Techno-Science Research Journal 10(2) (2022) 57-62

Content list available at ITC

Techno-Science Research Journal

Journal Homepage: http://techno-srj.itc.edu.kh/

Techno-Science Research Journal

Integration of RRT* Path Planning with Trajectory Tracking for Wheeled Mobile Robot

Sotheara Oum[1*], Sarot Srang[1], Phayuth Yonrith[1]

[1] *Department of Industrial and Mechanical Engineering, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia*

Received: 11 August 2022; Accepted: 18 October 2022; Available online: December 2022

Abstract: *Mobile robots have been around since the late 1960s. This kind of technology has caught the attention of many researchers since then. In the last decade, the applications of mobile robots are being applied in various sectors unleashing more benefits of automation and robotics to be captured to their maximum potential by mankind. As the tasks are getting more complex, it requires the mobile robots to be more advanced and autonomous. That is when motion planning comes into play. There are many types of wheeled mobile robots, one of which is called differential-drive. Although this type of robot has the benefits of simplicity over other types, it also has one downside, as it involves non-holonomic constraints. The problem of non-holonomic wheeled mobile robots in terms of path planning and tracking control are the big challenges for autonomous robot researchers throughout these years. Many approaches have been proposed and confirmed to have their own advantages and disadvantages in certain circumstances. This paper presents an integrated path planning and trajectory tracking control method for wheeled mobile robots allowing the robots to find the lowest-cost path while avoiding obstacles within a short computational time and move towards their goal by combining the RRT* path planning with Backstepping control. The performance of this integrated model is validated and implemented onto a two-wheel mobile robot which is a non-holonomic system subject to known static environmental obstacles. Assumptions and details on the test result are also described.*

Keywords: Non-holonomic mobile robot; Path Planning; RRT* Algorithm; Trajectory Tracking; Backstepping Control

1. INTRODUCTION

At present, autonomous mobile robots are being widely used for many applications ranging from household chores, delivery, defense, and planetary exploration. This type of robot requires motion planning, which in this work, can be considered as one word that describes the combination of two tasks; path planning and trajectory tracking control.

For the robot to navigate from one point to any desired point autonomously, it is required one crucial element which is a path that connects from the robot's starting point to the desired goal point. A robot that has been implemented with a path planning algorithm has the ability to determine its path entirely by itself. A variety of path planning algorithms have been proposed and tested in various situations. Each of them has been to be effective in a certain circumstance.

When choosing the right path planning algorithm, there is no algorithm can be considered better than others in all circumstances. The best algorithm depends on which factors are given more focus on, such as the type of the robot and the computing constraints, namely static/holonomic and dynamic/non-holonomic systems.

In the real environment, most mobile robots have size constraints and therefore use small microcontrollers to execute programs. These microcontrollers have limited computational ability. Among all other algorithms, the RRT*, shorts for Rapidly-exploring Random Tree Star, is a simple and computationally cost-effective planning algorithm suitable for dynamic environments [1], although it also works competently in static environments. The RRT* algorithm also produces lower-cost paths over its predecessor, the RRT [1]. In addition, combining this algorithm with a tracking controller create a motion planning that can be applied to almost any wheeled system because of its ability to cope with non-holonomic constraints [2,3]. Differential-drive robot has constraint on its kinematics which is call non-holonomic constraint. The robot

[*] Corresponding author: Sotheara Oum
E-mail: sotheara_oum@gscitc.edu.kh; Tel: +855-11 639 625

can only move forward and backward, it cannot turn sideway. If the path has many (desired) turning points next to each other, the robot will not be able to reach all the desired points as it needs to move to some distance about a radius (ICC) to change its orientation.

To allow the robot to follow the path generated by the path planning algorithm, it is required a trajectory tracking control method. There are many control algorithms available. Backstepping control is a nonlinear control technique that can deal with non-holonomic constraints, minimizing the error between the actual trajectory and the desired to zero [4]. The Backstepping is chosen because this approach is recommended owing to the fact that its structure is straightforward, and it is appropriate for applications that have small tracking errors. It was determined that the strategy was successful in solving the trajectory tracking control issue of differential-drive wheeled mobile robot. And lastly, the integration of these 2 methods is also needed.

