SC-627 Motion Planning & Coordination of Autonomous Vehicles

Assignment-1

Danish Behnal (190010018)
Instructor: Prof. Arpita Sinha
5 March 2022

BugBase Algorithm

In order to implement the BugBase algorithm we first define helper functions which aid us in the implementation of the algorithm. The functions in *helper.py* and their purpose is briefly described below:

- ***cal_distance()*:** returns the euclidean distance between two points.
- ***cal_angle()***: returns the angle between two vectors.
- ***ComputeLineThroughTwoPoints()***: returns the coefficients of line segment between two points.
- ***ComputeDistancePointToSegment()***: returns distance and a number. (0,1 or 2) depending on which region the point lies.
- ***computeDistancePointToPolygon()***: returns the minimum distance between a point and a polygon
- ***computeTangentVectorToPolygon()***: returns a unit vector tangent to a polygon.

BugBase Implementation

First thing we do is calculate the objective vector, this vector points from the bot's current location to the goal location.

Then we loop through all the obstacles and find the minimum distance from all the obstacles and store it.

Then we check if all the values in that array are less than the step size of the bot we continue to move along the objective vector towards our goal.

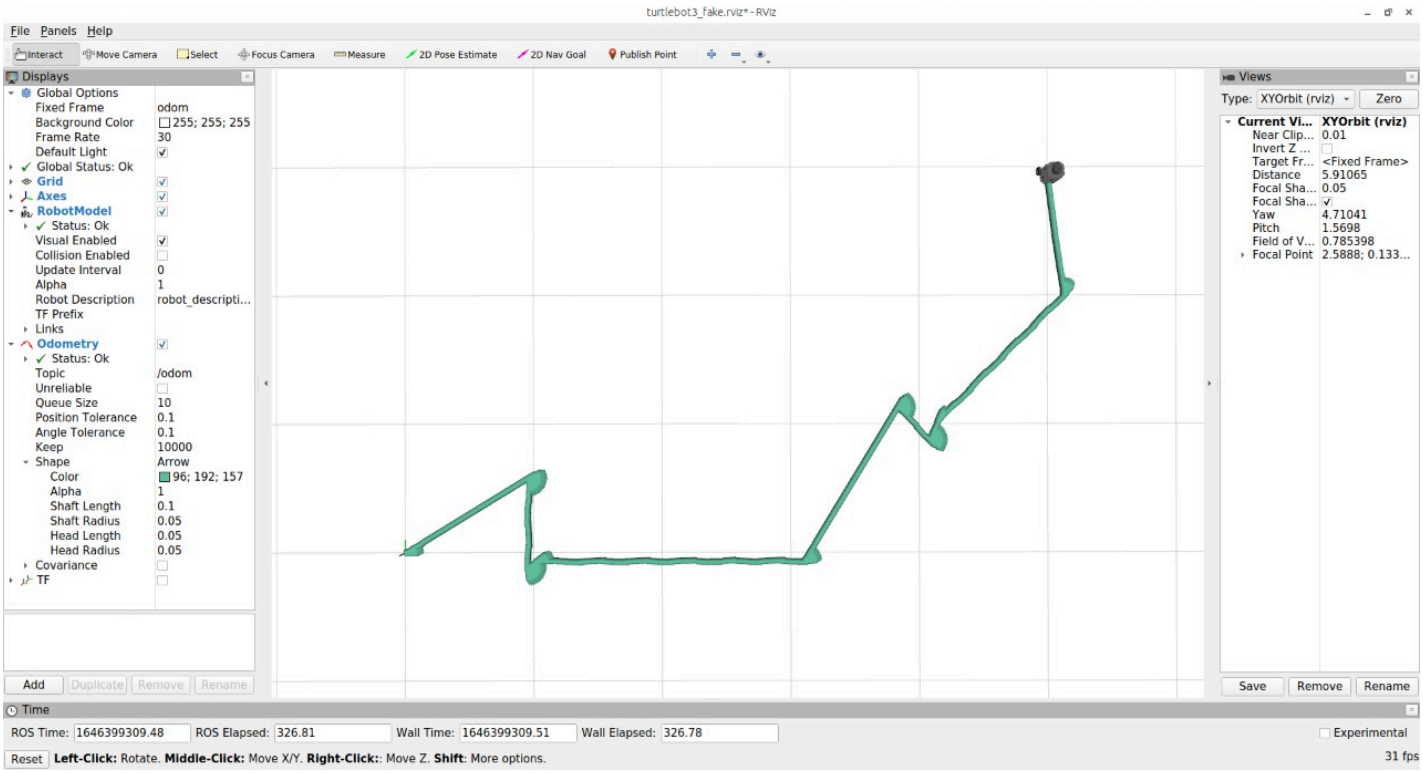
Once for any of the obstacle the above condition fails meaning the distance to the obstacle is less than the bot's step size we start to move along the tangent to the obstacle.

