An Integrated approach for Path Planning for Mobile Robot Using Bi-RRT

To cite this article: Faiza Gul and Wan Rahiman 2019 IOP Conf. Ser.: Mater. Sci. Eng. 697 012022

View the article online for updates and enhancements.
An Integrated approach for Path Planning for Mobile Robot Using Bi-RRT

Faiza Gul\textsuperscript{1}, Wan Rahiman\textsuperscript{2}
\textsuperscript{1}School of Electrical and Electronic Engineering, Universiti Sains Malaysia 14300 Nibong Tebal, Pulau Penang, Malaysia,
\textsuperscript{2}Sustainable Platform Air, Land and Naval (SPALAN) Research Group, Universiti Sains Malaysia
\textsuperscript{a}E-mail: faizagul@student.usm.my

Abstract. In this paper, an integrated effort has been exploited for path planning along with path smoothness for a mobile robot. An extension of randomly exploring tree is used to plan the path for robot along with path length smoothness. Our proposed system influences the concept of Bi-directional Rapidly Exploring Random Trees (Bi-RRTs) for path planning together with path smoothness for minimizing the processing time to search for the path length. We compared our system with standard method in trials with the autonomous robotic platform. Produced results show that the presented approach as high robustness and use less computational effort.

1. Introduction

Path planning and obstacle avoidance for mobile robot navigation has gain a lot of importance in recent years. It is considered as complex task yet very important. Autonomous robots are well known for their intelligent behaviour and capabilities. They have numerous applications in the field of industry, medical, rescue operations and transportation \cite{1}. Due to their intelligent capabilities’ researchers are working to further increase their tendency to behave autonomous. The expectations are to develop real time monitoring system to avoid the collision, interpret information collected from environment in right way and performs smooth and uninterpreted navigation. The path planning involves the collision free path for vehicle/robot to navigate in environment freely. It involves the safe delivery from one point to another point decreasing the path cost. Depending on this nature path planning is mainly divided into static (obstacles do not change position w.r.t time) and dynamic (obstacles do change position w.r.t time) environment \cite{2}. Up till now numerous techniques have been proposed and applied to solve the path planning and obstacle avoidance of mobile robot. For example, road map approach \cite{3}, cell decomposition \cite{4} and APF \cite{5}. These methods are considered as inefficient because of high computational complexity and ability to easily stuck in local minima problem. To ensure the stability and good tracking trajectory, path planning utilize control system strategy to steer the autonomous mobile robot. A robot must exploit all the forces to make a turn. Like, for example if a robot is turning at an angle of 90 degree, all external forces will act while taking a turn. Robot needs to position the tires with robotic trajectory direction. This act is contrast with human driver who will exploit all forces on vehicle as it is heading towards the end of a turn. The main idea of this work is based on the utilization of conventional Randomly Exploring tree to Bi-directional RRT. To overcome the nonlinear mapping, an integrated path planning and control strategy is incorporated that leverage’s the Bi-RRT path planner together with pure pursuit controller. The utilization of Bi-RRT works efficiently for path planning and to keep a track of mobile robot, pure pursuit controller is used.

The paper is arranged as follows: In section II describes the proposed method of path planning and control method. Section III presents the modelling of mobile robot wheels. Section IV proves the effectiveness of proposed method through experimental results. In Section V, performance analysis is done with APF and PRM methods. Section VI contains the conclusion.
2. Methodology

Our integrated control and path planning methodology depends on two major factors i.e. a path planner using Bi-RRT and steering controller to keep track of desired trajectory. The method is used to help the robot to navigate towards goal position following the optimal path via avoiding obstacles in its way.

2.1 Bi-RRT based Path Planning Navigation

Firstly, RRT was first proposed by Steven M. LaValle and James J. Kuffner Jr. in [6] to handle problems with obstacles and differential constraints. The algorithm was designed to help in efficient data structure to search for high dimensional spaces by randomly building space filling tree. The tree grows in space by joining random points in free space. If the selected node/tree found colliding with obstacles it will not be selected as newly selected node of expanding tree. Since it is a stochastic method.

