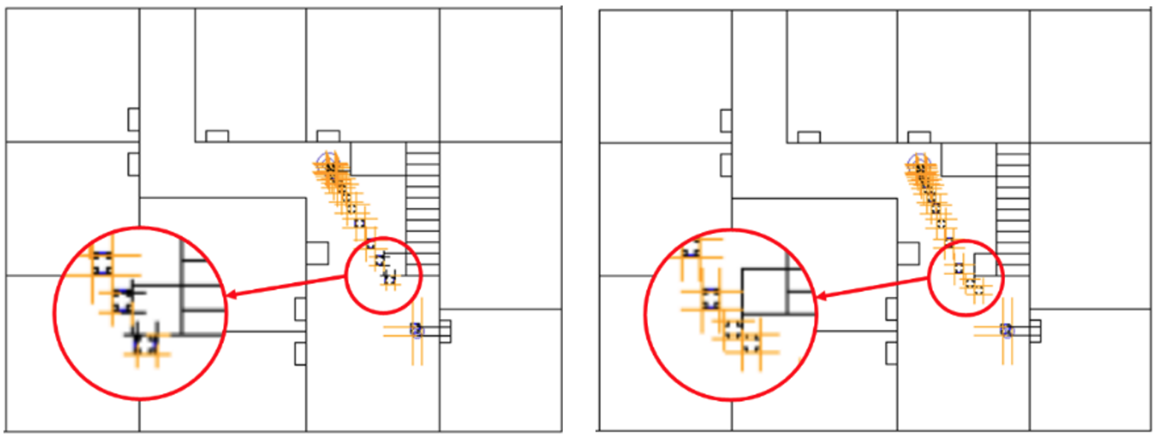


Figure 11. Snapshot of logistics delivery test.

4.2. Position Driver and Obstacle Avoidance (Simulation)

Before conducting the experiments in real conditions, the algorithm was first tested in the simulation. Figure 12 shows the simulation results where the size of the workspace is 380 × 500 pixels where one pixel in the simulation workspace represents 2 cm in the real workspace. The initial and target positions are (140, 100) and (220, 250), respectively. The mobile robot is depicted every 30 discrete time steps in this figure.



(a) Without obstacle avoidance (b) With obstacle avoidance

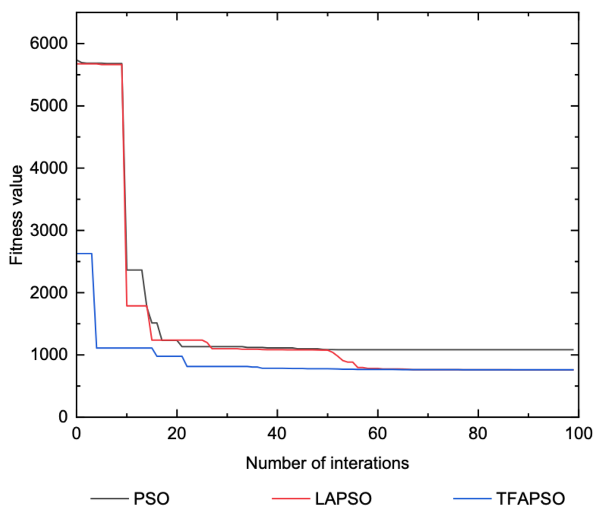
Figure 12. Simulation result of position driver.

From Figure 12 (a) the mobile robot collides an obstacle when it reaches the target if obstacle avoidance is not applied to the position driver, of course this is very dangerous when we apply it to the real mobile robot. However, in Figure 12(b) the mobile robot managed to avoid obstacles when heading toward the target if obstacle avoidance was applied to the position driver. Therefore, obstacle avoidance plays an important role in the mobile robot reaching the target point without colliding with obstacles or walls.

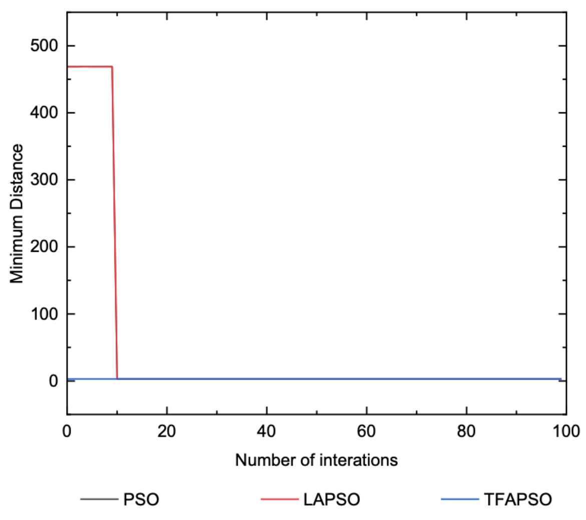
The case condition for the following figures and table is defined:

- Case 1:
  The radius of passing of each intermediate point was 20. The weights for the moving length, path length, minimum distance, and average danger were 0.5, 0.1, 1, 100.
- Case 2:
  The number of intermediate points and iterations were 2 and 100, respectively. The number of particles were 200. The radius for passing through each intermediate point was 20. The weights for the moving length, path length, minimum distance, and average danger were 0.5, 0.1, 1, 100.
- Case 3:
  The number of intermediate points and iterations were 2 and 100. The number of particles were 400. The radius of passing of each intermediate point is 20. The weights for the moving length, path length, minimum distance, and average danger were 0.5, 0.1, 1, 100.

