Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 7, No. 3, December 2021, pp. 368-379
ISSN: 2338-3070, DOI: 10.26555/jiteki.v7i3.21017

Path Tracking and Position Control of Nonholonomic Differential Drive Wheeled Mobile Robot

Muhammad Auzan, Roghib Muhammad Hujja, M. Ridho Fuadin, Danang Lelono
Universitas Gadjah Mada, Bulaksumur, Yogyakarta 55281, Indonesia

ARTICLE INFO

Article history:

Received June 30, 2021
Revised August 08, 2021
Accepted October 22, 2021

Keywords:

Control;
Differential Drive;
Kinematic;
Path Tracking;
Robot;
Wheeled Robot

ABSTRACT

Differential drive wheeled mobile robot (DDWMR) is one example of a robot with a constrained movement, Multiple Input Multiple Output (MIMO), and nonlinear system. Designing a low resource position and heading controller using linear MIMO methods such as LQR became a problem because of the linearization of robot dynamics at zero value. One of the solutions is to design a MIMO controller using a Single Input Single Output (SISO) controller. This work design a controller using PID for DDWMR Jetbot and selects the best feedback gain using different scenarios. The designed controller manipulates both motors by using calculated control signal to achieve a complex task such as path tracking with robot position in x-Axis, y-Axis, and heading angle as the feedback. The priority between position and heading angle can be adjusted by changing three feedback gains. The controller was tested, and the best gain was selected using Integral Absolute Error (IAE) metrics in a path tracking task with four different path shapes. The proposed methods can track square, circle, and two types of infinity shape paths, with the less well-formed shape being the four edges square path.

Corresponding Author:

Muhammad Auzan, Universitas Gadjah Mada, Bulaksumur, Yogyakarta and 55281, Indonesia
Email: muhammadauzan@ugm.ac.id

1. INTRODUCTION

Emerging new technology making robots enable to live beside the human. Even if robots can only work in controlled environments such as assembly lines, warehouses, or mapped environments, robot demands and supply still increase nowadays [1]. One of the rising robot types is a wheeled robot that can autonomously drive itself into a specific position [2], [3]. A wheeled robot driven only by two motors to move from one position to another is called a different drive-wheeled mobile robot (DDWMR). DDWMR is a robot with a nonholonomic system; it means the robot cannot move freely in all axes because there is movement constrain applied [4].

DDMWR is also a nonlinear and multiple-input multiple-output (MIMO) system [5]. Designing a controller for a nonlinear MIMO system is a challenging problem. One way to design the controller is by linearizing the robot dynamics, such as LQR methods [6]. Robot motion dynamics must be precisely modeled and linearized with the robot's real value [7]. However, it needs an exact knowledge of robot states, and such information is rarely available in some cases [8]–[11]. Therefore, this works to simulate a control system for nonlinear MIMO DDWMR without using linearization.

Controlling mobile robots can be done using model-based control or non-model based such as AI [12]–[15]. A model-based robot is an excellent approach to control a robot because it is easier to analyze its stability [16], [17]. However, it is hard to formulate the mathematical model representing all physical forces acting on the robot [18], [19]. The more manageable and less computation option is to model a robot using the kinematic models and ignore forces applied to the robot [20].

Another controller method that can be used and robust is PID, but it is a Single Input and Single Output System, and it is a challenging problem to design a MIMO system using PID [21], [22]. PID can also be divided into several control scenarios such as proportional only, PI, PD, and PID based on the system's needs [23].

Some research also designs a PID in cascading architecture to create double proportional and derivative control and combine PID with other control to overcome PID problems [24], [25].

Based on the previous research, this works will simulate a control system for the DDWMR Jetbot model using DDWMR kinematic to reduce the computation and simulate a position control using low resource proportional control. The proposed model also formulates the relation between right and left motor, linear velocity, and angular velocity to simulate the robot behavior accurately. The system will be tested using simulation in four different path shapes to find the best feedback using the Integral Absolute Error (IAE) metric and analyze the robot's behavior.

2. METHOD

This research designed a proportional control system to control Jetbot position and heading. There is some step to follow and the outcome needed to simulate the Jetbot control system, as shown in Fig 1. The first step of this work is Jetbot kinematic modeling, which contains two motor velocities as the input and robot position and heading as the monitored states. After the kinematic is formulated, the control system can be designed by formulating the feedback and input using the Lyapunov stability criterion. The kinematic and controller model was implemented into Simulink to simulate the response and find the best feedback gain using IAE metrics and grid search.

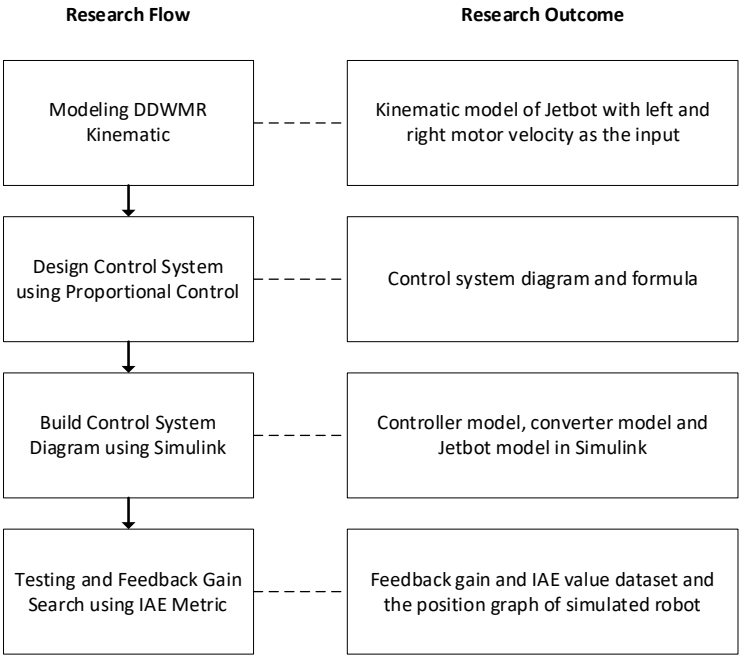

Fig. 1. Research Flow

The Control system in this works uses to control the robot position without considering the robot orientation. It means the robot only tracks a specific position in the x and y axes. The system consists of a trajectory generator that produces different goal positions for each time sampling, a position controller to calculate robot wheel speed, and a robot plant that consists of the robot kinematic model. Converter needed to convert the linear and angular velocity calculated by position control into each robot velocity using robot mechanical specification. Fig. 2 shows the entire system used in this works.

