Report on Non-Holonomic Planning Of A Car

By Kushal Jadhav

Siddish P Rao

Introduction: -

In this Project, we consider motion planning for a Non-Holonomic car (Dubin's Car) using RRT Algorithm. The objective is to develop and implement a path planning system that can be used for a car to move from an initial configuration to a goal configuration. Non-Holonomic Planning means planning with some differential constraints. The constraint in our case is that the car can only move in forward direction with constant velocity.

Description of Implementation: -

After assessing various algorithms available for non-holonomic planning, we decided to implement a Rapidly Exploring Random Tree Algorithm (RRT) with a Dubin's car model to implement this project.

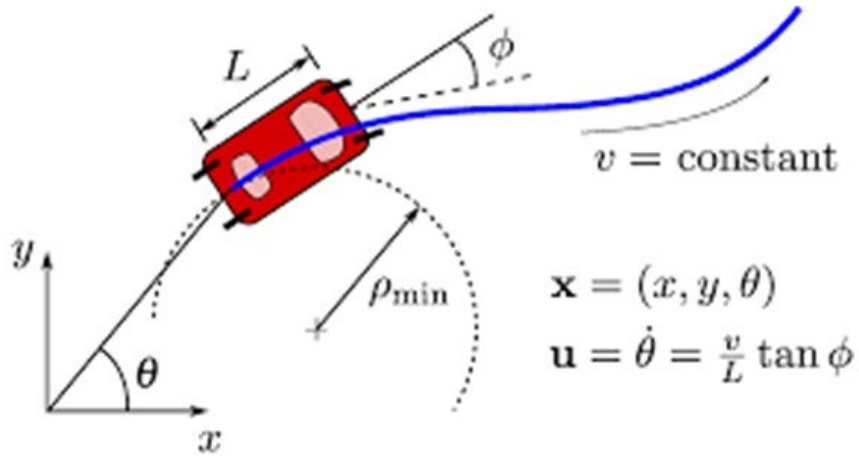
RRT: -

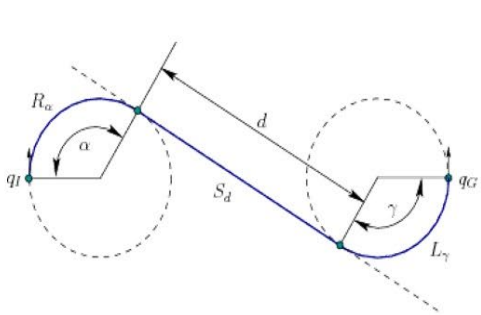
RRT is a Sampling-based motion planning approach which was introduced by LaValle and Kuffner in 1998. RRT is an algorithm which performs incremental sampling and searching and is mostly appropriate for single query problems. RRT is probabilistically complete and has a perfect balance between greedy search and exploration. RRT is constructed incrementally in a way that quickly reduces the expected distance of a randomly chosen point to the tree. RRT is particularly suited for path planning problems that involve obstacles and differential constraints. Usually, an RRT alone is insufficient to solve a planning problem. Thus, it can be considered as a component that can be incorporated into the development of a variety of different planning algorithms.

45 iterations 390 iterations

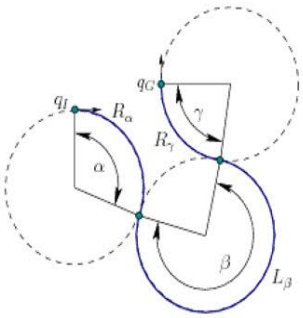
Dubin’s Car: -

Dubin’s Car was introduced by a famous mathematician and statistician Lester Dubin in the year 1957.It is a simplified mathematical model of a car that moves in the x and y plane. The car's location is specified by the location of the center of the car's rear axle and the orientation of the car. The car cannot move sideways because the rear wheels would have to slide instead of roll. The Dubin’s car model stipulates that the car moves forward at a constant speed and has a maximum steering angle that translates into a minimum turning radius. Dubin’s Car can only move forward and not backwards with constant velocity as 1 m/s. The car can only have three controls which are Turn Right at maximum, Turn Left at maximum and Go Straight.





$R_\alpha S_d L_\gamma$



$R_\alpha L_\beta R_\gamma$

Source: S. LaValle, Planning Algorithms

Method of Implementation: -

There are many ways in which we can implement the local planner in our RRT algorithm such as defining six scenarios for the Dubin's car which include {RSR,LSL,RSL,LSR,RLR AND LRL} and select the optimal scenario to traverse the path or apply random controls (or a sequence of random controls) for a fixed amount of time and select the control that leads you closest to the random configuration. In our Implementation, we have defined three controls (Left, Straight, Right) which are applied for a fixed amount of time and selected the control which has the minimum distance to the random configuration.

