Swarm Intelligence Autotune for Differential Drive Wheeled Mobile Robot

Muhammad Auzan[1], Roghib Muhammad Hujja[2], M Ridho Fuadin[3], Danang Lelono[4]
[1,2,3,4]Department of Computer Science and Electronics, Gadjah Mada University, Indonesia

Article Info

Article history:

Received Nov 22, 2021
Revised Aug 8, 2022
Accepted Aug 15, 2022

Keyword:

DDWMR
Autotune
Bird Swarm
PSO
Tracking

ABSTRACT

Differential Drive Wheeled Mobile Robot (DDWMR) is a nonholonomic robot with constrained movement. Such constraint makes robot position control more difficult. A closed-loop control system such as PID can control robot position. However, DDWMR is a Multiple-Input-Multiple-Output system. There will be many feedback gains to be tuned, and the wrong value will make the system unstable. Therefore this research proposes an offline autotune method to choose optimal feedback gain that minimizes a fitness function. The fitness function uses Integral Absolute Error (IAE) and Integral Time Absolute Error (ITAE). These works propose to autotune feedback gain for DDWMR Jetbot, which implements a PI control system with six feedback gains. The methods used to tune the feedback gain are Particle Swarm Optimization (PSO) and Bird Swarm Algorithm (BSA). There are four different scenarios to do the autotune. The autotune result performance shows that those two methods can find an optimal gain to make the robot follow four different continuous trajectories without much trajectory deformation. PSO and BSA can do an autotune PI gain with six variables to minimize the Integral Absolute Error (IAE) and Integral Time Absolute Error (ITAE)

Corresponding Author:

Muhammad Auzan,
Department of Computer Science and Electronics,
Gadjah Mada University,
Bulaksumur, Yogyakarta 55281, Indonesia.
Email: muhammadauzan@ugm.ac.id

1. INTRODUCTION

Differential drive wheeled robot (DDWMR) is one of the most common configurations of terrestrial robots. A DDWMR consists of two independently actuated wheels, each driven by a DC motor. It is also commonly equipped with a wheel that is not actuated but moves freely and supports the chassis. If the chosen output is the robot's position, its dynamics may vary depending on the chassis centre [1]–[5].

A robot needs to control its position using many kinds of an algorithm to ensure the robot position is correct. There are many control algorithms for robots to handle position in coordination in wheeled mobile robots [1], [6]–[9]. PID is one method that presents a cost-effective way for robots' DC motor drive [10].

The PID control model consists of proportional (Kp), integral (Ki), and derivative (Kd) feedback gains. PID control method is a linear system method. Because DDWMR robots have many states to control, the feedback gains became too many that it would be challenging to select them [10]–[12].

Some research uses autotune methods to solve the feedback gain's selection difficulties [13]–[18]. Autotune methods in a control system automatically choose a feedback gain to minimize a given fitness function. One method that can build an autotune is a bio-inspired algorithm such as Particle Swarm Optimization (PSO), Cuckoo Search and Bird Swarm Algorithm (BSA). Although many bio-inspired algorithms exist, BSA performs better than the simple PSO and Differential Evolution (DE) [19].

Designing an autotune system using a bio-inspired algorithm needs a fitness function. In a control system, some research uses transient response and tracking response. A robot that continuously moves to track a trajectory can use Integral Absolute Error (IAE) and Integral Time Absolute Error (ITAE) as the performance metric.

The control of DDDWMR starts with robot mechanic configuration either it's a 2-wheel type in [2], [13], [16] the 3-wheel type in [1], [6], [8], [10], [14]–[16], [20], or the 4-wheel type in [1], [4], [7], [21]. From mechanic configuration, we get a model and configure the kinematic and dynamic of the robot. Finally, the last is to design the control system such as PID. PID stands for: proportional, integral, and derivative, denoted P, I, D. It is the most commonly used feedback control technique. It minimizes the error by measuring the difference between the output value and the required set-point and adjusting the control inputs.

Control systems combine with optimization methods to improve quality and reduce development costs to achieve high-quality output [20]. In [12], PID gain parameters coefficients are tuned manually. Although the control technique is PID, [12] used PDI instead because the D-control term contributes more to the application's accuracy. The robot in [12] used an infrared sensor as a feedback to avoid obstacles and find cargo manually. Because of the stability of their system, finding the best gain would be a problem.

The cuckoo optimization method is used to design the PID controller, which updates the translation & angular velocities to track the path [9]. The objective function is evaluated by calculating individual errors in three (x, y, and theta) directions. The proposed method is validated by analyzing the simulations for different paths.

Cuckoo Search is an evolutionary optimization algorithm effective for solving non-linear problems with higher accuracy and convergence rate. To calculate PID parameters cuckoo algorithm is an effective method. When used in mobile robot velocity equations, the obtained PID parameters provide a path that is very much close to the desired result.

An advanced fuzzy immune PED-type control algorithm is proposed for robot path tracking. The tracking controller combines an organism's fuzzy control and immune feedback mechanism with conventional PID control. The proposed methods mixed connection of conventional PID and P-type immunity feedback controllers. So some parameters can point to their control performance individually.

Meanwhile, the controller's parameters are optimized by an immune genetic algorithm. The effectiveness of the proposed method is demonstrated by a series of simulation and comparison studies [20]. So a fuzzy compensator is added to improve control performance. Finally, the proposed control system has been evaluated through computer simulation to demonstrate the improved results [16]–[18].