This paper demonstrates the integration of the RRT* algorithm with Backstepping control, where the coordinates of the nodes on the path generated by RRT* are then used as inputs into the trajectory tracking control model using Backstepping controller. Simulation in MATLAB Simulink of a differential-drive mobile robot, assuming that the robot is operating in a 2D static environment is performed and has been implemented onto real hardware testing.

2. METHODOLOGY

The integrated framework can be compactly described as follows. First, in order to generate a path, the RRT* generates random nodes in the configuration space and connects them to the initialized tree until the goal is reached. The configuration of the coordinate x and y of each waypoint on the generated path are then taken as the reference/desired position of the robot $q_d = [x_d\ y_d\ \theta_d]^T$, where θ_d is calculated by $\theta_d = atan2(\frac{y_{i+1}-y_i}{x_{i+1}-x_i})$. This 3xn matrix is then employed as the input to the trajectory generation function to calculate the velocities required for the tracking control model. The backstepping controller minimizes the error between the desired and actual robot velocities and sends commands to the robot model, thus the robot moves towards each desired point of the reference trajectory until it reaches the final point. Fig. 1 illustrates this system block diagram. Details on the RRT* algorithm, the kinematic model of the robot, and Backstepping control will be discussed in the sections below.

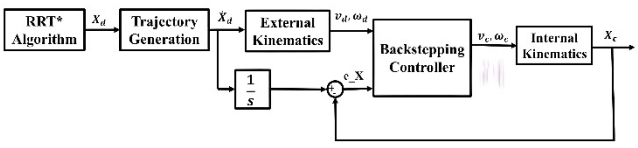

Fig. 1. System architecture

2.1. Path Planning: RRT*

RRT* is a random sampling-based planning algorithm. The two commonly used methods of planning are grid-based and sampling-based planning. A grid-based algorithm is particularly effective at solving low-dimensional topographical problems. The sampling-based algorithm, on the other hand, excels at solving high-dimensional problems.

The way this algorithm works can be divided into 4 main steps. Step 1: the tree is initialized at the start point/node q_{start} of the robot. Step 2: a random node q_{rand} is randomly generated inside the configuration space. Step 3: the algorithm tries to connect the nearest node $q_{nearest}$ on the tree to the new generated node while making sure there is no collision with obstacles as shown in Fig. 2. Step 4: after being connected to the tree, the new node q_{new} is being used as a substitute to the previous parent node(s) of the neighbor node(s) to compare their original cost to the new cost (when those nodes take the new node as their parent node). If the new cost is lower than the original cost, the neighbor node(s) will remove their branch(es) from the previous parent node(s) to connect with the new node instead and the tree will be re-shaped. Step 2, 3 and 4 will keep repeating until the goal is reached or the algorithm has generated random nodes as many as the number of nodes is set.

The process in step 3 and 4 improve the path quality. However, they can make convergence towards the goal slow as number of nodes in the tree increases [5]. In step 3, when a random vertex q_{rand} is generated, the RRT* will first tries to connect it with the tree by checking the distance of all nodes in the tree and select the one with the shortest distance to branch out (steer) towards q_{rand}. If the distance of q_{rand} is bigger than the stepsize EPS, a new node q_{new} will be added to the configuration space.

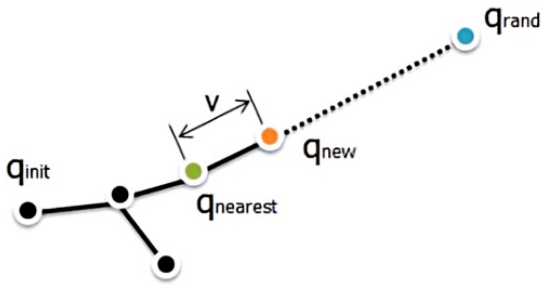

Fig. 2. Tree branching out towards q_{rand} [6]

After the q_{new} is added, the algorithm selects the best nearest neighbor as the parent for this new node. The process of choosing the parent node q_{parent} is as follows. First it considers all the nearby nodes q_{near} within a radius. The node with the lowest cost to reach the q_{new} will be selected as the parent as shown in Figs. 3a and b. This process in step 3 can be called finding parent, and the process in step 4 is called rewiring.

