Mobile Robot

by
Zhihao DAI
Keli ZHANG
Changrong CHEN
Suraj Kumar
Yihang AI

COP518 Robotics & Intelligent Systems
Coursework Report

Loughborough University

Feb. 2020

Abstract

This report will be discussing how we designed and implemented a robot that mimics Braitenberg behaviors, more specifically love, aggression, fear and curiosity. We will discuss the robot that was used and its specs and configurations alongside the way we implement it followed with testing and its results. The main focus of the report will be how we used the robot to mimic certain actions and behaviors. This report will also include how we made the robot track a specific object/human and how implemented a following/tracking action.

Contents

List of Figures

Chapter 1

Introduction

This project is based on a mobile robot that has been built with latest AI technology and various sensor capabilities. It could be programmed in Python. It has a latest Realsense camera from Intel, which could capture both colour RGB image and depth image and be processed with a Nvidia nano board which supports image processing and deep learning. It also has multiple proximity sensors at its front, sides and back.

The task we have done in this project is programming its behaviours and test them to see if it works fully functionally. There are two types of behaviour we have been working on, Reactive Behaviours and Following Behaviours.

1.0.1 Reactive Behaviours

In this part we have designed robot's behaviour simulate Braitenberg's vehicle behaviour. There are four different behaviours: Love, Fear, Aggression as long as curious. We have used a black backpack as our targeted object and used deep learning to train the robot using its camera to recognize that specific backpack in a complex environment. When the robot has detected the targeted object, it will initiate the behaviour that we have been assigned.

1.0.2 Following Behaviours

In this part we programmed the robot to follow a human that has been detected by its RGBD camera and not only follow its movement, but also keep a safe distance from targeted human.

For each behaviour, aspects of assumptions, system design, implementation and test & analysis are described in details.

Chapter 2

Robot Configuration

The robot we have used has many sensors and motors alongside accompanying software. We use the hardware to control the robot and to gather information of the robot's surroundings.

2.1 Hardware Components

2.1.1 Motors

The chassis of the robot is a Lynxmotion Aluminium A4WD1 Rover Kit (w/ Encoders) which contains built in motors and wheels. There are 4 different motors that are default with the rover kit which are 12.0vdc 30:1 gear head motor attached to 4.75" tires and wheels. Each motor can reach up to 7500 RPM and a Motor Driver HAT I2C interface has been installed to the rover kit so that in software we can drive two DC motors at the same time. This allows us to perform forward, forward-left, forward-right, left, right, backward-left and

Figure 2.1: Mobile Robot in This Coursework.

Figure 2.2: Camera on the Mobile Robot.

backward-right motion.

2.1.2 Camera

The camera that is on the robot is an Intel RealSense Depth Camera D435i which is a colour and depth camera with the addition of an inertial measurement unit (IMU). The IMU refines the camera depth awareness when the camera is moving (i.e. useful for when the robot moves). The depth camera has a depth output resolution and frame rate of 1280x720 active stereo depth resolution up to 90 FPS. The camera RGB feature has a resolution of 1920x1080 with a frame rate of 30 FPS.

There robot has an added Grove IMU 10DOF v2.0 installed at the bottom of the chassis so that we can see the angle of the camera. An issue with IMU is that there can be a drifting affect when stopping so by having two IMU's it compensates for the drifting for more accurate robot movement.

2.1.3 Wi-Fi Module

An Intel Dual Band Wireless-AC 8265 Wi-Fi card is installed onto the robot. This allows for remote connectivity and internet access on the robot.

2.1.4 Range Sensor

There is a range sensor added to the robot which is a VL53L1X Time-of-Flight Distance Sensor Carrier with Voltage Regulator. This sensor is a fast and accurate sensor that ranges up to 4 meters. It uses time of flight of invisible, eye-safe lasers to measure absolute distances independent of lighting conditions and target object characteristics such as colour and shape.

2.1.5 Jetson Nano Developer Kit

The NVIDIA Jetson Nano Developer kit is a small AI computer that is installed onto the chassis of the robot. This board contains everything needed for us to be able to program the robot and provides access to most of the library functions that were used to control the robot.

2.2 Software Components

2.2.1 Jupyter Notebook

The software we used to write the code for the robot is called Jupyter Notebook. This is a small cloud IDE that is connected to the robot. For the project we wrote all our Python code in the Jupyter Notebook including running the code, viewing the output of the robot and seeing the information provided by sensors and the camera.