Another method to use autotune is used Artificial neural network (ANN) [13]. BP neural network self-tuning PID controller combines BP neural network and the traditional PID control advantages, tuning PID three coefficients based on neural network in real-time online learning [2]. This controller has better robustness and adaptability than the traditional PID controller.

Each autotunes method has a different performance and produces different results in different cases. Even though not in autotune cases, BSA and PSO proved to optimize complex optimization [19]. In the DDWMR PID control system, many variables must be considered in the optimization. Therefore, based on the previous research, this paper implements and compares offline autotune methods performance in the DDWMR control system using BSA and PSO. The chosen feedback gain is then tested in a different trajectory to calculate and analyze the performance metric.

2. RESEARCH METHOD

2.1. DDWMR Kinematic Models

The model of DDWMR is a mathematic equation representing the relation between robot input and output of robot movement. The robot's input is the right and left wheels. In contrast, the output is robot velocity and position in translation and rotation spaces. Figure 1 and Figure 2 describe all robot kinematic variables.

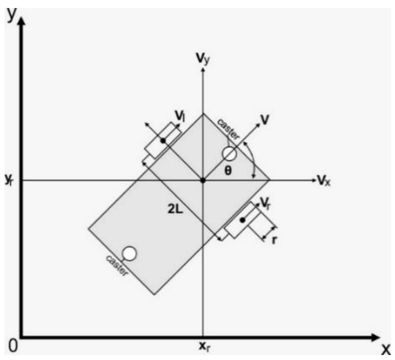


Figure 1. DDWMR Kinematics Variables from Top View

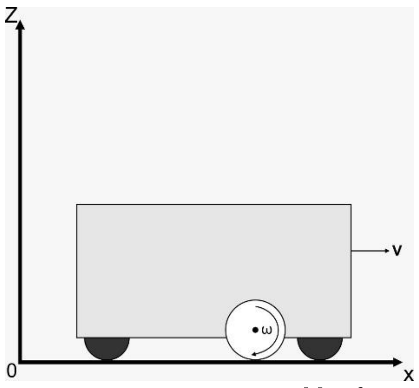


Figure 2. DDWMR Kinematics Variables from Side View

The robot can do a translational movement if there is any movement from the right and left wheels. The value of translation velocity and angular velocity is affected by right and left wheels velocity. The robot will move in straight lines if each wheel moves at the same velocity. If there is any difference between the right and left wheel's velocity, the robot will turn left or right based on the difference, as shown in Equations (1) and (2).

$$v = \frac{(Wr + Wl) * r}{2} \tag{1}$$

$$\omega = \frac{(Wr + Wl) * r}{L} \tag{2}$$

From equation (1), we can project the translation velocity into x-Axis and y-Axis to find the velocity in one axis, as shown in Equations (3) and (4).

$$vx = v \cos \theta \tag{3}$$

$$vy = v \sin \theta \tag{4}$$

Utilizing Equations (3) and (4), in discrete, we can write the position in x-Axis, y-Axis, and angular heading using Equations (5), (6), and (7).

$$x_{n+1} = x_n + v_x dt \tag{5}$$

$$y_{n+1} = y_n + v_y dt \tag{6}$$

$$\theta_{n+1} = \theta_n + \omega\, dt \tag{7}$$

Table 1 explains the symbol and unit for mentioned equations.

Table 1. DDWMR Kinematic Variable and Symbols

Symbol	Parameter	Unit
v	Translation Velocity	m/s
v_x	Velocity in x Axis	m/s
v_y	Velocity in y Axis	m/s
W_r	Right Wheel Angular Velocity	rad/s
W_l	Left Wheel Angular Velocity	rad/s
r	Wheel radius	m
L	Half the distance between wheels	m
θ	Robot angle calculated from the x Axis	rad
ω	Robot Angular velocity	rad/s
x	Robot position in x Axis	m
y	Robot position in y Axis	m

2.2. Particle Swarm Optimization

PSO is one bio-inspired algorithm to search for the optimal solution in a given function. PSO can be used to find an optimal value in a continuous problem such as autotune. Even though PSO is a simple algorithm, it has several hyperparameters that need to be tuned first. The objective of PSO is to minimize a function. The function can consist of one or more variables and an unknown global minimum.

PSO imitates the works of birds searching for food, which we call a particle. Each particle did not know where the global minimum was. However, each particle knows the fitness value from the given fitness function and the position in Equation (8).

$$P_i^t = \left[x_{0,i}^t\ x_{1,i}^t\ x_{2,i}^t\ x_{3,i}^t\ x_{4,i}^t \dots x_{n,i}^t\right] \tag{8}$$

Where,
n = particle-n
i = number of variable
t = time
Each of the particles moves with a velocity and updates its position as shown in Equation (9)

$$V_i^t = \left[v_{0,i}^t\ v_{1,i}^t\ v_{2,i}^t\ v_{3,i}^t\ v_{4,i}^t \dots v_{n,i}^t\right] \tag{9}$$

Three parameters affect each particle's movement: inertia or velocity, cognitive or personal intuition, and social or group knowledge. Each particle tries to find the global minimum using intuition, changing its speed as in Equation (10), and updating its position (11). However, it is also affected by group knowledge.

$$V_i^{t+1} = wV_i^t + c_1r_1\left(P_{best(i)}^t - P_i^t\right) + c_2r_2(P_{bestGlobal(i)}^t - P_i^t \tag{10}$$

$$P_i^{t+1} = P_i^{t+1} + V_i^{t+1} \tag{11}$$

The three parameters control the level of exploration and exploitation. Exploitation is the ability of particles to target the best solutions found so far. Exploration, on the other hand, is the ability of particles to evaluate the entire research space. Hyperparameter in Table 2 affects the exploration and exploitation ability.

