Mobile Robot Challenge

Group: Cyberpunk 2024

Fien van Wetten	Kacper Kadziolka	Evan Meltz	Anca-Mihaela Matei	Giulia Rivetti
s2269651	s4115945	s3817911	s4004507	s4026543

June 20, 2024

Abstract

For the *Mobile Robot Challenge*, we have decided to work with the Picar-4wd + RPi Camera v3 (wide) robot. The main goal of this assignment is to program the given mobile robot such that, firstly, it is capable to return to its home position after a manual drive, and secondly it can return home, after a manual drive and then after detecting and touching a small object.

1 First challenge

The goal of the first challenge is to have the robot begin at an unmarked position. From there, the robot is manually controlled until the "q-key" is pressed, at which point it should autonomously return to its starting location. For this challenge, we were only allowed to use the RGB camera of the robot.

1.1 Solution

The main idea for our solution is a backtracking solution. Each time the "w-key" (forwards) and "s-key" (backwards) are pressed, the movement and the time between the previous key pressed are stored in an array. When the "a-key" (turn left) and "d-key" (turn right) are pressed, the image of the current location is also stored in the array, along with the movement and time since the previous key was pressed. In this way, we keep track of the movement history, the times between movements and the images that give a reference to the point at which the robot should turn. When the "q-key" is pressed, the robot loops backwards through the movement history and performs the opposite movement found in the movement history. Thus, if the "w-key" is pressed, the system executes a backward motion. The loop delays execution based on the duration retrieved from historical movement data. If the history movement is a right or left turn, the robot takes pictures during the turn, and when the robot is aligned with the reference picture, it goes to the next movement in the history. In the end, the robot should stop close to the starting point.

1.1.1 Comparing images

An important step in in this strategy is the comparison between the current image and the reference image, which represents the known orientation of the turn. Using the OpenCV library (Olli-Pekka & OpenCV, 2016) the absolute difference between these images is computed. This results in an image highlighting the dissimilarities between the two images. Then the mean value of the absolute difference in pixel intensities between the current and reference image is calculated. This provides a measure of how much the two images differ on average. In the loop, the mean difference value is compared with a predefined threshold. This threshold determines the level of similarity required between the current and reference images for alignment. If the mean difference is below the threshold, it indicates that the robot is sufficiently aligned with the reference image. In this case, the loop breaks, and the alignment process is considered complete.

2 Second challenge

The objective of the second challenge is to have the robot begin at a specific position and be manually operated until the "q-key" is pressed. Upon activation, the robot must autonomously locate an object, drive towards it, and touch it. After completing this task, the robot should automatically return to its starting position. Again we where only allowed to use the RGB camera of the robot.

2.1 Solution

For this challenge we have reused most of the code from the first challenge. So we use much of the same solution as described in section 1.1. During the manual drive we use the same history tracking. When the "q-key" is pressed, the robot will first detect a red object. Within this loop the robot slowly rotates to the right while capturing images. Each captured image is evaluated for the presence of a red object using colour thresholding in HSV colour space to detect red regions within the image. By counting the non-zero pixels within the thresholded region, it determines whether a significant amount of red is present in the image. The loop continues until a red object is detected, at which point it breaks out, indicating successful detection. To reduce potential alignment problems, an additional rotation is performed after red object detection to refine the robot's orientation. The aim of this adjustment is to remove any misalignment of the robot that may have occurred during the initial detection process. Then the robot moves forward until a certain amount of red colour coverage is reached in the image. This is done by calculating the coverage of red pixels in the captured images and ensuring that the robot advances until a pre-defined threshold of red colour coverage is reached. Upon successful interaction with the red object, the robot returns to the initial position where it detected the red object. The robot first moves backwards for a duration equivalent to the forward movement performed during the approach of the red object. Finally, the robot undergoes a left rotation to align with its original orientation in the same manner as described in section 1.1.1. After this, the same backtracking to the home position is preformed by the robot as in challenge 1.

3 Limitations

The main problem we encountered in both challenges was the correct orientation of the robot when backtracking a right or left turn. This happens when the environment is too similar, making it harder for the robot to find the correct reference images. This could be solved by adjusting the threshold. But sometimes the robot could not find the reference images at all, and then the robot continued to rotate. The reason for this problem is that the image comparison strategy is sensitive to changes in the environment. So if someone passed by during the manual drive and was no longer there during the backtracking, then the robot would not be able to find the reference image because of the change in the environment.

4 Conclusions

In this assignment, we programmed the Picar-4wd robot to accomplish two tasks: autonomously returning to its starting position after a manual drive, and locating and touching a red object before returning to the starting position. For the first challenge, we have recorded the movements of the robot as well as the captured images during the manual driving, enabling it to retrace its steps accurately. This approach, however, showed some flaws in a non-various environment, even though in general it led to good results. In the second challenge, we additionally used object detection; with color thresholding in the HSV color space, the robot could identify and go close to a red object before returning to its initial position. The main challenge with this task was ensuring precise orientation during the return phase, which we managed with an additional alignment step. Despite that, the robot was able to achieve good performance even for the second task.

References

Olli-Pekka, H., & OpenCV, t. (2016, 9). *OpenCV Python packages.* Retrieved from https://github.com/opencv/opencv-python

A Code instruction

Connect to the robot.
We used the opencv2 libary this can be installed with the following command:

```
$ pip install opencv-python
```

A.1 challenge 1

For the execution of the challenge 1, navigate in the terminal to the map where challenge1.py is stored. Run the following command in the terminal:

```
$ python Challenge1.py
```

Then the robot is ready to drive with "awsd-keys" when "q-key" is pressed the robot should return to the start position.

A.2 challenge 2

For the execution of the challenge 2, navigate in the terminal to the map where challenge2.py is stored. Run the following command in the terminal:

```
$ python Challenge2.py
```

Then the robot is ready to drive with "awsd-keys" when "q-key" is pressed the robot should start finding a red object drive towards it and than return to the start position.