Path Planning

Algorithm Over view:

1. Initialize ROS 2 Node:

- Create a ROS 2 node named ‘soccer\_path\_planning’.

- Set up subscriptions for data scan --> either laser scan or occupancy grid or combination of both

- create publisher for the goal position on the robots coordinate frame

- Initialize parameters

2. scan the environment and find the ball (should be able to use the obstacle avoidance class)

3. scan to find the obsacles

4. get repulsive and attractive forces of the obstacles

5. implement path planning logic

6. send velovity commands

Pseudo code:

import rclpy

from rclpy.node import Node

from geometry\_msgs.msg import PoseStamped

from nav\_msgs.msg import OccupancyGrid

from nav2\_msgs.action import ComputePathToPose

From nav2\_msgs.msg import Path

From nav2\_msgs.srv import GetPlan

Class PathPlanningNode (Node):

Def \_\_init\_\_(self):

super().\_\_init\_\_(‘soccer\_path\_planning’)

self.get\_logger().info(‘Path Planning Node Started’)

self.map\_subscription = self.create\_subscription(OccupancyGrid, ‘/map’, self.map\_callback, 10)

Self.target\_publisher = self.create\_publisher(PoseStamped, ‘/goal\_pose’, 10)

Self.map\_data = None

Def map\_callback(self, msg):

self.map\_data = msg

self.get\_logger().info('Received Map Data')

# Once map is received, plan a path

self.plan\_path()

Def plan\_path(self):

If self.map\_data is not None:

#take in the camera data to find the object and get the goals location

# achieve the goal pose in 3D space

Goal\_pose = PoseStamped()

Goal\_pose.header.freame\_id = ‘map’

Goal\_pose.pose.position.x = X

Goal\_pose.pose.position.y = Y

Goal\_pose.pose.position.z = Z

Goal\_pose.pose.orientation.w = W

Self.follow\_3D\_path(goal\_pose)

#implement the kinematic and path planning

Def follow\_3D\_path():

Def main(args=None):

Rclpy.init(args=args)

Path\_planning\_node = PathPlanningNode()

rclpy.spin(path\_planning\_node)

Path\_planning\_node.destroy\_node()

Rclpy.shutdown()

If \_\_name\_\_ == ‘\_\_main\_\_’:

Main()