Navigation Robot, ROS project

Author:
koussaila KADI
Mickael Da Rocha Martins

Abstract

In this project, we worked on a turtleburger3 robot where we implemented the knowledge we developed during the practical sessions. The goal of this project is to program the robot in Python language, so that it performs certain tasks, such as following a line, avoiding obstacles, navigating in a corridor, and in an environment full of obstacles, and in the end the robot must reach a target. And among the knowledge that are developed and used during the realization, we site the image processing, signal processing, algorithmic.

Keywords: Robotics, ROS, Image processing, Simulation, Signal processing, navigation.

1 TABLE OF CONTENTS

2 SIMULATION STEPS

each challenge has its own launch file:
challenge1:
roslaunch challenge_project challenge1.launch
challenge2:
roslaunch challenge_project challenge2.launch
challenge3:
roslaunch challenge_project challenge3.launch

and at each challenge, the robot is positioned in the right place. with the spawn.

after each challenge, we should stop the execution by interrupting the process, and after that we run the next challenge.

Observation:

When copying the folder, be sure to make all nodes (python files) in the scr/scripts folder executable so that the launches run correctly.

3 CHALLENGE I, LINE FOLLOWING

The goal of this challenge, the robot must follow colored lines on the ground, and also must stop in front of a moving obstacle in the form of a barrier. The robot must follow its path as soon as the obstacle is no longer there. Then, the robot has to go around an immobile obstacle on its way.

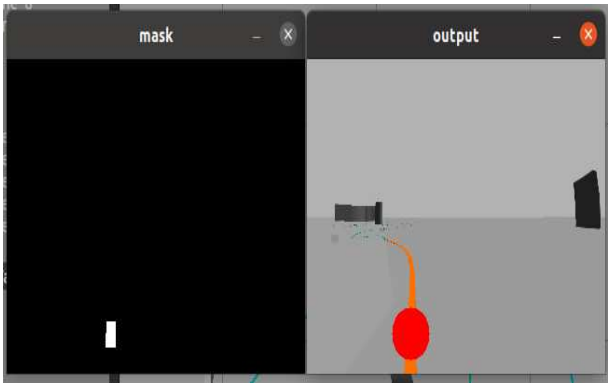

Figure 1: Image Processing

3.1 ARCHITECTURE

We create a ROS node **follow_line** which must subscribe to the topic '/camera/image' to retrieve the image data captured by the robot's camera, and also the topic '/scan' which returns the data captured by the lidar. This node publishes on the topic '/cmd_vel' to guide the robot.

The robot is equipped with a camera and a lidar.

1-Camera: returns real time images which are published on the topic '/camera/image' in simulation and /raspicam_node/image/compressed in real robot.

2-Lidar returns a vector of 360 values, each value corresponds to a degree on a unit circle.

3.2 ALGORITHM

In this part, we used a method of color detection on lines, which is based on image processing with openCV and CV_bridge. The idea is to use masks

on HSV represented images to limit the undesired effect of color intensity.

We used masks for each color (orange and blue) and then we binarize the images. so that at the end we get that the desired color. Then, we calculate the moment of the mask, and we recover its center (cx, cy) what is going to allow us to track the thread on the ground. and to publish the values adapted of command on "/cmd_vel" according to the color of the thread.

We have the blue line which corresponds to a path full of turns, which obliges us to reduce the speed of the robot compared to the orange line which has less turns.

All this we take into account if there is an obstacle or not on the path. If yes, then, we measure the size of the obstacle thanks to the data of the lidar, so we decide if it is a barrier to stop or a cylinder to go around it.

4 CHALLENGE II, MOVEMENT IN A CORRIDOR

The transition to challenge 2 is made when the robot arrives in front of a corridor. The robot must cross the corridor without hitting the surrounding wall.

4.1 ARCHITECTURE

We create the node **challenge2**, which will receive the data captured by the lidar via the topic '/scan', and will publish under certain conditions the appropriate movement commands for the robot, on the topic "/cmd_vel".

4.2 ALGORITHM

We use the data captured by the lidar, 360Â° around the robot. Then, we use this data vector to separate the data into four parts: front, right, left, back. This defines the vector sizes that we are interested in for data processing. For this challenge we needed the data from the walls in front, left and right of the robot to guide it.

So the robot goes straight as long as the average of the "front" vector we defined is not less than a dangerous distance from an obstacle. If the average of the front vector is less than this distance then the average of the right and left vectors are compared.

If "left" greater than "right" we go left and if "left" less than "right" we go right. We compare the data perceived by the lidar on the left and right front sides of our robot and then we go to the vector with the largest unobstructed area.

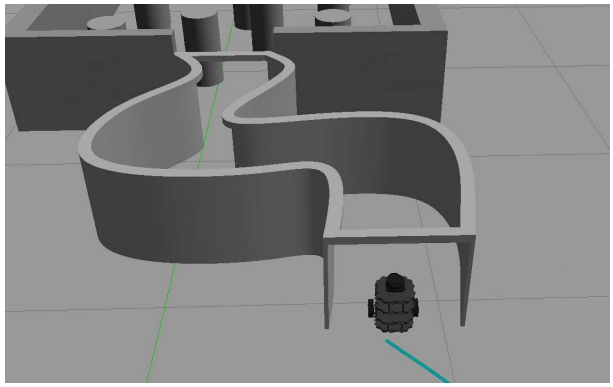

Figure 2: Lidar

5 CHALLENGE III, Navigation

The objective of this challenge is to make the robot travel through an area filled with obstacles, and reach the center of a colored target placed on the ground to stop.

5.1 ARCHITECTURE

we create the node "challenge3", which will receive the data captured by the lidar via the topic '/scan', and will publish under certain conditions a message for the commands of displacement on the topic "/cmd_vel". Same method of displacement as for the challenge2. We have not coded a method for detecting and stopping at the center of the target.

5.2 ALGORITHM

We used the same method as for challenge 2, except that we adapted the algorithm. In challenge 2 the robot is most of the time surrounded by obstacles. Here in challenge 3 the lidar will not always pick up the distance of the surrounding obstacles because they are located farther than the minimum distance to pick up the signal. For this part we only performed the navigation between the obstacles following the same technique as for challenge 2 by adapting the speed and the distance at which the obstacle becomes dangerous.

6 CONCLUSION

this project allowed us to work on real robots, and to simulate on gazebo and rviz environment. also it allowed us to understand and use all the equipments on the robot such as the camera for the recovery of the images, and the lidar to have information on the environment such as the obstacles

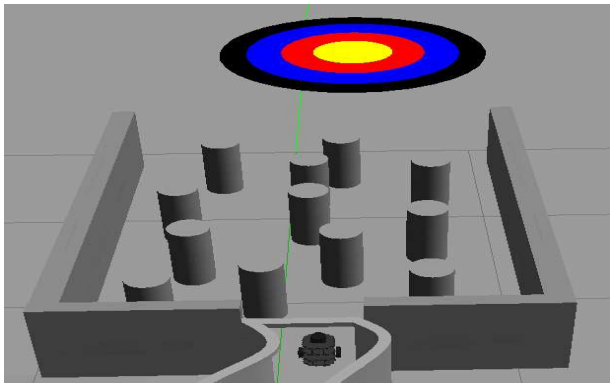

Figure 3: Navigation

References

[1] http://edu.gaitech.hk/turtlebot/line-follower.html.

[2] https://www.robot-advance.com/EN/art-turtlebot3-burger-1997.htm.

[3] vision_opencv - ROS Wiki. http://wiki.ros.org/vision_opencv (consulte le mai 15, 2021).