Mobile Robot Challenge

Team A
Bohan Wei (s3600246) , Yuying Xia (s3772233),
Shihang Yu (s3240924), Tian Xia (s3669254)

June 2023

1 Introduction

The purpose of this Mobile Robot Challenge is to use a Raspberry Pi-based smart car kit, SunFounder PiCar-4wd, with a wide camera v3 to complete the following three tasks:

- The task 1 is shown as Figure 1a, the robot need to detect unknown objects and estimate the distance.
- The task 2 is shown as Figure 1b, it requires our robot to perform obstacle avoidance after detecting the object, go around the object and continue in the same direction.
- The task 3 is shown as Figure 2, the robot searches for an unknown object and gets as close to it as possible.

All three tasks are are implemented based on camera functions, not ultrasound. The size of the unknown object is approximately $7cm \times 7cm \times 7cm$, and the starting position A of the robot is known (as shown in Figure 1 and Figure 2). The object is clearly visible.

2 Task 1: Unknown object detection and distance estimation

In this section, our task is to detect the unknown object and estimate the distance so that the robot moves towards the object and stops as close as possible to the object. The necessary packages we used to solve this task are: **libcamera** [1] to take photos, **openCV** [2] for vision tasks , **imutils** [3] for images processing and **picar_4wd** to control the movement of the robot.

[1] https://github.com/raspberrypi/libcamera
[2] https://pypi.org/project/opencv-python/
[3] https://github.com/PyImageSearch/imutils

2.1 Edge detection and contour finding

First, we process the images which the camera captured. By converting the original image to grayscale, removing the noises and apply **Canny** edge detection, all the edges in the image can be found. Object is always clearly visible according to known conditions, so we can easily find the contour with the largest area using the **findContours()** function in openCV and return the minimum area rectangle enclosing this contour. The width of this rectangle can be used to calculate the apparent width in pixels.

2.2 Distance estimation

We utilize **triangle similarity** to estimate the distance between the robot and the object which has the largest contour in view. First we take a picture of the object at the initial position, and measure the distance from the camera to the object as a marker. Then, we could calculate the focal length of the camera [4]:

$$F = (P \times D)/W \tag{1}$$

where F is the focal length, P is the apparent width in pixels which we have already calculate in section 2.1, D is the distance from the marker to our camera, W is known width of the maker.

After the marking is done, we move the car forward to a new position and take another picture of the camera at the new distance. At this point, we can calculate the distance between the robot and the object:

$$D' = (W \times F)/P' \tag{2}$$

where D' is the calculate distance between camera and object and P' is the new apparent width in pixels.

The unknown object detection and distance estimation results are shown as Figure 3.

2.3 Robot movement

After obtained the distance information in section 2.2, the robot moves towards the object through the **fc.forward()** function in picar_4wd package, and the moving distance is the obtained distance minus 10cm. The reason for minus 10cm is to effectively avoid some errors in distance calculation and prevent the robot from colliding with objects. Taking Figure 3 as an example, the robot should move 64cm forward and then stop. Because of the 3cm errors, the position where the robot stops is 7cm from the object instead of 10cm.

3 Task 2: Unknown object avoidance

After we have completed the unknown target detection and distance estimation, the next step is to make the robot realize the obstacle avoidance operation according to this distance. The main idea is that the distance that the robot advances per second is fixed. Calculate the time needed to move forward, and then perform the turn and straight operation in obstacle avoidance.

[4] https://pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/

3.1 Robot steering

In this task, we use Picar-4wd, which has a packaged steering code, and can directly call fc.turn_right and fc.turn_left to realize the turning of the car, but this turning method cannot control the robot well for 90 degrees of turning, which will greatly affect the effect of the robot in the process of obstacle avoidance. To this end, we adopt a strategy of multiple turns during the turning process, and pause for 1 second after each turn. By adjusting the time.sleep parameter value and Continuous experiments have determined the best parameters to ensure that the robot can move relatively safely when avoiding obstacles.

The entire obstacle avoidance workflow is shown in the Figure 4. The robot is placed at point A. First, the method used by task1 is used to detect the unknown target and estimate the target distance. After the robot car obtains the distance parameter, it advances a certain distance. In order to achieve 100% For safe obstacle avoidance, we have added a fault tolerance range: that is, when the robot is 10cm away from the target, it will start to make a 90-degree right turn and advance a certain distance. In actual tests, we found that this advance distance needs to be It is slightly larger than the object's width of 7cm, about 9cm, so that it will not touch the object during the turning process, then turn left, advance 9cm, turn right and return to the original route and continue straight to complete the obstacle avoidance operation.

4 Task 3: Unknow object search

In this section, our task is to find an object within a given scene range and control the robot to move towards the object and stop as close to the target as possible. The scene is shown in Figure 2, **picamera2** [5] is a necessary package to solve this task.

4.1 Object filtering and center calculation

To achieve this task, we need to implement real time object color filtering with picamera2 before the object searching. The positions of the robot and object are shown in Figure 2. We first convert each frame from BGR (Blue, Green, Red) color space to HSV (Hue, Saturation, Value) color space, the example is shown as Figure 5 [6]. By adjusting the three thresholds value of H, S, and V, the upper and lower color range of the object can be found. Once we defined the color range, we applied a mask to the frame to filter out all the pixels that fall within this range. The resulting masked frame will only contain the pixels that match the specified color range to isolating the object. After the target area is obtained by color filtering, we use the contour detection function **cv2.findContours()** in OpenCV to find the contour of the object and mark it by drawing a green rectangular box.

