An Approach for Dynamic Obstacle Avoidance in Autonomous Mobile Robots Operating in Unstructured Indoor Environments

Ngoc Tien Tran
School of Mechanical and Automotive Engineering, Hanoi University of Industry, Vietnam
tientn@haui.edu.vn

Thanh-Lam Bui
School of Mechanical and Automotive Engineering, Hanoi University of Industry, Vietnam
buithanhlam@haui.edu.vn

Van-Long Trinh
School of Mechanical and Automotive Engineering, Hanoi University of Industry, Vietnam
longtv@haui.edu.vn (corresponding author)

Received: 17 March 2025 | Revised: 14 April 2025 | Accepted: 19 April 2025

 DOI: https://doi.org/10.48084/etasr.10999

ABSTRACT

This paper deals with mobile robot navigation and obstacle avoidance, presenting a probabilistic search algorithm, Rapidly exploring Random Tree (RRT) method, to ensure stability and flexibility of robots in the face of unexpected events. To optimize the map updating process, the improved RRT automatically builds the map by combining global and local paths. The path is optimized using the Dijkstra algorithm to increase the real-time performance. As the robot moves, the map is continuously updated to detect dynamic obstacles. When an obstacle is detected, a new optimal path is generated to guide the robot to the goal. Experiments on the robot operating system have shown that the optimization works and the robot can automatically avoid static and dynamic obstacles in the simulated environment, map quickly, and avoid small spaces. The results of the study can be extended to movements in unstructured environments and can be used at boundary nodes.

Keywords-mobile robot; dynamic obstacles; unstructured indoor environments; RRT algorithm

I. INTRODUCTION

A mobile robot is a system capable of autonomous movement, using wheels or legs on terrain characteristics to perform various tasks in unstructured environments [1]. Due to their mobility and agility, mobile robots have become an integral part of numerous fields, such as industrial production, healthcare, military, and civilian applications for human needs and practical requirements [2]. These robots have the ability to be accurately and efficiently guided through uncertain and complex environments. The mobile navigation robot has witnessed significant progress through various innovative methods, such as proposing a navigation system that integrates global planner, Visual Simultaneous Localization and Mapping (V-SLAM) and local planner Deep Reinforcement Learning (DRL), categorizing reinforcement learning algorithms into value-based, policy-based, and hybrid approaches [3]. Their work highlights the strengths, limitations, and potential real-world applications of these methods, providing insights to address scalability and reliability challenges in robot control systems. Authors in [4] introduced a mapping method that creates volumetric topology maps from 3D point clouds, structuring environments into distinct hierarchical levels, regions, and volumes. This approach significantly improves robot interpretation and navigation in complex multi-level environments, such as buildings with irregular structures. Authors in [5] developed a motion tracking controller tailored for non-standalone robotic systems, while authors in [6] proposed a navigation model that combines supervised learning with reinforcement-based methods. When combined, these advances represent significant progress toward stable, reliable, and adaptive robotic navigation suitable for diverse real-world environments. Recently, researchers have made significant progress in the development of mobile robots using various techniques, such as Simultaneous Localization and Mapping (SLAM) [7], path planning [8], and motion control [9-11]. In particular, RRT algorithms have gained popularity among the path planning algorithms for their ability to find optimal ways for mobile robots. Authors in [12] extended the standard RRT to improve its efficiency and adaptability in dynamic

environments, ensuring real-time obstacle avoidance and optimal path generation. Through simulations, the study demonstrates the effectiveness of the enhanced RRT algorithm in navigating complex, changing environments. Authors in [13] showed that the RRT-RT algorithm can efficiently explore complex terrain regions. Authors in [14] introduced an improved RRT algorithm to reduce computation time in complex environments. By using grid map skeletonization to generate an initial solution, the method achieves faster and more robust performance compared to traditional random sampling techniques. The simulation results show that this approach not only reduces computation time, but also reduces variability, leading to more efficient path planning for mobile robots. Authors in [15] introduced an improved population initialization approach based on the Bidirectional Rapidly exploring Random Tree (Bi-RRT) algorithm to optimize both the path length and computation time. Authors in [16] developed an automatic path planning method for the robot controller using an improved RRT algorithm to increase planning efficiency. This improved algorithm addresses challenges, such as obstacle avoidance and computational efficiency, ensuring the generation of feasible and optimized paths for robot manipulators. The simulation results presented in the study demonstrate the effectiveness of the algorithm in navigating complex environments, highlighting its potential for practical industrial applications. Authors in [17] proposed the 1-0 Bias-goal Rapidly exploring Random Tree (Bg-RRT) algorithm, designed to reduce computational complexity and planning time compared to traditional RRT and Bg-RRT methods. The simulation results demonstrate that this algorithm effectively reduces both the computation time and generated path length, especially in complex scenarios. Authors in [18] proposed the Smoothly RRT (S-RRT) algorithm to increase the sampling speed and plan automatic obstacle avoidance paths for the robot. The present paper presents a differential robot model driven by two separate wheels and introduces an improved RRT algorithm that uses both local and global RRT techniques to guide the robot in steering away from static and dynamic obstacles. In addition, it determines the optimal scanning angle for the laser to create a map of the environment, applying Dijkstra's shortest path algorithm on this map to guide the robot. Combining these algorithms with the primary structure of the RRT algorithm enabled guiding the robot into different scenarios and avoiding obstacles.