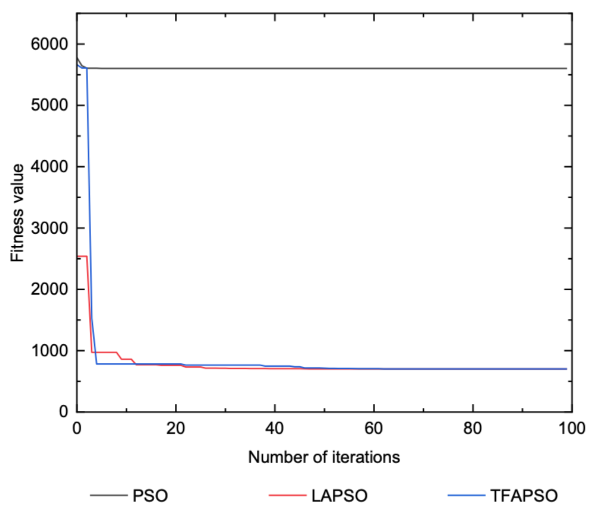
Figure 13 shows simulation result of comparison between PSO variants. Figure 13 (a) shows the fitness value in Case 1, 2 and 3. Figure 13 (a) shows minimum distance in Case 1, 2 and 3.



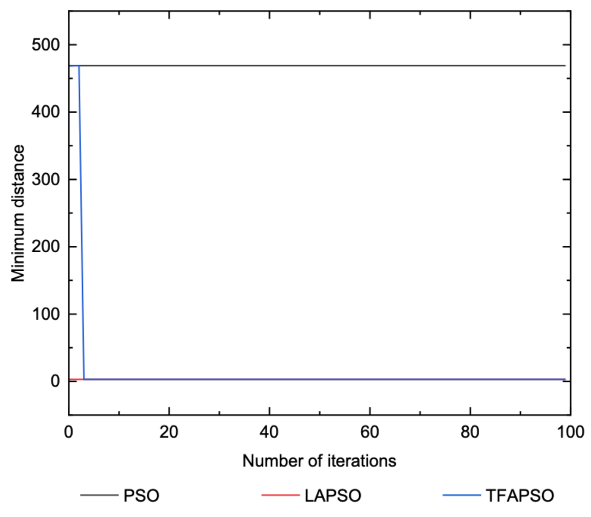
(a) Fitness value (Case 1)



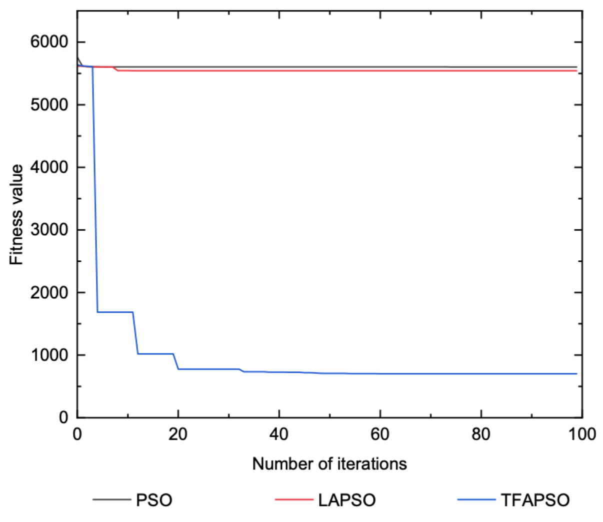
(b) Minimum distance (Case 1)



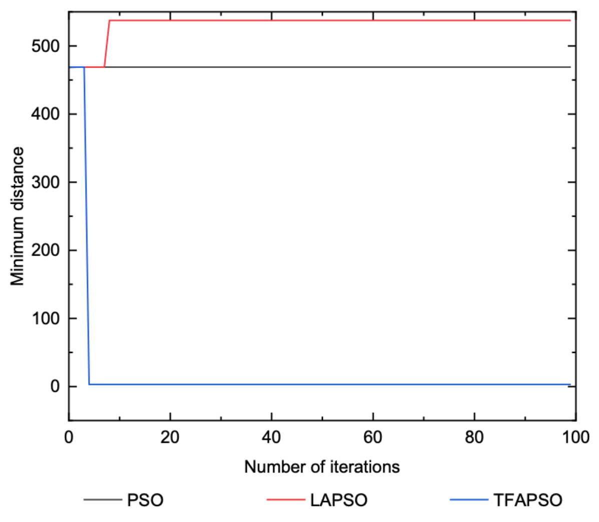
(a) Fitness value (Case 2)



(b) Minimum distance (Case 2)