In step 4, after the q_{new} is added to the tree, the RRT* compares the cost of these q_{near} back to the start point with the new cost of the RRT* uses this new node added to the tree to reconnect the tree. If the initial cost of the nearby nodes within a radius is higher (from start node to them) compares to when they connect to the q_{new}, the RRT* removes those nodes from their parent nodes and select this new added node as their parent node instead. Hence, the tree is rewired (see Figs. 3c and d).

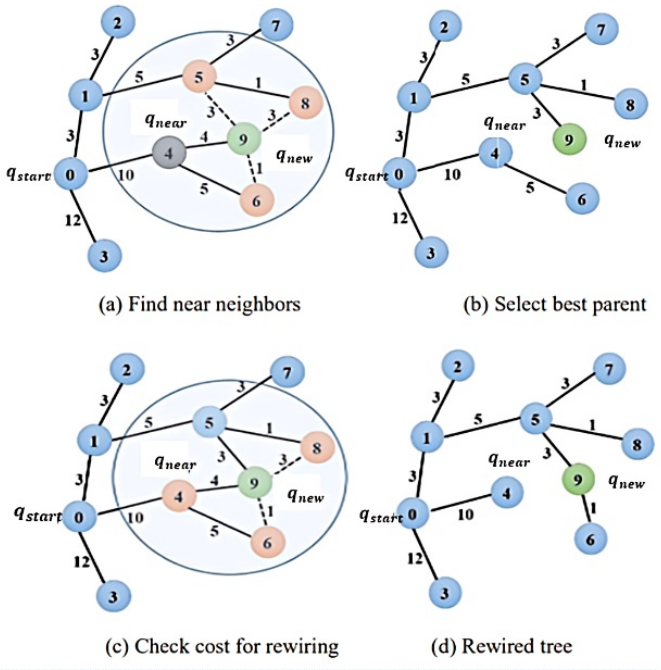

Fig. 3. New node connecting and rewiring process [5]

The cost of each node can be obtained by calculating the distance from each node to the start node, given the cost of start/initial node is 0 (zero). Knowing the position of each node, the distance can be calculated using Pythagorean theorem.

2.2. Robot Kinematic Model

The differential-drive robot is a type of mobile robot that is commonly used in both research and practical applications, although they have one disadvantage that involves non-holonomic constraints meaning that the controllable degree of freedom of their wheels is less than the total degree of freedom. In simple words, they are not able to move sideways like the mechanum or omnidirectional wheels mobile robots.

A robot has two frames; one is attached to its body which is called body frame and another one is the global coordinate frame as shown in fig. 4. The configuration of the robot in global frame is denoted by: $q = [x \quad y \quad \theta]^T$ which yields the velocity vector in global frame as: $\dot{q} = [\dot{x} \quad \dot{y} \quad \dot{\theta}]^T$.

The forward kinematic equation of the robot can be expressed as:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = J(q)V(t) = \begin{bmatrix}cos\theta & 0\\ sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \quad \text{(Eq. 1)}$$

where:

$v = \frac{v_r+v_l}{2}$ is linear velocity of the robot in robot frame (m/s)

$\omega = \frac{v_r-v_l}{L}$ is angular velocity of the robot in the robot frame (rad/s)

θ is the rotation angle of the robot frame with respect to global frame (rad)

J is Jacobian matrix transform

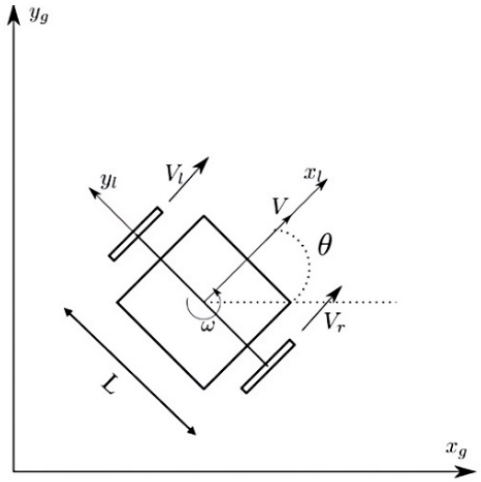

Fig. 4. Differential-drive wheel mobile robot in 2D coordinate

2.3. Trajectory Tracking

Backstepping controller reduces the error pose between the desired and robot trajectories to zero by taking the desired linear and angular velocities of the robot, and the pose error vector as the inputs to calculate the linear velocity v_c and the angular velocity ω_c required to tackle the error. The Backstepping control law [4] is given as:

$$\begin{bmatrix}v_c\\ \omega_c\end{bmatrix} = \begin{bmatrix}k_1 e_x + v_d cose_\theta\\ \omega_d + k_2 v_d e_y + k_3 v_d sine_\theta\end{bmatrix} \quad \text{(Eq. 2)}$$

where k is positive constant and e is error pose in global frame between desired and robot trajectory which can be calculated by:

$$\begin{bmatrix}e_x\\ e_y\\ e_\theta\end{bmatrix} = \begin{bmatrix}cos\theta & sin\theta & 0\\ -sin\theta & cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_d - x_c\\ y_d - y_c\\ \theta_d - \theta_c\end{bmatrix} \quad \text{(Eq. 3)}$$

where $[x_d \quad y_d \quad \theta_d]^T$ is desired pose in global frame, and $[x_c \quad y_c \quad \theta_c]^T$ is controlled pose in global frame.

3. RESULTS AND DISCUSSION

This integrated method has been validated through numerical simulation in MATLAB Simulink and also implemented in hardware testing using a differential-drive mobile robot.

Many test scenarios have been performed, one of which is when the robot is steering in a 6 $m \times 6$ m lobby as shown in Fig. 6 with initial configuration $q_{init} = [0 \quad 0 \quad 0]^T$ and the final goal $q_{end} = [4.8 \quad 4.2 \quad 0]^T$.

With 1200 number of nodes, eps size 0.2, the RRT* algorithm generates a path that avoid obstacles in the environment as shown in Fig. 5. The green rectangles area are obstacles that the robot cannot cross over. The red line is the path generated by RRT* algorithm.

Be noted that the path generated each time the model is executed is not always the same since the RRT* is a random sampling-based planning algorithm.

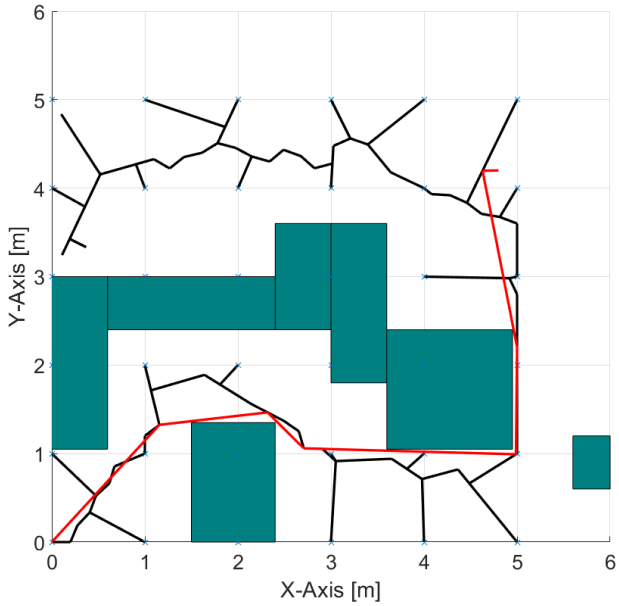

Fig. 5. Map and path generated in RRT*

The obstacles size and position in Figs. 5 and 6 are slightly different because the RRT* algorithm does not take robot size into consideration, meaning that it only focuses on the robot center, therefore, the obstacles size in the planning algorithm needs to be offset according to the real robot size to fit in the real practice.

The robot's maximum velocity due to hardware constraint is equal v = 0.88 m/s. Different velocity values have been tested with the robot. Fig. 7. shows the comparison between the desired trajectory and the controlled robot trajectory from the experiment with linear velocity v = 0.2 m/s and 0.5 m/s respectively.