2.2.2 Python & Libraries

The robot is programmed using Python and the library used to control the robot and its components is called JetBot which is a open-source robot based on NVIDIA Jetson Nano (that is installed on the robot).

Additionally, built in Python libraries such as Numpy and OpenCV are also used to make programming the robot easier.

Chapter 3

Reactive Behaviours

We are going to design a robot which could do multiple Braitenberg's behaviours, which has four specific behaviour, they are love, aggression, fear and curious. Firstly, finding a good targeted object is very important. It needs to be easy to recognize while robot is swing on the floor. By several testing we have decided that a backpack is the best static targeted object to use in labs. Because it not only contains different unique features, but also the size of the object is suitable for camera to recognize. Before it starts doing behaviour, it needs to wonder around to find targeted objects. Once it has targeted objected in its bounding box, it starts the behaviour that we designate. For each unique behaviour, we need to consider circumstances for each one of them.

3.1 Assumptions

Several assumptions are made in our design of reactive behaviours.

1. There is no obstacle between the object and the robot.
2. There is only one object of interest in the environment.

3.2 System Design

3.2.1 Flow Chart

The algorithm of reactive behaviour is shown in Figure 3.1. After we initiate the robot, the robot will rotate itself to scan around the area until it finds the targeted object. While RGBD camera successfully lock the object, it will put a blue bounding box at it, then it will

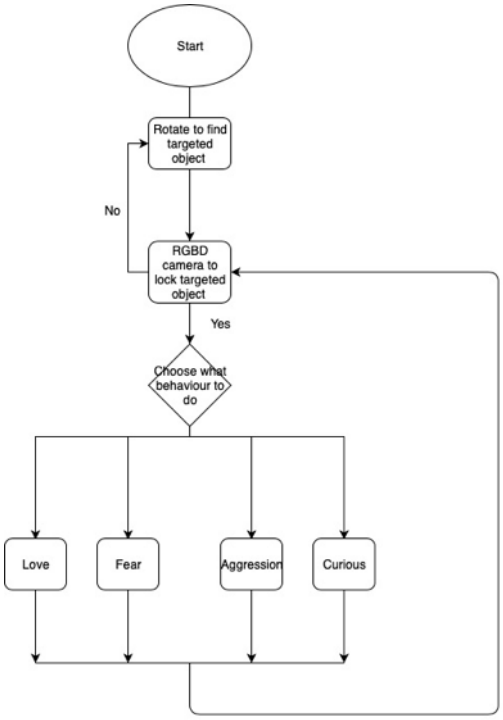

Figure 3.1: Flow Chart of the Mobile Robot with Reactive Behaviours.

further process the image that it has been read to make sure it is the real targeted object, at that time, bounding box will become red and it will process what behaviour task we have been given in our computer. After one behaviour has been finished, it will start the previous process again and wait for command.

3.2.2 Aggressive Behaviour

Aggression means feelings of anger or antipathy resulting in hostile or violent behaviour; readiness to attack or confront in English, so we want robot to react some critical movements after it successfully located target. A significant accelerate of speed will be the key point of aggressive behaviour. It also should be stop from a safe distance to avoid collision. So, Depth sensor function in camera will be useful in this circumstance.

Figure 3.2 describes Aggressive Behaviour in details. For Aggressive Behaviour, after robot taking instructions, it will initiate its motor to a stable speed, while depth camera has a reading that targeted object has beyond safe distance, it will centralise its direction towards object, then it will accelerate significantly towards targeted object. While depth camera has reading that it reached safe distance, the motor will stop to avoid collision to

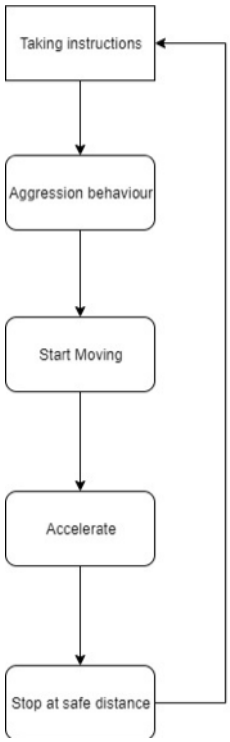

Figure 3.2: Aggressive Behaviour.

