SC627-Motion Planning & Coordination of Autonomous Vehicles

Assignment-3

Danish Behnal (190010018)
Instructor: Prof. Arpita Sinha
8 April 2022

Problem Statement

We are provided a dynamic environment with 3 obstacles and our goal is to plan a trajectory that takes the agent from (0,0) to (5,0) while avoiding collision.

Constraints:

- Robot and obstacle Diameter = 0.15
- Robot's maximum velocity = 0.15m/s
- Max Deviation = 10 Degrees

Implementation

My implementation includes 2 helper files and 1 plotter file as well as the main collision avoidance script. The process is explained below:

- First objective vector (i.e unit vector that takes us to the goal) is calculated using the Bot's current position and goal location. Subsequently we also obtain the angle of that unit vector.
- We use the *colcone_vec()* function to obtain tangent vectors(upper & lower) from each obstacle.
- We pass these tangents to *col_cone()* function that gives us all the angles lying in between the cone for each obstacle. We take union of all the 3 sets to obtain all the angles which our agent should avoid.
- Then we obtain a feasible set of angles which our agent can take without fear of collision. We then compare all the feasible angles to our objective angle and chose the angle which is closest to our objective.
- The strategy employed is the MV (Maximum Velocity) strategy.

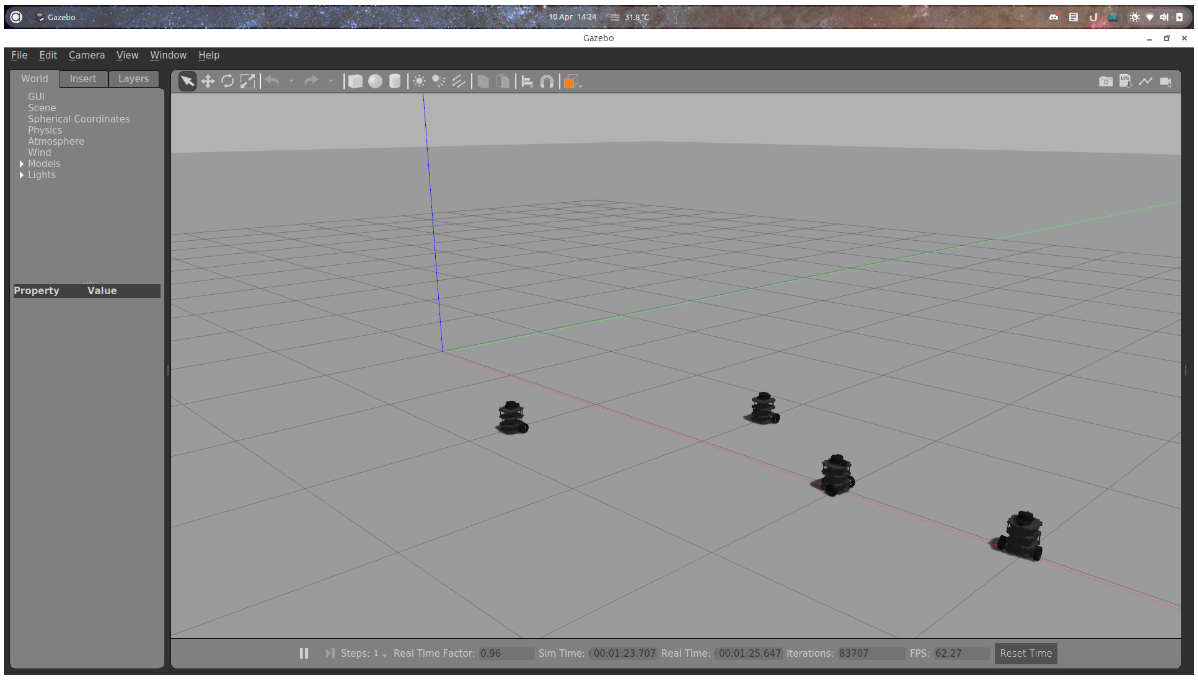
Explanation of Helper functions:

- ***nice_angle()***: This function takes in an angle and then clips it so that we always obtain angles from 0 to 2π
- ***colcone_vec():*** This function takes in position of the Bot, position of an obstacle as well as radius of obstacle as arguments and outputs two unit vectors which points in the direction of the two tangents which forms a cone of an obstacle.
- ***col_cone()***: This function takes in two tangent vectors (low and up tangents) and outputs an array of angles lying between those two tangents (0 to 360)

Changes to existing code:

- Removed the angle conversion in *if-else* loop in ***velocity_convert()***
- Included time.sleep() for 3 seconds in the code

Results:



Steps: 1 Real Time Factor: 0.96 Sim Time: 00:01:23.707 Real Time: 00:01:25.647 Iterations: 83707 FPS: 62.27