Table 2. Particle Swarm Optimization Parameter and Symbols

Symbol	Parameter	Explanation
w	Inertia	the ability of the swarm to change its direction
r1	Cognitive acceleration	Random personal acceleration weight at each iteration
r2	Social acceleration	Random group acceleration weight at each iteration
$c1$	Cognitive Hyperparameter	the ability of the group to be influenced by the best personal solutions
$c2$	Social Hyperparameter	the ability of the group to be influenced by the best global solutions

Based on each PSO step, the overall algorithm works by following the procedure in Figure 3. The first step is an initialization of the parameter in Table 2. At first, particle position and velocity are randomly generated in each iteration. Each iteration will consist of a fitness function calculation and velocity update based on the fitness function calculation. The last is position update using the calculated velocity.

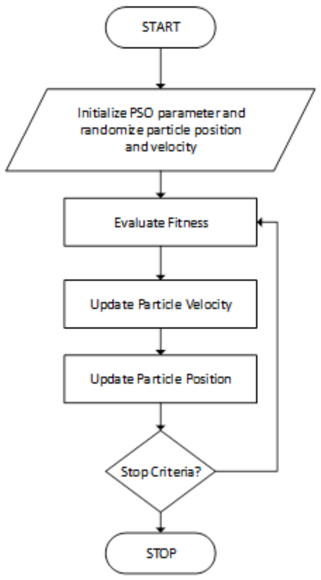


Figure 3. Particle Swarm Optimization Flowchart

2.3. Bird Swarm Algorithm

The BSA used in this research is based on [19]. The birds' swarm algorithm is another bio-inspired algorithm that could solve an optimization problem. The BSA's biological fundamental is birds' social interaction when searching for food and behaviour when encountering a predator.

The first thing birds can do is fly or on the ground eat for food. There are two roles when birds are not flying: foraging in their flock or vigilance. The purpose of foraging in the flock is to get more information. Whilst vigilance is to check for a predator.

While flying, there are two roles in a bird's flock: the producer actively searching for food and the scrounger who only feeds from the found food. Individual birds can switch between producer and scrounger. Figure 4 shows the flow of how the BSA works.

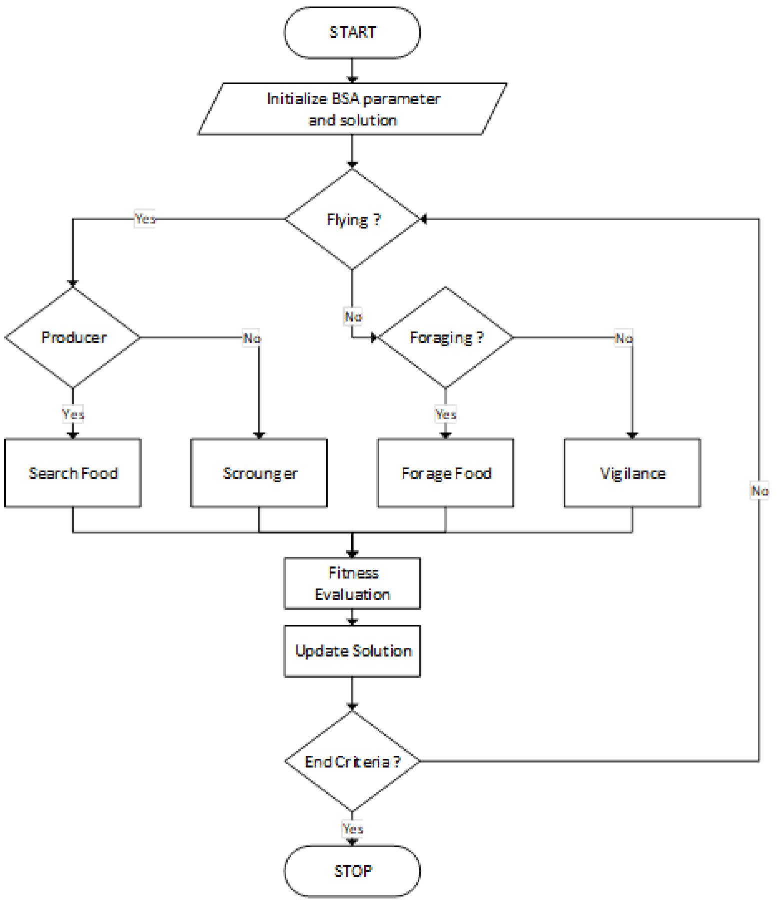


Figure 4. Bird Swarm Algorithm Flowchart

When the birds are foraging, the mathematical model that represents it is Equation (12)

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \left(p_{i.j} - x_{i.j}^{t}\right) \times C \times rand(0,1) + \left(g_j + x_{i.j}^{t}\right) \times S \times rand(0,1) \tag{12}$$

And for vigilance state shown in Equation (13)

$$x_{i,j}^{t+1} = x_{i,j}^{t} + A1\,\left(mean_j - x_{i.j}^{t}\right) \times rand(0,1) + A2\left(p_{k.j} - x_{i.j}^{t}\right) \times rand(-1,1) \tag{13}$$

When the bird is flying, if the bird is a producer, it can search for food with a model in Equation (14)

$$x_{i,j}^{t+1} = x_{i,j}^{t} + rand(0,1) \times x_{i.j}^{t} \tag{14}$$

While if the bird is a scrounger, the mathematic model is in Equation (15)

$$x_{i,j}^{t+1} = x_{i,j}^{t} + \left(x_{k,j}^{t} - x_{i,j}^{t}\right) \times FL \times rand(0,1) \tag{15}$$

Seven parameters must be considered at initialization based on all bird swarm equations. The parameter is explained and mentioned in Table 3.