II. MATERIALS AND METHODS

As shown in Figure 1, the nonholonomic mobile robot under consideration possesses two independently controlled wheels. The mathematical relationship between the robot's drive control values and pose is:

$$\begin{cases} \boldsymbol{p} = [x_Q \quad y_Q \quad \phi]^T \\ \dot{\boldsymbol{p}} = [\dot{x}_Q \quad \dot{y}_Q \quad \dot{\phi}]^T \end{cases} \tag{1}$$

where (x_Q, y_Q) are the robot's position coordinates and ϕ is its orientation angle. The right wheel's linear velocity (v_r) and the left wheel's linear velocity (v_l) are:

$$\begin{cases} v_r = r\dot{\theta}_r = v_Q + a\dot{\phi} \\ v_l = r\dot{\theta}_l = v_Q - a\dot{\phi} \end{cases} \tag{2}$$

where v_Q is the robot's linear velocity, r and $2a$ are the radius and distance between the two driving wheels, respectively, and $\dot{\theta}_r$ and $\dot{\theta}_l$ are the left and right wheels' angular velocities, respectively.

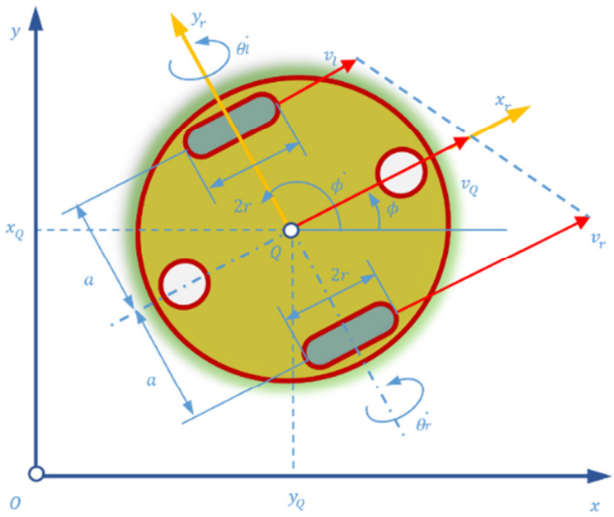

Fig. 1. Kinematic parameters of the robot.

The kinematic model of the robot can be described as:

$$\dot{\boldsymbol{p}} = \boldsymbol{J}\dot{\boldsymbol{q}} \leftrightarrow \begin{bmatrix} \dot{x}_Q \\ \dot{y}_Q \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}cos\phi & \frac{r}{2}cos\phi \\ \frac{r}{2}sin\phi & \frac{r}{2}sin\phi \\ \frac{r}{2a} & \frac{r}{2a} \end{bmatrix} \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{bmatrix} \tag{3}$$

where $\dot{\boldsymbol{q}}$ are the angular velocity vectors of the wheels and $\boldsymbol{J}$ is the robot's Jacobian. The inverse kinematic equation may be obtained by applying (3):

$$\dot{\boldsymbol{q}} = \tilde{\boldsymbol{J}}\dot{\boldsymbol{p}} = (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\dot{\boldsymbol{p}} \leftrightarrow \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{bmatrix} =$$

$$\frac{1}{r}\begin{bmatrix} cos\phi & sin\phi & a \\ cos\phi & sin\phi & -a \end{bmatrix} \begin{bmatrix} \dot{x}_Q \\ \dot{y}_Q \\ \dot{\phi} \end{bmatrix} \tag{4}$$

where the generalized inverse of $\boldsymbol{J}$ is denoted as $\tilde{\boldsymbol{J}}$. The research then advances by formulating a methodology for mapping and path planning for the robot, predicated on the previous equation. In order to create a simulated environment for the robot to operate in, a map was designed, including the following features: walls, paths, static obstacles and dynamic obstacles. The dimensions of the environment were 22 m x 22 m, designed in the gazebo environment, as displayed in Figure 2. In this environment, the presence of instantaneous obstructions, such as humans, is taken into consideration. The map learning strategy employed by the robot is based on the 5-point rule, which uses the initial four points to determine a quadrilateral area corresponding to the robot's scanning range. Upon completion of the scanning process, the robot will reach the final point on the map. In order to ensure that the robot is capable of traversing the entire region, the initial four points on

the map are situated at the robot's corners. As presented in Figure 3, the 5-point rule is based on five distinct points.

