Jurnal Teknologi

Full Paper

IMPROVING WALL-FOLLOWING ROBOT PERFORMANCE USING PID-PSO CONTROLLER

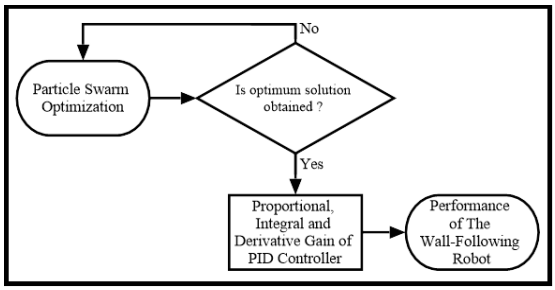
Andi Adriansyah[a], Heru Suwoyo[a,b*], Yingzhong Tian[b], Chenwie Deng[c]

[a]Departement of Electrical Engineering, Universitas Mercu Buana, Jakarta 11650, Indonesia
[b]School of Mechatronic Engineering and Automation of Shanghai University, Shanghai, 200072, P. R. China
[c]School of Information and Communication Engineering, Beijing Institute of Technology, Beijing Shi, P. R. China

Article history
Received
28 August 2018
Received in revised form
22 February 2019
Accepted
5 March 2019
Published online
18 April 2019

*Corresponding author
heru.suwoyo@mercubuana.ac.id

Graphical abstract



Abstract

A wall-following robot is one of the main issues in autonomous mobile robot behavior. However, a wall-following robot needs a robust controller to perform specific tasks accurately. This paper presents an optimization method termed Particle Swarm Optimization (PSO). It was used to automatically produce necessary parameters of the PID controller; henceforth, it was termed as PID-PSO Controller. A new technique of PSO was introduced to enhance the ability of a PID controller to maintain the linear velocity of a mobile robot. The PID-PSO controller was applied to a wheeled wall-following robot. A number of experiments were carried out, and the simulated results were adopted and performed in real applications. Based on several experimental results it can be obtained that the accumulative errors the robot use PID controllers tuned manually, tuned by GA and tuned by PSO are 0.7866, 0.78543 and 0.74619, respectively. Also, the convergence process of PID parameters using the proposed PSO is faster and more optimal than GA. Therefore, it can be said that the proposed system can improve the performance of wall-following robots by decreasing the accumulative error of up to 9%.

Keywords: Mobile robot, wall-following robot, PID controller, particle swarm optimization, PID-PSO

1.0 INTRODUCTION

The principal task of a wall-following robot is to follow the wall by maintaining its movement as it moves almost close to the wall. Therefore, the robot has to sense the distance between the wall and itself during the operation. Moreover, these distances are processed to generate an appropriate movement employing the involvement of a specific controller. In addition, the robot's successful task lies in its capability to maintain the predetermined path in terms of quickness and accuracy as applied in automatically-steered wheelchair, electronic transportation goods, and automated guided vehicle (AVG) [1- 6].

However, the accuracy of all sensors is affected by a limited exactness; even the best sensors still have a degree of imprecision to a certain extent. Consequently, the controller plays an important role in maintaining the movement, and that becomes a crucial solution to the problem. There are several control methods that have been employed to

address the common issues of wall-following robot [7], such as Fuzzy Logic [8, 9], Genetic Algorithm, Neural Network or their hybrid [10]. Nevertheless, most of these controllers have to utilise the orientation sensor featured in a robot (compass, GPS, etc.) [11].

The proposed method presented in this paper utilised a PID controller. It was used to reduce the wobble caused by unexpected velocity. In general, the PID controller's highly-successful performance in producing the linear velocity lies on the accurate setting of three unknown parameters. The produced velocity was transferred to each driven wheel pursuant to the rule of pulse width modulation (PWM) [12 -15]. It is not easy to set the optimum PID parameters manually getting better accurate velocity values. Therefore, a certain method is needed to determine the best PID parameters.

Accordingly, various methods have been proposed and employed to search for these parameters automatically. In the past, PID tuning used Ziegler-Nichols Algorithm [16]. However, the tuning method required a difficult robot modeling to achieve a satisfactory result. On another occasion, the necessary parameters were obtained automatically by involving an algorithm that could develop the original data into a proper optimum parameter concerning the movement. Fuzzy-PID was one of the PID tuning methods which used Fuzzy Logic [12]. Camci *et al*. [17] and Fister [18] also proposed a method to automatically tune PID by using a Neural Network, and reactive nature-inspired algorithms. Unfortunately, these approaches require too much computational work and made the control process is difficult to operate.