Table 3. Birdswarm Algorithm Parameter

Symbol	Parameter	Value
C	Constant parameter	1
S	Constant parameter	1
$A1$	Constant parameter	1
$A2$	Constant parameter	1
FL	Followed Coefficient	Random*0.2 + 0.8
P	Probability of Foraging	Random*0.4+0.5
FQ	Frequency of bird flight behaviour	10

2.4. Control System

The Control system in this paper consists of a plant or the jetbot, converter, control, and trajectory generator. The jetbot is the plant we want to control with the right and left wheel's angular velocity input. A converter is a function to convert from robot translation velocity and angular velocity into robot right and left wheels velocity. Control is to generate a signal that will control the robot's position. The trajectory is a function to produce a continuous trajectory. Figure 5 shows the control system diagram explained before.

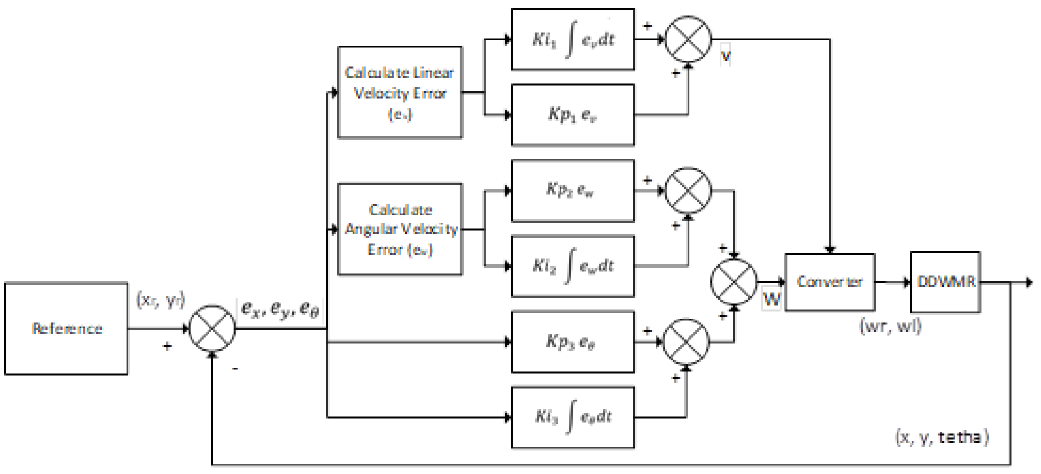


Figure 5. Control System Diagram

There is three error the system calculates that is translation velocity error, angular velocity error, and heading error. Those three inputs of the PI control system consist of proportional gain and integral gain. As such, there is six feedback gain in total in Figure 5.

There are three errors processed in this control system: error position in x-Axis, y-Axis, and heading error calculated with Equations (16), (17) and (18).

$$e_x = x_r - x \tag{16}$$

$$e_y = y_r - y \tag{17}$$

$$e_\theta = atan2(e_y, e_x) \tag{18}$$

From the calculated error, we can get the straight distance of the robot and the reference using Equation (19)

$$d = \sqrt{e_y^2 + e_x^2} \tag{19}$$

Calculation of robot velocity error shown by Equation (20) and the velocity control signal in Equation (21)

$$e_v = d \cos e_\theta \tag{20}$$

$$v = Kp_1 * e_v + Ki_1 \int e_v dt \tag{21}$$

Equation (22) and Equation (23) is the mathematical equations to calculate the angular velocity.

$$e_w = \cos e_\theta \sin e_\theta \tag{22}$$

$$\omega = Kp_2 * e_w + Ki_2 \int e_w dt + Kp_3 * e_\theta + Ki_3 \int e_\theta dt \tag{23}$$

Equations (24) and (25) can change from translation and angular velocity into robot right and left wheels angular velocity.

$$W_r = \frac{(2v + \omega L)}{2R} \tag{24}$$

$$W_l = \frac{(2v - \omega L)}{2R} \tag{25}$$

All the error is an important parameters to be considered. This paper uses IAE and ITAE in Equation (26) and Equation (27) as the performance metric.

$$IAE = \int |e_x| + |e_y| \tag{24}$$

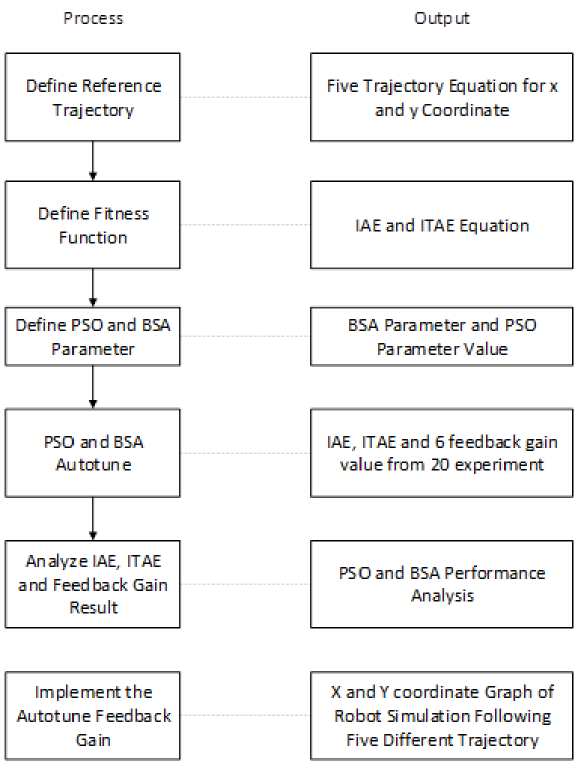
$$ITAE = \int t(|e_x| + |e_y|) \tag{25}$$

2.5. Autotune

Autotune in this paper is an offline autotune. Offline autotune means the gain calculated before being implemented into the robot. As such robot cannot update the gained feedback value when moving.