On the other hand, we also need to calculate the center coordinates of the object and the frame. By calculating the difference between the coordinates of the object center and the coordinates of the frame center, and comparing it with the threshold value of 50 (used to determine the tolerance range between the center of the object and the center of the frame), it can be determined whether the position of the object is in the center of the frame.

[5] https://github.com/raspberrypi/picamera2
[6] https://github.com/freedomwebtech/colortracking

4.2 Object searching

The search task is based on capturing real-time video by Raspberry Pi camera module. We first turn on the camera at the initial position A of the robot, and then judge whether the camera detects the object framed in section 4.1. If the camera does not capture any object, the robot turns right 15 degrees (the steering method is in section 3) and continues the target search. If the camera captures the target, it needs to determine whether the object is in the center of the frame, when the object is not in the center of the frame, the robot still turns 15 degrees to the right. When the object is in the center of the frame, the robot moves forward and approaches the object. As the robot gets closer and closer to the object, the target will gradually deviate downward from the center position, and the robot stops moving.

5 Conclusion

In summary, our group completed three challenges using edge detection and color detection methods, which obtained good results with acceptable margin of errors. However, both methods are easily affected by factors such as lighting, latency of real-time captured video etc. In future works, we can continue to optimize and tune parameters, or fine-tune fault tolerance thresholds and employ robust filtering techniques to minimize errors.

Appendix

A Introduction

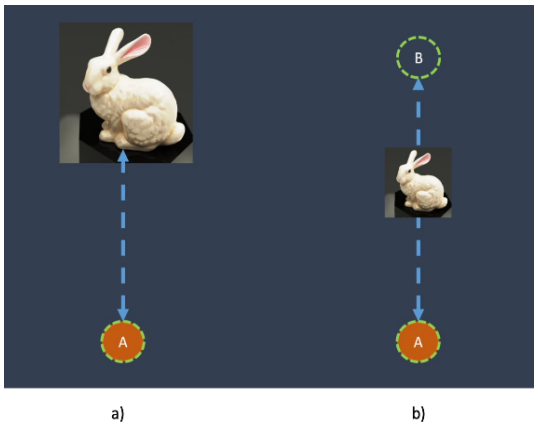

Figure 1: a) Unknown object detection and distance estimation. b) Unknown object avoidance.

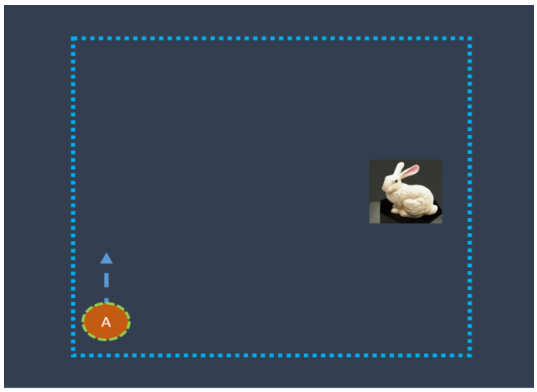

Figure 2: Unknown object search.

B Task 1

Figure 3: The object width is known as 7cm (1) and the distance of the left figure is 120cm. The figure on the right is the result of unknown object detection and distance estimation. The object is framed by a green box, and the distance is displayed as an integer in centimeters. The actual distance was 71cm, which has a 3cm error from our measured 74cm.

C Task 2

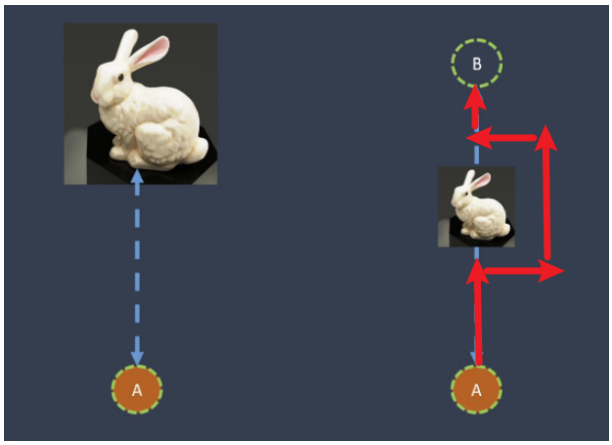

Figure 4: Schematic diagram of robot obstacle avoidance. The size of the unknown target object is 7*7*7. After starting the robot and the obstacle avoidance program, the robot automatically drives to the front of the target object, and automatically avoids the object and continues to move forward, The red arrow indicates the route of the robot

workflow about oa

D Task 3

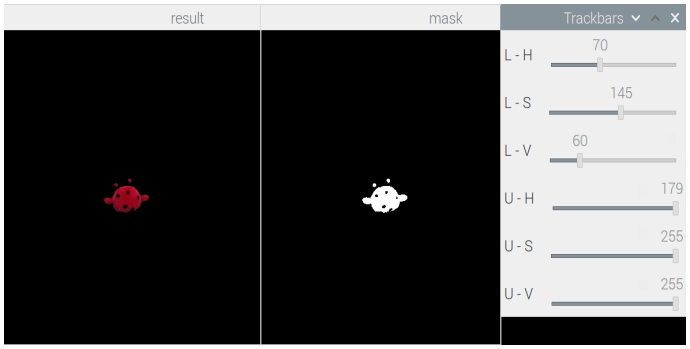

Figure 5: The result of object color filtering. By adjusting the values of the six trackbars, we can define the lower and upper color ranges, which is used to find the object with this color range in the frame. L represents the lower threshold, and U represents the upper threshold. H, S, and V in the HSV color space correspond to hue, saturation, and lightness, respectively. By adjusting these parameters, different color ranges can be selected for filtering. The example object is a red ladybug toy.