This paper proposed a tuned-PID controller to improve the performance of a wall-following robot. Unlike the classical controller, the PID controller was automatically tuned using Particle Swarm Optimization (PSO). Henceforth, it was called the PID-PSO controller. PSO is a population-based stochastic optimization technique inspired by bird flock or the fish school social behaviour [19]. The robot used a self-designed robot equipped with some distance sensors. These sensors were installed and placed to be able to accurately sense the left-side, the right-side, and the front-side of the wall.

The rest of this paper is organized as follows: The related basic knowledge is presented in Section 2. Section 2 also describes the design flow of the proposed method. Section 3 elaborates the experimental results and discussion. Last but not least, Section 4 elaborates the conclusion of the experiment.

2.0 METHODOLOGY

In order to implement the proposed method, a self-designed robot was used. It consisted of 2 driven wheels separately mounted on the back part of the robot's body. Both were connected to the DC motor equipped with an odometer. Just like any other wall-following robots, this robot was equipped with 3 proximity sensors mounted in front, right and left side. The sensor mounted on the front-side was aimed at making the robot capable of detecting the distance of any features in the environment, whereas the sensors mounted on the sides served as the main sensor used to sense the distance between the robot and the wall.

Based on these descriptions, the robot's movement was only affected by a different speed transferred to each odometer. Ideally, the robot moves in a forward direction if there are no differences between the right and left wheel velocities. This difference can be recognised from the command sent to the main controller through the odometer which was formerly produced by a certain controller such as a PID [14-18]. Therefore, the differential steering system and the odometry system was deemed a proper technique in this case. By definition, since the model was a built-up one, the movement characteristics can be recognised as well. At this point, we can conclude that the robot's movement can be effectively monitored after the model was initiated.

The model was inspired by the wheelchair movement principles. It was aimed at easing and providing the completeness to conduct a simulation. This model was well-termed as the kinematic configuration. Moreover, in reference to its movement factors, both the linear and the angular velocities could be generated based only on speed differences. This model is shown in Figure 1. It shows the steering system of a nonholonomic-wheeled mobile robot. To apply this theorem, the only required information was the radius of the wheel denoted by *R*, and the half distance between the separated wheels, indicated by *L*. In this experiment, both were given in reference to the actual information that we had mentioned earlier where R = 3.5 cm and L = 10 cm.

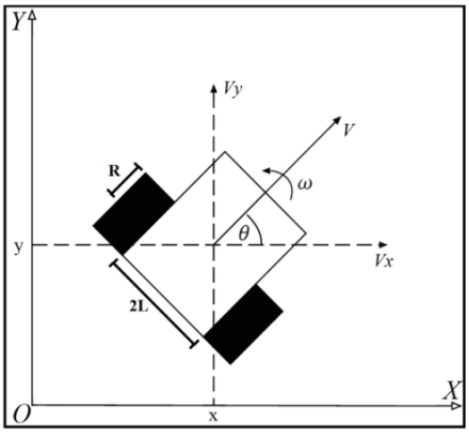


Figure 1 Robot pose in a global coordinate system

Firstly, supposed the robot was operated in a planar environment, the robot pose p comprises two-dimensional planar coordinates (x,y), and orientation of its heading θ. Thus, the robot's current pose can be denoted in Equation 1 [20 - 24]:

$$p_{(t)} = [x_{(t)} \quad y_{(t)} \quad \theta_{(t)}]^T \tag{1}$$

Then by executing the command from the odometer, the linear *v* and angular velocity *ω* can be deemed as the main influencer. Applying them to the system; then, the movement transition denoted by $\dot{p}$ can be expressed as follows:

$$\dot{p} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta_{(t)} & 0 \\ \sin\theta_{(t)} & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

Thus, since Equation 2 affected the initial position; the summation of Equations 1 and 2 will give us the current position.

$$p_{(t+1)} = p_{(t)} + \dot{p} \tag{3}$$

The formulation shown in Equation 3 illustrates that the position in time *t* changed because of the transition effect, where t indicates the time-step index. Next, taking into consideration that both right angular velocity ω_r , and left angular velocity ω_l can be produced from the command transferred to the odometer, and the linear velocity of right and left wheels are expressed by $v_r = \omega_r R$ and $v_l = \omega_l R$, respectively, then the complete equation representing both the linear, and the angular velocities of the robot is summarised as follows:

$$v = \frac{(\omega_r + \omega_l)R}{2} \tag{4}$$

$$\omega = \frac{(\omega_r - \omega_l)R}{2L} \tag{5}$$

$$\omega_r = \frac{(v + \omega L)}{R} \tag{6}$$

$$\omega_l = \frac{(v - \omega L)}{R} \tag{7}$$

At this point, the completeness of the differential steering systems had already been gained To produce effective movement, a popular technique called PID controller will be used in the system. PID controller is an algorithm that was used to determine the precision of an instrumentation system in the presence of the feedback. Generally, PID controller processed the error value as the initial input. In this case, the error value was deemed a different value between the system feedback caused by the actual movement, and the predetermined value known as setpoint. In reference to the setpoint given by the operator, PID controller was able to keep the feedback as close as that of its value. This explains why the PID controller accuracy is represented by zero error. Therefore, it can be noted that the smaller the error generated, the better quality the PID controller will offer. To meet its accuracy, three basic parameters namely the proportional, the integral, and the derivative gains that should be properly adjusted. Characteristically, since three nonzero coefficients were predetermined [12-15] by the operator, the PID controller can be formulated as follows.

