Advanced Production and Industrial Engineering
R.M. Singari and P.K. Kankar (Eds.)

doi:10.3233/ATDE220744

Action Plan for a Two Wheeled Mobile Robot Navigation with Controlled Velocity Approach on MATLAB

Abhishek Jain[1], Manik Singhal and Mansi Jhamb
University School of Information Communication and Technology (USIC&T), GGSIPU, New Delhi (India)

Abstract. This paper presents an "Autonomous Navigation" Action plan for a 2 wheeled, single axis rotary mobile robot with controlled (or constant) velocity approach. Through this analysis we will step-by-step understand how we can navigate a mobile robot from a 'start point' to a 'goal point' with "obstacle avoiding" utility and constant velocity provision (using different controllers). This paper also presents the comparison of different types of controllers used to control the velocity and different algorithms to plan as well as track the path.

Keywords. Mobile robot system, Probabilistic Roadmaps, Rapidly Exploring Random Tree, Pure pursuit controller, PID, MATLAB

1. Introduction

Mobile robot system (MRS) are robots that can move in the surroundings either manually or autonomously to carry out the assigned task. Nowadays MRS are being extensively used at personal as well as industrial level [1]. A general action plan for Autonomous Navigation from a start point to a goal point with constant/controlled velocity consists of Path planning with obstacle avoidance, Path tracking and controlled robot movement [2-3]. Main factors for navigational purpose of robot are (i) Path taken and (ii) Target velocity of robot with which it will move from one waypoint to another. This paper involves simulation-based analysis performed on MATLAB and Simulink software.

2. Path Planning

Path planning also known as Motion planning is a logical/computing problem to generate a sequence of valid waypoints to reach from a start to a goal location, such that it avoids hitting any obstacle while travelling. For path planning a map is needed for localization. In this paper, simulations are performed on (i) 'Binary Occupancy Map', where 2 states are present – occupied and unoccupied and (ii) 'Occupancy Map', where probability values are used to create the map representation.

Path planning algorithms are classified as – (i) Grid Based, where the entire map is divided into several grid cells and each cell has an associated cost depending on distance, and the path having least cost is selected and (ii) Sampling based, where random sampled

[1] Corresponding Author, Undergraduate Student, University School of Information Communication and Technology, New Delhi (India); E-mail: abhishek.2216412819@ipu.ac.in, jain27854@gmail.com

nodes are generated in a map [4]. Grid based algorithms are not efficient for applications that has high degree of freedom or when the map size is very large, therefore Sampling based algorithms are more preferred. Here two Sampling based algorithms are discussed – (a) Probabilistic Roadmaps (PRM) and (b) Rapidly exploring random tree (RRT).

2.1. Probabilistic Roadmaps (PRM) Method

PRM method involves generation of a network graph of different possible paths by joining various random points in a given map on the basis of occupied (obstacle) and un-occupied spaces. It generates random points and certain paths around those points, and then try to find the most efficient path. It starts by generating a random node/point in the map. Then it checks whether that point lies in unoccupied space, if it does then it is added to the graph network. In similar way multiple random points are sampled in the given state space & after this is done, they are inter-connected, ensuring that these connections don't lie in occupied space [5]. MATLAB provides inbuilt function for PRM simulation.

Drawback: PRM does not guarantee the most optimized path every time. This is because PRM algorithm is based on random point generation and connections lying in unoccupied space. This increases the chance of many voids that are lying between two or more closely placed obstacles of not being used for node connection, which otherwise could have contributed to make a more optimized path to reach the destination.

2.2. Rapidly Exploring Random Tree (RRT) Method

RRT method generates an unoccupied space filling tree/path from start to a goal location. RRT plans the path in a state space so that it can get an idea of the map, in which it is going to plan the path. A State defines the robot's position & its orientation, whereas the State space defines the various robot states possible. The State space representation used in this paper is a 'SE(2)' State space, where the robot localizes in a 2D map & each state of the robot has a 3-parameter vector (comprising of x, y and theta) associated with it [6].

RRT starts the path generation process from an initial tree root (i.e., a random sample network) that has start point as one of its nodes. A random point is generated in the state space, then a line is drawn between this and the nearest node within the initial tree and then a new point is generated at 'delta' distance from the nearest node on this drawn line. This connection is saved in the network if it lies in unoccupied (i.e., obstacle free) space. The process of expanding the tree goes on till the random node generated coincides with the goal location [6]. MATLAB provides inbuilt function for RRT simulation.

