Cylinder: Jurnal Ilmiah Teknik Mesin
Volume 12, Nomor 01, April 2026
ISSN 2252-925X (Online)

Path-Planning Module for Tricycle Mobile Robot

Tommy, Christiand*, Frederikus Wenehenubun
Program Studi Teknik Mesin, Universitas Katolik Indonesia Atma Jaya
E-mail: christiand@atmajaya.ac.id

ABSTRAK

Modul *path planning* merupakan bagian yang sangat penting untuk navigasi sistem-sistem otonom. Penelitian ini bertujuan untuk mengembangkan modul *path planning* menggunakan bahasa pemrograman Python berbasis algoritma A* yang dijalankan dalam perangkat lunak V-REP untuk mensimulasikan jalur lintasan (*path*) robot. Algoritma A* digunakan dalam penelitian ini untuk menghasilkan lintasan robot roda tiga (*tricycle mobile robot*). Simulasi maupun eksperimen fisik dilakukan untuk mengevaluasi kinerja modul *path planning* yang telah dikembangkan. Pada eksperimen yang dilakukan, robot bergerak berdasarkan masukan (*input*) berupa kecepatan dan sudut kemudi (*steering angle*) yang diberikan pada setiap interval waktu dan dalam bentuk hasil regresi polinomial. Perbedaan antara hasil simulasi dan eksperimen fisik disebabkan oleh gaya inersia yang bekerja pada robot serta faktor-faktor lainnya. Dominasi sudut kemudi pada robot ketika bergerak diperkirakan mempengaruhi *error* posisi robot.

Kata kunci :
Robot Roda Tiga, *Path Planning*, Algoritma A*, V-REP, Simulasi.

ABSTRACT

The path planning module is essential for the navigation of autonomous systems. This research aims to develop a path-planning module in Python programming language using the A* algorithm in conjunction with V-REP software which is used to simulate the path-planning outcome. The A* algorithm is used in this research to implement path-planning for a tricycle mobile robot. Both simulated and physical studies were performed to evaluate the module's performance. In the experiments, inputs of velocity and steering angle are given to the mobile robot at every period as obtained from the simulation regression. The discrepancy between the simulated and physical experimental results was attributed to the inertia force acting on the robot and other factors. The steering angle dominance on a mobile robot also affects the final position of the robot.

Keywords :
Tricycle Mobile Robot, Path Planning, A* Algorithm, V-REP, Simulation.

1. INTRODUCTION

A mobile robot can travel from one site to another within its operational environment. This mobility allows the robot to execute diverse activities, including material handling, inspection, service delivery, and exploration. A crucial capability facilitating this functionality is autonomous navigation, enabling a robot to operate independently without ongoing human involvement. Several critical components, including localization, mapping, and path planning, must function cohesively inside the autonomous navigation system. The path planning module is essential, as it establishes the route the robot will take to reach its objective effectively and safely.

Path planning is the process of creating a viable route from a starting location to a destination while accounting for constraints such as obstructions, robotic kinematics, and environmental factors. When the environment is known, path-planning algorithms can determine collision-free trajectories by evaluating spatial data and the distribution of obstacles [1]. Environmental

obstacles can appear in either static or dynamic forms. Static obstacles like walls or furniture are immobile, while dynamic obstacles, including individuals or moving items, add complexity owing to their erratic behavior [2]. In environments characterized by static obstacles numerous algorithms have been developed, including cell decomposition methods, heuristic-based strategies, and evolutionary techniques like genetic algorithms. Heuristic-based algorithms are predominantly preferred for their equilibrium between computing efficiency and solution optimality. The A* algorithm is one of the most used heuristic-based path planning methods. This algorithm integrates the benefits of Dijkstra's algorithm and greedy best-first search by considering both the cost of the traversed path and an estimation of the cost to attain the objective [3]. Consequently, A* can identify an optimal or near-optimal path with minimal computational effort, rendering it adequates for real-time applications in organized environments. Its adaptability and efficacy have resulted in its broad utilization in robotics and autonomous systems. Nonetheless, despite its advantages, the application of A* in actual robotic systems necessitates meticulous attention to issues such grid resolution, heuristic formulation, and computational limitations.

In the realm of autonomous vehicles, especially those with tricycle configuration, path-planning is rendered more complex due to the vehicle's distinctive kinematic restrictions. In contrast to differential-drive robots, tricycle-type vehicles possess steering devices that restrict turning radius and maneuverability. This indicates that not all mathematically viable courses are physically realizable by the robot. Consequently, a path-planning module must guarantee that the esulting paths are both collision-free and kinematically viable [4]. This necessitates the incorporation of motion constraints, like steering angle restrictions and velocity profiles, into the planning procedure.