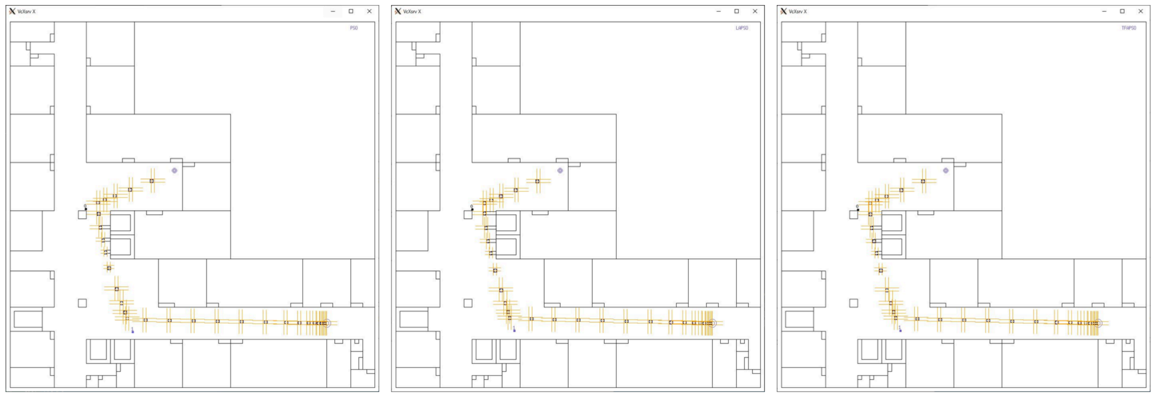
(a) Fitness value (Case 3)



(b) Minimum distance (Case 3)

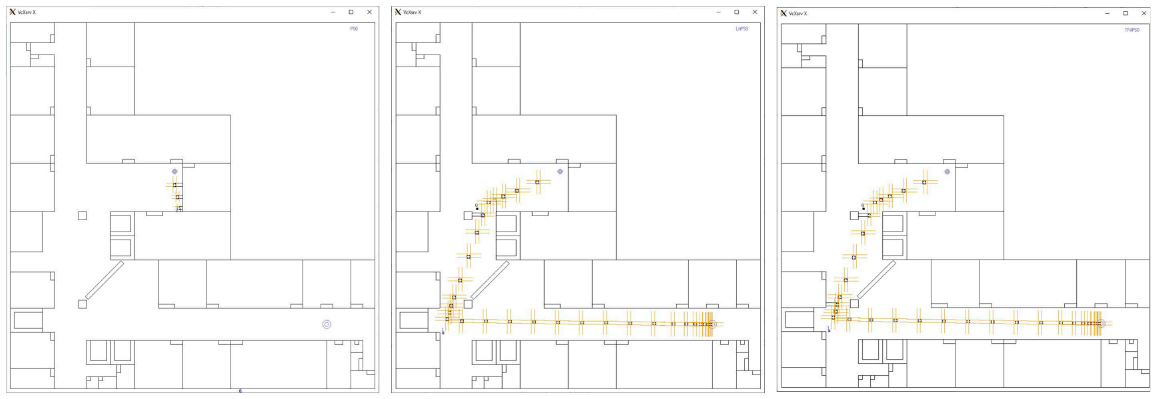
Figure 13. Simulation result of comparison between PSO variants

Figure 14 shows the simulation comparison between the Standard PSO, Linear Adaptive PSO (LAPSO) and Trigonometric Function Adaptive PSO (TFAPSO) in Case 1, Case 2 and Case 3. The simulation resolution was similar to that of real conditions. The obstacle is given to the environment, for testing so that the mobile robot can manage the other trajectory in order to reach the target. The mobile robot managed to avoid obstacles when heading toward the target if obstacle avoidance was applied to the position driver. Therefore, obstacle avoidance plays an important role in the mobile robot reaching the target point without colliding with obstacles or walls.



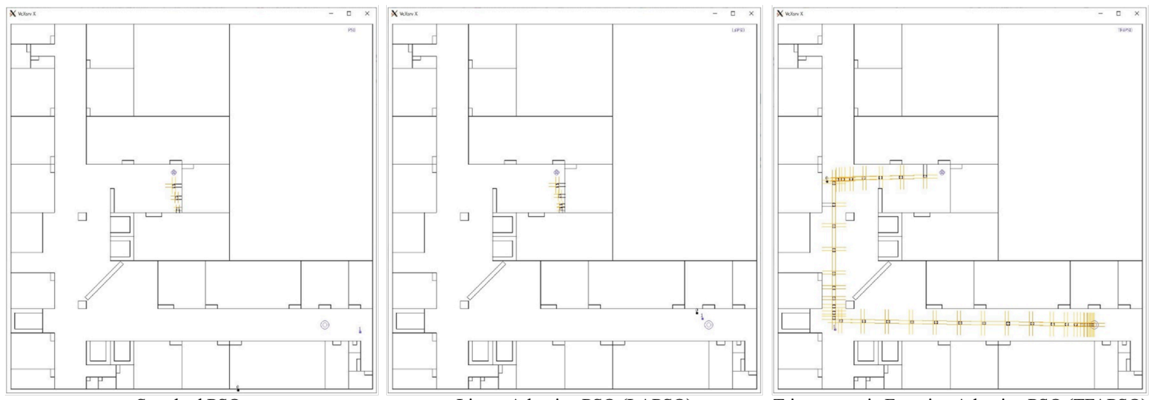
Standard PSO　Linear Adaptive PSO (LAPSO)　Trigonometric Function Adaptive PSO (TFAPSO)

(a) Case 1



Standard PSO　Linear Adaptive PSO (LAPSO)　Trigonometric Function Adaptive PSO (TFAPSO)

(b) Case 2



Standard PSO　Linear Adaptive PSO (LAPSO)　Trigonometric Function Adaptive PSO (TFAPSO)

(c) Case 3

Figure 14. Simulation comparison of position driver and obstacle avoidance between Standard PSO, Linear Adaptive PSO (LAPSO) and Trigonometric Function Adaptive PSO (TFAPSO) in Case 1, Case 2 and Case 3.