The first process for autotuning using the bio-inspired algorithm is to define the reference trajectory and fitness function. This paper uses five different reference trajectories: circle, square CW, square CCW, lemniscate of Bernoulli, and point that form a square. The fitness function calculates the IAE and ITAE from the difference between robot position and the reference trajectory.

The next step is to do a feedback gain search using PSO and BSA. The result of PSO and BSA optimization is the six feedback value for each iteration and the IAE and ITAE values. The result of each method is then compared and analyzed in Figure 6 before being implemented into the simulation.


Process
Output
Define Reference Trajectory
Five Trajectory Equation for x and y Coordinate
Define Fitness Function
IAE and ITAE Equation
Define PSO and BSA Parameter
BSA Parameter and PSO Parameter Value
PSO and BSA Autotune
IAE, ITAE and 6 feedback gain value from 20 experiment
Analyze IAE, ITAE and Feedback Gain Result
PSO and BSA Performance Analysis
Implement the Autotune Feedback Gain
X and Y coordinate Graph of Robot Simulation Following Five Different Trajectory
Figure 6. Autotune Process Flow Chart

3. RESULTS AND DISCUSSION

This work does an offline autotune process using two optimization methods to tune six feedback gains in the DDWMR position control problem. Autotune is a real-world problem; as such, it is continuous, so this work use method that can handle continuous spaces. The chosen optimization methods are BSA and PSO.

This section will explain three research results. The first one is the result and explanation of several autotune schemes. Second is the test of every feedback gained from the first result in a different kind of trajectory. The last is the robot position graph.

This work has six parameters tuned with four schemes that combine two methods and two fitness functions, as shown in Table 4. It gives a different result of feedback gain, especially the integral gain. The quality of the autotune can be seen from the IAE and ITAE values. The smaller value of IAE and ITAE means better performance of feedback gain.

Error correction mainly uses the proportional feedback gain. In this work, the maximum proportional value is 10. Proportional gain affects the robot input value in equation error. It multiplies it by the proportional gain, which means a more significant error correction. Although more significant proportional gain gives better error correction, it also means more work for the right and left motor and more significant acceleration.

There is no saturation in the motor velocity model; the motor velocity is in continuous value. Some research provides a dynamic model of the actuator. However, Jetbot by Waveshare has no exact value of motor DC parameter. The proportional gain value will approach the maximum search value even if the search range increases because the motor velocity is not saturated.

Unlike the proportional value, the integral value plays a steady-state correction in the tracking problem. The integral will calculate robot input value based on the accumulated error, the more significant the impact of integral control. In the tracking problem, integral control takes a vital role in correcting the robot position if the robot position is left behind the moving coordinate.

The chosen integral value for Ki_1 is essential. Given proportional gain near the value of 10, the robot is still left behind by the reference trajectory. As such, tuning an integral gain Ki_1 is an excellent approach for making a reasonable DDWMR tracking control and minimizing the fitness function.

Ki_2 and Ki_3 is an integral gain that affects the angular velocity of the robot with a different error input. The angular velocity integral feedback gain is more significant than the translation velocity feedback gain. It means there is more possibility of angular velocity left behind than translation velocity.

The importance between translation velocity and angular velocity is difficult to determine. From the kinematic models, the angular velocity affects the robot's right and left wheel's distance. That information only gives us more wheel distance; the robot needs more work to turn. However, it does not give us more information to determine the feedback gain. Here are the needs for autotune for the user that does not have much knowledge of the plant.

Table 4. PI Feedback Gain Autotune Result

Method	Fitness Function	Best IAE	Kp_1	Kp_2	Kp_3	Ki_1	Ki_2	Ki_3
PSO	IAE	41.19	10	9.855239	9.99951	3.550765	5.616164	9.839051
BSA	IAE	43.686203	10	10	10	2.790655	0.25397	4.378453
PSO	ITAE	137.6	9.947472	9.562999	9.993383	3.229908	8.456699	9.434133
BSA	ITAE	129.0633	9.992228	7.635334	9.956129	3.208382	6.839019	9.974985

Table 4 shows the research using IAE and ITAE as the fitness function. IAE is the commonly used metric to determine the quality of tracking response. ITAE was used to emphasize the most updated error than the previous error.

Table 5 shows the tracking performance of every feedback gain in Table 4 for five different trajectories. There is only a slight difference between the result of IAE and the ITAE as the fitness function in autotune. It means the importance of time is insignificant to calculate as the fitness function.

The continuous trajectory gives a good and stable result. The only important point is that the more and more significant the turn will increase the IAE. It is a gap that is interesting to improve further.