$$u(t) = Kp.e(t) + Ki.\int_0^t e(\tau)d\tau + Kd\frac{d}{dt}e(t) \tag{8}$$

Where, *u* refers to the system output, *Kp* is proportional gain, *Ki* is the integral gain, and *Kd* is derivative gain. By definition, the proportional component depends only on the difference between the setpoint, and the process variables. The proportional gain determined the ratio of the output response to the error signal. The error caused the integral component to be increased, whereas the derivative component caused the output to be decreased if increment of the process variable occurred. The derivative response was proportional to the rate of change of process variables. In order to get the exact value for each coefficient required a tuning step. This step can be completed either manually or automatically. Figure 2 shows the block diagram of a PID controller.

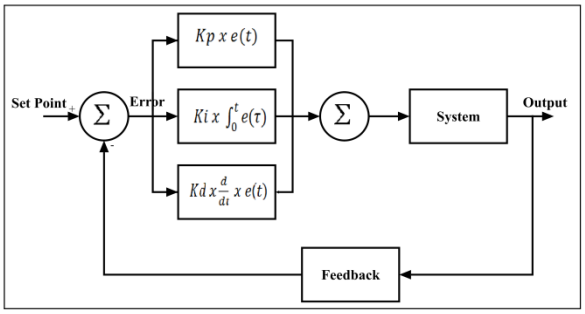


Figure 2 The block diagram of a PID controller

Next, taking into consideration that manual tuning was not recommended, PSO-based tuning method was proposed in this paper. Particle swarm optimization (PSO) is a population-based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995. It was inspired by a bird flock or a fish school social behaviour. It adopted the scenario of a flock of birds randomly searching for food in an unknown environment where only one area was searched, with no knowledge of the exact location of the food, and taking into consideration that only very few of them know the distance to the food. Thus, in order to reach the food, they followed the bird which has the closest distance to the food [19].

A slightly different form of genetic algorithm, each with a potential solution represented by a single bird is called a particle was spread in the searched space. Then using the system performance condition information, a decision is made. The condition information is familiarly called the fitness function. Based on the fitness function, the best value can be analyzed and observed. Therefore, since the

information of each bird is represented by its position and its velocity when it observed the location of the food, a decision was considered in terms of updating these parameters as well.

Similar to evolutionary optimization, a random particle is initially generated. Each particle has certain values which represent the prior unknown parameter of the PID controller. Then, they are confirmed through the system. In reference to the error represented by the fitness function in Equation 12, where the optimum solution was deemed to be the base. It illustrates the first step. The base solution followed two optimum values known as Pbest and Gbest. Pbest represents the best fitness achieved, and is indicated in the iteration index j, whereas, Gbest is the global best representing the particle swarm optimizer. In this case, Pbest is the error carried by the fitness function, and Gbest is the posterior unknown parameter of the PID controller. Both are stored and deemed as the achieved base solution. The particles continuously fly through the problem space by using certain considerations to follow the current optimum particles. Mathematically, these analogies are expressed by Equations 9 and 10.

$$v_{j+1}^{i} = wv_j + c_1 r_1 \left(pbest^i - x_j^i\right) + c_2 r_2 \left(gbest_j - x_j^i\right) \quad (9)$$

$$x_{j+1}^{i} = x_j^i + v_{j+1}^i \quad (10)$$

where v_j^i is the velocity of the ith particle at the jth iteration, x_j^i is the current solution of the ith particle at the jth iteration, r_1, and r_2 are random numbers generated uniformly from 0 up to 1, c_1 and c_2 are self-confidence (cognitive) factor and swarm-confidence (social) factor, respectively, and w is the inertia factor that takes sigmoid decreasing values downward from 1 to 0 according to the predefined number of iterations as recommended in [25, 26] to emphasize the optimization process as shown in Figure 3.