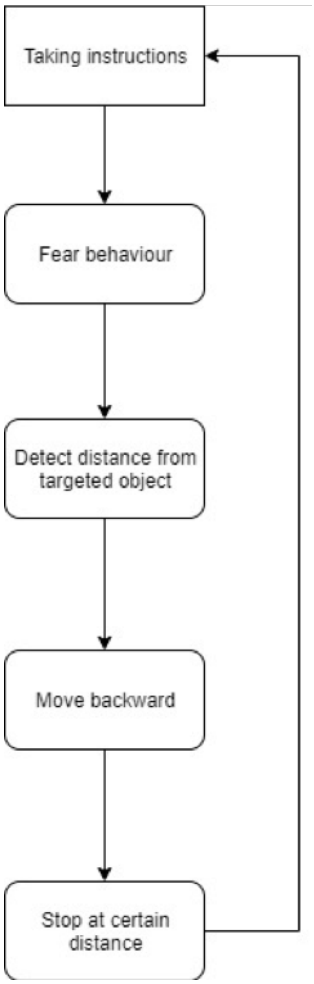

Figure 3.3: Fear Behaviour.

the target.

3.2.3 Fear Behaviour

In opposite to Aggression, fear behaviour should keep a distance from the target. So RGBD camera should be use as well. While robot has detected the targeted object, it firstly using depth camera to read distance between object and itself, then it will move backward in a significant speed. The speed should also decrease as the distance between targeted object increase.

Figure 3.3 describes Fear Behaviour in details. In contrast to Aggressive Behaviour, Fear is doing opposite movement, it also starts by wondering around. While targeted object has

been detected and confirmed, it will quickly move backwards from the target. There is also a deceleration of speed that we have designed at this part, which means the farther away the robot is from the targeted object, the slower it moves.

3.2.4 Curious Behaviour

Curious means eager to know or learn something, as a robot with only movement physical function, we define curious as a movement that contains moving forward and moving backward constantly. While targeted object is detected in camera's sight, it will seek if the target is at left side or right side, then it will move left forward or right forward correspondingly.

Figure 3.4 describes Curious Behaviour in details. Curious Behaviour uses same direction recognition technique as 'Love' behaviour, by defining what location of targeted object that robot is facing; For example If targeted object is on left side of the robot, robot will move left forward towards the target, while it getting closer, it will moving backward, and moving towards the target again until it reaches the safe distance.

3.2.5 Love Behaviour

Love is hard to define for a simple physical movement robot since it's an emotional vocabulary. We defined love as move slowly and shake its head while moving. When targeted object has been detected, it will turn left, then go forward a bit, then move right, forward a bit. It terminates the movement after reaching the safe distance from object.

Figure 3.5 describes Love Behaviour in details. When t robot is taking instructions of doing 'Love' behaviour, it will firstly find targeted object, after successfully put the bounding box on the targeted object, it will define whether object is on its left or on its right, then it will turn its direction to the opposite, then it will move forward a bit and turn its direction again, to do this movement is simply because wants robot to combine this series of movement, after-all the robot will shakes its head as he walks towards the target.

3.3 Implementation

Task 1 was to implement the Braitenberg behaviours which are love, fear, aggression and curiosity. Each of the behaviours the robot needs to first find the target we have specified. This is done within the process function which first calculates all the detected objects in the view of the camera.

```
1 detections = model(imgsized)
```

We then filter out the detections found to give us an array of only objects that we are interested in.

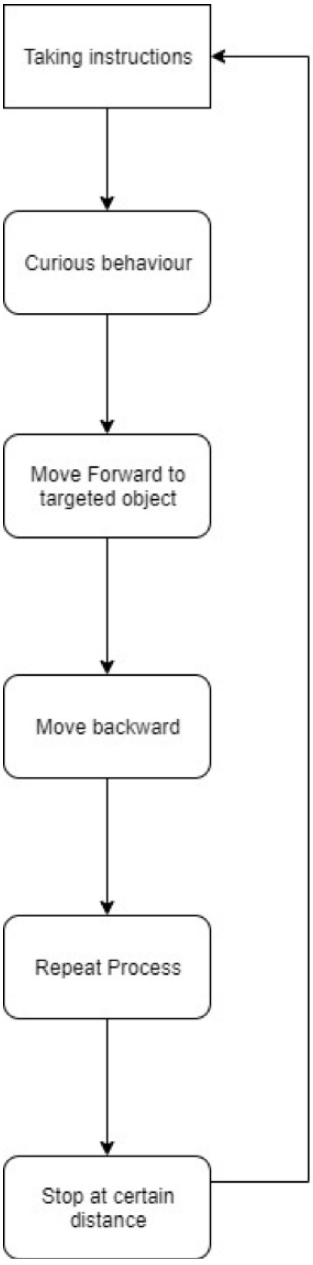

Figure 3.4: Curious Behaviour.