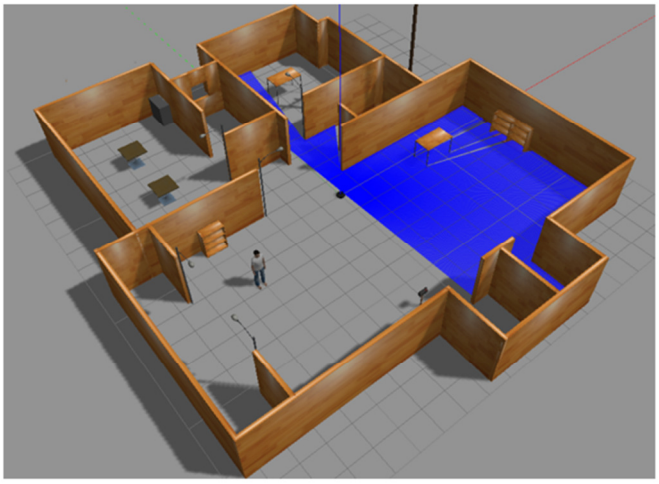

Fig. 2. Map building with mobile robot in dynamic environment.

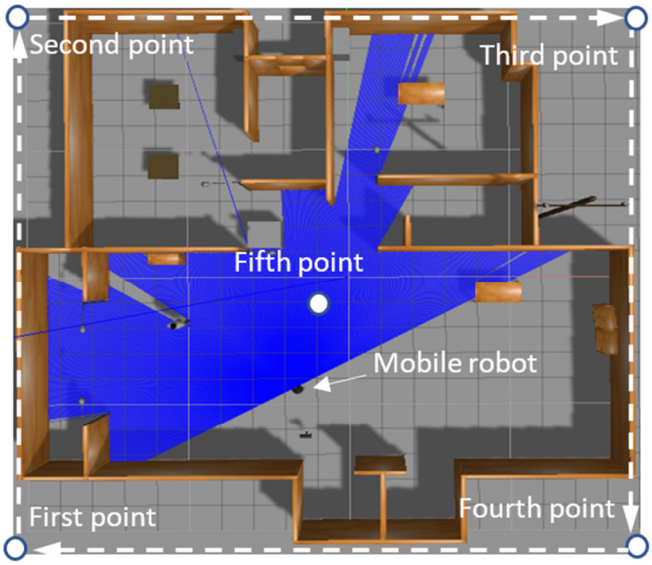

Fig. 3. The 5-point rule in learning maps of robots.

It is important to consider that the configuration space is $C \subseteq \mathbb{R}^d$ ($C \in \mathbb{N}, d \geq 2$) and C_{obs} is the obstacle region. A general setting, denoted as τ and belonging to C, represents the state of the robot, with the configuration being $Robot(\tau)$. Thus, this can be expressed as:

$$\begin{cases} C_{free} = \{\tau \in C | Robot(\tau) \cap \} C_{obs} = \emptyset \\ C_{obs} = C \backslash C_{free} \end{cases} \quad (5)$$

where C_{free} is the obstacle-free space, and C_{obs} is the obstacle region. The robot starts from an initial point v_{init}, which is an element of C_{free}. It moves along a path $\sigma: [0,1] \rightarrow \mathbb{R}^d$ in C_{free}, which is a continuous path, towards the goal region V_{goal} (V_{goal} is an open subset of C_{free}). Therefore, a path planning is defined by a triplet $\{C_{free}, v_{init}, V_{goal}\}$. The path σ is called collision-free path $\sigma(\tau) \in C_{free}, \forall \tau \in [0,1]$. The path σ is considered feasible if $\sigma(0) = v_{init}$ and $\sigma(1) = V_{goal}$. Let Σ_{free} be the set of all collision-free paths when $c: \Sigma_{free} \rightarrow \mathbb{R}^{\geq 0}$ is a cost function where each collision-free trajectory is mapped to a non-negative. To determine a feasible path $\hat{\sigma}$, a cost function c must be chosen where $c(\hat{\sigma}) = min\{c(\sigma): \sigma\ is\ feasible\}$. In C_{free}, the robot moves from the initial point v_{init} to various destination points V_{goal}. The branches of RRT algorithm are generated by continuously adding random points v_{rand} ($\{C_{free}, v_{init}, V_{goal}\} \subseteq C_{free}$). Initially, the tree $T = (V, E)$ starts with $V = \{v_{init}\}$ and $E = \emptyset$ (E is the edge set of the two linked points, and V is the set of nodes). At each iteration, a random point v_{rand} is generated, and the nearest vertex to v_{rand} is selected from V, denoted as $v_{nearest} \in V$. The function $Nearest: (T(V, E), v^*) \longrightarrow \tilde{v} \in V$ returns the vertex in $V (v^* \in C_{free})$ that is closest to $\tilde{v}$:

$$Nearest: (T = (V, E), v^*) \coloneqq argmin_{\tilde{v} \in V} \|v^* - \tilde{v}\| \quad (6)$$

The $Steer(v_1, v_2)$ function takes two points $v_1, v_2 \in C$ and returns a point $\tilde{v}$:

$$\begin{aligned} & f = \|\tilde{v} - v_2\| \rightarrow min \\ & s.t.: 0 < \|\tilde{v} - v_1\| \leq \eta \end{aligned} \quad (7)$$

where η is the tree growth rate. A significant distinction from the standard RRT approach is the frequent resetting of the tree during exploration, as presented in algorithms 1 and 2. The employment of the local RRT has been demonstrated to result in a substantial prolongation of map-learning time, given the algorithm's restart after each cut-off point. Furthermore, the robot's rapid tree growth may result in the oversight of specific map edges, as depicted in Figure 4. The global RRT model is employed to solve this problem. As with the local RRT method, the global RRT algorithm does not reset the tree to its initial position based on the robot's location at the conclusion of each iteration. Instead, it undergoes continuous expansion as it is explored.