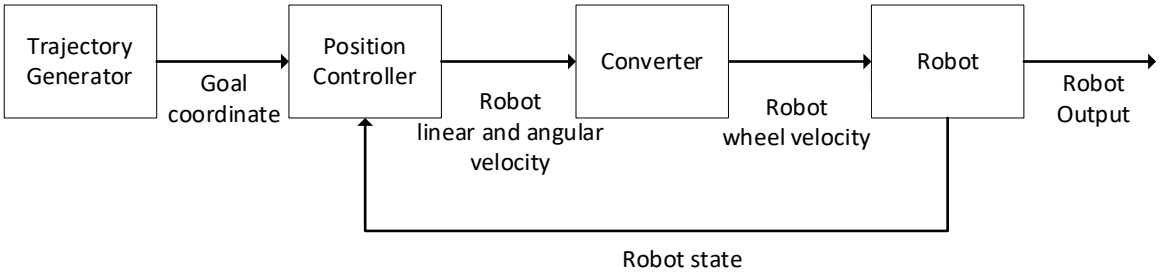

Fig. 2. Differential drive two-wheeled robot control system diagram.

System diagram in Fig. 2 created in Simulink to simulate and observe the robot's entire behavior in Fig. 3. Robot states observed are local acceleration, local velocity, global velocity, angular velocity, and global orientation. The Simulink model consists of four subsystems: trajectory, position controller, converter, and jetbot kinematics. The trajectory is a subsystem built using Matlab code that takes times as the input and positions in the x-axis and y-axis as the output. The position controller is the control subsystem to provide the jetbot model with a proper control signal. Jetbot subsystem takes a left and right motor velocity as the input; it needs a converter to convert the robot translation and angular velocity into robot right and left motor velocity, as shown in Fig 3.

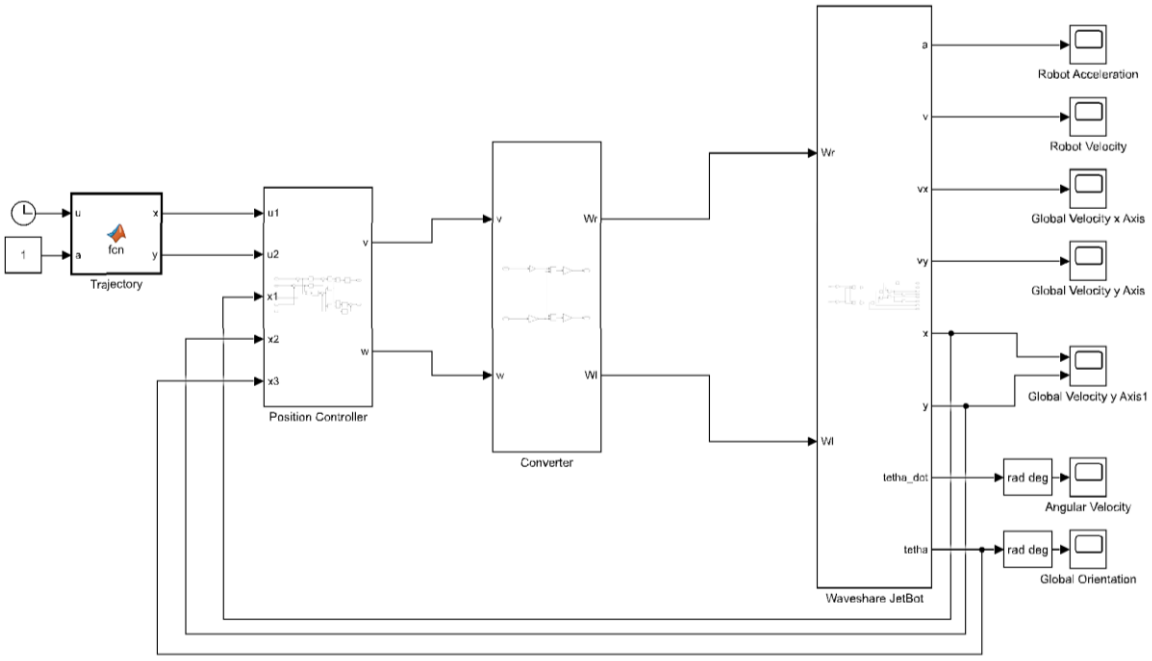

Fig. 3. Differential drive two-wheeled robot system Simulink model.

The Simulink model runs from 0s until 10s with a fixed-step solver. To simulate an actual continuous model but with discreet computation, the simulation used 0.01s as the step size. The ode1 (Euler) solver used in this works to compute the integration and differentiation. The summary of the simulation configuration is shown in Table 1.

Table 1. Simulink simulation solver configuration

Solver Parameter	Value
Start time	0.0 s
Stop time	10.0 s
Solver type	Fixed-step
Solver	ode1 (Euler)
Fixed-step size	0.01 s

2.1. Jetbot Robot Specification

The robot used in this works is based on JetBot Robot by Waveshare powered by Nvidia Jetson Nano. The robot mechanical consists of two active wheels powered by a DC motor on both left and right sides and two passive wheels in front and back for balancing the robot. JetBot has a wheel with a radius (r) of 0.035 meters, and the distance between two wheels (2L) is 0.0575 meters, as shown in Table 2. Wheel radius will affect the translation velocity of the robot, and the distance between two wheels will affect the angular velocity of the robot. The bigger the wheel's radius, the robot's maximum velocity will be immense, and the distance between two wheels affects the influence between two wheels.