Due to the rising demand for autonomous systems across many applications, the creation of a robust path-planning module specifically designed for tricycle-type mobile robots is important. Current methodologies frequently emphasize generic robot models, neglecting the particular restrictions inherent to tricycle kinematics. This establishes a disparity between theoretical path planning outcomes and practical execution.

The research work presented in this article seeks to create a path planning module utilizing the A* algorithm with V-REP software, tailored for autonomous tricycle vehicles. This advancement is crucial for enhancing navigation reliability, ensuring motion feasibility, and facilitating the widespread implementation of autonomous robotic systems in practical settings.

2. PATH PLANNING MODULE BASED ON V-REP SOFTWARE

This path planning module mostly relies on simulations conducted with V-REP software. Python functions as a programming language that handles data and runs path-planning algorithms. V-REP offers a virtual environment for the realistic simulation of path planning scenarios tailored to the specific real-world physical environment. The virtual environment comprises all entities that replicate the real environment.

2.1 V-REP

V-REP, often referred to as CoppeliaSim, is an advanced robot simulation platform widely employed for the modeling, analysis, and validation of robotic systems in a controlled virtual environment. The program features a physics-based engine that simulates robot kinematics, dynamics, and sensor interactions, facilitating realistic system behavior without the need for physical prototypes.

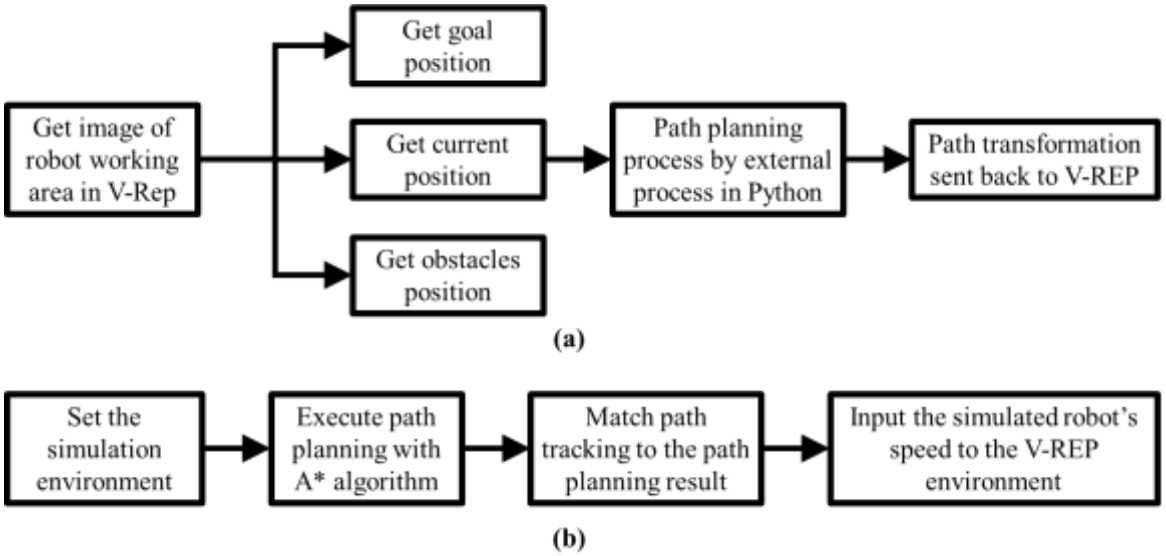

Figure 1. Steps for Simulating Path in V-REP : (a) Path Planning by External Process in Python, (b). Main Process in V-REP.

In mobile robotics, it enables the integration and assessment of navigation algorithms, encompassing path-planning and control strategies. It is frequently preferred over other simulation platforms due to its exceptional flexibility, compatibility with various programming interfaces including Python, C++, and MATLAB, and its proficient management of intricate simulation scenarios, rendering it especially appropriates for research and rapid prototyping applications [5].