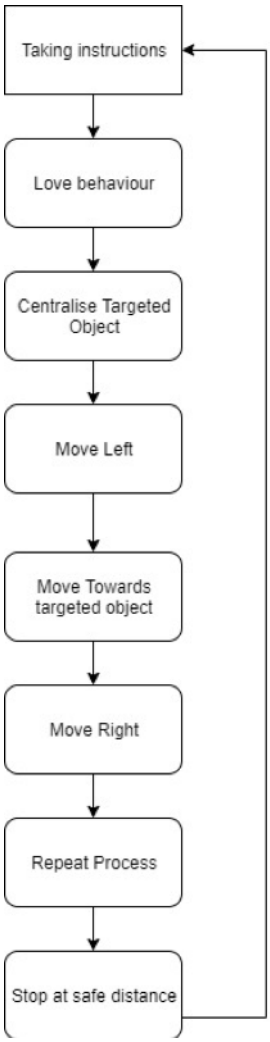

Figure 3.5: Love Behaviour.

```
matching_detections = [d for d in detections[0] if d['label'] == int(
    label_widget.value)]
```

Finally, each detection has a confidence field which is used to show the confidence (0-1) that the object is the one we want. If the confidence is less than 0 (i.e. not our object) we draw a blue bounding box around it otherwise if the object is what we want, we draw a red bounding box around it.

Additionally, we keep a count of the number of objects that the camera sees so we have a counter of all the objects on the left-hand side of the screen, centre of the screen and right-hand side of the screen.

This is done with the following code:

```
if (bbox[0]+bbox[2])/2 < 2/5.0:
                left_count = left_count + 1
            elif (bbox[0]+bbox[2])/2 > 3/5.0:
                right_count = right_count + 1
            else:
                center_count = center_count + 1
```

When the robot does not find the object we want, the robot does a 360 rotation (starting left or right depending on how many objects there are on that side, so if there are more objects on the right hand side, the robot will start turning towards the right). Once the object has been detected, the emotion that has been selected will be executed.

A dropdown widget has been added to allow users to specify the behavior they want. By default this is set to love.

3.3.1 Aggressive Behaviour

The aggression behavior is implemented by first having the robot move towards the side of the screen that has the most objects. Once the target object has been located, the robot will move towards the object increasing its speed the closer it gets to the object. Once the robot has reached a distance of 1200 or less, the robot stops. The gradual speed up from slow to fast going to the object is how we have implemented aggression.

The code for agression:

```
    if distance < 1200:
        robot.stop()
    elif left_count > right_count and left_count > center_count:
        robot.forward_left(max((5000-distance)/2500*2+0.3, 0.3))
    elif right_count > left_count and right_count > center_count:
        robot.forward_right(max((5000-distance)/2500*2+0.3, 0.3))
    else:
        robot.forward(max((2000-distance)/2000*2, 0.3))
```

The code shows how the robot stops if too close and how it moves left, right or center depending on what side of the screen the object is on. The speed is increased as the distance is shortened.

3.3.2 Fear Behaviour

Fear is implemented by first havimg the robot robot move in the direction where there are the most objects (left, centre or right) then once it detects the target object and is less than 2500mm in distance, the robot goes backwards in reverse quckly. The robot can be observed to slow down the further away it is from the object.

This is done with the following code:

```
            if distance < 2500:
                robot.backward(max((3000-distance)/3000*2, 0.3))
```

We choose that 3000 is the max distance we can do this from and that the speed can range between 0-1. Thus this gives the affect of fear with the control of speed.

3.3.3 Curious Behaviour

For curiosity we have made the robot go towards the area that has the most amount of objects. This is because we believe that where there are more objects, the robot is more curious. The robot will stop if the distance between the object is less than 800mm otherwise it will either go left, right, or center depending on object count on that side. The code to see the behavior is in the curious function were a direction is specified and the movement happens accordingly. For the curious behavior we have made the robot go forward and backwards towards the target object once it has been located.

3.3.4 Love Behaviour

The way love is implemented is once the robot has found the target object, the robot moves left, moves forward, moves right and then moves forward again. This results in the robot moving in a sort of short ziggzag fashion towards the object. This can be observed in the love function. Additionally, to keep the robot on track towards the object, in the process function we check if the distance is bigger than 100mm and depending on what side of the screen the most amount of objects are, we turn the robot towards that direction (i.e. if there are more objects on the left, we make the robot turn left). Once the robot reaches a safe distance of 900mm the robot stops.