Table 2. Differential drive wheeled robot mechanical parameter

Robot Parameter	Symbol	Value
Wheel radius	r	0.035 m
Half distance between wheels	L	0.0575 m

In addition to mechanical parameters, some parameters were observed or controlled, such as both wheels and robot velocity, robot position, and robot angular position, as shown in Table 3. Those parameters were used to build the robot kinematic model and control system. The observed parameter is robot linear velocity, angular velocity, position, and orientation. At the same time, the controlled variable is the robot wheel's angular velocity.

Table 3. Controlled and observed robot parameters.

Robot Parameter	Symbol	Unit
Robot translation velocity	v	m/s
Right wheel linear velocity	v_r	m/s
Left wheel linear velocity	v_l	m/s
Right wheel angular velocity	ω_r	rad/s
Left wheel angular velocity	ω_l	rad/s
Robot velocity in the x-axis	v_x	m/s
Robot velocity in the y-axis	v_y	m/s
Robot angle calculated from the x-axis	θ	degree
Robot angular velocity	$\dot{\theta}$	degree/s
Robot position in x-axis	x	m
Robot position in y-axis	y	m

Robots move forward or backward from the resultant of right and left wheels. If there is any difference between the left and right wheel velocity, the robot will turn left or right. All robot parameters drawn in cartesian coordinate are shown by Fig. 4 and Fig. 5.

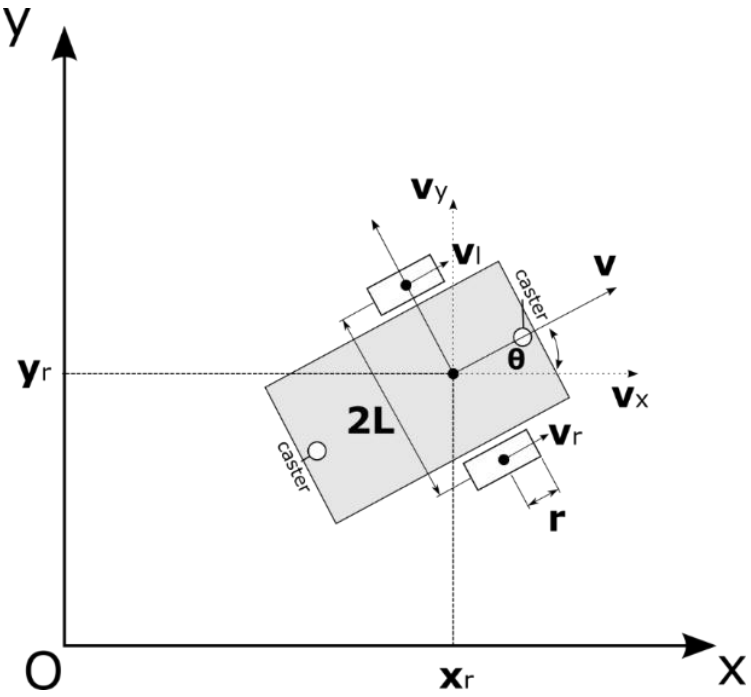

Fig. 4. Robot parameter in the transverse plane

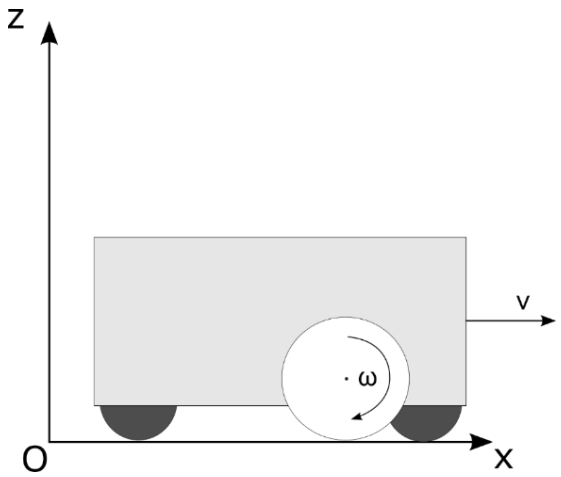

Fig. 5. Robot parameter in the sagittal plane

2.2. Differential Drive Wheeled Mobile Robot Kinematic

The robot moves by utilizing the left and right wheel's linear velocity in two-wheeled robots with a differential drive system. Even if the two motors move on their own, it affects the entire system states. The robot wheel's linear velocity can be calculated from motor angular velocity and wheel radius in (1) and (2).

$$v_r = \omega_r r \tag{1}$$

$$v_l = \omega_l r \tag{2}$$

Robot linear velocity and angular velocity can be calculated by using each left and right wheel velocity. Robot linear velocity is the average of each wheel's linear velocity as in (3). Robot angular velocity can be calculated from the difference between right and left wheel linear velocity as in (4). If the right wheel velocity is bigger than the left, the robot will turn left and vice versa. The distance between two wheels also affects the value of robot angular velocity.

$$v = \frac{v_r + v_l}{2} \tag{3}$$

$$\dot{\theta} = \frac{v_r - v_l}{L} \tag{4}$$

Substituting (1) and (2) into (3) and (4), we got an equation that explains the relationship between each wheel angular velocity and robot linear and angular velocity in (5) and (6).

$$v = r\left(\frac{\omega_r + \omega_l}{2}\right) \tag{5}$$

$$\dot{\theta} = r\left(\frac{\omega_r - \omega_l}{L}\right) \tag{6}$$

We can project the robot's linear velocity into the x-axis and y-axis in global coordinate to calculate the robot's global x-axis velocity (7) and y-axis velocity (8). However, angular velocity in global coordinate and local coordinate has the same value as the local robot angular velocity (9).

$$v_x = v\cos\theta \tag{7}$$

$$v_y = v\sin\theta \tag{8}$$

$$\dot{\theta} = \frac{v_r + v_l}{L} \tag{9}$$

The Simulink model was created using (5) until (9) to simulate the robot behavior with the mechanical parameter of JetBot in Table 2. The robot will take the angular velocity of each motor as input and have eight states to be observed and as a feedback variable, as shown in Fig. 6.