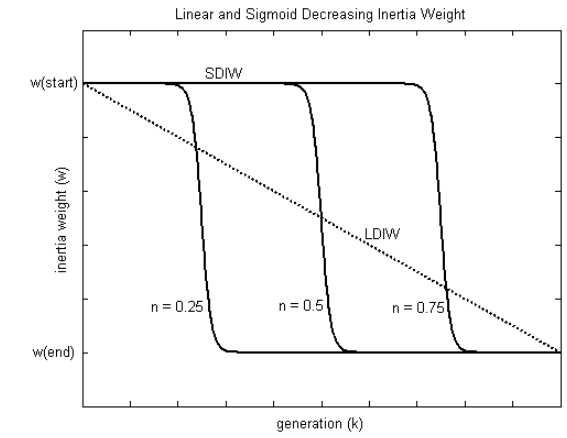


Figure 3 Sigmoid and linearly decreasing inertia weight

Now, by taking into consideration that the robot is steered based on the different speeds of the robot wheels, and the linear v and angular velocity ω are these derived speeds based on Equations 4-7, the operator should update both linear and angular velocities based on the sensed actual distance. However, there is a difference value between the setpoint and the actual distance. The difference is called the feedback value. Accordingly, the approximate error can be formulated as follows.

$$e(t+1) = sp - pv(t) \quad (11)$$

Where sp is the setpoint, and pv is the feedback.

Once Equation 11 was set, the next step was gaining the three basic variables of the PID controller automatically. It was conducted by utilizing PSO. Initially, the random particles were randomly generated with a special boundary for each variable referring to the described analogies. These particles represented the potential solution including the gain of proportional Kp, integral Ki, and derivative Kd. Moreover, its particle also has an initial velocity randomly generated. Then, each particle is checked through the fitness function in Equation 12 which is the representation of the performance of the wall-following robot. By checking all the generated values of the proportional gain, the integral gain, and the derivative gain in each generation, the algorithm stored the best solution of the posterior unknown parameters of PID controller, and the best fitness performance of wall-following robot. There were no exact ways on how to detect the performance of the wall-following robot except by knowing the error total. This error total represented the summation of single error in Equation 11for all iteration index of the wall-following robot process. This error was used as the fitness function shown below.

$$\sum_{it=0}^{max} e(it)^2 \quad (12)$$

Finally, to get the optimum solution, the particles and velocity were updated by referring to Equation 12, and using Equations 9 and 10. The iteration of generation was increased after Pbest and Gbest were obtained. Then, by setting the maximum generation index as the limit, the last generation had to store the expected optimum solution. In this experiment, the limit was set to be equal to 200. The changes of the generated Pbest represented the best quality of the optimum solution in a single generation with respect to the fitness function. We observed that if the change did not exist when the optimization was being processed, the last Pbest and Gbest were deemed to be the expected value. The complete optimization process is shown in Figure 4.

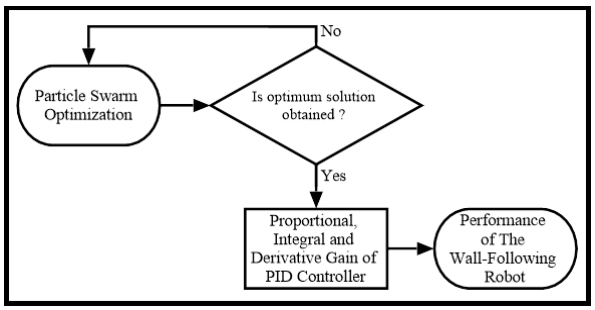


Figure 4 Performance Optimisation Process of PID Controller using PSO

Furthermore, by applying Kp, Ki, and Kd to Equation 8, we could conclude that the PID controller would perform the robust tuned gains. Finally, since the PID-PSO controller produced a value denoted by *u*, the defined velocity was affected. The defined velocity denoted vst was initially defined. It was defined to respond to the zero-error condition. The zero-error condition was achieved when the robot position was on the setpoint value.

$$v(t) = v_{st} - u(t) \tag{13}$$

3.0 RESULTS AND DISCUSSION

The real wall-following robot had been designed in is shown inFigure 5. The proposed method of PID-PSO controller was tested by simulating the robot kinematic model using MATLAB, where a PID controller automatically calculated and presented the velocity in each of the steps -- both in the right and the left wheels. In this experiment, the velocity ranged from 1.5 cm/s to 3.5 cm/s. This arrangement was taken into consideration based on a prior manual testing utilizing odometer and manual movement referring to the command launched to the main controller.

Figure 5 Design wall-following robot

The main controller used in this experiment was Arduino Mega with 16MHz crystal oscillator. It processed the input of the proximity sensor HC-SR04 which had a coverage ranging from 1 cm to 400 cm, and an angle of 30 degrees with a 0.3-cm resolution, and controlling the 12V DC motor through the odometer with an integrated quadrature encoder providing a resolution of 16 counts per revolution of the motor shaft, which corresponded to 2096 counts per revolution of the gearbox's output shaft.