The love function:

```
    def love():
```

```
        global love_flag
        if love_flag == 0:
            robot.left(0.5)
        elif love_flag == 1:
            robot.forward(0.3)
        elif love_flag == 2:
            robot.right(0.5)
        elif love_flag == 3:
            robot.forward(0.3);
        love_flag = (love_flag + 1) % 4
```

The love_flag is used to keep track of the movement that had been performed. This is so that we can control the movement of the robot in stages.

3.4 Test & Analysis

Our group test the task 1 in the computer master students Lab. In this Lab, we put the robots on the carpet. Because there are too many chairs and people. So, we decide to use the cup first. But is too small to recognize. Then we test the keyboard, but it still hard for robot's camera to recognize. Finally, we decide to use the black bag as the object. Because it is big enough to be easy to recognize.

3.4.1 Aggressive Behaviour

When we test this function, after choosing the aggressive. The robot will approach target slowly. When it close to the object, the robot will accelerate significantly. In order not to damage the robot, we design to stop in front of the target. Instead of going randomly, we design the if the robot cannot find the object. The robot will rotate in place, until find the object.

From the capture in the video shown in Figure 3.6, we can find that the process of accelerate is very clearly. It can show this function is accomplished perfectly. During the test, we found that the need some distance to show this function obviously.

3.4.2 Fear Behaviour

When we test this function, after choosing the fear. The robot will leave the object quickly. Just like the real fear. At first, we set the stop distance as 2500. During the test, the 2500 is too far away sometimes the camera will lose the object because the distance is too big, then the robot will rotate in place to find the object again. In order to this situation. We change the distance as 1800.

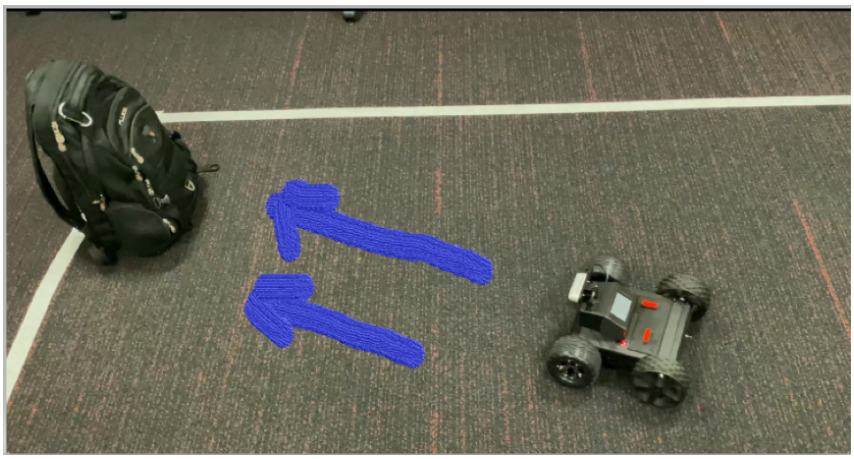

Figure 3.6: Testing of Aggressive Behaviour.

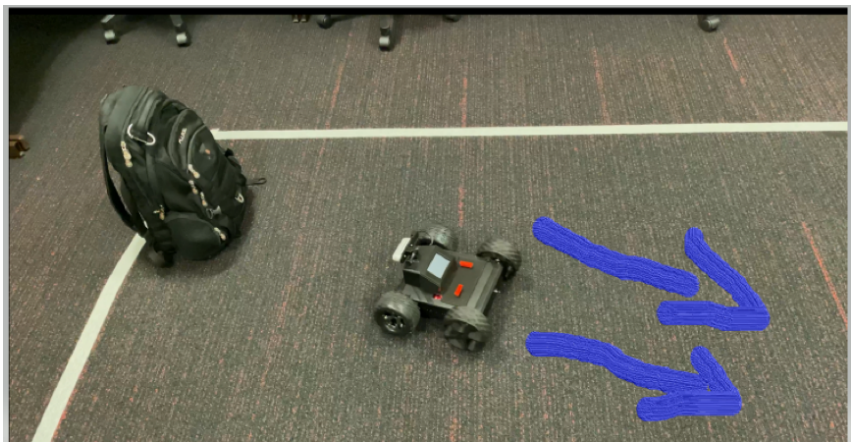

Figure 3.7: Testing of Fear Behaviour.