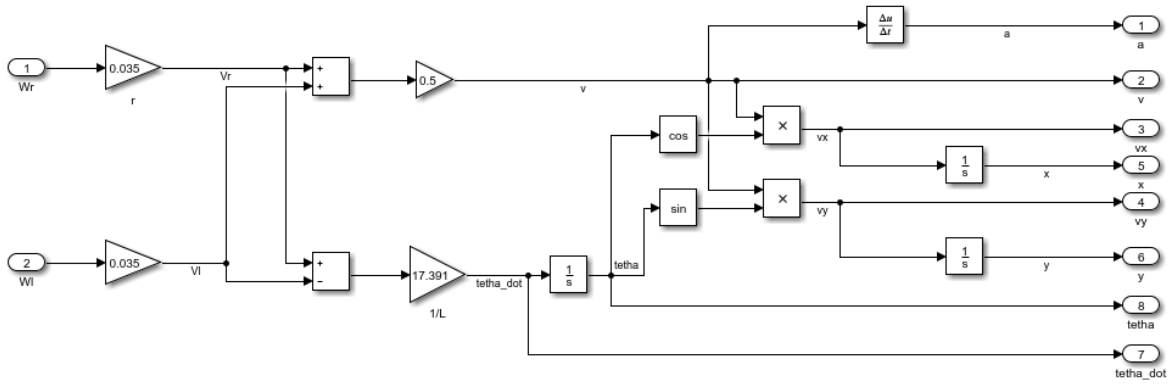

Fig. 6. Differential drive two-wheeled robot Simulink model

2.3. Position Controller

Position control aims to calculate robot linear velocity and angular velocity to move into the desired position. Position controls calculate robot linear velocity and angular velocity using error between robot states and desired states from trajectory generator and control gain that prioritize either position or angle error correction.

Position controllers accept goal coordinate as an input that consists of position in the x-axis and position in the y-axis symbolled as u in (10).

$$u = [x_u \quad y_u]^T \tag{10}$$

The position controller also needs a robot state to calculate the error. The robot state observed by the position controller is the current robot position in the x and y-axis, also robot orientation in angle symbolled as x in (11).

$$x = [x_r \quad y_r \quad \theta_r]^T \tag{11}$$

Position control calculates the position error as in (12), (13), (14) to produce desired robot wheel velocity and distance between robot and goal position as in (15).

$$e_x = x_u - x_r \tag{12}$$

$$e_y = y_u - y_r \tag{13}$$

$$e_\theta = \tan^{-1}\left(\frac{y_g - y_r}{x_g - x_r}\right) - \theta_r \tag{14}$$

$$d = \sqrt{(x_u - x_r)^2 + (y_u - y_r)^2} \tag{15}$$

Linear and angular velocities were calculated to produce v and ω as in (16) and (17) using Lyapunov stability. K_p is the gain for distance correction and K_θ is the gain for orientation correction.

$$v = K_p d \cos e_\theta \tag{16}$$

$$\omega = K_p(\cos e_\theta \sin e_\theta) + K_\theta(e_\theta) \tag{17}$$

The Simulink model was created by utilizing (12) until (17), as shown in Fig. 7. Using the proposed method position controller can calculate the linear velocity and angular velocity or Jetbot Robot while considering each state and its relationship.

Position controller outputted robot linear velocity and angular velocity. However, the robot only accepts angular velocity for each motor. Therefore, a converter needs to convert the linear and angular velocity into wheel linear velocity based on robot mechanical specification. The converter equations in (18) and (19) can be obtained by rearranging and substituting (5) and (6) in terms of wheel linear velocity. The Simulink converted model in Fig. 8 was created using (18) and (19) to simulate the system.

$$\omega_r = \frac{2v + \dot{\theta}L}{2R} \tag{18}$$

$$\omega_l = \frac{2v - \dot{\theta}L}{2R} \tag{19}$$

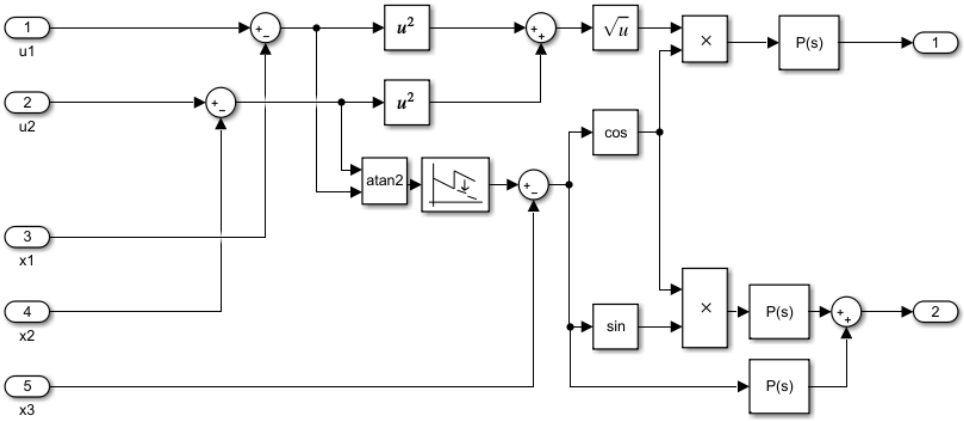

Fig. 7. Position controller Simulink model

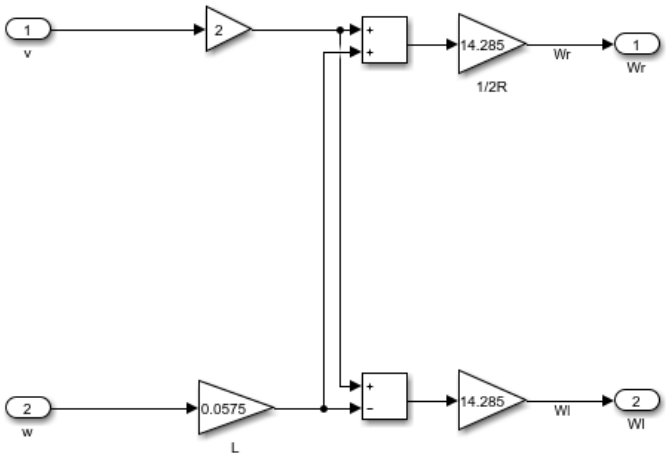

Fig. 8. Converter Simulink model

2.4. Trajectory Generator

A trajectory generator is a function that generates the desired robot position in the global x-axis and y-axis. These works use four kinds of trajectory to see the behavior of the simulated robot. The trajectory function utilizes the trigonometry function to create a continuous path for the robot.