Fig. 6. Real test setup

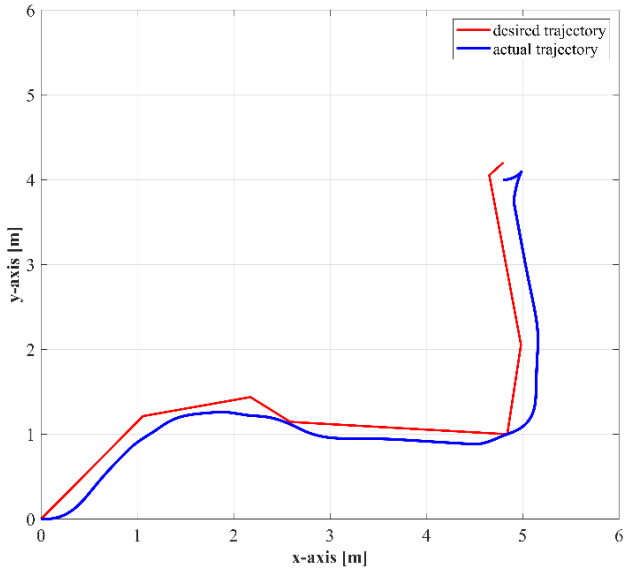

Fig. 7. Desired and actual trajectories of the robot (v = 0.2)

The results show that the shape of the desired and actual path is very similar although there is noticeable error in position. The two graphs in Fig. 9 illustrates the position error at each timestep. It can be seen that for v = 0.2 m/s, at the final goal, the error is 0.2 m in the y-axis and 0 m in the x-axis.

However, the error in actual robot position can significantly differ from the data captures by encoder when the velocities of the robot increase, which is caused by wheel slippery. Fig. 10 shows the actual position of the robot operating at velocity v = 0.2 m/s. The error in x-axis is 0.15 m and y-axis is 0.25 m which is slightly different from the error as shown in Fig. 9.

The actual velocity graphs are bumpy since they are calculated using numerical derivative. The actual velocity cannot track the desired velocity when it suddenly jumps due to the

switch from a current reached point to a next one. However, the convergence still can be achieved when the desired velocity becomes constant. This means that the control algorithm is robust although the tracking performance is not very accurate for higher maximum linear velocity

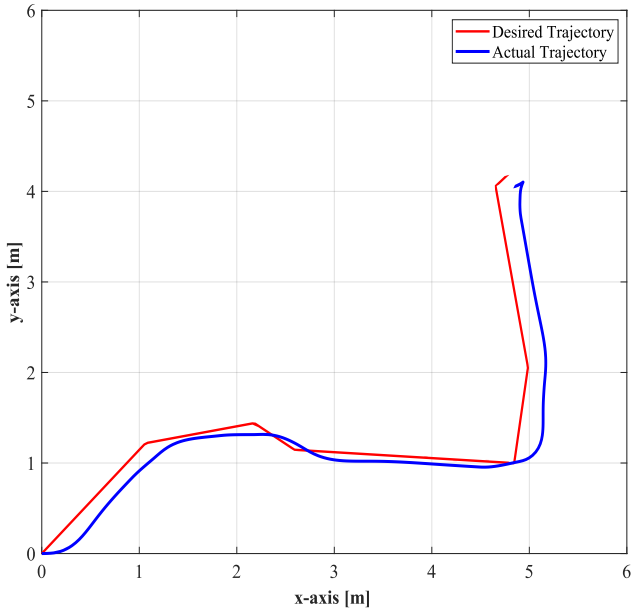

Fig. 8. Desired and actual trajectories (v = 0.5)