Once the bot turns around a vertex of the obstacle it again measures its distance from the polygon obstacle, eventually the distance between the bot and the obstacle is big enough that it stops following the wall of the obstacle

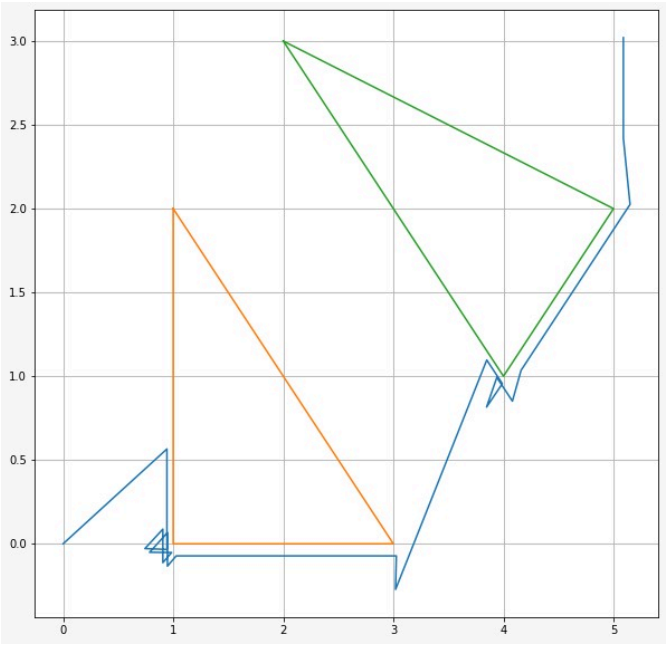
and start to move along the objective vector. This happens again if it encounters another obstacle till our bot has reached its final goal location.

Results

Results for the given environment for BugBase algorithm



Turtlebot3 BugBase



Python Plot for comparison

Bug_1 Algorithm

Bug_1 algorithm is a slightly advanced version of BugBase. This algorithm will make use of all the helper functions defined for the BugBase. The difference between the two algorithm is that in Bug_1 when we encounter an obstacle we circumnavigate around it and store the location around the obstacle where the distance to the goal was minimum. Then we move to that point and then proceed towards the target.