![Bi-Directional Random Exploring Tree](image)

**Figure 1.** Bi-Directional Random Exploring Tree

The main principle of Bi-RRT is shown in Fig.1. The summary can be made in few steps. The working principle of Bi-RRT is same as RRT the only difference comes in the space configuration is decomposed into two trees, as $T_a$ and $T_b$. Define the start point $X_{init}$ and end point $X_{target}$. Set $X_{init}$ as root of the tree. Select some random points in the space configuration $X_{rand}$. Expansion is done by expanding the nearest node in tree towards the random state by small segment of length $e$, where $X_{near}, X_{new}$ is a segment on the line joining the nearest point on tree. If no obstacles are found add the $X_{new}$ as a new point on tree. The algorithm will work until it reaches to target. The brief algorithm of Bi-RRT is described in Fig 1 and 2:

```
RRT_BIDIRECTIONAL($x_{init}$, $x_{goal}$)
1 $T_a$-init($x_{init}$); $T_b$-init($x_{goal}$);
2 for $k = 1$ to $K$ do
3 $x_{rand}$ ← RANDOM_STATE();
4 if not (EXTEND($T_a$, $x_{rand}$) = Trapped) then
5 if (EXTEND($T_b$, $x_{new}$) = Reached) then
6 Return PATH($T_a$, $T_b$);
7 SWAP($T_a$, $T_b$);
8 Return Failure
```

**Figure 2.** Algorithm for Bi-RRT

Several advantages and disadvantages of RRTs and Bi-RRT are discussed below:

[1.] A tree rapidly explores the whole space configuration, instead of circling around the start node.
[2.] It is easy to implement so lead to faster analysis to search for a path.
[3.] It can be used for solving different optimization problems.
[4.] It can be applied to Kino dynamic and nonholonomic planning.
[5.] It is considered as consistent due to its distribution of vertices approaches sampling distribution.
[6.] It is incorporated with different path planning systems, so it is considered as important module of path planning.

The major drawback of RRT and Bi-RRT algorithm are:
[1.] It is not deterministic and optimal.
[2.] It depends on the probability of picking up a point from the space which makes it harder to explore smaller areas and find path through them.
[3.] With numerous numbers of obstacles in space and choosing a point that does not lie on obstacle and then connect it to the tree is very small.
[4.] With sharp edged shaped obstacles path finding is not easy.

3. Path Tracking

After planning a path for robot path tracking is equally important for its motion control process. As the path planning helps in generating collision free path in space occupied by different obstacles, path process schedules the motion of mobile robot along a desired trajectory. Trajectory following is essential in motion control of mobile robot system. Therefore, a robust control law is needed to provide stabilization of an autonomous mobile robot.

Pure pursuit is a trajectory tracking algorithm for path planning purpose [7]. The pure pursuit algorithm enables a smoother steering control and enhance the robot ability to handle curved paths with lesser halts and short cuts. In this controller the vehicle moves along the circular trajectory when taking turns rather than following the straight lines, hence following up a circular arc is more natural than tracking a straight line. This could be analogizing with a car driver who follows a point further ahead and then steers smoothly toward that point therefore driving in a circular arc on a road.

![Figure 3: Pure Pursuit](image_url)

Pure pursuit algorithm is demonstrated in Fig 3, described in vehicle coordinate system, where z and y axis depict downward, and forward direction of vehicle and x axis shows a right handed coordinate.
system. The point \((x, y)\) is a look ahead distance from origin. The goal is to calculate and measure the curvature of the arc the joins the origin to point \((x, y)\) of length "l". In the figure, "D" is the distance between robot current position and target. \(\gamma\) is the curvature of the circular arc and \(\frac{1}{\gamma r}\) is defined as the radius of the circle that goes from the robot start position to goal position. The derivation of curvature and radius of the arc connecting to end point and start point of robot/vehicle can be made as follows:

If the point \((x_c, y_c)\) is on a look ahead distance \(D\) from the start position of robot/vehicle.

\[
x_c + y_c = D^2
\]
\[
\Delta x + d = r
\]
whereas, \(\Delta x\) is change in \(x\) of the goal to origin. The radius and curvature are derived as follows:

\[
d = r - x
\]
\[
(r - x)^2 + y^2 = r^2
\]
and by re-arrange Equation (4) as

\[
r^2 + 2rx + x^2 + y^2 = r^2
\]
\[
2rx = D^2
\]
and again, re-arrange Equation (6) as

\[
r = \frac{D^2}{2x}
\]
\[
\gamma r = \frac{2x}{D^2}
\]

Equation (1) represents the circle of radius \(D\) about robot start point and Equation represents the relationship between radius and vehicle/robot current point.