2.2 Simulation Steps

In the preliminary phase of the simulation process, a virtual environment is created within the V-REP simulator to depict the robot's operational workspace, figure 1.(b). This configuration entails specifying the spatial arrangement, including the location of fixed barriers, the robot's starting position, and the goal position. A 3D model of the mobile robot, specifically designed with tricycle kinematics, is loaded into the environment to verify that its motion behavior complies with realistic physical rules. The initialization of this environment is an essential step as it directly affects the precision and validity of the path-planning outcomes. By carefully adjusting these settings, the simulation can replicate authentic navigation scenarios, facilitating a solid evaluation of the developed path-planning module.

2.3 Path Planning Steps with A* Algorithm

After the simulation environment is set up correctly, the A* algorithm is used to carry out the path-planning procedure, figure 1.(a). This algorithm analyzes the given environment by assessing potential routes from the robot's starting place to the goal while avoiding obstacles. This stage yields a series of waypoints that represent a collision-free trajectory. A path-tracking process is also developed to guarantee that the robot follows the designated route. In this case, the path-tracking controller generates motion commands based on the discrepancy between the robot's present position and the reference path. These commands encompass velocity inputs and steering modifications, which are continuously updated to reduce position errors.

The simulation workflow includes an image-based data collecting procedure that interfaces the V-REP environment with an external processing module created in Python programming language. The simulator first acquires image of the environment. The images are subsequently sent to the Python program for further analysis. The Python

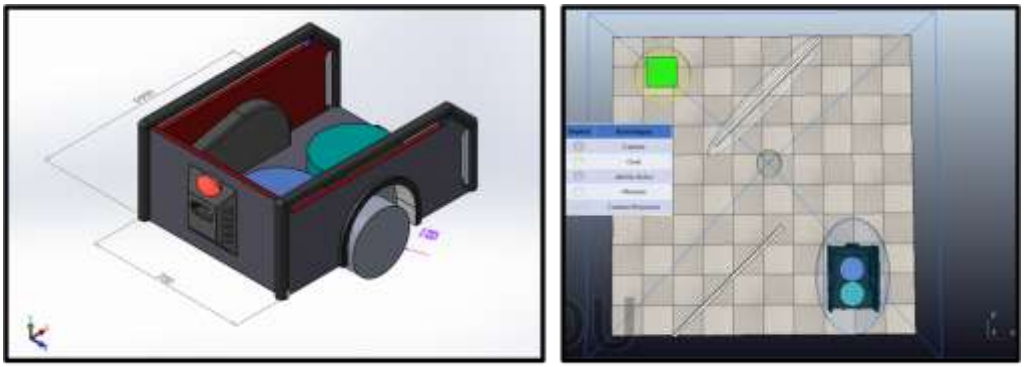

Figure 2. Virtual Environment of V-REP : (a) 3D Model of Tricycle Mobile Robot, (b) Working Area of the Robot with Obstacles.

program obtains essential data, including the robot's position, the goal position, and the obstacles. This is accomplished by image processing methods, namely by differentiating objects through color segmentation and determining their geometric centroids. This method is computationally efficient means of interpreting the simulation environment without depending on direct coordinate extraction from the simulator.

After acquiring the necessary positional data, the Python program does path planning with the A* algorithm. The generated path is subsequently refined via a path smoothing procedure to minimize abrupt directional changes and yield a more natural trajectory. The smoothing process is particularly crucial for tricycle-type robots, which face steering limitations and struggle to perform sharp turns effectively. The smoothed path is finally converted from the Python coordinate system to the V-REP coordinate frame. The path is subsequently transmitted to the simulator, where it is executed by the robot using the path tracking controller. This integrated workflow illustrates a comprehensive cycle of perception, planning, and control, guaranteeing that the robot can travel effectively inside the simulated environment.

2.3 Simulated Robot

The three-dimensional model of the tricycle mobile robot is developed to provide a simplified yet representative physical structure for simulation within the V-REP environment, figure 2.(a). The robot adopts a tricycle configuration consisting of a single front steering wheel and two rear wheels, enabling the study of nonholonomic motion behavior. Rather than focusing on detailed internal components, the model emphasizes the outer perimeter and overall dimensions of the robot body. This outer boundary is particularly important, as it defines the collision envelope used by the simulator during path-planning and navigation processes. By accurately representing the robot's physical footprint, the simulation can reliably evaluate obstacle avoidance and path feasibility.