Figure 14 Case 1 shows the trajectory of the mobile robot using the intermediate points generated by several variants of PSO in Case 1. The mobile robot can reach the target point by passing the generated intermediate points. Differ from the Case 1, the second intermediate point is further away from the target due to an additional obstacle or blockade that requires the second intermediate point to find another position so that the mobile robot can avoid the obstacle and reach the target.

Figure 13 (a) shows a comparison of the fitness values of several PSO variants for Case 1. Figure 13 (b) shows the minimum distance of the mobile robot's position to the target point in Case 1. The mobile robot reached the target at iteration 10 for PSO, iteration 10 for LAPSO, and iteration 0 for TFAPSO.

Table 1 Case 1 shows the comparison of each fitness function element. From these results it can be seen that TFAPSO and LAPSO is better at finding the optimal solution in Case 1 compared to PSO because it has the lowest fitness value (minimization problem).

Table 1. Simulation results of the path planning.

	Case 1		
	PSO	LAPSO	TFAPSO
Fitness Value	1083.50881	760.37469	760.72364
Moving Length	1190.75288	1168.38170	1169.58586
Path Length	1205.74148	1221.1841	1221.82083
Minimum Distance	2.95770	2.94937	2.96559
Average Danger	3.64601	2.94937	2.96559
	Case 2		
	PSO	LAPSO	TFAPSO
Fitness Value	5601.59663	701.93917	701.93992
Moving Length	10094.44424	1270.95071	1270.94604
Path Length	836.51866	1311.03071	1310.99829
Minimum Distance	468.85061	2.95882	2.95882
Average Danger	0.77159	0.50395	0.50398
	Case 3		
	PSO	LAPSO	TFAPSO
Fitness Value	5603.19464	5542.77517	701.93874
Moving Length	10094.44367	10000.00000	1270.94964
Path Length	990.22125	537.40115	1311.02936
Minimum Distance	468.85027	537.40115	2.95882
Average Danger	0.77220	0.00000	0.50395

Figure 14 Case 2 shows the trajectory of the mobile robot using the intermediate points generated by several variants of PSO in Case 2. For PSO, the mobile robot cannot reach the target point because the generated intermediate points are in the wrong position, so it makes the mobile robot trapped to the local area surrounded by the wall or obstacle. It means the generated intermediate points is not good solution because the solution is trapped to the local optima. But the for the LAPSO and TFAPSO, the mobile robot can reach the target by passing the generated intermediate points. So, it means the generated intermediate points is a good solution.

Figure 13 (a) shows a comparison of the fitness values of several PSO variants in Case 2. Figure 13 (b) shows the minimum distance between the mobile robot position and the target point in Case 2. From these two graphs we can observe that the trajectory of the mobile robot based on the intermediate points generated by PSO in this setting was trapped in the local optima, which means that the mobile robot cannot reach its target.

We can see from Table 1 Case 2 that a large penalty is added to the moving length for PSO because the mobile robot was trapped at the dead end. But LAPSO and TFAPSO can find the best solution so it means the mobile robot can reach its target.

Figure 14 Case 3 shows the trajectory of the mobile robot using the intermediate points generated by several variants of PSO in Case 3. For PSO and LAPSO, the mobile robot cannot reach the target point because the generated intermediate points are in the wrong position, so it makes the mobile robot trapped to the local area surrounded by the wall or obstacle. It means the generated intermediate points is not good solution because the solution is trapped to the local optima. Only for the TFAPSO, the mobile robot can reach the target by passing the generated intermediate points.

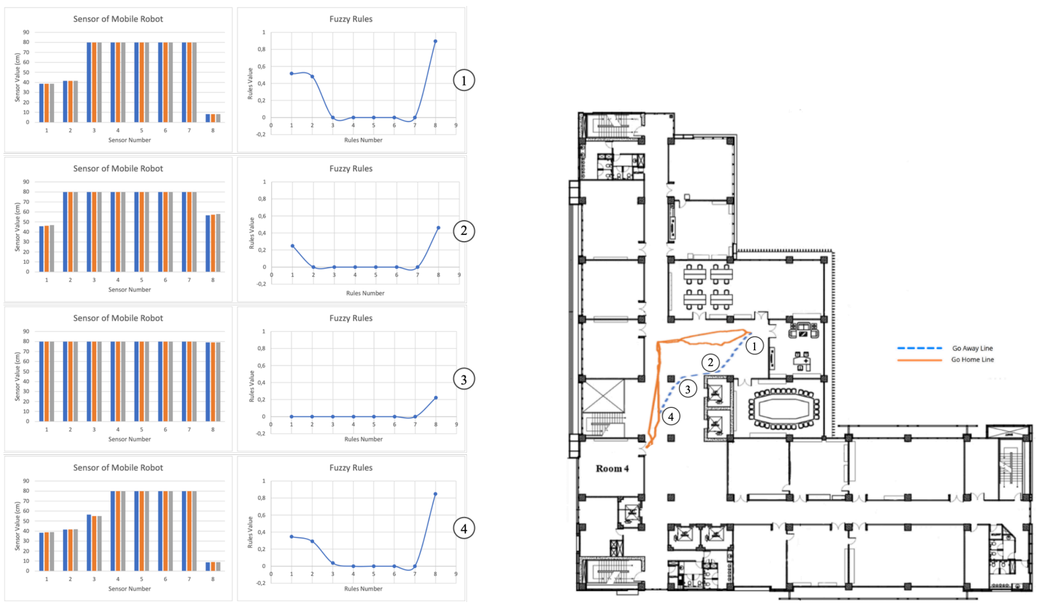
Figure 13(a) shows a comparison of the fitness values of several PSO variants for Case 3. Figure 13(b) shows the minimum distance between the mobile robot position and the target point in Case 3. From these two graphs we can see that the trajectories of the mobile robot based on intermediate points generated by PSO and LAPSO in this setting were trapped in the local optima. They could not find the optimal solution, which means that the mobile robot could not reach their target.