Gazebo Environment showing bot at Goal location

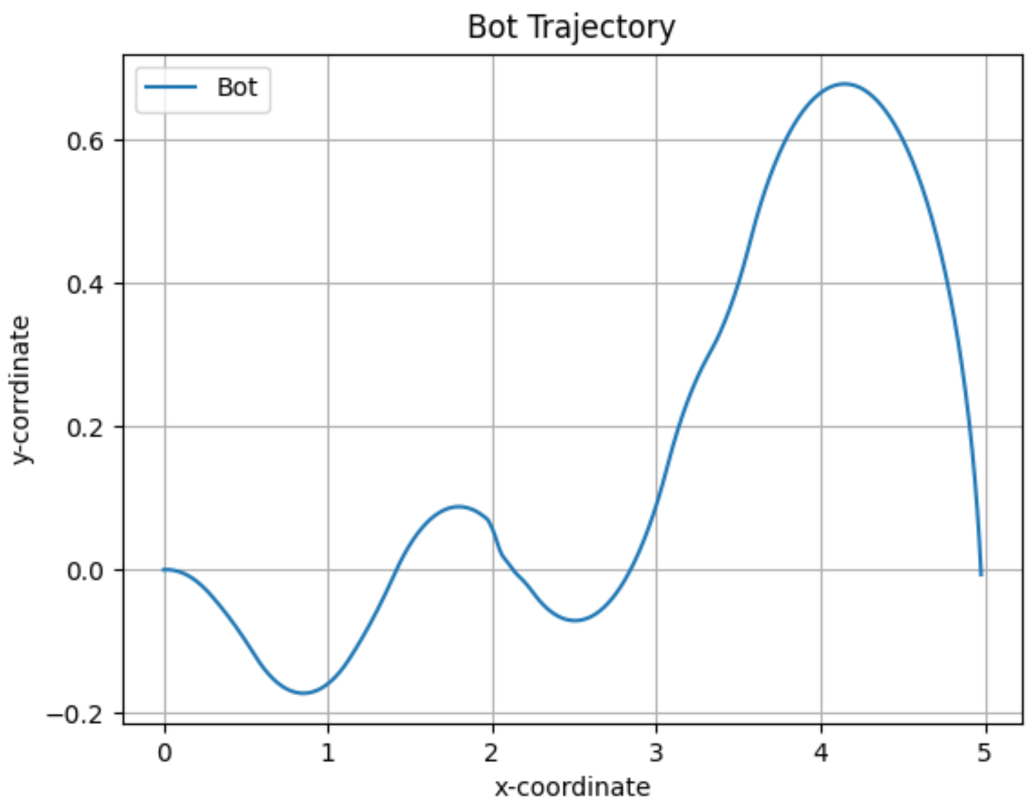


Fig2. Bot's Trajectory

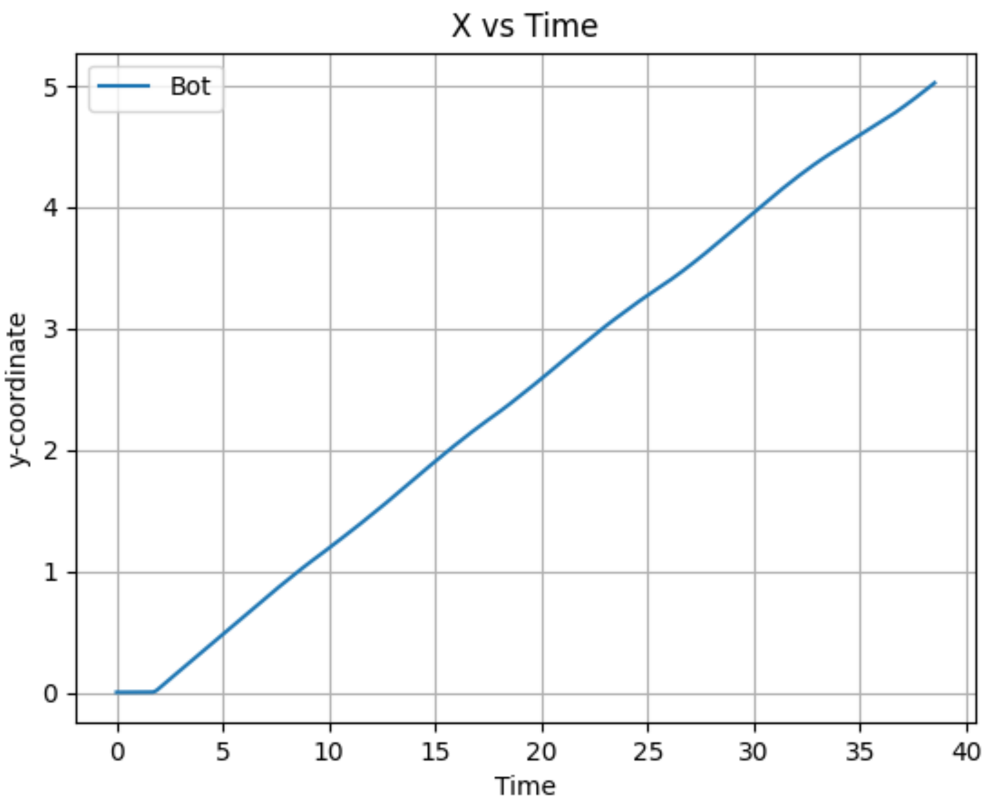


Fig 3. x-coordinate vs time