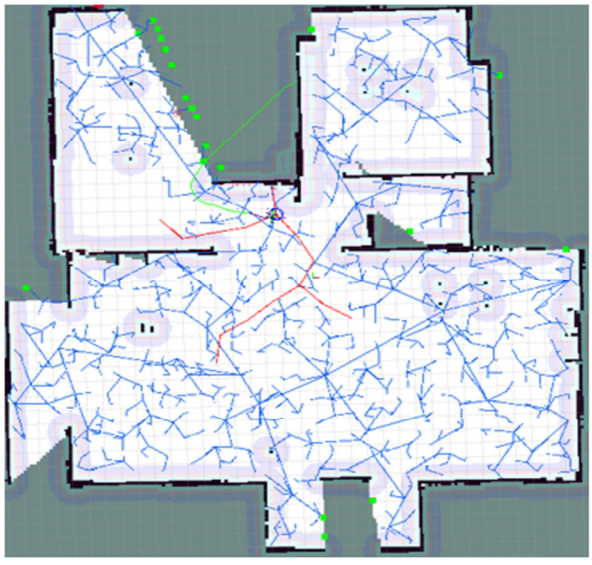

Fig. 4. Robot mapping strategy with local RRT and global RRT.

```
Algorithm 1 – Local RRT
1 V1:{z_init}; E ← ø
2 while True do
3      v_rand ← Free
4      v_nearest ← Nearest (T (V, E), v_rand)
5      v_new ← Steer (v_nearest, v_rand)
6      if ObsCheck (v_nearest, v_rand) = –1 then
7            PublishPoint (v_new)
8            V ← {v_new}; N ← ø
9      else if ObsCheck (v_nearest, v_rand) = 1
then
10           V ← V U{v_new}
             E ← E U{(v_nearest, v_rand)}
11     end if
12 end while
```

The RRT is designed to detect position limitations in regions distant from the robot as well as throughout the map. Algorithm 2 presents the global RRT algorithm. Conversely, as the tree's size increases, its growth rate tends to decrease. Consequently, the limit point detection process is slower than the growth tree in Algorithm 1. Therefore, the two algorithms complement each other, improving map learning time and avoiding missing edges in the map.

```
Algorithm 2 – Global RRT
1 V1:{v_init}; E ← ø
2 while True do
3      v_rand ← Free
4      v_nearest ← Nearest (T (V, E), v_rand)
5      v_new ← Steer (v_nearest, v_rand)
6      if ObsCheck (v_nearest, v_rand) = –1 then
7            PublishPoint (v_new)
8      else if ObsCheck (v_nearest, v_rand) = 1
then
9            V ← V U{v_new}
E ← E U{(v_nearest, v_rand)}
10     end if
11 end while
```

The robot is capable of recognizing objects and landmarks, while its current location can be calculated based on its position, whether in a local, global, or motion-based context. In this study, the laser sensor's scanning angle parameter is set at 180 degrees. The PC used for the scanning procedure is equipped with characteristics of Gazebo version 7.0.0, an i7-9700K CPU, 32GB of RAM, and ROS Kinetic running Ubuntu 16.04.3. The robot's physical specifications include a weight of 3 kg, a base radius of 0.178 m, a height of 0.23 m, and a driving wheel radius of 0.035 m. As illustrated in Figure 5, the experiment yielded the following: Figure 5(d) presents a stable environmental map that was generated at a scanning angle of 180. While the scanning angle has been demonstrated to influence map quality, this section examines the impact of RRT tree growth rate, controlled by parameter η (7). The global edge detection method demonstrates optimal efficiency at higher values of η (η=10), while the local method exhibits optimal performance at moderate values of η (η=6). While a faster growth rate reduces the time required for initial scanning, it increases the probability of missed edges, necessitating additional scans.