In this experiment, the proposed method produced proper parameters of a PID controller using PSO. Since the robot was assigned to follow the wall, the robot had various linear velocities. In order to validate the effectiveness of the proposed method, a number of experiments were presented. They were categorized based on the controller for the wall-following robot, and elaborated as follows. First, this paper presented the simulated normal movement of the wall-following robot, the wall-following robot controlled by a non-tune PID controller, the wall-following robot controlled by a PID-GA, and the wall-following robot controlled by a PID-PSO. Then, to show their striking differences, the mobile robots with different characters were positioned at the same coordinate position (8.4, 1, $\pi/4$) by means that 10 cm from the wall in the actual distance, then all of them were operated at the same time span of 100 seconds or 100 movement steps. Each performance was observed using Equation 13. Then, taking into consideration that a smaller error made by a simulated performance of the wall-following robot had a better quality, the wall-following robots' best performance was made by those controlled by PID-PSO.

Figure 6 shows the latest position of the wall-following robot located in coordinate 9.1986. It means that the robot moved successfully. Moreover, it reached 8.1986 m away from the initial position. Besides, the accumulative error of the wall-following robot was identified after we implemented Equation 13, and it equaled 0.83625 m.

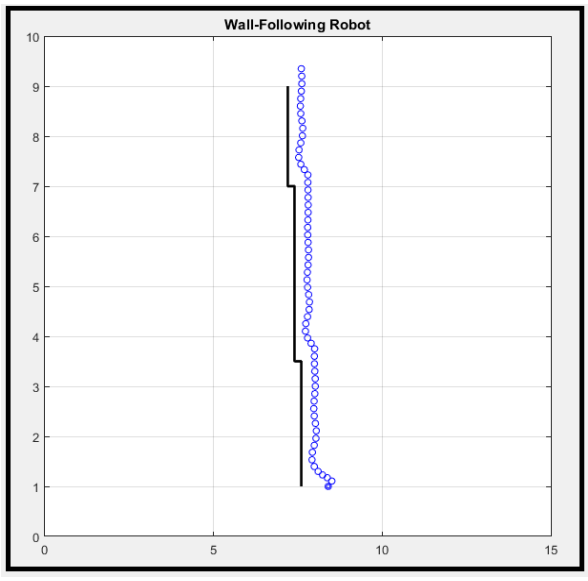


Figure 6 Normal movement of wall-following robot

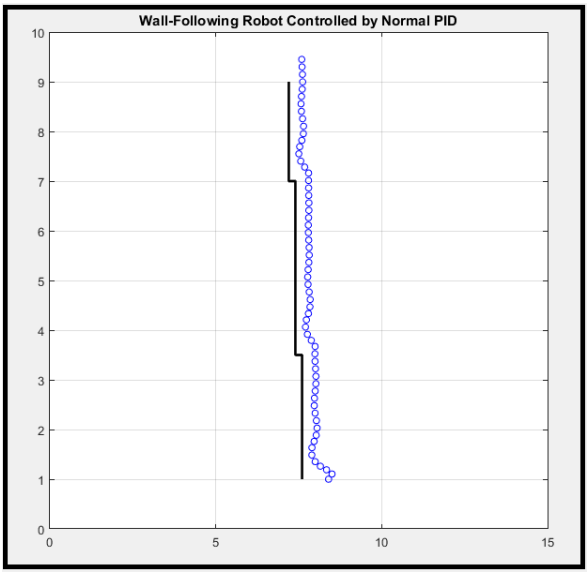


Figure 7 Performance of wall-following robot controlled by a non-tuned PID

Figure 7 shows that by adding a specific controller such as a PID controller, the accumulative error is 0.78666 m. This value means that the accumulative error had successfully decreased. Furthermore, the movement's performance's had also increased. It was proven by the latest position reached by the wall-following robot in the coordinate 9.2987 or 8.2987 m away from the initial point.