The first trajectory is a square-shaped trajectory calculated using (20) and (22) with rules (21) and (23). The symbol a denotes the radius of the square calculated from the center.

$$x(t) = \sin t \tag{20}$$

$$x(t) = \begin{cases} a, & x(t) > 0 \\ -a, & x(t) < 0 \end{cases} \tag{21}$$

$$y(t) = \cos t \tag{22}$$

$$y(t) = \begin{cases} a, & y(t) > 0 \\ -a, & y(t) < 0 \end{cases} \tag{23}$$

The second trajectory is a circle-shaped path calculated using (24) and (25). The symbol a also denotes the radius of the circle.

$$x = \sin(t)\, a \tag{24}$$

$$y = \cos(t)\, a \tag{25}$$

The third trajectory is an infinity-shaped trajectory calculated using Lemniscate of Gerono in (26) and (27).

$$x = \cos(t)\, a \tag{26}$$

$$y = \frac{\sin(2t)}{2} a \tag{27}$$

The last trajectory is also an infinity-shaped geometry symbol but using Lemniscate of Bernoulli in (28), (29), and (30). The difference between the Lemniscate of Bernoulli and the Lemniscate of Gerono is the scale variable that makes the Bernoulli shape smoother than the Lemniscate of Gerono trajectory.

$$scale = \frac{2}{3 - \cos(2t)} \tag{28}$$

$$x = scale * \cos(t)\, a \tag{29}$$

$$y = scale \frac{\sin(2t)}{2} a \tag{30}$$

Testing proses was conducted by comparing the integral absolute error (IAE) results from square and circle trajectory and trajectory generated by Lemniscate of Bernoulli and Lemniscate of Gerono. Square trajectory has a very sharp turn compared with circle-shaped trajectory. Lemniscate of Gerono has a sharper turn than Lemniscate of Bernoulli trajectory, although not as sharp as square trajectory.

3. RESULTS AND DISCUSSION

The simulation took four different paths, and the feedback gain was chosen by using integral absolute error (IAE) as the metric to explore the simulated robot's behavior. The smaller IAE means the robot has better performance at tracking the given trajectory. Robots start at position zero and face the x-axis direction in the global x-axis and 0 in the global y-axis and will track the given path with the same radius from point O.

Fig. 9 until Fig. 12 shows that the increasing position feedback gain reduces the overall IAE for every trial. The increase of feedback gain also increases the robot priority to correct its position more significantly than the smaller position feedback value. Although there is some case in square trajectory trial that 9.5 gain produce better IAE than gain with a value of 10 as shown in Table 4.

Table 4. Differential drive wheeled robot-controlled and observed parameter.

Robot Parameter	Trajectory Type			
	Square	**Circle**	**Gerono**	**Bernoulli**
Position Feedback gain	9.5	10	10	10
Orientation Feedback gain	0.5	10	10	10
x axis IAE	88.782	65.365	70.469	71.724
y axis IAE	97.757	76.28	65.485	46.84

That happened because of too much gain, creating a more significant overshoot when the robot turned sharply, especially in a square-shaped trajectory in Fig. 9. The significant overshoot makes the robot need to do more works to correct its position. Although it can touch the desired position, the entire shape of the trajectory becomes distorted.

This case also happens in the Lemniscate of Gerono trajectory in Fig. 12 with a sharper turn than the Bernoulli trajectory Fig. 11. However, in Lemniscate of Gerono trajectory, the turn is not as sharp as the squared-shape trajectory, and so the orientation error is smaller, making the overshoot damped. The proposed control method is good enough to follow turns not over 90 degrees as in square trajectory.

The proposed algorithm can only correct it when the robot detects an error and gives linear reaction with proportional feedback gain. Thereby high velocity near a turn will produce a significant overshoot. The overshoot effect can be reduced by decreasing the proportional feedback gain but with the cost of the robot's ability to track goal position.

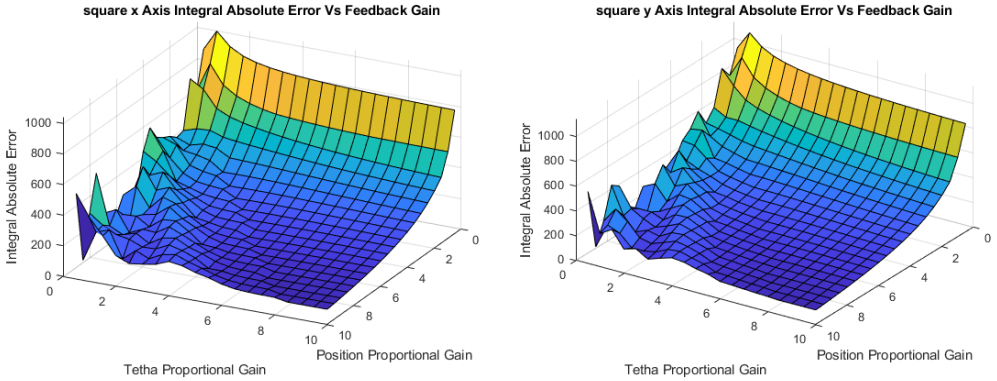

Fig. 9. Variation of gain versus IAE result in square-shaped trajectory test surface graph