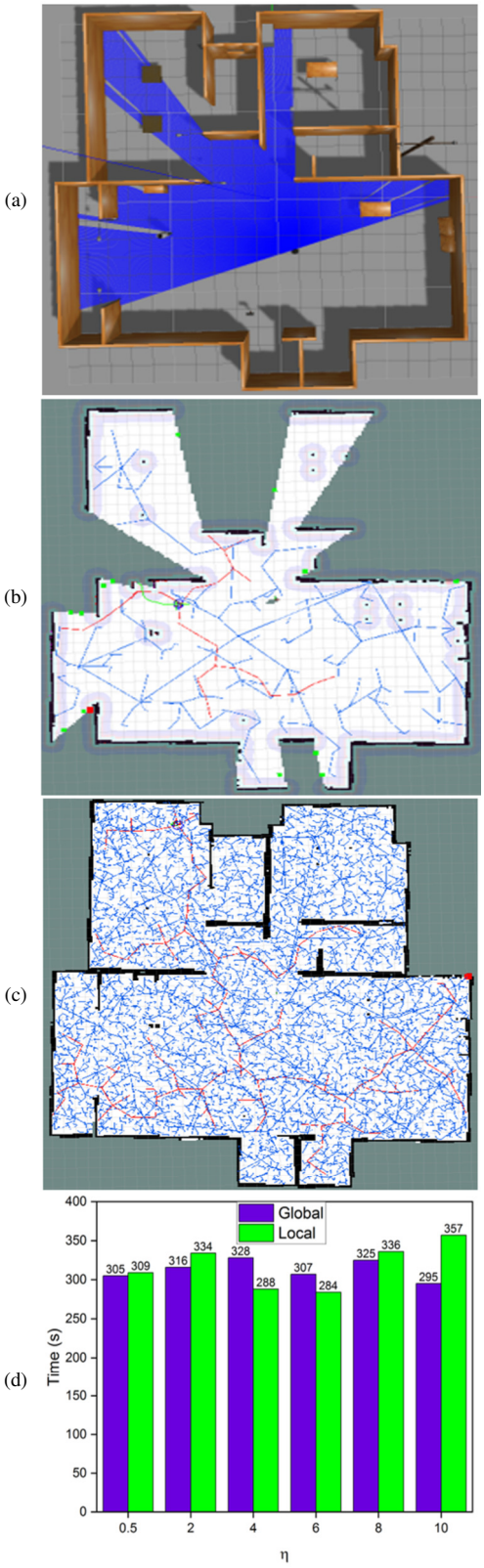

Fig. 5. Results of mapping investigation according to laser scanning angle: (a) scanning angle in the gazebo environment, (b) map of the robot's scanning process, (c) map results, (d) mapping time when changing tree growth rate.

In scenarios characterized by narrow angles, it is proposed to opt for a higher η when implementing global edge detection. This approach is intended to minimize the occurrence of missed edges and to reduce the duration of mapping. These insights underscore the necessity of meticulously calibrating the parameter η to achieve a harmonious balance between the velocity and thoroughness of the mapping process.

III. RESULTS AND DISCUSSION

This section presents a path planning scenario in which the robot is initially positioned at a designated starting point and aims to reach a specified target point. The scenario is set within a map that includes obstacles that the robot must navigate to successfully complete the task. Two path planning algorithms are used: The RRT algorithm is applied to identify exploratory targets, specifically points that have not yet been explored. The Dijkstra algorithm is used as a path planner to ascertain the shortest path between the robot's current position and the target destination. Dijkstra model begins at a node x_{start}. Let Q be a set of unvisited nodes and $Path$ be a set of paths. Initially, set: $Q \leftarrow \emptyset$, $Path \leftarrow \emptyset$. The cost of each node is denoted as $C(x)$, which is the sum of distances from the node to the source. Thus, initially, $C(x) \leftarrow \infty$ and $C(x_{start}) \leftarrow 0$. Randomly initialize nodes and evaluate their costs from those nodes to x_{start}. Upon receiving the node with the smallest cost, add it to the path: $u \leftarrow x \subseteq Q\ s.t.\ C(x) = C_{min}$. Other nodes are removed from the path $Q \leftarrow (Q \backslash u)$, but their distance data are retained. Subsequently, the cost from each following node is accumulated from previously determined nodes: $cost \leftarrow C(u) + C(u, x_n)$. Whenever a node generates the smallest cost along any path, the cost is updated and the node is added to the path: $C(x_n) \leftarrow cost$, $Path(x_n) \leftarrow u$.

```
Algorithm 3 – Dijkstra algorithm
1  Q← ∅; Path ← ∅
2  for x ⊆ V do
3       C(x) ← ∞
4       Q ← Q ∪ {x}
5  end for
6  C (x_start) ←0
7  while Q ≠ ∅ do
8       u ← x ⊆ Q s.t. C(x) = C_min
9       Q ← Q\u
10      for each x_n ⊆ Neighbor(Q,u) do
11      cost ← C(u) + C(u, x_n)
12      if cost < C(x_n) then
13      C(x_n) ← cost
14      Path (x_n) ← u
15      end if
16      end for
17 end while
18 return Path, C
```

The present study presents three scenarios for robot navigation: Scenario 1 involves the robot moving on a map without any obstacles. In Figure 6 (a), the robot's trajectory is displayed, traversing an unobstructed course from its initial position to its designated goal location.