2.3 Simulated Environment

A structured workspace is created within the V-REP simulation to examine the effectiveness of the path-planning module under controlled conditions, figure 2.(b). The environment consists of a flat, bounded area where the tricycle mobile robot navigates toward a predefined goal position. Several obstacles, primarily in the form of walls or elongated barriers, are placed to create constrained motion and potential collision scenarios. These obstacles remain unchanged, enabling the algorithm to produce an optimal and viable route. A virtual overhead camera is employed to capture the complete picture, supplying visual data for the analysis of the robot's and obstacles' positions. The architecture is

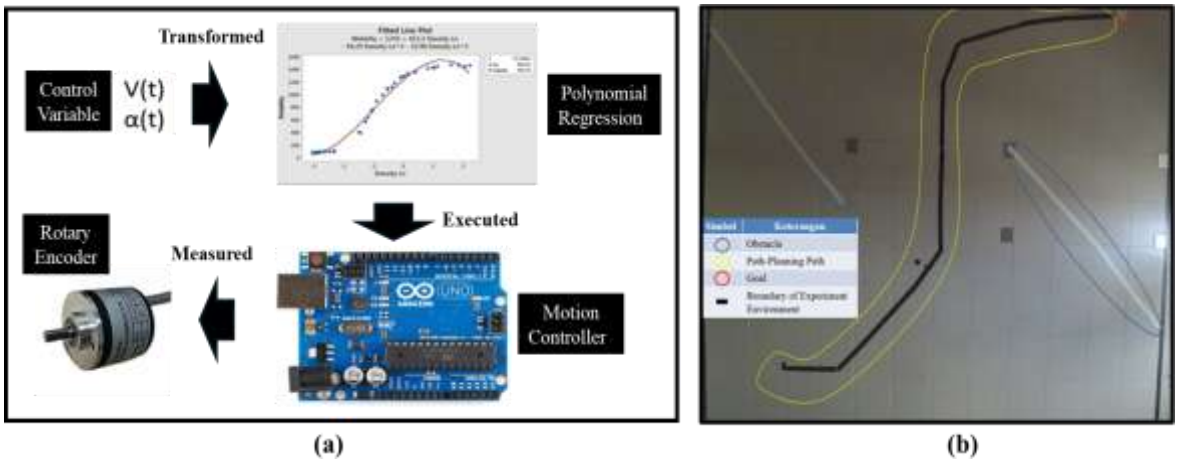

Figure 3. (a) process of executing the control variables and measuring the robot's position, (b) real physical environment of the experiment.

constructed to replicate the simplified real-world settings while maintaining necessary complexity to evaluate navigation performance.

3. RESEARCH METHOD

3.1 Tricycle Robot Kinematics

In the implementation of the proposed path-planning module, the differential-drive kinematics model is utilized as a mathematical representation to approximate the motion behavior of a tricycle-type mobile robot. This approach is motivated by the availability of a well-established differential-drive framework within the simulation environment, which provides a tractable basis for modeling robot motion [6]. Through appropriate kinematic adaptation, the differential-drive formulation is employed to capture the essential characteristics of tricycle locomotion, particularly in relating wheel velocities to steering behavior.

To establish this relationship, the steering angle $\alpha(t)$of the tricycle model is derived from the right and left wheel velocities, $v_r(t)$ and $v_l(t)$, of the differential-drive system, equation (1). The steering angle serves as the key parameter that enables the transformation of differential motion into an equivalent tricycle representation, and can be expressed in equation (1), where d denotes the lateral distance between the wheels and L represents the wheelbase of the tricycle configuration.

$$\alpha = \tan^{-1}\left(\frac{2d\left(v_l(t)-v_r(t)\right)}{L\left(v_l(t)+v_r(t)\right)}\right) \quad (1)$$

The analytical evaluation indicate that the values of $v(t)$ obtained from both formulations are equivalent when the derived steering angle $\alpha(t)$ is applied. This confirms that the differential-drive kinematics model can be effectively used to derive the steering angle and represent the motion behavior of the tricycle robot. Consequently, the proposed approach enables consistent application of the path-planning module, while maintaining compatibility between the simulation framework and the physical robot configuration.

3.2 Design of Experiment

The following is an explanation of the stages involved in applying the proposed path-planning module to the experimental system, outlining the process from data acquisition to validation, figure 3(a):

1. Data in the form of steering angle $\alpha(t)$ and robot velocity $v(t)$ are obtained for each time step.
2. The steering angle $\alpha(t)$ and robot velocity $v(t)$ data are plotted against time to generate their respective profiles. Subsequently, polynomial regression is performed to determine the trendline equation that optimally represents the steering angle and velocity as function of time.