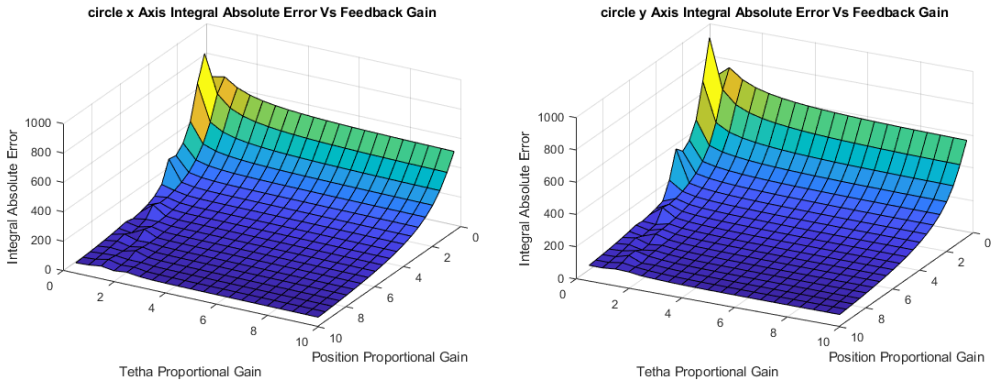

Fig. 10. Variation of gain versus IAE result in circle-shaped trajectory test surface graph

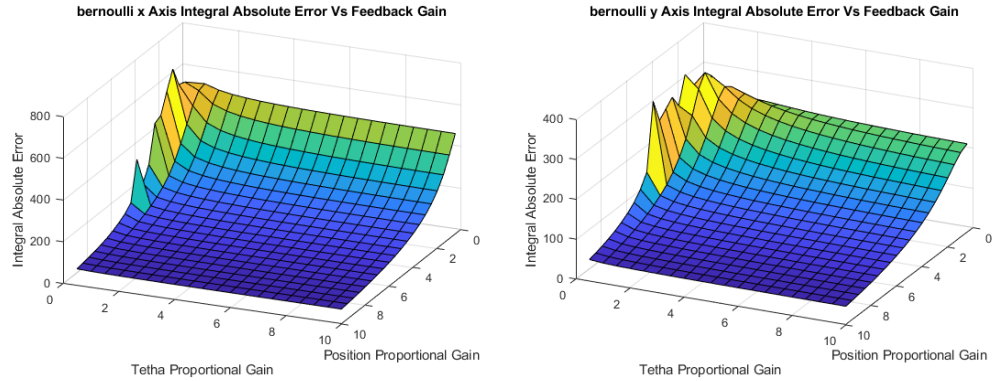

Fig. 11. Variation of gain versus IAE result in Leminscent of Bernoulli trajectory test surface graph

Circle, Gerono, and Bernoulli trials give an excellent result in tracking and producing correct trajectory shapes. The robot starts at point O and then tracks the goal position even if the goal position is always moving. The robot can follow Circle, Gerono, and Bernoulli trajectories asymptotically, while in square trajectory robot can follow the trajectory with deformation on every side Fig. 13.

Deformation shape in square trajectory trial happens because of the correction when robot turn at the corner. When the robot reaches the corner, the orientation error will become more prominent than the position error as it will prioritize correcting its orientation over the position. The more significant error will make a more significant overshoot. As such, the robot orientation will slightly be misaligned. The robot will keep moving even if there is any misalignment in robot orientation, and when the robot moves forward with misaligned orientation, it makes a deformed robot path. Even if the robot finally corrects its position and orientation, it will repeat in the next square corner.

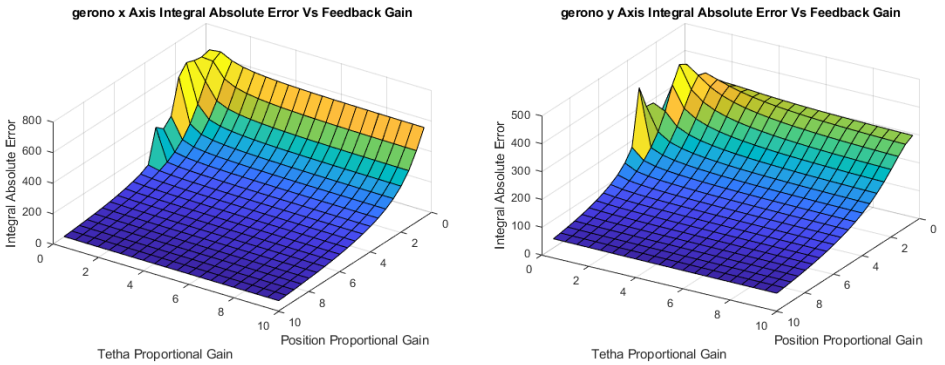

Fig. 12. Variation of gain versus IAE result in Lemniscent of Gerono trajectory test surface graph

The possible solution to reduce the big overshoot problem when robots encounter sharp turn is by reducing the orientation feedback gain. Reducing the orientation feedback gain will reduce the overshoot. The best feedback gain for a square trajectory is 9.5 for position feedback and small orientation feedback gain that is 0.5 instead of 10, like other trajectory trials. 9.5 feedback gain for position feedback is used to correct robot position, and 0.5 orientation feedback gain is correct the orientation problem with the smallest possible overshoot.