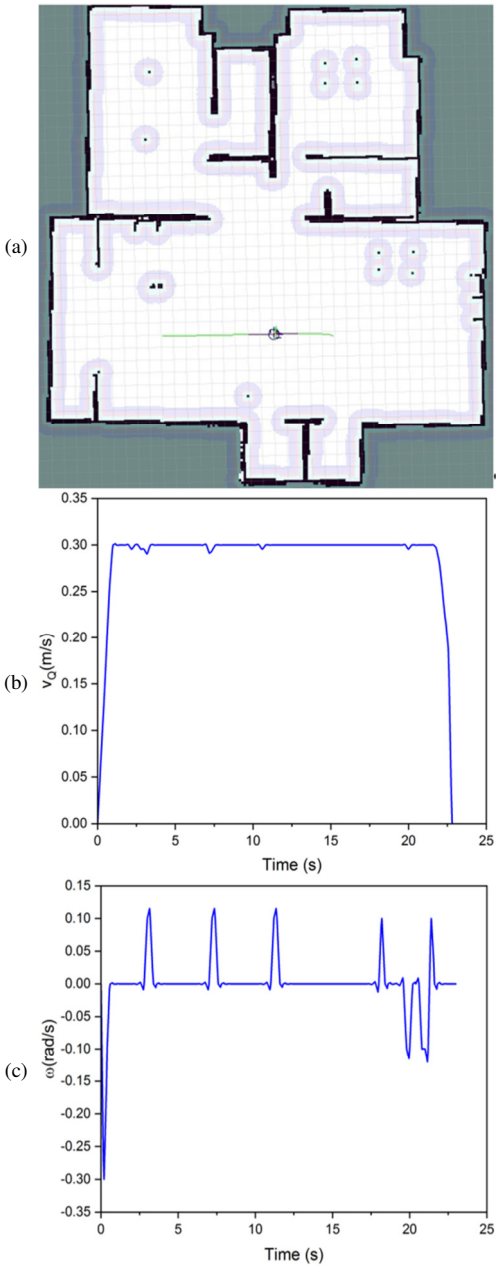

Fig. 6. (a) Motion of the robot on an obstacle-free trajectory, (b) linear velocity of robot, (c) angular velocity of robot.

The robot moves along a nearly straight path, maintaining stable linear and angular velocities, as shown in Figure 6 (b) and (c). This indicates that the algorithm effectively converges on the shortest possible route. In scenario 2, the robot adjusts its trajectory on two occasions, exhibiting angular velocity peaks of 0.3 rad/s and 0.75 rad/s, respectively. These adjustments correspond to right turns, a maneuver designed to avert collisions, as portrayed in Figure 7. In scenario 3, the robot swiftly adapts its trajectory upon the detection of an obstacle, closely following the boundaries of the obstacle and continuously adjusting its trajectory to ensure the shortest travel time, as presented in Figure 8. As expected, navigation in

dynamic environments slightly increases travel duration compared to obstacle-free scenarios (38.8 s vs. 36.4 s), thereby highlighting the importance of adaptive path planning, as illustrated in Figure 9. The robot's linear velocity, designated as $v_{(Q2)}$, undergoes two abrupt alterations at 15 s and 20.2 s, respectively. In the range of 10 s and 25 s, the robot's angular velocity undergoes continuous fluctuations. These fluctuations are the consequence of the robot's attempt to plan a new trajectory when it encounters instantaneous obstacles.

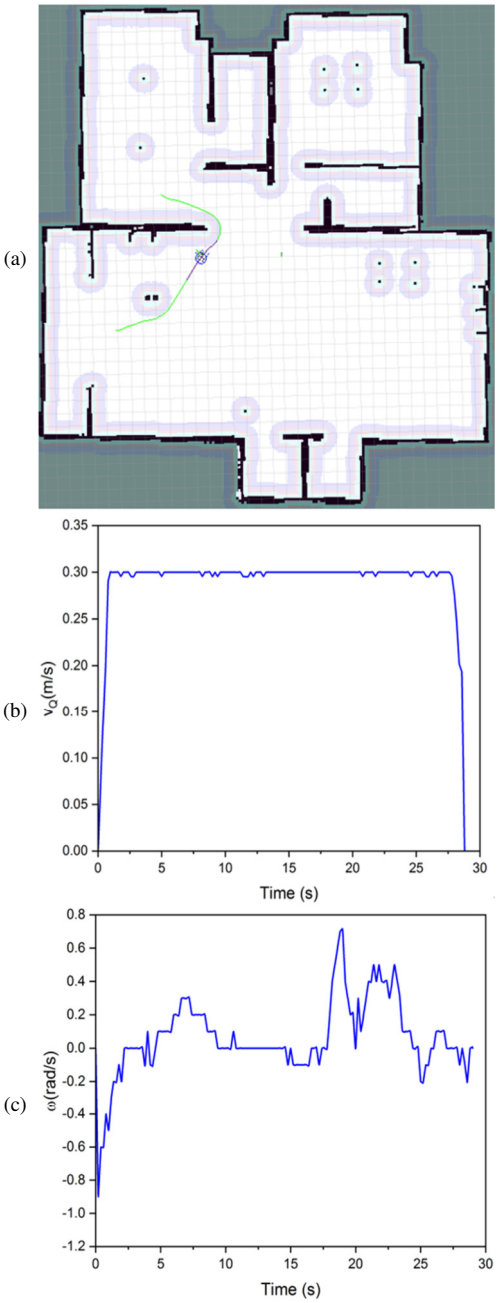

Fig. 7. (a) The motion of robot with static obstacles, (b) linear velocity, and (c) angular velocity.