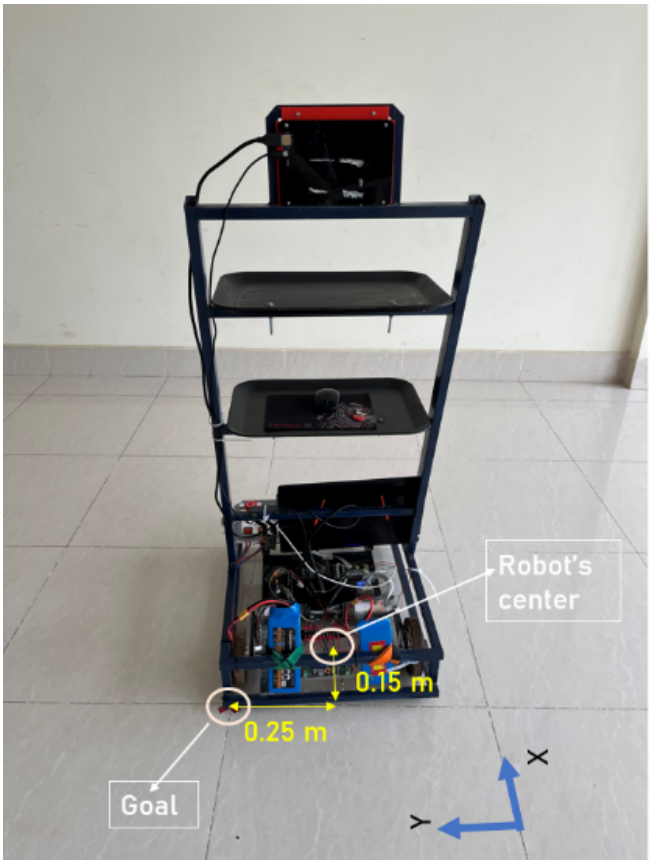

Fig. 10. Robot's actual position

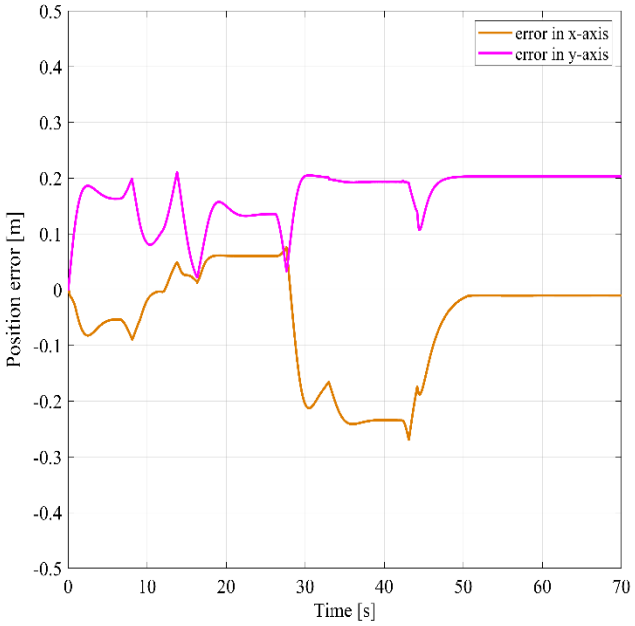

Fig. 9. Position error between desired and actual trajectory

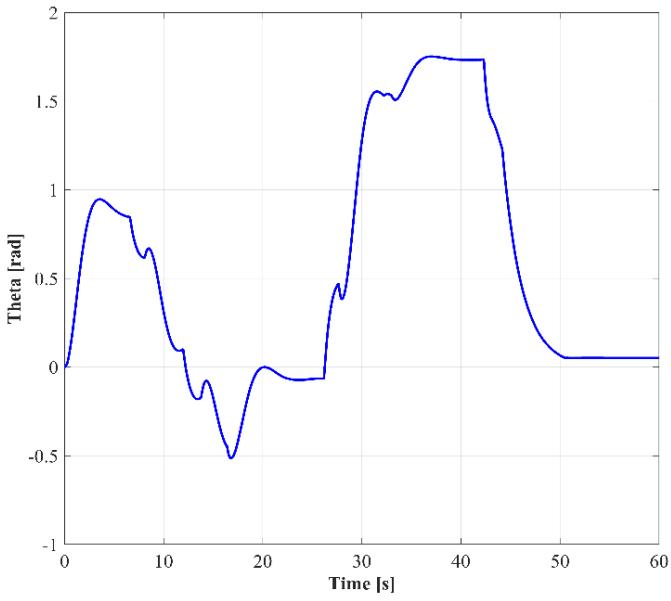

Fig. 11. Angle of rotation of the robot along the path

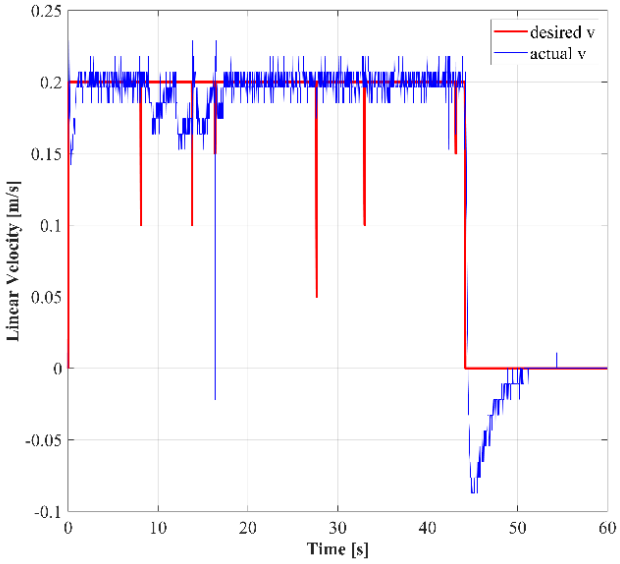

Fig. 12. Desired and actual linear velocity

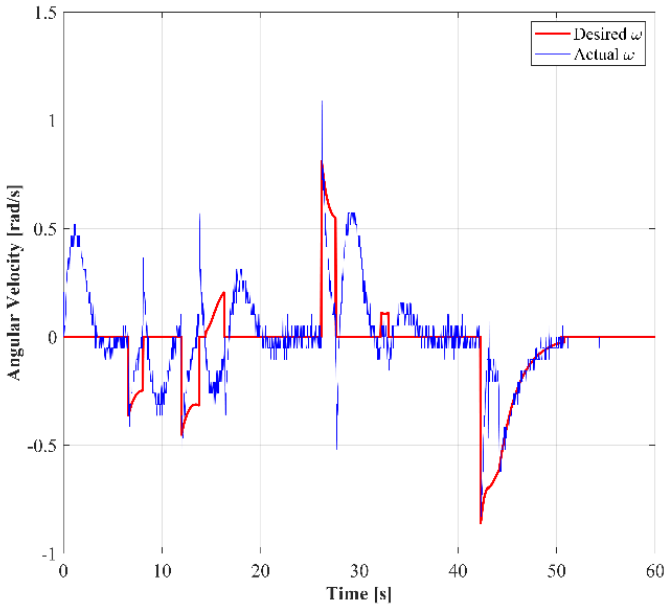

Fig. 13. Desired and actual angular velocity

4. CONCLUSION

In conclusion, the implementation of RRT* algorithm combining with Backstepping controller, a target tracking control technique, onto a non-holonomic mobile robot is presented. The results from many hardware testing show that this path planning technique works adequately with the Backstepping controller, allowing the robot to reach the goal point autonomously with position error increasing in accordance with higher value of desired maximum linear velocity. In addition, it also proved that this approach can be employed in development of service/delivery robot operating in a static environment. To further achieve a complete autonomous system in future work, mapping, localization and object detection should be included to allow the robot to generate a map of surrounding in each real-time execution.

ACKNOWLEDGMENTS

The authors would like to thank members and alumni of Dynamics and Control Laboratory who have been offering their kind help and support throughout this work.

REFERENCES

[1] Connell, D., & La, H. M. (2017, October). Dynamic path planning and replanning for mobile robots using RRT. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 1429-1434). IEEE.

[2] Li, J., Liu, S., Zhang, B. and Zhang, X. (2014). RRT-A Motion planning algorithm for non-holonomic mobile robot. In 2014 Proceedings of the SICE Annual Conference (SICE) (pp. 1833-1838).

[3] LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

[4] Zidani, G., Drid, S., Chrifi-Alaoui, L., Benmakhlouf, A., & Chaouch, S. (2015, April). Backstepping controller for a wheeled mobile robot. In *2015 4th International Conference on Systems and Control (ICSC)* (pp. 443-448). IEEE.

[5] Noreen, I., Khan, A., & Habib, Z. (2016). A comparison of RRT, RRT* and RRT*-smart path planning algorithms. *International Journal of Computer Science and Network Security (IJCSNS)*, *16*(10), 20.

[6] Jayasree, K. R., Jayasree, P. R., & Vivek, A. (2017, April). Dynamic target tracking using a four wheeled mobile robot with optimal path planning technique. In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)* (pp. 1-6). IEEE.