GENERATE_RRT($x_{init}, K, \Delta t$)
1 $\mathcal{T}$.init(x_{init});
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 $\mathcal{T}$.add_vertex(x_{new});
8 $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
9 Return $\mathcal{T}$

Figure 1. Pseudo code of an RRT

Steps to implement the Algorithm: -

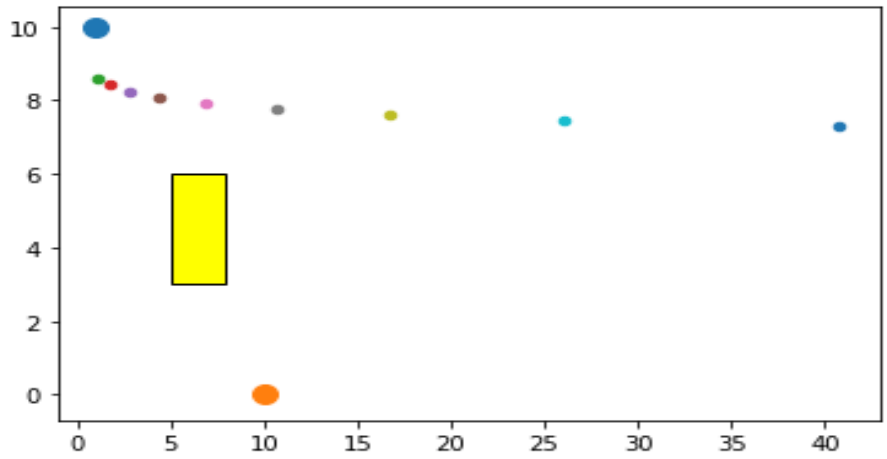
1. Initialize the start and goal node.
2. Select a random node from the configuration space.
3. Search for a nearest node from the random space.
4. Steer the nearest node to a new node using the left, right and straight controls.
5. Check the distance between the new node and the random node for all the three controls and select the control with the minimum distance as the feasible node.
6. Check for collision with the feasible node.
7. Check if the feasible node is near the final node.
8. Generate the final path.

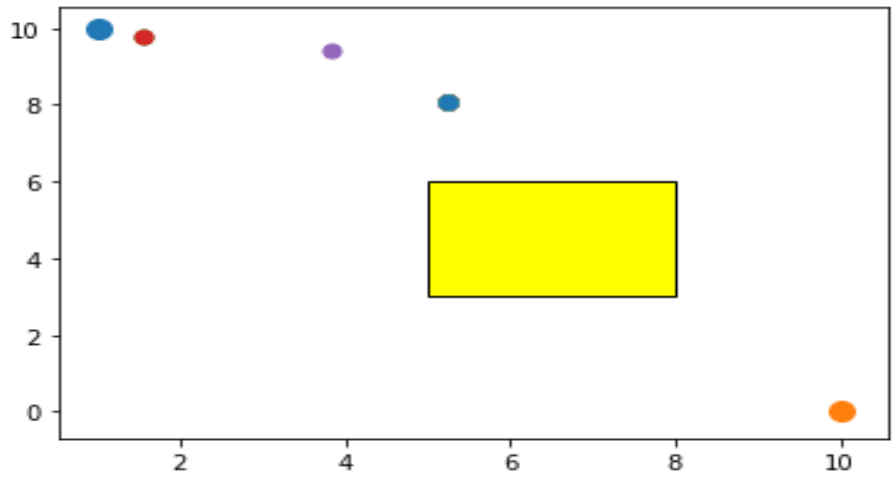
Status of Code: -

We implemented all the steps as mentioned in the Pseudo Code above. Plot for the configuration space with obstacles, start node and end node and the traversed path was successfully implemented. We were unable to find out the right expanded node which had the minimum distance from the expanded node to the random node. The three expanded nodes were implemented using the three controls left, right and straight. This further restricted us from getting the desired path from start configuration to goal configuration.

Results: -

We made changes to various parameters such as time step, window time size, value of theta and following are some of the results we obtained:





Future Work: -

RRT can help us find out the feasible path for the car to travel and not the optimal path. RRT Algorithm also cannot incorporate additional cost information such as smoothness or length of the path to solve a path planning problem. For this reason, it is desirable to use variants of the RRT that converges to the optimum path such as RRT* to make it more dynamic. The use of Dubin's cars as model to represent a non-holonomic vehicle model is seen to be restrictive and a misrepresentation of an actual car model. Relaxing the vehicle path constraints by using a Smooth Car model would be necessary to find a more optimal path.

3rd Party Libraries and Code Used: -

1.math

2.matplotlib

3.numpy

4.random

References: -

1) http://lavalle.pl/planning/ch5.pdf
2) https://gieseanw.files.wordpress.com/2012/10/dubins.pdf
3) http://planning.cs.uiuc.edu/node821.html
4) https://www.python.org/doc/
5) https://www.youtube.com/watch?v=rVqMwXUJWGQ&feature=youtu.be
6) https://www.geeksforgeeks.org/euler-method-solving-differential-equation/