3. The regression values are commanded to the Arduino-based motion controller to regulate the BLDC motor and the steering servo.
4. Experimental velocity (v) and position (x, y) data are then acquired using a rotary encoder for validation purposes.

3.3 Design of Physical Environment

The experimental environment is designed to evaluate the performance of the developed path-planning module derived from V-REP simulation in a real-world setting, figure 3(b). The test area consists of a flat tiled surface enclosed within a predefined boundary, marked by a black trajectory line that represents the feasible navigation corridor. The mobile robot is initialized at a designated starting position and is required to reach the specified goal position (red dot) by following the planned path (yellow line). Obstacles are represented by elongated white-striped regions positioned diagonally across the workspace, creating constrained passages that challenge the robot's navigation capability. These obstacles are strategically placed to imitate the conditions similar to environment condition in simulation. It ensures consistency between the simulation and experiment settings. The setup enables observation of the robot's ability to follow the planned trajectory, avoid collisions, and maintain motion stability under realistic operating conditions.

4. RESULT AND ANALYSIS

Data from the experiment are presented in the form of velocity, position, and steering angle information.

4.1 Velocity

The velocity performance is evaluated by comparing the reference polynomial regression with the experimental measured using a rotary encoder. The reference data exhibits a smooth and consistent trend that reflects the ideal motion generated by the model, figure 4(a). In contrast, the experimental velocity shows noticeable fluctuations, particularly during acceleration, turning, and deceleration. These discrepancies are mainly caused by real-world factors such as actuator response delays, friction, wheel slip, and sensor noise.

In several segments, the experimental velocity demonstrates overshoot and undershoot relative to the reference, indicating imperfect tracking performance. Although the overall trend follows the reference profile, the observed variations highlight the difference between ideal simulated behavior and actual system dynamics, emphasizing the influence of physical and measurement limitations on velocity accuracy.

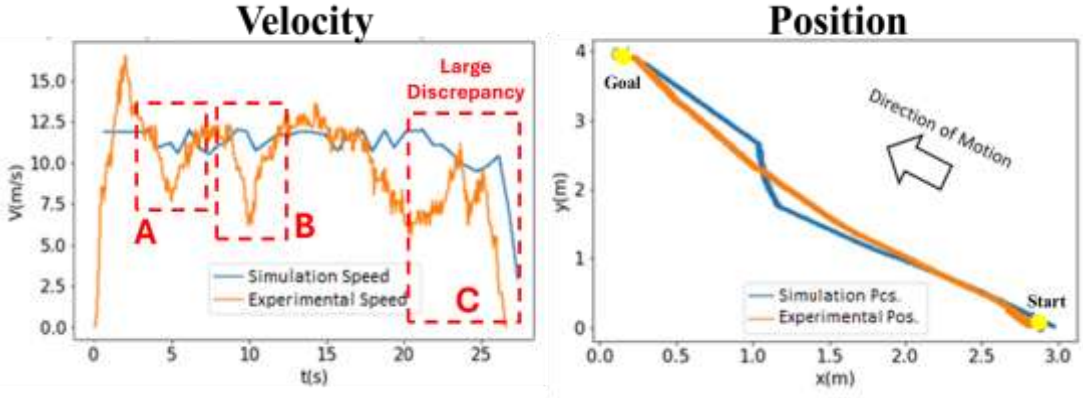

Figure 4. Result of Simulation vs Experiment : (a) Robot Velocity, (b) Robot Position.

4.2 Position

The position data shows that the primary performance limitation lies in the accumulated position error between simulation and experiment, figure 4(b). While both trajectories start with minimal discrepancy, the error becomes larger in the section (around x ≈ 1–1.5 m). This suggests a systematic drift rather than random noise. It is likely caused by wheel slip, uneven surface interaction, or bias in the rotary encoder measurements used for odometry.

Additionally, small steering inaccuracies and actuator delays can grow over time, leading to accumulated position error. When robot approaches the goal position, the error reduces. This indicates some corrective behavior in the control or path-following mechanism. Overall, the error pattern reflects cumulative effects of sensing and actuation imperfections rather than a fundamental issue in the path-planning algorithm itself.

4.3 Steering Angle

The steering angle graph in figure 5 shows that the experimental data closely follows the simulated reference, but with noticeable deviations that contribute to tracking error. The experimental curve slightly overshoots or undershoots the simulated angle at time 7–10 s and 14–17 s. This indicates actuator lag or limited steering precision. These discrepancies become more critical during sharp turns, figure 4(a) and figure 5. The data shows that greater steering angle dominance correlates with increasing of position error.