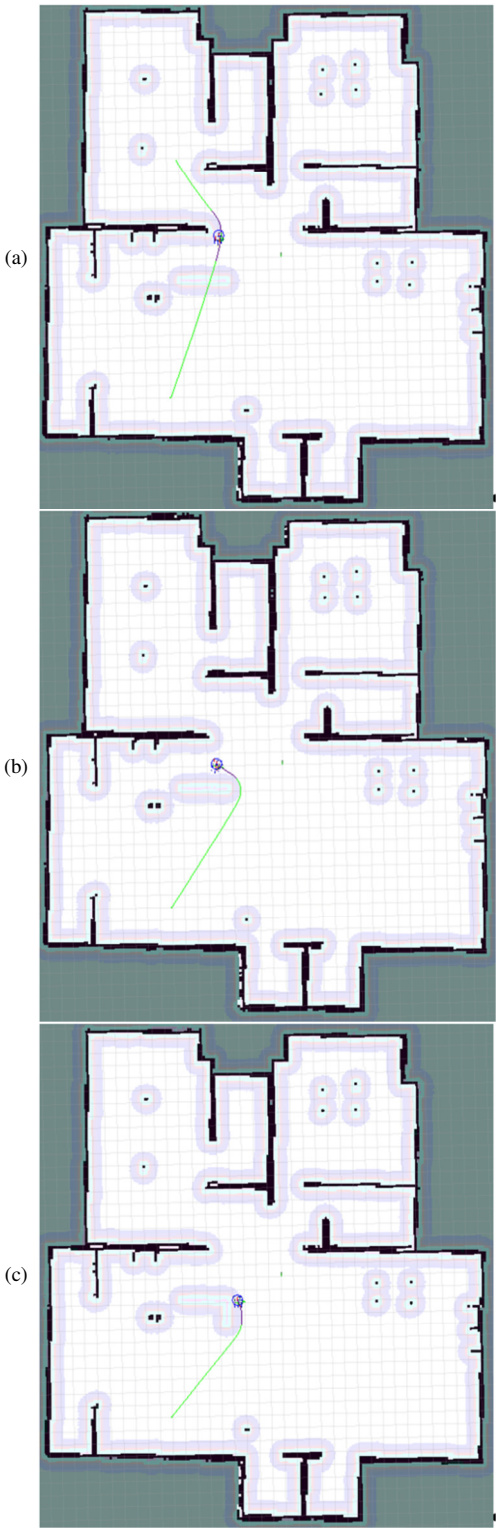

Fig. 8. Motion of robot with dynamic obstacle.

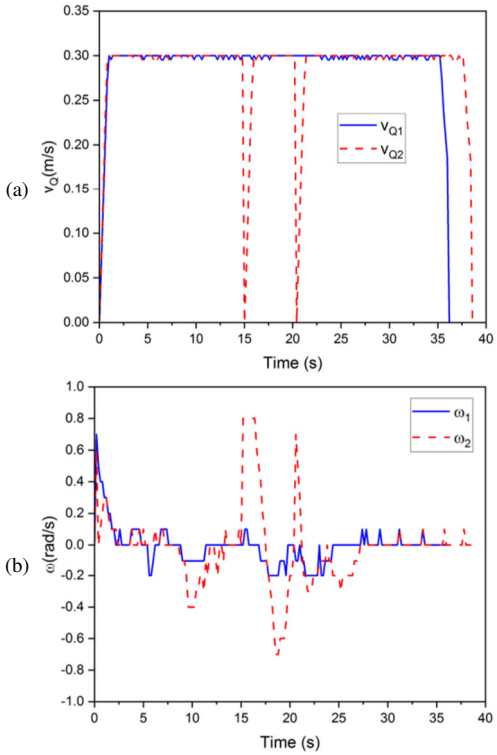

Fig. 9. Linear and angular velocity of the robot in motion with dynamic obstacles.

IV. CONCLUSIONS

In this study, the influence of the scanning angle and the growth rate of the Rapidly exploring Random Tree (RRT) on the performance of a mapping algorithm for mobile robots was examined. Through comparative evaluation, it was shown that these parameters significantly affect both mapping time and accuracy. In particular, the global edge detection method achieves superior performance at very small or large growth rates, while the local edge detection method is more effective at moderate growth rates. Furthermore, in narrow-angle environments, assigning a higher growth rate to the global method improves overall performance. The proposed algorithm exhibits an inherent preference for straight trajectories, but is able to dynamically adapt to avoid obstacles by maintaining linear velocity while varying angular velocity during complex maneuvers. These results highlight the importance of integrated trajectory planning and real-time obstacle detection to ensure safe and efficient robot navigation. This approach provides a more adaptive and efficient framework for map building and varying environmental conditions.

ACKNOWLEDGMENT

This work was sponsored by Hanoi University of Industry, Hanoi, Vietnam.

REFERENCES

[1] J. Liao, Z. Chen, and B. Yao, "Model-Based Coordinated Control of Four-Wheel Independently Driven Skid Steer Mobile Robot with Wheel–Ground Interaction and Wheel Dynamics," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1742–1752, Mar. 2019, https://doi.org/10.1109/TII.2018.2869573.

[2] Y. Saidi, A. Nemra, and M. Tadjine, "Robust mobile robot navigation using fuzzy type 2 with wheel slip dynamic modeling and parameters uncertainties," *International Journal of Modelling and Simulation*, vol. 40, no. 6, pp. 397–420, Nov. 2020, https://doi.org/10.1080/02286203.2019.1646480.

