Autonomous Maze Solving in ROS

CTE Workshop Sem 1 2021

Final Assignment

Prerequisites

1) **Dual booting**
   Documentation -
   https://linuxconfig.org/how-to-install-ubuntu-20-04-alongside-windows-10-dual-boot

   Youtube video -
   How to Dual Boot Ubuntu 20.04 LTS and Windows 10 [2020] | UEFI - GPT Method
   How to Dual Boot Ubuntu 20.04 LTS and Windows 10 [2020]
2) **Installing ROS**
   ROS wiki documentation -
   http://wiki.ros.org/noetic/Installation/Ubuntu (For Ubuntu 20.04)
   http://wiki.ros.org/melodic/Installation/Ubuntu (For Ubuntu 18.04)

   Youtube video -
   ROS1 - How to Install ROS Noetic on Ubuntu 20.04
3) **Omnibase repository**
   You need to clone this omnibase repository as a package in your workspace. Go through the read me of this repository to properly install it in your workspace. Make sure to change “melodic” to “noetic” depending on whatever your ros version is.
4) **Shapely Library**
   Installation-
   ```
   $ pip install shapely
   ```

   If you don’t have pip installed-
   ```
   $ sudo apt install python3-pip
   ```

   You don’t need to understand how shapely works, but you need to have it installed for the obstacle detection code we wrote to work.
5) **Matplotlib Library (optional but recommended)**
   Installation-
   ```
   $ pip install matplotlib
   ```

   Matplotlib is useful to visualize the path planned by your path planner.

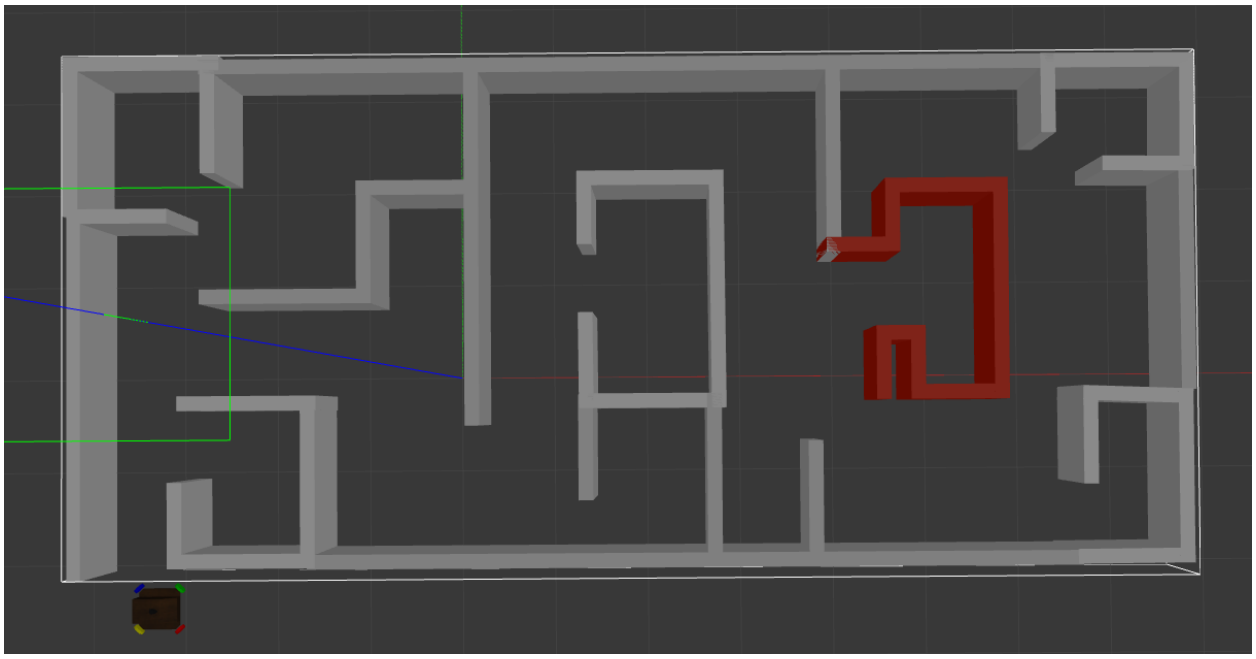
6) **Open CV**
   Installation-
   `$ pip install opencv-python`

   Tutorials-
   OpenCV: OpenCV Tutorials
   ROS Developers LIVE Class #86: How to use OpenCV with ROS - YouTube
   vision_opencv/Tutorials - ROS Wiki
7) **Go through CTE slides ;)**

Assignment

- The robot has to navigate through a given maze autonomously without colliding with any walls and to reach a final goal area which is marked in some color and after reaching, the robot has to detect the color of the walls surrounding the goal area.

- You need to clone the package from the git repository



- The goal region is the region surrounded by the coloured walls. The goal point provided is (3.2, 0.27). This is where the robot should finally stop.

- The robot spawns at the location (-5.2, -2.6). That is, this is the origin point in your tree which you will use for your path planning

- Due to some differences between how the coordinate system works, you must change all the coordinates that are planned by the path planner in the following manner.
  If (X, Y) is a node in your planned path to the goal, while feeding these values to the controller, change it to (X + 1.843515, Y + 0.6581).
  Basically, when you plan your path using the above source and goal points, you need to make these adjustments. Because, the robot is not really at (-5.2, -2.6). It's actually at (-5.2 +1.843515, -2.6 + 0.6581). You just need to make these changes to all the points (including the goal point as well).

- Do not account for the dimensions of the robot in the path planner, it has already been accounted for in the obstacles. That is, treat the robot as a point which has no dimensions.

- What we have provided in obstacle_detection.py:
  A function called isValidPoint(parentX, parentY, childX, childY)
  While plotting a point in **sampling based algorithms**, you can call this function to check if that point is valid or not, by passing the arguments - x,y of parent node (the existing node in tree that the child node is going to connect to) and, x,y of child node.
  The function will return true, if it is a valid point and false if it is not.

We have created a package to help you get started out with assignment. You can find the package her : https://github.com/pranavgo/Autonomous_maze_solving_assignment

You have to make the following in the package maze_solver:

1) **path_planner.py**
   A path planner only a **sampling based algorithm** (RRT, RRT*).
2) **controller.py**
   A PID controller to have the robot follow the path planned by your path planner.
3) **colour_detection.py**
   You can either write this in your controller or write a different script for this and call it separately when you reach the goal.

What you need to submit

You need to make a github repository of your workspace and upload all the scripts that you make in this repository. The following is the google form link in which you need to submit the link to your github repository:
https://forms.gle/99JnVWDnpV1q5DUV8

The deadline for submission is 14th Feb. For any queries contact any of the course instructors/mentors.