B3B33LAR - Semester project
COLUMN FINDER

Students Yevhenii Kubov, Mikhail Poludin

***Abstract*—Semester project description. Project task - given a Turtlebot and a virtual world in a Gazebo simulation, find the needed column standing somewhere on the map, calculate path and bump into it without touching any obstacles.**

April 20, 2021

I. Principles of algorithms

THERE are a couple separate problems needed to be solved in this task. Robot vision and image processing, controlling the current position of the robot, mapping the environment around to create a map, path search, movements of the robot and the combination of all the above in one piece of Python code.

A. Robot vision, detecting columns

Implemented using libraries such as OpenCV, Numpy, Pillow and Turtlebot control library. The main task is to detect columns around the bot and find their location. Image is taken from the robot's camera and also has a depth sensor. Threshold of the image

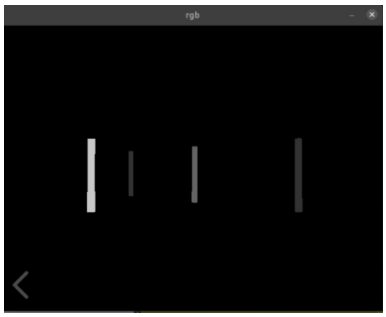

Fig. 1. Threshold of the cameras image.

by given colors of the columns gives us a picture with connected regions of colors. We can determine if a region is a column by comparing the relation of height and width of that component(for our columns it should be around 10:1).

B. Robot movements

Turtlebot library already has functions for moving and rotating the bot with a certain velocity. However, we wrote a couple help-functions for easier and more precise control. The algorithm of moving by a certain angle/distance is implemented using pretty precise robots odometry. It estimates current position of the robot relative to the last time odometry was reset.

C. Mapping of the environment

Using robot's depth sensor we can estimate the position of each column relative to static robot. Creating a map of 100x100 pixels, each representing 5cm square in real world. By calling mapping functions we place all the visible columns on it relative to robot's position(center of the bottom of the map). For not goal columns we draw a restricted area around them (10*10 pixels, 5cm each), so that robot doesn't touch them when moving. Robot itself is counted as a dot for simplification. The map is used in path finding algorithm.

D. Path search

For robot to find a path to a desired column we use A* algorithm that finds the optimal way to go. It avoids restricted regions on the map and tries to minimize the length of the path. After finding the path, we draw it on our map.

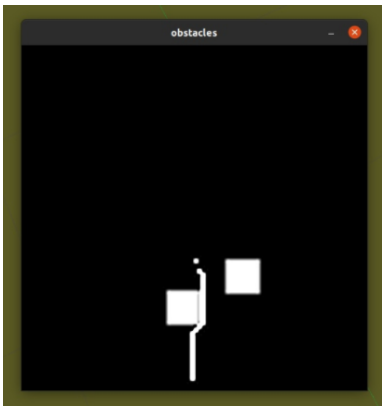

Fig. 2. Path representation.

E. Combining all of the above algorithms

To correctly evaluate the actions we used Finite-state machine principle. Main file contains multiple states such as START, ROTATED, PATH FOUND, DONE etc. When the last state is set, program only shows image and map windows, robot stands still.

II. Compiling and starting the program

To start the program, please:

- Start Gazebo simulation
- Go to a folder with the project
- In terminal: $python\ lar\_project.py < arg >$, where $< arg >$ is a color of a column to find $< arg > \in \{red, blue, green\}$.

III. Typical run of the robot algorithm

1. Once started, Turtlebot will rotate itself until column of needed color is found, stops when this column is in the middle of the robot's view.

2. When column is found, robot starts creating the map: puts columns on it and then draws restricted zones.

3. When the map is ready, A* algorithm is initialized and returns an array, where each element contains coordinates and direction.

Direction is represented as integer value, with a step of 45 degrees per unit. So, 0 equals 0 degrees, 2 equals 90 degrees etc.

4. Next step is iterating through the path array and moving step-by-step until goal is reached.

5. State is set to ”DONE” and robot stops.

IV. Results of the work

We managed to manipulate robot in the way that it can precisely go to a column and bump into it with assumptions:

- There is only one goal column of a given color on the map.
- The goal column is visible from a robots position without any need of moving except rotation.
- The distance between any of two columns is not less than 500mm.
- There is always a valid path for the robot to go.