We can see from Table 1 Case 3 that a large penalty is added to the moving length for the PSO and LAPSO variants because the mobile robot were trapped at the dead ends. Only TFAPSO can find the best solution, which means that the mobile robot can reach its target.

4.3. Fuzzy Controller on Real Condition

The next experimental results describe the additional testing of the mobile robot's logistics delivery task, namely, to move and carry them, and to deliver them to the targeted place. The purpose of this test is to ensure that the mobile robot can carry out its function for logistics delivery tasks.

Figure 15 shows a comparison of the sensors of the mobile robot and fuzzy system rules that were used to move away from Room 4 to Home. Figure 15 (a) shows that the input from eight ultrasonic sensors (left side) can affect the dangerous membership function and fuzzy rule (right side). Each number shows the position as shown in Figure 15 (b), when the robot returned from Room 4 to Home. This input affects the rules that will proceed by the robot to drive the robot. Figure 15 (b) shows the real trajectory movement of the robot generated by PSO.



(a) Sensor of mobile robot and fuzzy system rules (b) Real robot trajectory movement from Home-Room 4-Home

Figure 15. Comparison of sensor of mobile robot and fuzzy system rules that used on going away from Room 4 to Home.

Figure 16 shows a snapshot of the real mobile robot experimental results from Home to the closest room (Room 1), going away, and going home. Figure 16 (a) shows a snapshot without obstacle avoidance and Figure 16 (b) shows a snapshot with obstacle avoidance. The number below each image show the sequence of the movement. From the snapshot, we can see that on the real movement holonomic mobile robot movement, the robot can reach the target and return back to Home well as shown in Figure 16 (a). From Figure 16 (b) the robot can detect the obstacle (even if the human is moving) through sensory network perception. The perceive of sensory network are represent the robot 'vision'. When a human is moving around the robot, the robot will avoid the human and try to find the trajectory generated by PSO again. The robot can reach the target room and return to the Home from the targeted room.

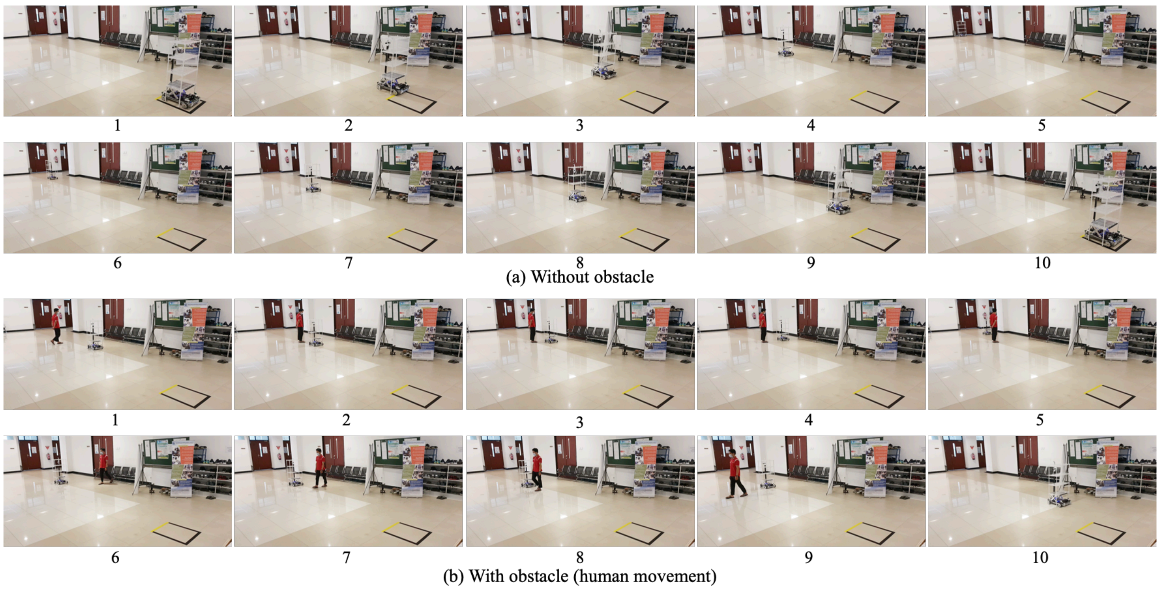


Figure 16. Snapshot of the real mobile robot experimental results from Home to the closest room (Room 1), going away and going home.

Figure 17 shows the real robot movement trajectory, moving away from the Home to the targeted room and returning back to Home from the targeted room. We selected rooms that represented the "L" shape to the right (Home-Room 2-Home), "L" shape to the left (Home-Room 4-Home) and "U" turn (Home-Room 3-Home). All trajectories, of course generated by PSO and the perception of the sensory network and the robot decision when obstacle avoidance is by fuzzy controllers.