Pure pursuit algorithm work as follows:

1. Firstly, calculate the current location of the vehicle.
2. Calculating the point closest to the vehicle.
3. Find the look ahead distance (goal points) from the robot start position.
4. Convert goal points into vehicle coordinates system.
5. Calculate the curvature of the circular arc.
6. Determine the steering angle and vehicle position.

The look ahead distance is used as the main attribute of this controller because it can show how far along the pathway of a robot/vehicle can look from the start position of robot for calculating the angular velocity.
The effect of altering the lookahead parameter results in how a robot/vehicle tracks the trajectory. These parameters help in achieving two goals i.e. maintaining the path and retrieving the path. As shown in figure 4, smaller lookahead distance makes robot to follow the path quickly but it results in oscillation due to overshoot. But these oscillations can be decreased by applying large lookahead distance, but it can also cause large curvatures neat to end corners.

4. Kinematic Modelling of Differential Drive Robot

We have considered a differential drive robot with 2 wheels, and it kinematic model can be described with global coordinate system and body frame of mobile robot. The physical configuration of mobile robot is depicted in Fig 5. The theta represents the rotation about axis, the velocity of robot can be represented in form of vectors for forward and backward motion, and angular velocity represents the rotation of mobile robot. Linear and angular velocity of mobile robot can be defined as following:

\[ v = \frac{r_r}{2} w_r + \frac{r_l}{2} w_l \]  \hspace{1cm} (9)
\[ w = \frac{r_r}{d} w_r - \frac{r_l}{d} w_l \]  \hspace{1cm} (10)

where \( r_r \) and \( r_l \) represents the wheel radii of right and left wheels and \( d \) is the width. Kinematics model of mobile robot can be transferred into global coordinate system. The system is represented in Fig 6 with the motion of equation is given as follows:

\[ \dot{x} = v \cos \theta \]  \hspace{1cm} (11)
\[ \dot{y} = v \sin \theta \]  \hspace{1cm} (12)
\[ \dot{\theta} = w \]  \hspace{1cm} (13)

the position of robot can be found by integrating these equations in time differential from anytime time \( T_1 \) to \( T \). An offset can be added to give a kick start to robot if it does not start from origin.

\[ x(T) = \int_{T_1}^{T} v(t) \cos(\theta(t)) dt + C_0 \]  \hspace{1cm} (14)
\[ y(T) = \int_{T_1}^{T} v(t) \sin(\theta(t)) dt + B_0 \]  \hspace{1cm} (15)
\[ \theta(T) = \int_{T_1}^{T} w(t) dt + \theta \]  \hspace{1cm} (16)
5. Experimental Result

We used the path planner Bi-directional RRT to mobile robot in an unspecified area crowded with different shape obstacles. Five test runs were taken in completely different environment with unknown locations of obstacles. For testing the performance limits of proposed methods, in all navigation the environment and positions of obstacles were kept completely unknown. The start and target position were already set in advance for the robot. Dimensions of environment are 510*510 in all cases. Because it is Bi directional RRT the algorithm starts its search from both the ends of map (say it there are two trees which will start exploring the configuration space from both the ends of a map to reduce the processing time for search of shortest path as early as possible), the blue and red lines indicates the two trees and green line shows the intersection of two trees and indicates the end of search for algorithm. The increasing blue line indicates the Bi-RRT and obstacles are shown by black objects. The experimental results of simulation are shown in Fig 5,6,7:

![Figures](image)

**Figure 5.** Different setup of obstacle shapes (a) Test 1 with rounded obstacles, (b) Test 2 with rectangular block obstacles, (c) Test 3 with various shapes of obstacles