Unlike the continuous trajectory result, the point trajectory shows a significant increase in IAE. Increasing IAE in point trajectory affects the motor speed increase until the robot reaches the point. However, it is more challenging to do a break because the velocity is already high. The robot must turn in a more significant curve to compensate for the position error. This behaviour always happens on every square edge.

The chosen feedback gain from the proposed offline autotune scheme is not appropriate to implement chosen feedback gain for tracking points with significant position differences. Integrator is not a good option for tracking a far separated position because it will significantly increase the control signal.

Table 5. Feedback Gain Result in Different Trajectory

Method (Fitness Function)	Trajectory	IAE	ITAE
PSO (IAE)	Bernoulli	24.61	111.58
	Circle	29.76	127.87
	Square CCW	37.04	133.93
	Square CW	35.19	132.84
	Square Point	201.17	1079.59
BSA (IAE)	Bernoulli	24.49	111.01
	Circle	29.64	127.25
	Square CCW	36.90	133.18
	Square CW	35.04	132.09
	Square Point	201.24	1079.99
PSO (ITAE)	Bernoulli	24.61	111.58
	Circle	29.76	127.87
	Square CCW	37.04	133.93
	Square CW	35.19	132.84
	Square Point	200.76	1077.28
BSA (ITAE)	Bernoulli	24.63	111.48
	Circle	29.77	127.47
	Square CCW	37.52	134.88
	Square CW	35.40	133.64
	Square Point	198.41	1060.53

Two different optimization methods were used in this research. The performance between the two methods can be seen in Figure 7 for the IAE fitness function and ITAE fitness function. Although PSO's initial result is more significant than BSA, the PSO algorithm gives smaller IAE and ITAE values than the BSA. PSO also takes fewer iterations to convergence than BSA.

The number of iterations is not a parameter that determines good performance in the offline autotunes process. As such, we can set it aside. However, the importance of offline autotune is the IAE and ITAE value. We can see that the PSO algorithm gives a better result from two different fitness functions.

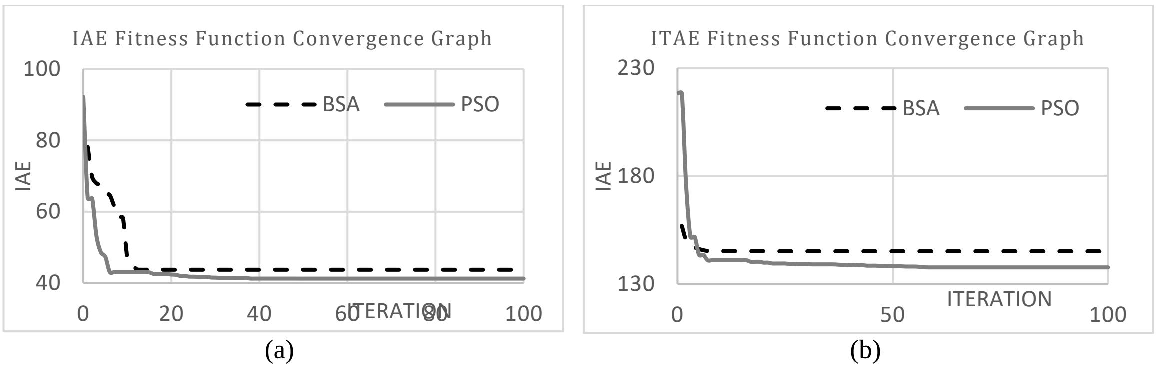


Figure 7. (a) IAE Fitness Function Convergence Graph (b) ITAE Fitness Function Convergence

Figure 8 shows the response of the tracking trajectory for curved reference. There is no sharp turn in a curved trajectory. Although the starting point and the start reference are separate for 0.2 meters in x-Axis and y-Axis, the control system can create infinite and circle symbols after several steps.

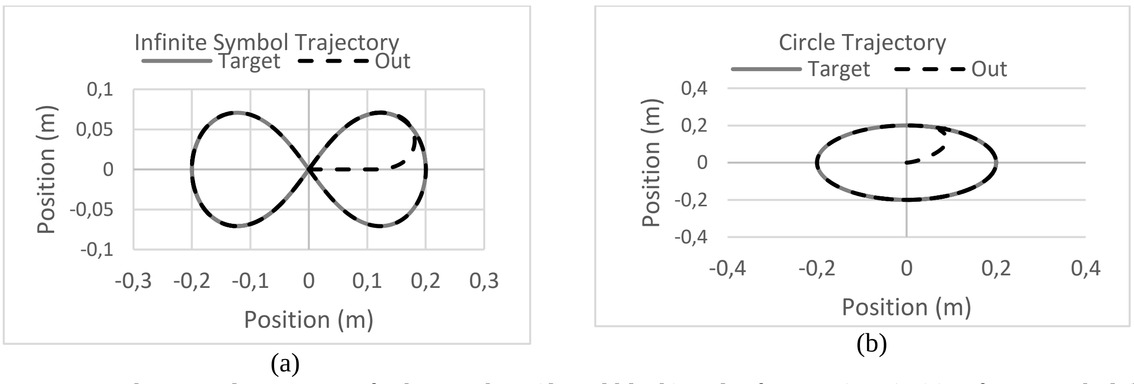
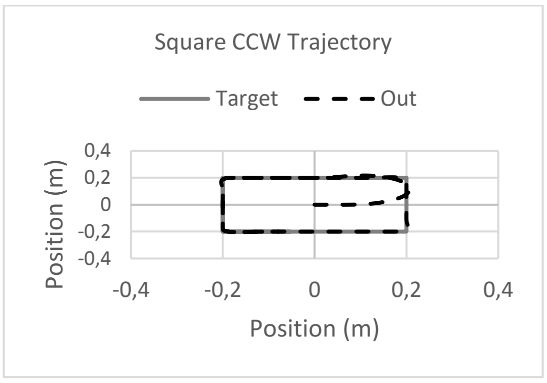
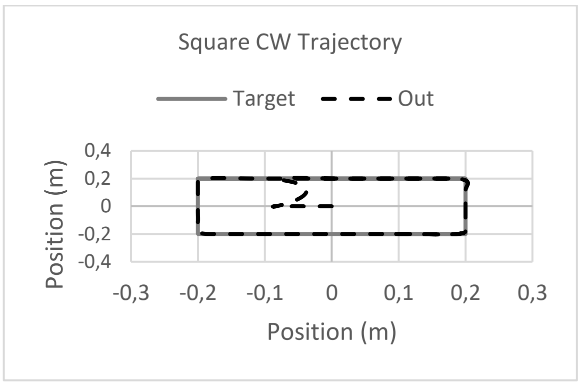