[3] W. Xue, P. Liu, R. Miao, Z. Gong, F. Wen, and R. Ying, "Navigation system with SLAM-based trajectory topological map and reinforcement learning-based local planner," *Advanced Robotics*, vol. 35, no. 15, pp. 939–960, Aug. 2021.

[4] Z. He, H. Sun, J. Hou, Y. Ha, and S. Schwertfeger, "Hierarchical Topometric Representation of 3D Robotic Maps," *Autonomous Robots*, vol. 45, no. 5, pp. 755–771, Jun. 2021, https://doi.org/10.1007/s10514-021-09991-8.

[5] J. Fu, F. Tian, T. Chai, Y. Jing, Z. Li, and C.-Y. Su, "Motion Tracking Control Design for a Class of Nonholonomic Mobile Robot Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 6, pp. 2150–2156, Jun. 2020, https://doi.org/10.1109/TSMC.2018.2804948.

[6] D. Wang, Y. Hu, and T. Ma, "Mobile robot navigation with the combination of supervised learning in cerebellum and reward-based learning in basal ganglia," *Cognitive Systems Research*, vol. 59, pp. 1–14, Jan. 2020, https://doi.org/10.1016/j.cogsys.2019.09.006.

[7] T. Du, Y. H. Zeng, J. Yang, C. Z. Tian, and P. F. Bai, "Multi-sensor fusion SLAM approach for the mobile robot with a bio-inspired polarised skylight sensor," *IET Radar, Sonar & Navigation*, vol. 14, no. 12, pp. 1950–1957, 2020, https://doi.org/10.1049/iet-rsn.2020.0260.

[8] P.-C. Song, J.-S. Pan, and S.-C. Chu, "A parallel compact cuckoo search algorithm for three-dimensional path planning," *Applied Soft Computing*, vol. 94, Sep. 2020, Art. no. 106443, https://doi.org/10.1016/j.asoc.2020.106443.

[9] C. Ren, X. Li, X. Yang, and S. Ma, "Extended State Observer-Based Sliding Mode Control of an Omnidirectional Mobile Robot With Friction Compensation," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, pp. 9480–9489, Dec. 2019, https://doi.org/10.1109/TIE.2019.2892678.

[10] B. Kasmi and A. Hassam, "Comparative Study between Fuzzy Logic and Interval Type-2 Fuzzy Logic Controllers for the Trajectory Planning of a Mobile Robot," *Engineering, Technology & Applied Science Research*, vol. 11, no. 2, pp. 7011–7017, Apr. 2021, https://doi.org/10.48084/etasr.4031.

[11] H. Medjoubi, A. Yassine, and H. Abdelouahab, "Design and Study of an Adaptive Fuzzy Logic-Based Controller for Wheeled Mobile Robots Implemented in the Leader-Follower Formation Approach," *Engineering, Technology & Applied Science Research*, vol. 11, no. 2, pp. 6935–6942, Apr. 2021, https://doi.org/10.48084/etasr.3950.

[12] R. Seif and M. A. Oskoei, "Mobile Robot Path Planning by RRT* in Dynamic Environments," *International Journal of Intelligent Systems and Applications*, vol. 7, no. 5, p. 24, Apr. 2015.

[13] A. Tahirovic and G. Magnani, "A Roughness-based RRT for Mobile Robot Navigation Planning," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 5944–5949, Jan. 2011, https://doi.org/10.3182/20110828-6-IT-1002.03351.

[14] H. Ryu and Y. Park, "Improved Informed RRT* Using Gridmap Skeletonization for Mobile Robot Path Planning," *International Journal of Precision Engineering and Manufacturing*, vol. 20, no. 11, pp. 2033–2039, Nov. 2019, https://doi.org/10.1007/s12541-019-00224-8.

[15] D. Lin, B. Shen, Y. Liu, F. E. Alsaadi, and A. Alsaedi, "Genetic algorithm-based compliant robot path planning: an improved Bi-RRT-based initialization method," *Assembly Automation*, vol. 37, no. 3, pp. 261–270, Aug. 2017, https://doi.org/10.1108/AA-12-2016-173.

[16] H. Zhang, Y. Wang, J. Zheng, and J. Yu, "Path Planning of Industrial Robot Based on Improved RRT Algorithm in Complex Environments," *IEEE Access*, vol. 6, pp. 53296–53306, 2018, https://doi.org/10.1109/ACCESS.2018.2871222.

[17] Y. Gan, B. Zhang, C. Ke, X. Zhu, W. He, and T. Ihara, "Research on Robot Motion Planning Based on RRT Algorithm with Nonholonomic Constraints," *Neural Processing Letters*, vol. 53, no. 4, pp. 3011–3029, Aug. 2021, https://doi.org/10.1007/s11063-021-10536-4.

[18] K. Wei and B. Ren, "A Method on Dynamic Path Planning for Robotic Manipulator Autonomous Obstacle Avoidance Based on an Improved RRT Algorithm," *Sensors*, vol. 18, no. 2, Feb. 2018, Art. no. 571, https://doi.org/10.3390/s18020571.