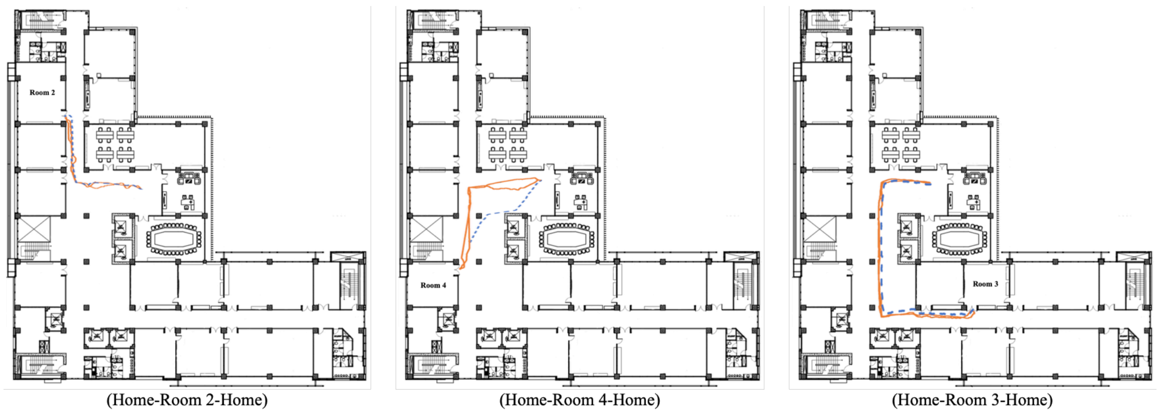


Figure 17. Real robot movement trajectory, moving away from the Home to the targeted room and returning back to Home from the targeted room.

The last figure as shown in Figure 18 is a snapshot of the real mobile robot movement experimental results from Home to the most distant room, go and away, and with and without obstacle avoidance. This is a "U" turn trajectory. The number below each image shows the sequence of movement. The holonomic mobile robot took a long journey and reached its target. Despite the obstacles surrounding the robot, the robot can avoid them and try to find the previous trajectory generated by the PSO to reach the target. Moreover, the robot could return back to the Home well. Figure 17 the right one (Home-Room 3-Home) shows the real robot movement trajectory.

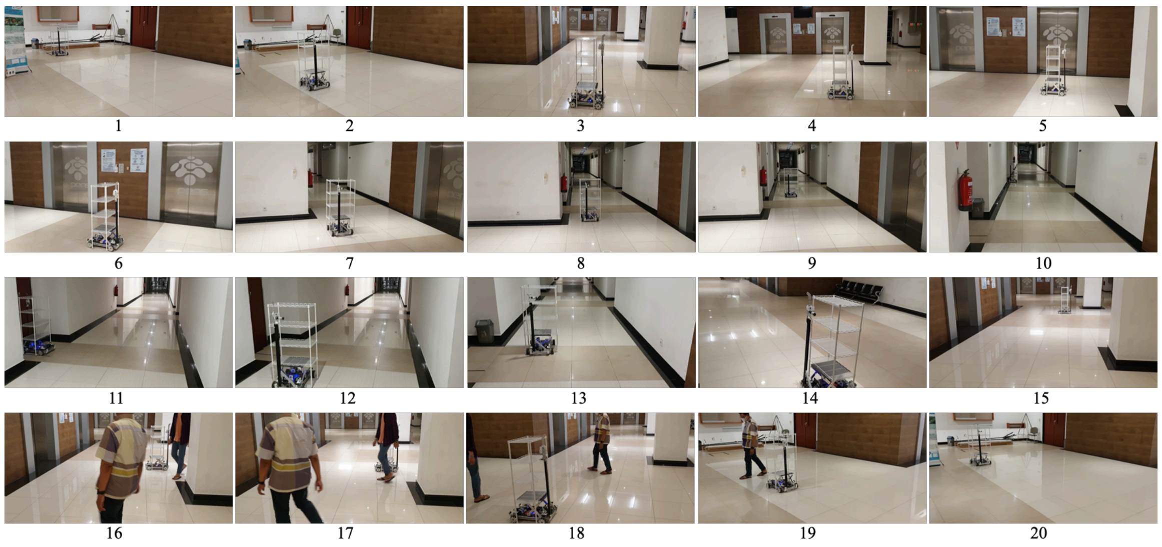


Figure 18. Snapshot of the real mobile robot experimental results from Home to the most far room, go and away, with and without obstacle avoidance.

5. Conclusion

Based on the experimental results, the following conclusions can be drawn from this study:

1. The holonomic motion model applied to the mobile robot allow it to move freely in all directions and function in a narrow area.
2. The position driver and fuzzy controller with the rules can make the mobile robot avoid obstacles when reaching the target.
3. Path planning plays a crucial role in navigation because without path planning, a mobile robot is often trapped in a local area when the environment is unknown. The use of particle swarm optimization (PSO) as a path planning algorithm can overcome these problems. There are 3 types of PSO were compared, and it was found that the Trigonometric Function Adaptive PSO (TFAPSO) is better for finding the most optimal solution compared to Linear Adaptive PSO (LAPSO) and standard PSO based on the safest, shortest path, and lowest time.
4. The proposed navigation system uses a position driver that has been able to make the mobile robot go to and reach the desired target position and uses obstacle avoidance with a fuzzy controller that has been applied which has succeeded in enabling the mobile robot to avoid obstacles around it. Thus, the mobile robot can carry out logistics delivery tasks that minimize contact between dangerous infectious disease patients and health workers.