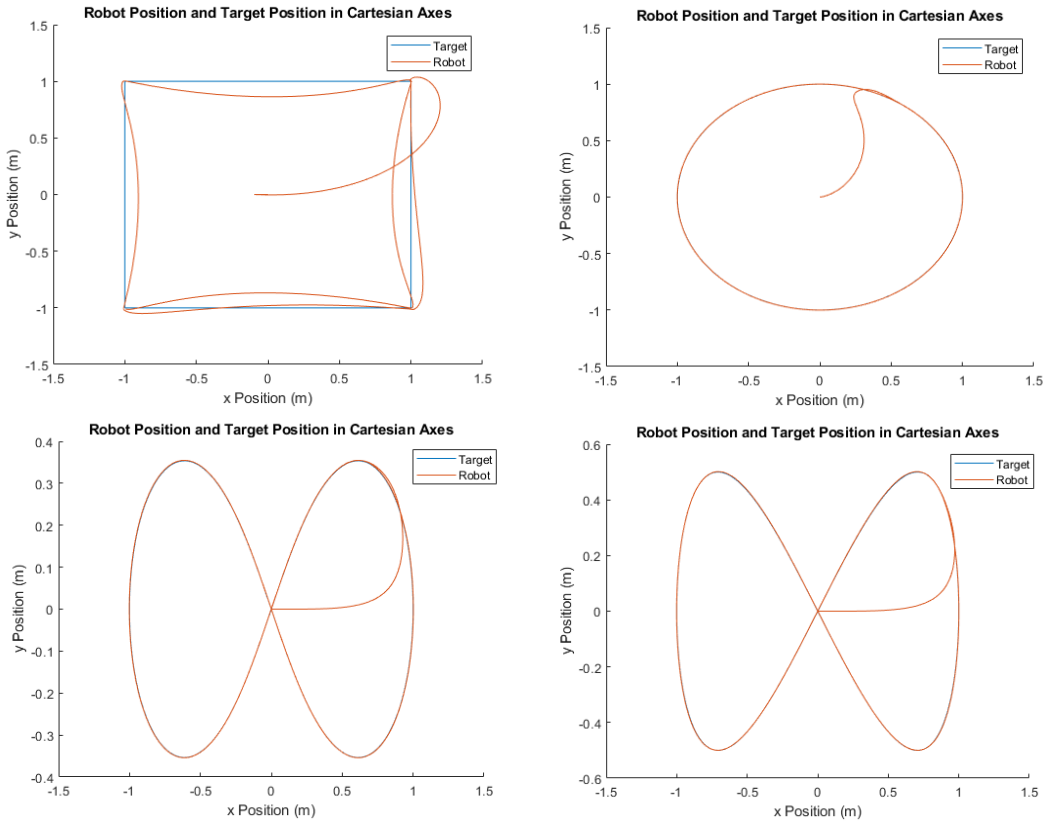

Fig. 13. The square-shaped trajectory (top-left), circle-shaped trajectory (top-right), a trajectory from Leminscent of Bernoulli (left), and trajectory from Lemniscent of Gerono (right)

4. CONCLUSION

The proposed controller can track a different trajectory type, such as continuous turn created from Circle, Gerono, Bernoulli, and sudden turn in a square-shaped trajectory. The controller can prioritize position tracking or orientation correction by adjusting the position feedback gain and orientation feedback gain. Increasing position feedback gain and orientation feedback gain will decrease the IAE when the robot tracks a continuous turn and continuous moving goal position. For tracking a sudden turn, the smaller orientation feedback is better

to reduce the orientation correction overshoot. The proposed control algorithm still needs improvement, especially when the robot faces a sudden turn, such as the square shape trajectory tested in this work.

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] M. M. A. De Graaf, A. Dragan, B. F. Malle, and T. Ziemke, "Introduction to the Special Issue on Explainable Robotic Systems," *ACM Trans. Human-Robot Interact.*, vol. 10, no. 3, pp. 3–6, 2021. https://doi.org/10.1145/3461597

[2] M. S. Kaiser, S. Al Mamun, M. Mahmud, and M. H. Tania, "Healthcare Robots to Combat COVID-19," *Lecture Notes on Data Engineering and Communications Technologies*, pp. 83–97, 2020. https://doi.org/10.1007/978-981-15-9682-7_10

[3] C.-C. Chang, Y.-H. Juan, C.-L. Huang, and H.-J. Chen, "Scenario Analysis for Road Following Using JetBot," in *2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, Oct. 2020, pp. 403–406. https://doi.org/10.1109/ECICE50847.2020.9302017

[4] B. M. Yousuf, A. Saboor Khan, and S. Munir Khan, "Dynamic modeling and tracking for nonholonomic mobile robot using PID and back-stepping," *Adv. Control Appl.*, Jun. 2021. https://doi.org/10.1002/adc2.71

[5] A. Safaei and M. N. Mahyuddin, "Optimal model-free control for a generic MIMO nonlinear system with application to autonomous mobile robots," *Int. J. Adapt. Control Signal Process.*, vol. 32, no. 6, pp. 792–815, Jun. 2018. https://doi.org/10.1002/acs.2865

[6] E. Kuantama, I. Tarca, and R. Tarca, "Feedback Linearization LQR Control for Quadcopter Position Tracking," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, Apr. 2018, pp. 204–209. https://doi.org/10.1109/CoDIT.2018.8394911

[7] F. N. Martins, M. Sarcinelli-Filho, and R. Carelli, "A Velocity-Based Dynamic Model and Its Properties for Differential Drive Mobile Robots," *J. Intell. Robot. Syst. Theory Appl.*, vol. 85, no. 2, pp. 277–292, 2017. https://doi.org/10.1007/s10846-016-0381-9

[8] N. K. Goswami and P. K. Padhy, "Sliding mode controller design for trajectory tracking of a nonholonomic mobile robot with disturbance," *Comput. Electr. Eng.*, vol. 72, pp. 307–323, Nov. 2018. https://doi.org/10.1016/j.compeleceng.2018.09.021

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

[10] M. Yallala and S. J. Mija, "Path tracking of differential drive mobile robot using two step feedback linearization based on backstepping," in *2017 International Conference on Innovations in Control, Communication and Information Systems (ICICCI)*, Aug. 2017, pp. 1–6. https://doi.org/10.1109/ICICCIS.2017.8660858

[11] N. V. Tinh, N. T. Linh, P. T. Cat, P. M. Tuan, M. N. Anh, and N. P. T. Anh, "Modeling and feedback linearization control of a nonholonomic wheeled mobile robot with longitudinal, lateral slips," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 996–1001. https://doi.org/10.1109/COASE.2016.7743512

[12] T. T. Mac, C. Copot, R. De Keyser, T. D. Tran, and T. Vu, "MIMO Fuzzy Control for Autonomous Mobile Robot," *J. Autom. Control Eng.*, vol. 3, no. 6, pp. 65–70, 2015. https://doi.org/10.12720/joace.4.1.65-70