4.4 Error Analysis

Overall, the experiment data shows that the path planning module is potentially applied to the limited real-world scenario. However, discrepancies between simulation and experiment remain evident. These discrepancies can be attributed to several factors:

1. Encoder measurement error
   Inaccuracies arise from imperfect encoder readings, including possible slip on the encoder shaft or misalignment during speed transmission.
2. Control loop timing limitation
   The experimental velocity profile terminates earlier ($t < 26$ s) due to inconsistencies in the Arduino loop execution, where delayed final iterations cause the robot to stop prematurely.
3. Inertial effects
   Instances where experimental velocity exceeds the simulated reference are influenced by the robot's inertia, exacerbated by the absence of braking mechanisms and closed-loop control.
4. Motor asymmetry
   Differences in hardware characteristics and processing delays between left and right motors introduce uneven motion response.
5. Mechanical wear
   Degradation of the steering wheel shape (non-ideal circular form) affects motion consistency and accuracy.
6. Transient braking during steering
   Changes in steering angle induce momentary braking effects, leading to temporary reductions in velocity, figure 4(a) and figure 5.

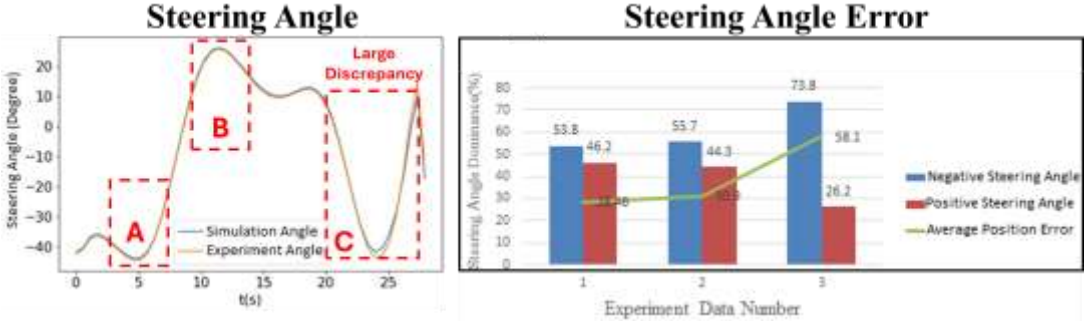

Figure 5. Result of steering angle and its error

7. Regression approximation
   The implemented regression model uses approximated values rather than exact parameters, introducing modeling error.
8. Computational limitations
   Limited numerical precision in Arduino, particularly in handling exponential calculations, leads to rounding errors and reduced accuracy.

5. CONCLUSION

This research proposes and evaluates the applicability of path planning module by V-REP simulation and A* algorithm. The main conclusions are as follows:

- The A* based path planning algorithm can be implemented on a mobile robot, despite discrepancies between simulation and experimental results. That because the A* core function generating collision-free paths remains reliable, and deviations typically arise from execution-level factors rather than the planning logic itself.
- Larger steering angles (left or right turns) lead to increased tracking error due to system and actuation limitations.
- Robot motion can be effectively controlled using time-dependent equations of velocity and steering angle derived from regression.
- Differential-drive kinematics can approximate tricycle motion, provided that specific constraints, such as pivot movement, are properly considered.

REFERENCES

[1] P. Zhang, C. Xiong, W. Li, X. Du, and C. Zhao. “Path Planning for Mobile Robot Based on Modified Rapidly Exploring Random Tree Method and Neural Network,” *International Journal of Advanced Robotic Systems*, vol. 15 no. 3, 2018.

[2] H. Zhang and M. Li. “Rapid Path Planning Algorithm for Mobile Robot in Dynamic Environment”. *Advances in Mechanical Engineering*, vol. 9, no. 12, 2017.

[3] H. Y. Zhang, W. M. Lin, and A. X. Chen, 2018. “Path Planning for the Mobile Robot: A Review,” *Symmetry*, vol. 10, no. 10, Art. no. 450, 2018.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[5] E. Rohmer, S. P. N. Singh, and M. Freese, “V-REP: A versatile and scalable robot simulation framework,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1321–1326, 2013.

[6] J. C. Montesdeoca Contreras, D. Herrera, J. M. Toibero, and R. Carelli, “Controllers design for differential drive mobile robots based on extended kinematic modeling,” in *Proc. IEEE European Conference on Mobile Robots (ECMR)*, 2017.