2.3. Comparison between PRM and RRT

We performed simulations in MATLAB using inbuilt 'timeit' function to determine the time taken for the generation of the most optimized path by PRM and RRT method as shown in Fig 1. From the results obtained (shown in Table 1) it was concluded that RRT is much faster than PRM for path planning.

Table 1. Comparison between time taken by PRM and RRT

	X coordinate (meters)	Y coordinate (meters)	PRM Simulation Time (seconds)	RRT Simulation Time (seconds)
Start	2.8	1.4	0	0
Goal	11.2	0.7	0.1818	0.0933

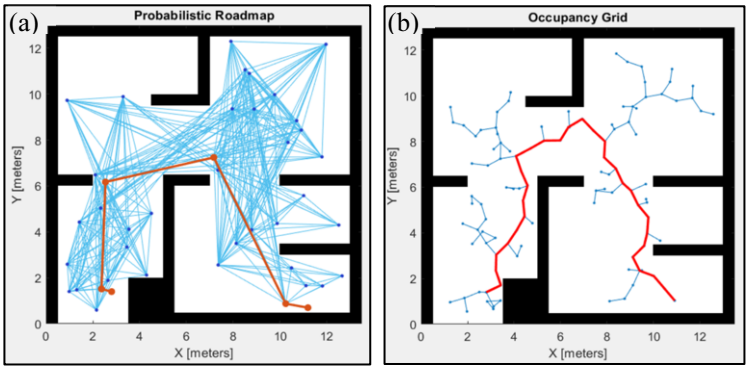

Figure 1. (a) PRM algorithm-based path (b) RRT algorithm-based path

3. Path Tracking - Pure Pursuit controller

After planning the path, the robot needs to trace it. Pure Pursuit is a path tracking algorithm. It tracks the path that is already created (set of waypoints), by calculating the required target velocity and orientation for the robot between 2 waypoints. Pure pursuit controller has certain lookahead distance & kinematic calculations associated with it [7].

3.1. Look Ahead Distance

Pure pursuit algorithm makes the robot move from current waypoint to the next waypoint (destination) by determining the curvature of path. This destination also called Look-ahead point is a point taken at a fixed distance on the reference path ahead of the robot's current position. An arc is generated by joining the current point and the Look-ahead point; and the chord between these two points is called the Look-ahead distance [7]. The robot needs to reach this point & for this it uses a steering angle (δ) as defined below:

$$\delta = arctan\left(\frac{2Lsin\alpha}{l_d}\right) \quad (1)$$

Where, l_d=look-ahead distance, L=car length, α=target direction angle.

There are 2 types of look-ahead distance, based on which the path tracking way changes for the robot – (a) Small Look-Ahead Distance: Here a short look-ahead distance is taken. This results in a faster speed but leads to unstable oscillations and higher deviation from the path (b) Large Look-Ahead Distance: Here a larger look-ahead distance is taken. This overcomes the problem of oscillations along the path but results in larger radius of curvature near corners (waypoints) and doesn't pass through them [7].

3.2. Kinematics calculation

Robot has individual target velocities for each point lying on the path, to reach them. The robot takes the nearest point target velocity to calculate the wheels angular velocity. While calculating the target velocity for a point, the maximum velocity allowed for the path and the curvature at the point needs to be taken care of [7]. Velocity at each point is

$$V_i = \text{min (max path velocity, } k\text{/curvature at point)} \quad (2)$$

Where, k is a constant (range of 1-5), based on how slow the robot should go around the turns. This ensures that robot's target speed never exceeds the maximum velocity and it also reduces the robot's speed around turns. The robot should also ensure that it doesn't exceed a maximum acceleration. For calculating the target velocity to reach the next point, robot is accelerated at maximum acceleration along the path, with its velocity limited by the maximum velocity calculated in Eq. (2). Kinematic equation is used to calculate the target velocity:

$$\text{Target linear velocity, } V_f = \sqrt{V_i{}^2 + 2 * a * d} \tag{3}$$

Where, V_i = velocity at prior point, a = maximum acceleration, d = distance between two points. Target angular velocity is calculated by:

$$\text{Target angular velocity, } \omega = R_{bot} * V_f \tag{4}$$

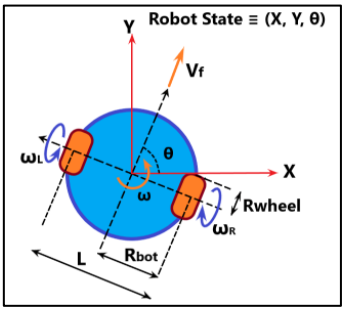

Figure 2. Two wheeled robot state diagram and its parameters representation

Where, R_{bot}= radius of robot chassis, V_f = Target velocity at a point as shown in Fig 2. After getting the target linear and angular velocity for the robot, the angular velocity for each wheel of robot can be calculated using inverse kinematics [8].

$$\omega_L = \frac{V_f - \frac{\omega L}{2}}{R_{wheel}} \quad \& \quad \omega_R = \frac{V_f + \frac{\omega L}{2}}{R_{wheel}} \tag{5}$$

4. Controlled Angular velocity

For a robot to perfectly trace the path it is necessary that its wheels run with controlled angular velocity, otherwise the uncontrolled angular velocity might lead to irregular path tracing. Here, three feedback algorithms are compared. But before that let's understand the Motor structure & equations that are used in the mathematical modelling of the MRS.

4.1. Permanent Magnet Direct Current (PMDC) Motor Structure and equation

MRS discussed in this paper includes a PMDC circuit. A PMDC motor is a DC motor with 2 permanent magnets (opposite poles) fitted inside the metal body in-front of each other. These two magnets along with external body of motor forms the 'Stator', whereas the inner electronic & mechanical circuit forms the electromechanical part of the motor [9]. When the circuit is given voltage supply, an electromagnet is formed, and it starts rotating inside the fixed field loop; and thus, the motors rotate. The circuit equations are:

- Applying KVL in Electrical part:

$$V_{in} = I_a R_a + L_a \frac{dI_a}{dt} + K_e \omega \implies \boldsymbol{I_a = \int \frac{1}{L_a} (V_{in} - K_e \omega - I_a R_a)\, dt} \quad (6)$$

- Rotational mechanical system equation:

$$J \frac{d\omega}{dt} = K_t I_a - B\omega - \tau_{Load} \implies \boldsymbol{\omega = \int \frac{1}{J} (K_t I_a - B\omega - \tau_{Load})\, dt} \quad (7)$$

Where, *Ra* = Armature resistance, *La* = Inductance in armature, *Ia* = Current in armature, *E(t)* = Back EMF, V_{in} = Supplied input voltage, τ_{load} = Torque, θ = Angular displacement, *B* = Rotational Damping/Friction coefficient, *J* = Moment of Inertia, K_e= Electrical/Back-emf constant, K_t= Torque constant

4.2. Transfer function and Modelling

Transfer function of the system is obtained by converting the Eqs. (6) and (7) into frequency domain and rearranging them, given in Eq. (8). Here, the input is the supply voltage and the output is angular velocity of the robot's wheel [9]. The control system plant model for Eq. (8) was constructed in Simulink and used for simulation.

$$G(S) = \frac{\omega(S)}{V_{in}(S)} = \frac{K_t}{[(L_a J)s^2 + (R_a J + B\, L_a)s + (R_a B + K_t\, K_e)]} \quad (8)$$

4.3. Open loop system

It is a control system where the output is dependent on input only, and there is no feedback signal from the output to the input. In this system a supply voltage is given as input and the angular velocity is received. An additional input torque represents any external resistance (a step signal) that may obstruct the movement of motors, but our aim is to get controlled angular velocity. In open loop configuration the output is adversely affected by external torque/resistance, therefore it doesn't ensure desired output.

4.4. Closed Loop Controlled system and Controllers

It is a control system where the output is dependent on system input as well as on feedback input. In this, feedback is taken from the output and added in negative phase with system input resulting in actuating error signal which is then fed into the plant. To further enhance the desired output probability, a controller is used in the control system, which is a mathematical algorithm that is used to manipulate the operating conditions of the input that is fed into the plant [10]. There are various types of controllers available, three of them are discussed here – (A) Proportional Integral (PI): Here the plant is fed with combination of proportional & integral controller output of actuating error signal (B) Proportional Derivative (PD): Here the plant is fed with combination of proportional & derivative controller output of actuating error signal (C) Proportional Integral Derivative (PID): Here the plant is fed with combination of proportional, integral & derivative controller output of actuating error signal.