Figures on the left corners shows the projection of Bi-RRT into the environment along with the shortest path for the robot. The Bi-directional RRT expands from the opposite sides to explore the corners of a square. Since it is Bi-directional RRT, it joins to the goal node in few iterations. Different environmental maps were checked on algorithm. After running the same algorithm on different maps, it was observed that produced solutions were optimal, and near optimal. During the testing of the algorithm, it was observed Bi-RRT can find solutions in all kind of environment in a specified time. After the path is found, pure pursuit controller is applied to track the motion of robot/vehicle as the path curvature. During this time period, linear velocity is held constant and angular velocity changes. The spacing between the peaks in the depicts that the rotating wheels with larger spacing results in slower speed as compared to smaller spacing which results in faster speed. This means that the proposed method guides the robot/vehicle comfortably and smoothly through difficult turns to target. During conducting experiment, it was found that the robot reaches the target in a safest way. In real time we would want that robot reaches target as early as possible.
6. Performance Analysis and Conclusion

The proposed method has been tested on three different environments occupied with distinctively sized obstacles together with other algorithms such as APF, RRT and PRM. The analysis has been made based on Path length (total distance covered from start to end) and Processing Time (time taking by the algorithm for execution). MATLAB 2017b software is used to test the performance of algorithm. The obtained results are summarized as follows

Table 1. Experimental Results from Test 1, Test 2 and Test 3 according to Fig. 5

|       | Test 1               | Test 2               | Test 3               |
|-------|----------------------|----------------------|----------------------|
|       | Path length          | Processing Time      | Path length          | Processing Time      | Path length          | Processing Time      |
| Bi-RRT| 7.775438e+02         | 1.172861e+01         | 7.452761e+00         | 1.021773e+00         | 7.885617e+00         | 1.382988e+01         |
|       | 2                    | 1                    | 2                    | 1                    | 2                    | 1                    |
| APF   | 7.784429e+02         | 2.322934e+00         | 7.926919e+00         | 1.018852e+00         | 8.14402e+00          | 2.835975e+00         |
|       | 2                    | 0                    | 2                    | 1                    | 0                    | 0                    |
| RRT   | 7.83738e+02          | 5.101757e+00         | 8.39837e+00          | 3.514516e+00         | 8.565363e+00         | 1.723281e+00         |
|       | 0                    | 0                    | 2                    | 0                    | 2                    | 1                    |
| PRM   | 7.951940e02          | 7.685013e+00         | 7.573221e+00         | 3.543044e+00         | 7.936524e+00         | 4.308038e+00         |
|       | 0                    | 0                    | 2                    | 0                    | 2                    | 0                    |

Table 1 shows that Bi-RRT is the fastest of all the other algorithms. Though APF requires less computational time but it suffers from local minima whereas PRM may uses optimal path but uses a lot of time and though RRT is quite similar to Bi-RRT (in terms of algorithm) but it is not as fastest as bidirectional one is.

References

[1] Gul, F., Rahiman, W., & Nazli Alhady, S. S. (2019). A comprehensive study for robot navigation techniques. Cogent Engineering, 6(1), 1632046.

[2] P. K. Mohanty and D. R. Parhi, “Controlling the motion of an autonomous mobile robot using various techniques: a review,” Journal of Advance Mechanical Engineering, vol. 1, no. 1, pp. 24–39, 2013.

[3] J. M. Keil, “Decomposing a polygon into simpler components,” SIAM Journal on Computing, vol. 14, no. 4, pp. 799–817, 1985.

[4] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in Autonomous robot vehicles. Springer, 1986, pp. 396–404.

[5] D. Gallardo, O. Colomina, F. Florez, and R. Rizo, “A genetic algorithm for robust motion planning,” in Tasks and Methods in Applied Artificial Intelligence, pp. 115–121, Springer, 1998.

[6] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[7] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.
[8] M. Samuel, M. Hussein, and M. B. Mohamad, “A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle,” International Journal of Computer Applications, vol. 135, no. 1, pp. 35–38, 2016.

Acknowledgement

The authors would like to thank to the Universiti Sains Malaysia for financial assistance under Research University Individual grant (1001\PELECT\8014097).