In future work, we will develop a mobile robot that can move faster and more effectively and can work well in unexpected environments. The health workers must be maintained as a first and prime priority so they can provide better services to patients.

Acknowledgments: This work was partially supported by JST [Moonshot Research and Development][Grant Number JPMJMS2034].

References

1. Ghebreyesus T., "WHO Director-General's opening remarks at the media briefing on COVID-19-2020," 2020.
2. COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University, "Daily new confirmed COVID-19 deaths," Our World in Data, 23 July 2021. [Online]. Available: https://ourworldindata.org/covid-deaths. [Accessed 23 July 2021].

3. Special Expert Group for Control of the Epidemic of Novel Coronavirus Pneumonia of the Chinese Preventative Medicine Association, "An update on the epidemiological characteristics of novel coronavirus pneumonia (COVID-19)," Chin J Epidemiol, vol. 41, pp. 139-144, 2020.
4. Calderon C. A. A., Mohan R. E. and Zhou C., "Eldery Tele-care in the Singapore Context," in International Conference on Human-Robot Interaction, Lausanne, 2011.
5. Al-Araji A. S., "Development of Kinematic Path-Tracking Controller Design for Real Mobile Robot via Back-Stepping Slice Genetic Robust Algorithm Technique", Arabian Journal for Science and Engineering, Vol. 39, No. 12, pp. 8825-8835, 2014. DOI 10.1007/s13369-014-1461-4.
6. Al-Araji, A.S., Ahmed, A.K. and Hamzah, M.K., "Development of a Path Planning Algorithms and Controller Design for Mobile Robot", The 2018 3rd Scientific Conference of Electrical Engineering, SCEE 2018, pp. 72–77, 2018.
7. Sulistijono I. A., Rois M., Yuniawan A., Binugroho E. H., Teleoperated Food and Medicine Courier Mobile Robot for Infected Diseases Patient, IEEE Region 10 Annual International Conference, Proceedings/TENCON 2021, pp. 620-625, 2021.
8. Tzafestas S. G., "Mobile Robot Control and Navigation: A Global Overview," Journal of Intelligent & Robotic Systems, vol. 91, no. I, pp. 35-38, 2018.
9. Li W. and Xiong R., "Dynamical Obstacle Avoidance of Task Constrained Mobile Manipulation Using Model Predictive Control," IEEE Access, vol. 7, pp. 88301-88311, 2019.
10. Bonyadi M. R. and Michalewicz Z., "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review," Evolutionary Computation, vol. 25, pp. 1-54, 2017.
11. Brandstatter B. and Baumgartner U., "Particle swarm optimization - mass-spring system analogon," IEEE Transactions on Magnetics, vol. 38, no. 2, pp. 997-100, 2002.
12. Tzafestas S. G., Introduction to Mobile Robot Control, Athens: Elsevier, 2014.
13. Yunwang L., Sumei D., Yuwei Z., Feng T. and Xucong Y., "Modeling and Kinematics Simulation of a Mecanum Wheel Platform in RecurDyn," Journal of Robotics, vol. 2018, pp. 1-7, 2018.
14. Zadeh L. A., "Fuzzy sets," Information and Control, vol. 8, no. 3, pp. 338-353, 1965.
15. Mamdani E. H., "Applications of Fuzzy Algorithms for Control of a Simple Dynamic Plant," Proc. of IEEE, vol. 121, no. 12, pp. 1585-1588, 1974.
16. King P. J. and Mamdani E. H., "The Application of Fuzzy Control to Industrial Processes," Automatica, vol. 13, pp. 235-242, 1977.
17. Mamdani E. H. and Assilian S., "An Experiment in Liguistic Synthesis with a Fuzzy Logic Controller," Int. J Man-Machines Studies, vol. 7, pp. 1-13, 1975.
18. Sugeno M. and Murakami K., "An Experimental Study on Fuzzy Parking Control Using a Model Car," North-Holland, 1985.
19. Yagishita O., Itoh O. and Sugeno M., "Application of Fuzzy Reasoning to Water Purification Process," North-Holland, 1985.
20. Kennedy J. and Eberhart R., "Particle Swarm Optimization," Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, 1995.
21. Shi Y. and Eberhart R., "A Modified Particle Swarm Optimizer," Proceedings of IEEE International Conference on Evolutionary Computation, pp. 69-73, 1998.
22. Kennedy J., "The Particle Swarm: Social Adaptation of Knowledge," Proceedings of IEEE International Conference on Evolutionary Computation, pp. 303-308, 1997.
23. Kennedy J. and Eberhart R., Swarm Intelligence, San Francisco: Morgan Kaufmann, 2001.
24. Nurlaili R., Sulistijono I. A. and Risnumawan A., "Mobile Robot Position Control Using Computer Vision," in International Electronics Symposium (IES), Surabaya, 2019.
25. Sulistijono I. A., Kuswadi S., Setiaji O., Salfikar I. and Kubota N., "A Study on Fuzzy Control of Humanoid Soccer Robot EFuRIO for Vision Control System and Walking Movement," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 16, no. 3, pp. 444-452, 2012.
26. Fukuda T. and Kubota N., "An Intelligent Robotic System Based on a Fuzzy Approach," Proceedings of the IEEE, vol. 87, no. 9, pp. 1448-1470, 1999.
27. Kubota N., Masuta H., Kojima F. and Fukuda T., "Perceptual System and Action System of A Mobile Robot with Structured Intelligence", The 2002 IEEE World Congress on Computational Intelligence, Honolulu, Hawaii, USA, 2002.