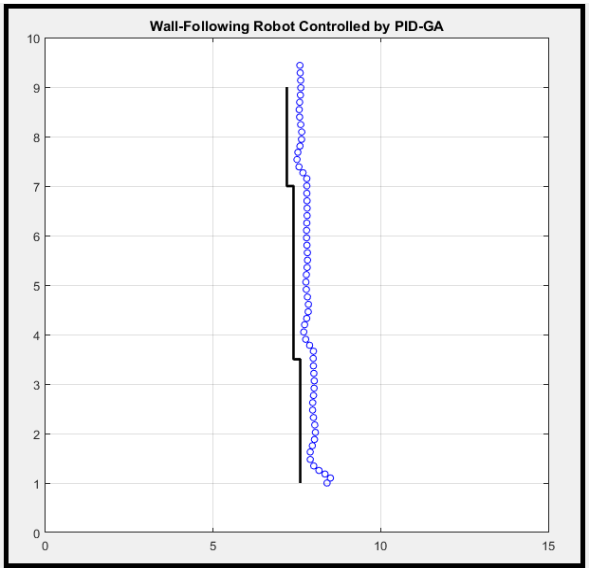


Figure 8 Performance of wall-following robot controlled by PID-GA

Figure 8 shows a wall-following robot controlled by a PID-GA. It showed that it had reached the latest position in coordinates 9.2867, and its accumulative error was 0.78543 m. We observed that adding a PID optimized by a GA improved the robot's performance as indicated by the latest position 8.2867 m away from the initial position.

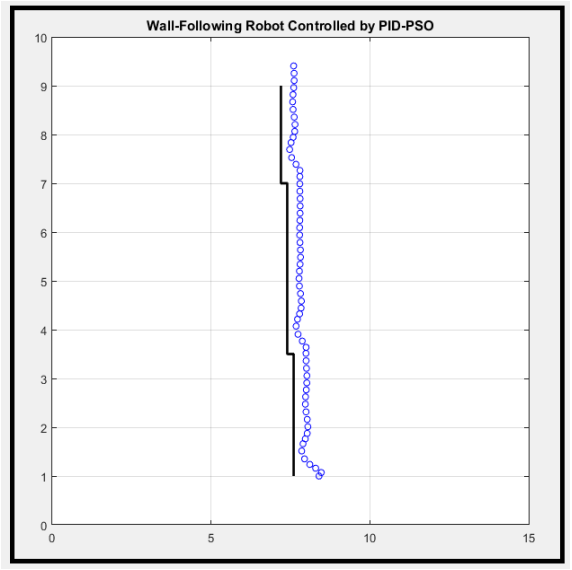


Figure 9 Performance of wall-following robot controlled by a PID-PSO

The latest performance was made by a wall-following robot controlled by a PID-PSO. It is shown in Figure 9. The latest position was in coordinate 9.2528, and accumulative error amounted to 0.74619 m.

Furthermore, a comparison of these performances is shown Table 1. It showed the effectiveness of the proposed method in producing a proper velocity for a wall-following robot in terms of its Dynamic Error. This term represented the average error of the whole robot process from the initial process to the end point. The term dynamic was involved due to the uncertainty error caused by the motion in each step. The dynamic error improvement was 9%.

Table 1 Comparison between dynamic error and final position

Characteristic	Dynamic Error	Improvement (%)	Final Position (s)
Normal Movement	0.83625	0	9.1986
PID Only	0.78666	4.959	9.2987
PID-GA	0.78543	5.082	9.2867
PID-PSO	0.74619	9.006	9.2528

In addition to our understanding of the controlled wall-following robot's performance in its graphical and error representation, the comparison between the GA, and the PSO approaches to optimize PID parameters are shown in Figures 10 and 11. Both figures confirmed the effectiveness of a PSO.

Figure 10 showed the capability of genetic algorithm of gaining the lowest accumulative error generated by the wall-following robot controlled by a PID controller.

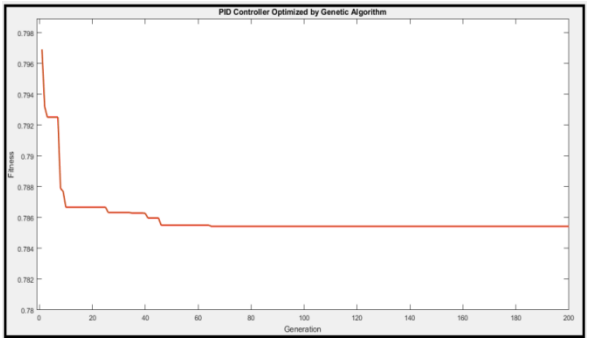


Figure 10 Achieved fitness value by genetic algorithm

Figure 11 shows the PSO ability to gain the optimum solution for a PID controller.

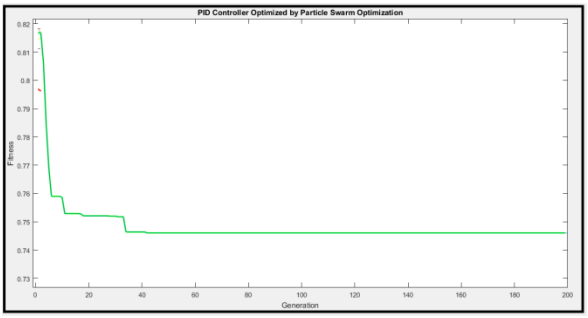


Figure 11 Achieved fitness value by particle swarm optimisation

Unlike genetic algorithm, PSO showed sudden changes in achieved fitness (see Figure 11). It showed that PSO had a better working speed than GA. Moreover, the final result of the optimum solution using PSO is better than GA. The PSO reach the final result in 0.745 at 42nd generation, while GA reach it in 0.785 at 65th generation (as seen in Figures 10 and 11).