MATLAB Simulation time response output for open loop, PI, PD, and PID systems are shown in Fig 3. In this analysis, a step signal is taken as the input torque condition.

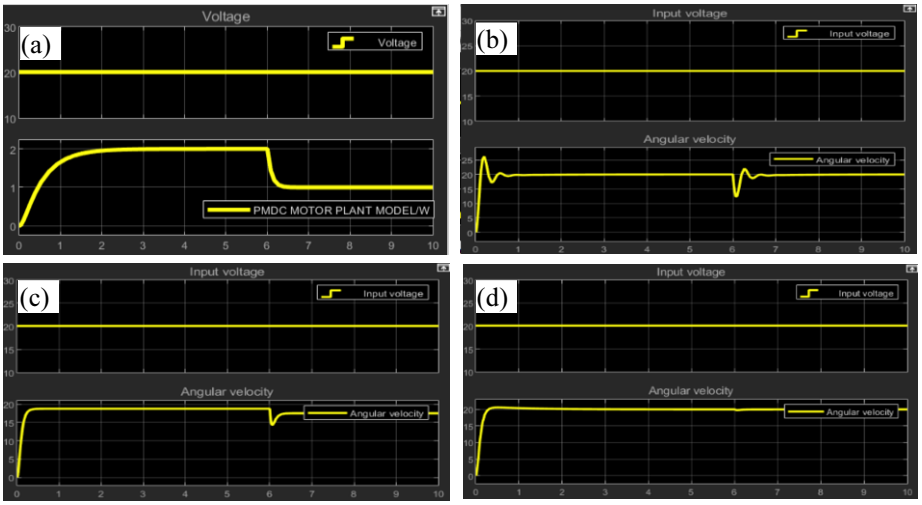

Figure 3. Time response output for (a) Open-loop (b) PI (c) PD (d) PID

Observation: From Fig 3 and obtained results it can be observed that PID is the best algorithm among the three, as it extracts the best features of Proportional, Integral and Derivative controller modelling; and also gives the least steady state error.

5. Conclusions

Action plan began with path planning-PRM and RRT. Out of them RRT was found more optimized and faster, therefore we further traced the RRT path waypoints by Pure Pursuit algorithm. This algorithm was used to calculate the curvature, target robot velocity, and wheel angular velocity with which the robot will travel between two waypoints. To control Robot's wheel angular velocity, PID controller was found most effective.

References

[1] Nehmzow U. Mobile Robotics: A Practical Introduction. 1st ed.: Springer, London; 1999.

[2] Armah , Yi , Abu-Lebdeh T. Implementation Of Autonomous Navigation Algorithms On Two-Wheeled Ground Mobile Robot. American Journal of Engineering and Applied Sciences. 2014; 7: 149-164.

[3] Mnubi SA. Motion Planning and Trajectory for Wheeled Mobile Robot. International Journal of Science and Research. 2016 January; 5(1): 1064-1068.

[4] Lynch KM, Park FC. Modern Robotics: Mechanics, Planning, and Control; 2017.

[5] Kavraki LE, Svestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional config spaces. IEEE Transactions on Robotics and Automation. 1996; 12(4): 566-580.

[6] LaValle SM,&KJJ. Randomized Kinodynamic Planning. IJRR. 2001 May;: 378-400.

[7] Counter RC. Implementation of the Pure Pursuit Path Tracking Algorithm. Tech. Report. CMU, 1992.

[8] Hirpo BD, Zhomgmin PW. Design and Control for Differential Drive Mobile Robot. International Journal of Engineering Research & Technology. 2017; 6(10): 327-334.

[9] Sharmal S, Jain S. Speed control of Mobile Robotic System using PI, PID and Pole Placement Controller. In IEEE ICPEICES; 2016. p. 1-5.

[10] Khan H, Khatoon S, Gaur P. Comparison of various controller design for the speed control of DC motors used in two wheeled mobile robots. IJIT, Springer. 2021; 13: 713-720.