Figure 8. The curved trajectory of robot tracking(dotted black) and reference (grey). (a) Infinite symbol (b) Circle

As for the square trajectory in Figure 9, the reference result shows a slight deformation at first and at the corner. Response for the CW square trajectory is slightly different from the CCW and previous curved trajectory. The first response of the robot is to move straight backwards following the reference in the back. That behaviour happens because the heading error is small compared to the position error. As such, the robot tried to correct the position first. After the reference move in front, the robot starts to align the y axis and turn to the reference point. The response shows that the autotune and control method can handle continuous automatically.



(a)



(b)

Figure 9. The square trajectory of robot tracking(dotted black) and reference (grey). (a) Square CCW (b) Square CW

Significant deformation can be seen in the square point trajectory in Figure 10. Although the robot reaches the edge of the square with a small error, the robot trajectory shapes not form a square shape. When the robot reaches the square edge, the wheel's speed is too big to do a sudden break. The high speed happens because the integral feedback continuously increases the velocity until it reaches the destination. The accumulated gain from integral feedback still increases every time, making the response chaotic, the velocity increasing, and the error bigger every second.

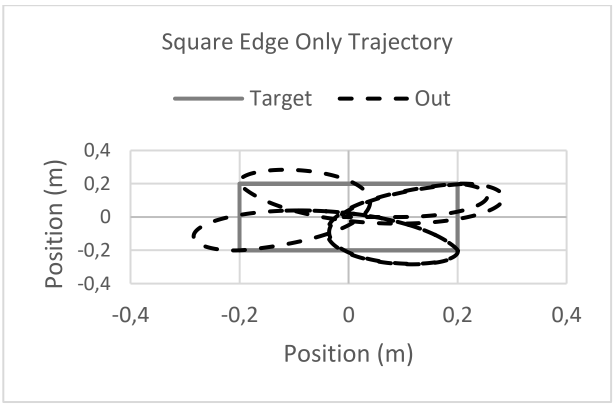


Figure 10. The square edge point only trajectory of robot tracking(dotted black) and reference (grey).

Proposed methods can autotune for six parameters consisting of three proportional gains and three integral gains. The chosen gain value can make the robot track a continuous trajectory with a slight difference in each step. However, the proposed autotune scheme is unsuitable for tracking a point with a significant distance because of the big integral gain.

4. CONCLUSION

In this work, the autotune using BSA and PSO is implemented in the DDWMR PI control system and can make a robot follow four continuous trajectories without much trajectory deformation. The autotune found two proportional and integral gains for each *x* position feedback, *y* position feedback and theta feedback with IAE and ITAE as the minimized function. Although, in previous research, BSA has a better performance than PSO, the IAE and ITAE value of BSA and PSO autotune result in this work is not too different. BSA also has more parameters than PSO.

ACKNOWLEDGMENTS

The authors would like to thank the UGM Research Directorate for providing "financial support" to this research through funding assistance from legal entities of Hibah Peningkatan Kapasitas Dosen Muda Universitas Gadjah Mada 2021 with contract number 2258/UN1.P.III/DIT-LIT/PT/2021.

REFERENCES

[1] F. G. Rojas-Contreras, A. I. Castillo-Lopez, L. Fridman, dan V. J. Gonzalez-Villela, "Trajectory tracking using continuous sliding mode algorithms for differential drive robots," *2017 IEEE 56th Annu. Conf. Decis. Control. CDC 2017*, vol. 2018-Janua, no. Cdc, hal. 6027–6032, 2018, doi: 10.1109/CDC.2017.8264571.

[2] D. Diaz dan R. Kelly, "On modeling and position tracking control of the generalized differential driven wheeled mobile robot," *2016 IEEE Int. Conf. Autom. ICA-ACCA 2016*, no. 1, hal. 1–6, 2016, doi: 10.1109/ICA-ACCA.2016.7778500.

[3] A. Stefek, T. Van Pham, V. Krivanek, dan K. L. Pham, "Energy Comparison of Controllers Used for a Differential Drive Wheeled Mobile Robot," *IEEE Access*, vol. 8, hal. 170915–170927, 2020, doi: 10.1109/ACCESS.2020.3023345.

[4] X. Li, S. Jia, J. Fan, L. Gao, dan B. Guo, "Autonomous mobile robot guidance based on ground line mark," *Proc. SICE Annu. Conf.*, hal. 1091–1095, 2012.