As discussed above, the arrangement of the unknown PID parameters was adjusted by using a PSO. This arrangement was attracted to the real implementation that was deemed to be the center of the new arrangement. The ranges of Kp, Ki, and Kd were set to have either 0.25 less or more than the arrangement used in the simulation. Then, the PSO calculated these values in the optimization process of 25 generations. This process was shorter than that performed in the simulation aimed at not affecting the accuracy. It was well-known as a fast PSO [25], [26].

4.0 CONCLUSION

This paper has shown the performance of a wall-following robot controlled by a PID-PSO. The role of the PSO was to produce proper values of three unknown parameters of the PID controller. The PID controller was utilised to produce suitable velocity. Different approaches were simulated and compared. Based on the comparative results, we can conclude that the decrement of the total error showed the effectiveness, and the working speed showed the accuracy of the proposed method.

Acknowledgement

The authors would like to thank the parties who supported this research namely, Beijing Institute of Technology, Shanghai University and the University of Mercu Buana, Jakarta.

References

[1] Lee, Y. T., Chiu, C. S., and Kuo, I. T. 2017. Fuzzy Wall-following Control of a Wheelchair. *Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*, Otsu, Japan. 1-6. DOI: http://dx.doi.org/10.1109/IFSA-SCIS.2017.8023223.

[2] Pasteau, F., Narayanan, V. K., Babel, M., and Chaumette, F. 2016. A Visual Servoing Approach for Autonomous Corridor Following and Doorway Passing in a Wheelchair. *Robotics and Autonomous Systems.* 75(A): 28-40. DOI: http://dx.doi.org/10.1016/j.robot.2014.10.017.

[3] Wardana, A. A., Widyotriatmo, A., and Turnip, A. 2013. Wall Following Control of a Mobile Robot without Orientation Sensor. *3rd International Conference on Instrumentation Control and Automation (ICA)*. Ungasan, Indonesia. 212-215. DOI: http://dx.doi.org/10.1109/ICA.2013.6734074.

[4] Adriansyah, A., Gunardi, Y., Badaruddin, B., and Ihsanto, E. 2015. Goal-seeking Behavior-based Mobile Robot using Particle Swarm Fuzzy Controller. *TELKOMNIKA*. 13(2): 528-538. DOI: http://dx.doi.org/10.12928/telkomnika.v13i2.111.

[5] Lin, C. J. and Lin, H. Y. 2017. Mobile Robot Wall-following Control using a Fuzzy Cerebellar Model Articulation Controller with Group-based Strategy Bacterial Foraging Optimization. *International Journal of Advanced Robotic Systems.* 14(4): 1-13. DOI: https://dx.doi.org/10.1177/1729881417720872.

[6] Priyandoko, G., Wei, C. K., and Achmad, M. S. H. 2018. Human Following on ROS Framework A Mobile Robot. *SINERGI.* 22(2): 77-82. DOI: http://10.22441/sinergi.2018.2.002.

[7] Pandey, A., Pandey, S., and Parhi, D. R. 2017. Mobile Robot Navigation and Obstacle Avoidance Techniques: A Review. *International Robotics & Automation Journal.* 2(3): 00022. DOI: http://dx.doi.org/10.15406/iratj.2017.02.00023.

[8] Emmanuel, I. 2017. Fuzzy Logic-Based Control for Autonomous Vehicle: A Survey. *International Journal of Education and Manajement Engineering (IJEME).* 7(2): 41-49. DOI: http://dx.doi.org/10.5815/ijeme.2017.02.05.

[9] Varma, N. P., Aivek, V., and Pandi, V. R. 2017. Intelligent Wall Following Control of Differential Drive Mobile Robot Along with Target Tracking and Obstacle Avoidance. *International Conference on Intelligent Computing,*

Instrumentation and Control Technologies (ICICICT), Kannur. 85-91.
DOI: http://dx.doi.org/10.ICICICT1.2017.8342539.

[10] Singh, N. H. and Thongam, K. 2017. Fuzzy Logic-genetic Algorithm-neural Network for Mobile Robot Navigation: A Survey. *International Research Journal of Engineering and Technology (IRJET)*. 4(8): 24-45.