28. Bonyadi M. R. and Michalewicz Z., "A hybrid particle swarm with velocity mutation for constraint optimization problems," Genetic and Evolutionary Computation Conference (GECCO), pp. 1-8, 2013.
29. Lian J., Yu W., Xiao K. and Liu W., "Cubic Spline Interpolation-Based Robot Path Planning Using a Chaotic Adaptive Particle Swarm Optimization Algorithm," Mathematical Problems in Engineering, vol. 2020, pp. 1-20, 2020.

Short Biography of Authors

Dr.Eng. Indra Adji Sulistijono Indra Adji Sulistijono received his B.S. degree in Mechanical Engineering from the Institut Teknologi Sepuluh Nopember Surabaya, Indonesia, in 1993. He received his M.Eng. degrees from Department of Human and Artificial Intelligent Systems of the University of Fukui, Fukui, Japan in 2005. He subsequently received his Dr.Eng. degrees from the Department of System Design of Tokyo Metropolitan University, Tokyo, Japan in 2008. Since 1994 to the present he has beens a lecturer and researcher of Mechatronics Engineering Division in Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. His research interests include intelligent mechatronics and robotics, computational intelligence, and robot vision.

Andy Yuniawan Andy Yuniawan received the B.S. degree in the Mechatronics Engineering from Politeknik Elektronika Negeri Surabaya (PENS), Surabaya, Indonesia, in 2020 and the M.S. degree in electrical engineering from Politeknik Elektronika Negeri Surabaya (PENS), Surabaya, Indonesia, in 2022. Since 2022 to the present, he works as a researcher at PT Halia Teknologi Nusantara (HTN), Jakarta, Indonesia.

Aliridho Barakbah, PhD. Aliridho Barakbah received his Bachelor degree from Department of Informatics, Sepuluh Nopember Institute of Technology, Indonesia in 1997. He received the PhD degree from Graduate School of Media and Governance, Keio University, Japan, in 2011. Since 2001, he works with Politeknik Elektronika Negeri Surabaya (PENS, known as Electronic Engineering Polytechnic Institute of Surabaya or EEPIS), Indonesia, at the Department of Information and Computer Engineering. Currently, he is a director of PENS. He is also a head of education affairs of Indonesian National Polytechnics Director Forum. His research interests are Clustering, Intelligent Computing, Semantic Image Retrieval, Socio-Cultural Computing, Reinforcement Learning and Knowledge Engineering. He is a head of Knowledge Engineering Research Group in PENS. For scientific activities, he involves in some joint-works and collaborative researches with both National and International institution partners. He also is often invited annually as a guest researcher and visiting professor in Keio University, Musashino University, Kanagawa Institute of Technology, and some universities in Japan, Thailand, and Australia.

Dr.Eng. Zainal Arief Zainal Arief received his B.S. degree in the Electrical Engineering from Institut Teknologi Sepuluh Nopember Surabaya, Indonesia, in 2002 and his M.S. degree in the Electrical Engineering from Institut Teknologi Sepuluh Nopember Surabaya, Indonesia, in 2006. He received the Ph.D. degree in electrical engineering from Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2012. Since 1993 to the present he has beens a lecturer of Electronics Engineering Division in Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. Formerly, he was a director of PENS from 2013 to 2021. Currently, he is a director of Politeknik Teknologi Nuklir Indonesia.

Prof. Naoyuki Kubota Naoyuki Kubota graduated from Osaka Kyoiku University in 1992. He received his M.Eng. degree from Hokkaido University, Hokkaido, Japan, in 1994, and his D.E. degree from Nagoya University, Nagoya, Japan, in 1997. He was an Assistant Professor and Lecturer at the Department of Mechanical Engineering, Osaka Institute of Technology, Japan, from 1997 to 2000. In 2000, he joined the Department of Human and Artificial Intelligence Systems, the School of Engineering, Fukui University, Japan, as an Associate Professor. He joined the Department of Mechanical Engineering, the Graduate School of Engineering, Tokyo Metropolitan University, Japan, as an Associate Professor in 2004. He was an Associate Professor from 2005 to 2012, and a Professor from 2012 at the Graduate School of Systems Design, Tokyo Metropolitan University, Japan. He was a Visiting Professor at University of Portsmouth, UK, in 2007 and 2009, and was an Invited Visiting Professor at Seoul National University from 2009 to 2012, and others. He is currently a Professor in the Department of Mechanical Systems Engineering, the Graduate School of Systems Design, and Director of Community-centric System Research Core, Tokyo Metropolitan University, Japan. His current interests are in the fields of topological mapping, coevolutionary computation, spiking neural networks, perception-based robotics, robot partners, and informationally structured space. He has published more than 500 refereed journal and conference papers in the above research fields. He received the Best Paper Award of IEEE IECON 1996, IEEE CIRA 1997, MHS 2011, WAC 2012, HSI 2016, and so on. He was an associate editor of the IEEE Transactions on Fuzzy Systems from 1999 to 2010, the IEEE CIS Intelligent Systems Applications Technical Committee, Robotics Task Force Chair from 2007 to 2014, IEEE Systems, Man, and Cybernetics Society, Japan Chapter Chair since 2018, Vice Director, Tokyo Biomarker Innovation Research Association, Japan from 2020, and others.