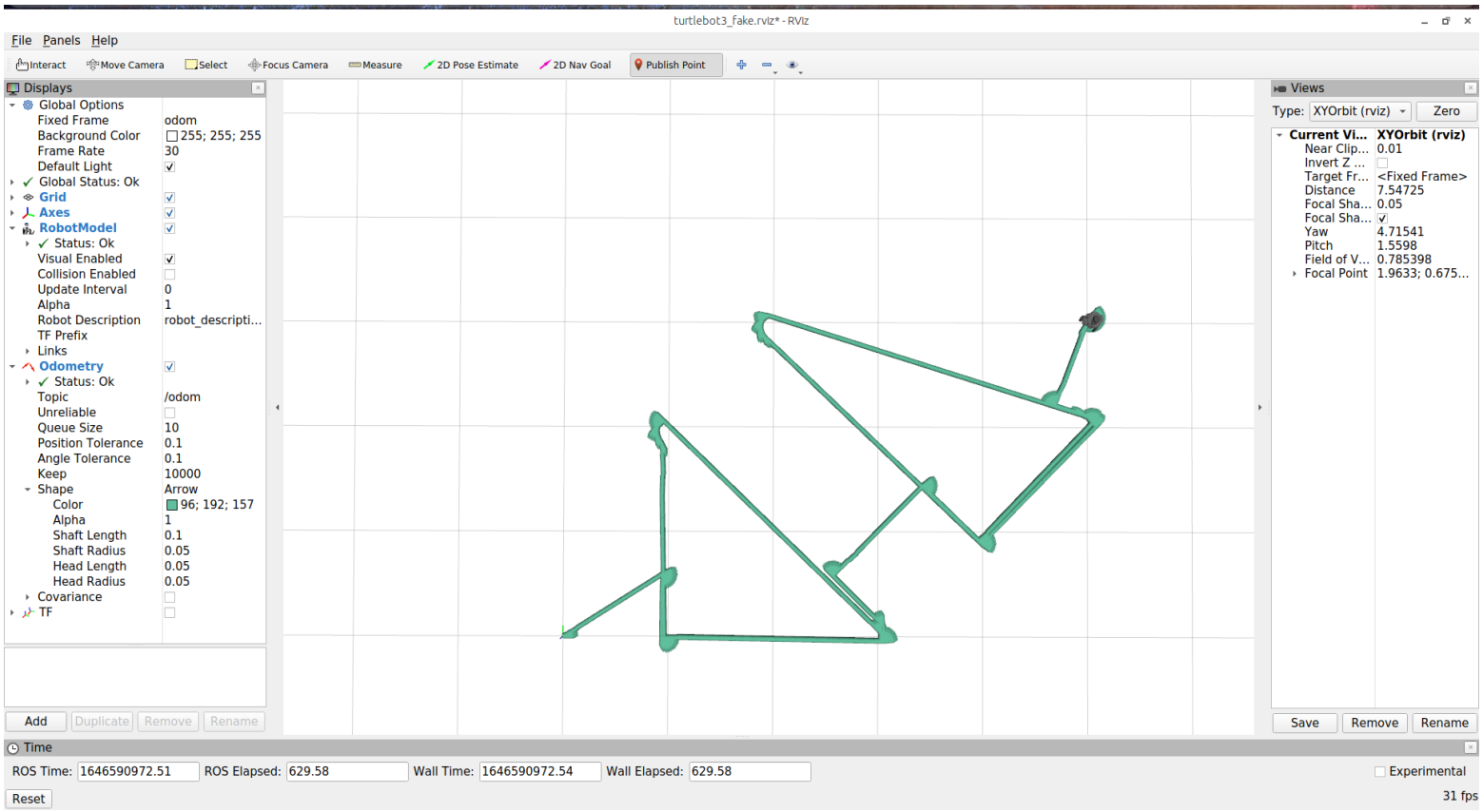
Bug_1 Implementation

The main challenge in this implementation was the task to make the bot identify the location where the distance was minimum. This was difficult because every time the bot turns it either moves far from the edge of the polygon and towards the edge of the polygon. Even if we are able to find the location at which distance is minimised, the bot never reaches it again due to reason mentioned above.

So as a quick fix a if statement is included in the Bug_1 code which ensures that the distance between the obstacles boundary and the bot stays constant as it circum-navigates it. The constant distance is the distance between the obstacle's boundary and the point when the collision between the bot and the obstacle is detected. This gave me good results although this can be worked upon as this fix has reliability issues.(For example if the bot moves inside the obstacle etc)

Results

The bot is able to reach the goal precisely!



—END—