[5] U. Zangina, S. Buyamin, M. S. Z. Abidin, M. S. A. Mahmud, dan H. S. Hasan, "Non-linear PID controller for trajectory tracking of a differential drive mobile robot," *J. Mech. Eng. Res. Dev.*, vol. 43, no. 7, hal. 255–269, 2020.

[6] R. Singh, G. Singh, dan V. Kumar, "Control of closed-loop differential drive mobile robot using forward and reverse Kinematics," *Proc. 3rd Int. Conf. Smart Syst. Inven. Technol. ICSSIT 2020*, no. Icssit, hal. 430–433, 2020, doi: 10.1109/ICSSIT48917.2020.9214176.

[7] Y. Kume, Y. Hirata, K. Kosuge, H. Asama, H. Kaetsu, dan K. Kawabata, "Decentralized control of multiple mobile robots transporting a single object in coordination without using force/torque sensors," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 3, hal. 3004–3009, 2001, doi: 10.1109/ROBOT.2001.933078.

[8] H. Zhou, "DC servo motor PID control in mobile robots with embedded DSP," *Proc. - Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2008*, vol. 1, hal. 332–336, 2008, doi: 10.1109/ICICTA.2008.426.

[9] A. Shukla, H. Goyal, S. Agarwal, S. Nema, dan P. K. Padhy, "Path control of mobile robot using Cuckoo-PID," *IEEE Int. Conf. Comput. Commun. Control. IC4 2015*, hal. 1–5, 2016, doi: 10.1109/IC4.2015.7375635.

[10] R. R. Carmona, H. G. Sung, Y. S. Kim, dan H. A. Vazquez, “Stable PID Control for Mobile Robots,” *2018 15th Int. Conf. Control. Autom. Robot. Vision, ICARCV 2018*, no. 1, hal. 1891–1896, 2018, doi: 10.1109/ICARCV.2018.8581132.

[11] C. T. Lee, B. R. Su, C. H. Chang, T. Y. Hsu, dan W. Der Lee, "Applications of Taguchi method to PID control for path tracking of a wheeled mobile robot," *Proc. 4th IEEE Int. Conf. Appl. Syst. Innov. 2018, ICASI 2018*, hal. 453–456, 2018, doi: 10.1109/ICASI.2018.8394283.

[12] C. L. Chen, Y. De Guo, dan J. H. Zhang, "Motors synchronization for mobile robots using PID control," *Proc. - 2014 Int. Symp. Comput. Consum. Control. IS3C 2014*, hal. 844–847, 2014, doi: 10.1109/IS3C.2014.223.

[13] S. G. Cui, H. L. Pan, dan J. G. Li, "Application of self-tuning of PID control based on BP neural networks in the mobile robot target tracking," *Proc. - 3rd Int. Conf. Instrum. Meas. Comput. Commun. Control. IMCCC 2013*, no. 2, hal. 1574–1577, 2013, doi: 10.1109/IMCCC.2013.350.

[14] X. Su, C. Wang, W. Su, dan Y. Ding, "Control of balancing mobile robot on a ball with fuzzy self-adjusting PID," *Proc. 28th Chinese Control Decis. Conf. CCDC 2016*, hal. 5258–5262, 2016, doi: 10.1109/CCDC.2016.7531938.

[15] I. Ardiyanto, "Task oriented behavior-based state-adaptive PID (Proportional Integral Derivative) control for low-cost mobile robot," *2010 2nd Int. Conf. Comput. Eng. Appl. ICCEA 2010*, vol. 1, hal. 103–107, 2010, doi: 10.1109/ICCEA.2010.27.

[16] K. H. Bae, Y. B. Kim, dan Y. K. Choi, "A fuzzy compensated PID controller for formation control of mobile robots," *Proc. 2014 Int. Conf. Model. Identif. Control. ICMIC 2014*, hal. 123–128, 2015, doi: 10.1109/ICMIC.2014.7020739.

[17] Y. Lei, L. Du, dan Z. Feng, "Motion control of robot based on fuzzy adaptive PID algorithm," *Proc. 2013 3rd Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 2013*, hal. 1310–1313, 2014, doi: 10.1109/ICCSNT.2013.6967342.

[18] T. Y. Wang dan C. Der Chang, "Hybrid Fuzzy PID Controller Design for a Mobile Robot," *Proc. 4th IEEE Int. Conf. Appl. Syst. Innov. 2018, ICASI 2018*, no. 1, hal. 650–653, 2018, doi: 10.1109/ICASI.2018.8394340.

[19] X.-B. Meng, X. Z. Gao, L. Lu, Y. Liu, dan H. Zhang, "A new bio-inspired optimization algorithm: Bird Swarm Algorithm," *J. Exp. Theor. Artif. Intell.*, vol. 28, no. 4, hal. 673–687, Jul 2016, doi: 10.1080/0952813X.2015.1042530.

[20] L. Yu, Z. Cai, Z. Jiang, dan Q. Hu, "An advanced fuzzy immune PID-type tracking controller of a nonholonomic mobile robot," *Proc. IEEE Int. Conf. Autom. Logist. ICAL 2007*, no. 2, hal. 66–71, 2007, doi: 10.1109/ICAL.2007.4338532.

[21] Q. Tian, L. Cheng, K. Wang, dan Y. Liu, "Research on motion control of mobile robot with fuzzy PID arithmetic," *ICEMI 2009 - Proc. 9th Int. Conf. Electron. Meas. Instruments*, hal. 3363–3366, 2009, doi: 10.1109/ICEMI.2009.5274284.