[11] Mohamed, A., El-Gindy, M. and Ren, J. 2018. Advanced Control Techniques for Unmanned Ground Vehicle: Literature Survey. *International Journal of Vehicle Performance*. 4(1).
DOI: http://dx.doi.org/10.1504/IJVP.2018.10009708.

[12] Wardoyo, A. S., Hendi, S. Sebayang, D. Hidayat, I., and Adriansyah, A. 2015. An Investigation on the Application of Fuzzy and PID Algorithm in the Two Wheeled Robot with Self Balancing System Using Microcontroller. *2015 IEEE International Conference on Control, Automation and Robotics (ICCAR)*. Singapore. 64-68.
DOI: http://dx.doi.org/10.1109/ICCAR.2015.7166003.

[13] Pandey, A., Pandey, S., and Parhi, D. R. 2017. Mobile Robot Navigation and Obstacle Avoidance Techniques: A Review. *International Robotics & Automation Journal*. 2(3): 1-13.
DOI: http://dx.doi.org/10.15406/iratj.2017.02.00023.

[14] Dagher, K. E. and I-Araji, A. 2014. Design of a Nonlinear PID Neural Trajectory Tracking Controller for Mobile Robot based on Optimization Algorithm. *Engineering & Technology Journal*. 32(4): 973-986.

[15] Varma, N. P., Aivek, V., and Pandi, V. R. 2017. Intelligent Wall Following Control of Differential Drive Mobile Robot Along with Target Tracking and Obstacle Avoidance. *International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. Kannur, 2017. 85-91.
DOI: http://dx.doi.org/10.1109/ICICICT1.2017.8342539.

[16] Prakash, C. B. and Naik, R. S. 2014. Tuning of PID Controller by Ziegler-Nichols Algorithm for Position Control of DC Motor. *International Journal of Innovative Science, Engineering & Technology*. 1(3): 379-382.

[17] Camci, E., Kripalani, D. R., Ma, L., Kacayam, E., and Khanesar, M. A. 2018. An Aerial Robot for Rice Farm Quality Inspection with Tyoe-2 Fuzzy Neural Networks Tuned by Particle Swarm Optimization-Sliding Mode Control Hybrid Algorithm. *Swarm and Evolutionary Computation*. 41: 1-8.
DOI: http://dx.doi.org/10.1016/j.swevo.2017.10.003.

[18] Fister, D., Fister Jr., I., Fister, I., and Safaric, R. 2016. Parameter Tuning of PID Controller with Reactive Nature-inspired Algorithms. *Robotics and Autonomous Systems*. 84: 64-75.
DOI: http://dx.doi.org/10.1016/j.robot.2016.07.005.

[19] Sengupta, S., Basak, S. and Peter II, R. A. 2019. Particle Swarms Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. *Machine Learning and Knowledge Extraction*. 1(1):157-191. DOI: http://dx.doi.org/10.3390/make1010010.

[20] Differential Drive Mobile Robot Platform for use in constrained environments. *International Journal of Innovations in Engineering Research and Technology [IJIERT]*. 2(6): 1-10.

[21] Chou, C. Y. and Juang, C. F. 2018. Navigation of an Autonomous Wheeled Robot in Unknown Environments Based on Evolutionary Fuzzy Control. *Inventions*. 3(3): 2-14.
DOI: http://dx.doi.org/10.3390/inventions301003.

[22] Myint, C. and Win, N. N. 2016. Position and Velocity control for Two-Wheel Differential Drive Mobile Robot. *International Journal of Science, Engineering and Technology Research (IJSETR)*. 5(9): 2849-2855.

[23] Gunardi, Y., Hanafi, D., Supegina, F, and Adriansyah, A. 2018. Mathematics Base for Navigation Mobile Robot Using Reachability Petri net. *Journal of Telecommunication, Electronic and Computer Engineering*. 10(1-9): 65-69.

[24] Adriansyah, A. Sulle, B., Ihsanto, E., and Gunardi, Y. 2017. Optimization of Circular Robot Size using Behavior based Architecture. *Journal of Telecommunication, Electronic and Computer Engineering*. 9(3-7): 67-72.

[25] Rathore, A. and Sharma, H. 2017. Review on Inertia Weight Strategies for Particle Swarm Optimization. In Deep, K. *et al*. (Eds). Proceedings of Sixth International Conference on Soft Computing for Problem Solving. *Advances in Intelligent Systems and Computing*. 547.
DOI: http://dx.doi.org/10.1007/978-981-10-3325-4_9.

[26] Gupta, I. K., Choubey, A., and Choubey, S. 2017. Particle Swarm Optimization with Selective Multiple Inertia Weight. *8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. Delhi, India. 1-6.
DOI: http://dx.doi.org/10/1109/ICCCNT.2017.8204132.