[13] A. Pandey, V. S. Panwar, M. E. Hasan, and D. R. Parhi, "V-REP-based navigation of automated wheeled robot between obstacles using PSO-tuned feedforward neural network," *J. Comput. Des. Eng.*, vol. 7, no. 4, pp. 427–434, Aug. 2020. https://doi.org/10.1093/jcde/qwaa035

[14] P. T. Jardine, M. Kogan, S. N. Givigi, and S. Yousefi, "Adaptive predictive control of a differential drive robot tuned with reinforcement learning," *Int. J. Adapt. Control Signal Process.*, vol. 33, no. 2, pp. 410–423, Feb. 2019. https://doi.org/10.1002/acs.2882

[15] N. Y. Allagui, F. A. Salem, and A. M. Aljuaid, "Artificial Fuzzy-PID Gain Scheduling Algorithm Design for Motion Control in Differential Drive Mobile Robotic Platforms," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–13, Oct. 2021. https://doi.org/10.1155/2021/5542888

[16] U. Zangina, S. Buyamin, M. S. Z. Abidin, M. S. A. Mahmud, and H. S. Hasan, "Nonlinear PID controller for trajectory tracking of a differential drive mobile robot," *J. Mech. Eng. Res. Dev.*, vol. 43, no. 7, pp. 255–269, 2020. http://eprints.utm.my/id/eprint/90651/1/UmarZangina2020_NonLinearPIDControllerforTrajectoryTracking.pdf

[17] R. L. S. Sousa, M. D. do Nascimento Forte, F. G. Nogueira, and B. C. Torrico, "Trajectory tracking control of a nonholonomic mobile robot with differential drive," in *2016 IEEE Biennial Congress of Argentina (ARGENCON)*, Jun. 2016, pp. 1–6. https://doi.org/10.1109/ARGENCON.2016.7585356

[18] S. K. Malu and J. Majumdar, "Kinematics, Localization and Control of Differential Drive Mobile Robot Global Journal of Researches in Engineering: H Kinematics, Localization and Control of Differential Drive Mobile Robot," *Type Double Blind Peer Rev. Int. Res. J. Publ. Glob. Journals Inc*, vol. 14, no. 1, 2014. https://engineeringresearch.org/index.php/GJRE/article/view/1233

[19] M. Elsayed, A. Hammad, A. Hafez, and H. Mansour, "Real Time Trajectory Tracking Controller based on Lyapunov

Function for Mobile Robot," *Int. J. Comput. Appl.*, vol. 168, no. 11, pp. 1–6, 2017. https://doi.org/10.5120/ijca2017914540
[20] J. García-Sánchez, S. Tavera-Mosqueda, R. Silva-Ortigoza, V. Hernández-Guzmán, J. Sandoval-Gutiérrez, M. Marcelino-Aranda, H. Taud, and M. Marciano-Melchor, "Robust Switched Tracking Control for Wheeled Mobile Robots Considering the Actuators and Drivers," *Sensors*, vol. 18, no. 12, p. 4316, Dec. 2018. https://doi.org/10.3390/s18124316
[21] S. Memon and A. N. Kalhoro, "Design of Multivariable PID Controllers: A Comparative Study," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 21, no. 8, pp. 377–384, 2021. https://doi.org/10.22937/IJCSNS.2021.21.9.2
[22] I. Carlucho, M. De Paula, and G. G. Acosta, "Double Q-PID algorithm for mobile robot control," *Expert Syst. Appl.*, vol. 137, pp. 292–307, Dec. 2019. https://doi.org/10.1016/j.eswa.2019.06.066
[23] P. Panahandeh, K. Alipour, B. Tarvirdizadeh, and A. Hadi, "A self-tuning trajectory tracking controller for wheeled mobile robots," *Ind. Robot Int. J. Robot. Res. Appl.*, vol. 46, no. 6, pp. 828–838, Oct. 2019. https://doi.org/10.1108/IR-02-2019-0032
[24] M. N. Shauqee, P. Rajendran, and N. M. Suhadis, "An effective proportional-double derivative-linear quadratic regulator controller for quadcopter attitude and altitude control," *Automatika*, vol. 62, no. 3–4, pp. 415–433, Oct. 2021. https://doi.org/10.1080/00051144.2021.1981527
[25] T. A. Mai, T. S. Dang, D. T. Duong, V. C. Le, and S. Banerjee, "A combined backstepping and adaptive fuzzy PID approach for trajectory tracking of autonomous mobile robots," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 43, no. 3, p. 156, Mar. 2021. https://doi.org/10.1007/s40430-020-02767-8

BIOGRAPHY OF AUTHORS

Muhammad Auzan is a junior lecturer Universitas Gadjah Mada since 2021. He graduated with an electronics and instrumentation major in 2017 and computer science master's degree in 2020. Auzan specializes in robotics and the internet of things; his main research topic is robotics such as wheeled robots, humanoid robots, and manipulators. Email: muhammadauzan@ugm.ac.id

Roghib Muhammad Hujja is junior lecturer Universitas Gadjah Mada. He graduates with an electronics and instrumentation major and a computer science master's degree in Universitas Gadjah Mada. Hujja specializes are in sensors, signals, and the internet of things, with his main research topic is the internet of things. Email: roghib.muh@ugm.ac.id

M. Ridho Fuadin is a master degree student in Universitas Gadjah Mada. He took a computer science major in 2020. He graduates from an electronics and instrumentation major and already has working experience in a control system company. His topic for his master's degree thesis is robotics. Email: m.ridho.f@mail.ugm.ac.id

Dr. Danang Lelono is a lecturer with many years of experience in Universitas Gadjah Mada. Danang specializes in intelligent sensors, intelligence systems, instrumentation & control, embedded system, and electronic nose. Danang is also active in assisting the government and has already got many research grants from the government. His main research topic is electronic noses and already published many papers in national and international journals. Email: danang@ugm.ac.id