Assignment 3
SC627

Bhavini Jeloka
18D100007

I. Introduction

While interacting with the real world a robot often comes across moving obstacles. It must be able to plan in advance so as to avoid intersecting with them in any future time step. In this report, we implement one such method - the velocity obstacles approach. In this report, we employ the velocity magnitude and directions of the robot as well as the obstacles in order to obtain a feasible next move. Simultaneously, we also optimise using heuristics so that the next move brings the robot closer to its desired goal position.

II. Dynamic Collision Avoidance - Velocity Obstacle Approach

The flowchart given below summarizes the algorithm that we have implemented (assuming that a path exists). From the figure it is clear that the algorithm is capable of avoiding obstacles in order to reach the goal as long as such a path exists.

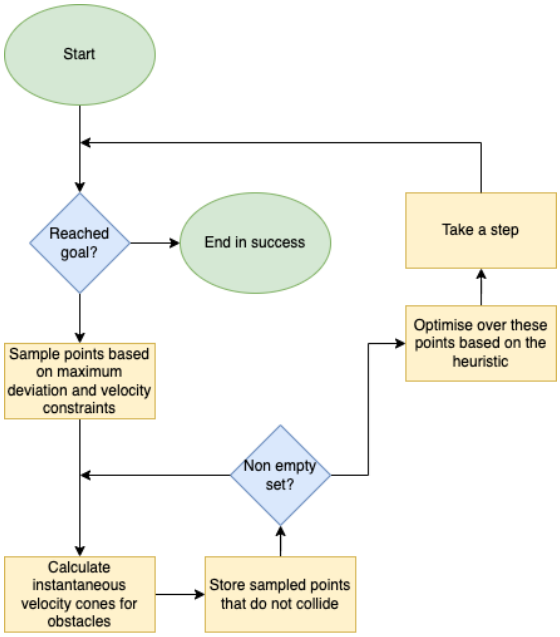

Fig. 1. Dynamic Collision Avoidance

A. Algorithm Initiation

We define a class that contains the position, velocities (angular and linear) and orientation of the robot via the odometer. It also receives the data regarding the velocities and positions of the obstacles. After the file is launched, we begin taking readings and this is further used in deciding the next step of motion. At the first step, the robot is at the origin and is at rest.

B. Dealing with Obstacles - Collision Cones and Heuristics

We have a constraint on the maximum deviation and the velocities due to the actuator dynamics. Therefore, the attainable velocities significantly reduce. In order to calculate the range of velocities, we discretize the angles (+/- max deviation from heading, step size: 1 degree) and the velocities (upto max velocity, step size: 0.01). This gives gives us the next velocity, with respect to the current velocity.

Now, we just need to work with the velocities that do not result in collision. For this, we calculate the velocity cone using the current position. This cone is essentially the angle made by the robot (in configuration space, therefore represented as a point) and the tangents to an obstacle. We store this angle for each cone. In order to check whether a sampled velocity is within the cone or not, we find its deviation from the line joining the centre of the obstacle and the robot. If it is less than half the cone angle, we conclude that it lies within the velocity obstacle. These velocities are discarded from our heuristic calculations. In our implementation, we note two important points. Firstly, according to the reference provided, to the velocity cone with originating at the bot, we add the velocity of the obstacle (translation of the cone)

in order to get the spatial representation and decide the feasible motion. For our case, we implement the opposite (albeit equivalent computation). We shift the sampled point by the velocity of the obstacle and then check for collision. The second thing to note is with respect to calculating the swept angles for the velocity obstacles. Instead of only adding the radius of the robot to that of the obstacle to get the tangents to the circle, we multiply the aforementioned value by 2 for padding and protection against collisions (especially while implementing it on ROS). This can prevent the robot from grazing the obstacle and moving. The end product of this computation is the feasible regions that ensure a collision free movement for the next time step.

Lastly, we decide the robot's next step based on a heuristic. In our case, we calculate the deviation of the feasible velocities from the direction towards the goal and then move in the direction with least deviation.

C. Termination

Since the angular deviation from the current heading is limited, the bot takes a long time to reorient towards the goal. Once it reorients, it incurs the obstacles once again thereby adding to the time complexity. In order to yield faster (and near optimal) results, we terminate the program once the robot is within a δ ball of the goal.

III. Numerical Results

We test our algorithm in a 2D environment with three obstacles. Some key features of the environment are given below along with plots and visualisations of the results obtained.

1) Start Position: [0,0]
2) Goal Position: [5,0]
3) Robot and Obstacle Diameters: 0.15m
4) Maximum Velocity of the Robot: 15m/s
5) Maximum Deviation in Orientation (of the Robot): 10 degrees

Although the goal is along the x-axis, there is an obstacle that resides on the axis itself. In order to avoid this, the robot needs to move in the y-direction and this is indeed the case as seen in figure 2.

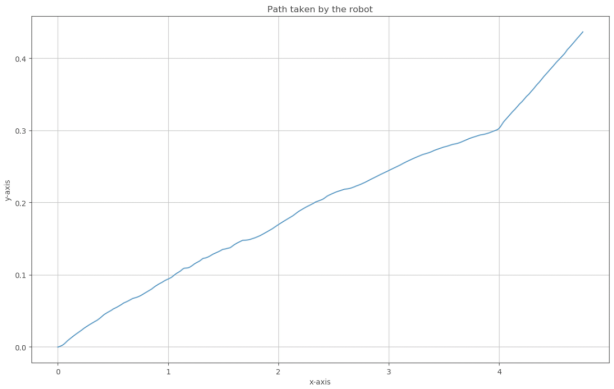

Fig. 2. Path Taken by the Robot

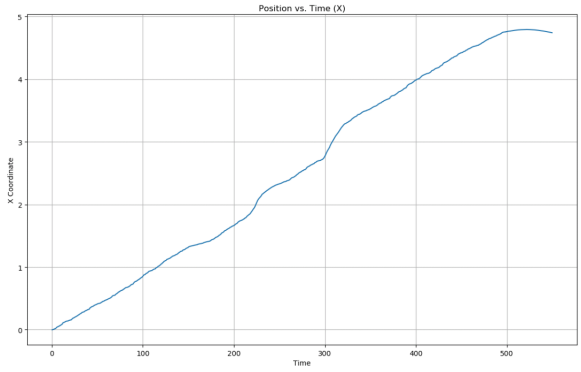

Fig. 3. x Coordinate as a Function of Time