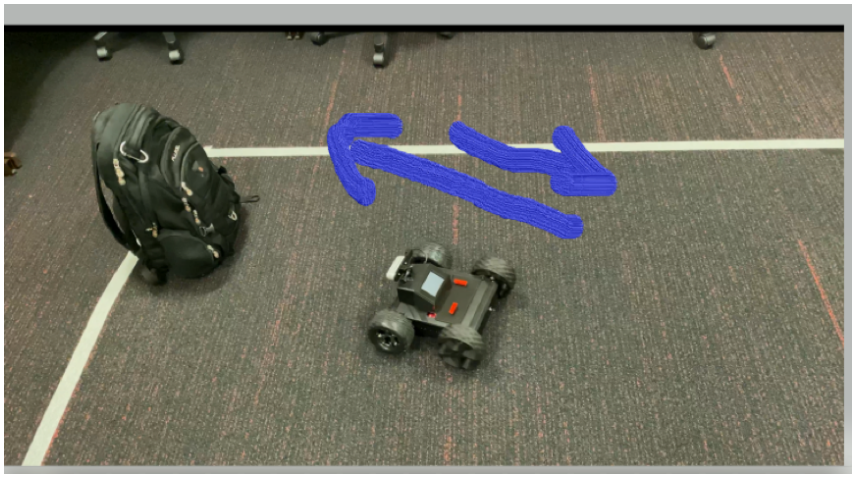

Figure 3.8: Testing of Curious Behaviour.

From the capture in the video shown in Figure 3.7, we can find that the distance 1800 is pretty appropriate for the test. The leave process is obvious. The function is completely accomplished.

3.4.3 Curious Behaviour

When we test this function, after choosing the curious. The robot will find where is the object, then forward and backward during the forward process. There are some questions when we test this function. At first, we set the forward or backward speed to high. Then the camera also easy to lose the object. So, we turn down the speed. After that, the result is better than before.

From the capture in the video shown in Figure 3.8, the robot forward and backward and during this process, the camera still can get the object. So, this function is successful.

3.4.4 Love Behaviour

When we test this function, after choosing the love. The robot will find where is the object, then turn left and turn right, like shake the robot. During the test we find some question, the robot is hard to find the project during turn left and right. It is easy to lose the object when the process of making a turn. At first, we segment the frame to 3 part, left, right and center. Then we found it is easy to lose the object. So, we change the frame and segment it to 5 part. The left two parts as left. The middle part as the center. And the right two parts as the right. The robot turns left or turn right by in bounding box which part contain the most objects. After do this, the robot is better to find the object during the turning

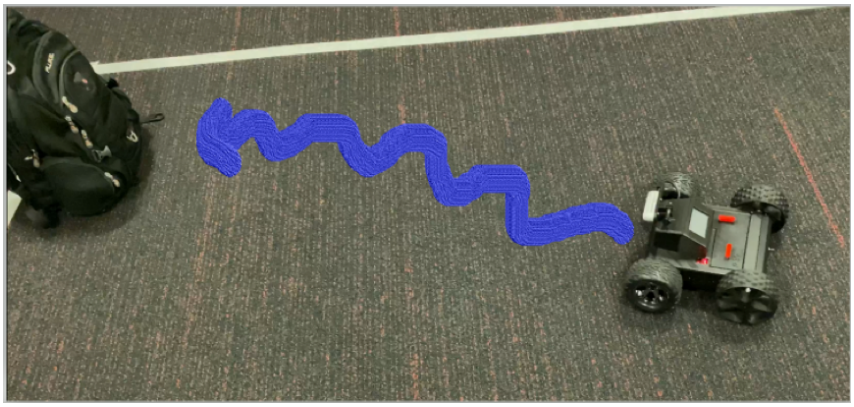

Figure 3.9: Testing of Love Behaviour.

process.

From the capture in the video shown in Figure 3.9, we can find that the camera can recognize the object from beginning to the end. And perfectly complete the love function turn left and turn right during the forward process.

3.4.5 Facing Backward to the Object

In order to test the stability of the function. We also test some special situation not test before. For example, the robot is facing back to the object, side to the object and no object at all.

In this situation, we put the black bag in the back of the robot, the robot will make a turn in place for about 180 degrees angle to find the object in the back of the robot. After find the object, the expected outcome is the same with the actual outcome. The robot can find the object quickly.

Just like the capture in the video shown in Figure 3.10, we can find it can find the object in the back of the robot.

3.4.6 Facing Sideways to the Object

In this situation, we put the black bag in the side of the robot, the robot will make a turn in place for about 90degree angle to find the object in the side of the robot.

From this capture shown in Figure 3.11 we can find the robot can find the object in the side of the robot.

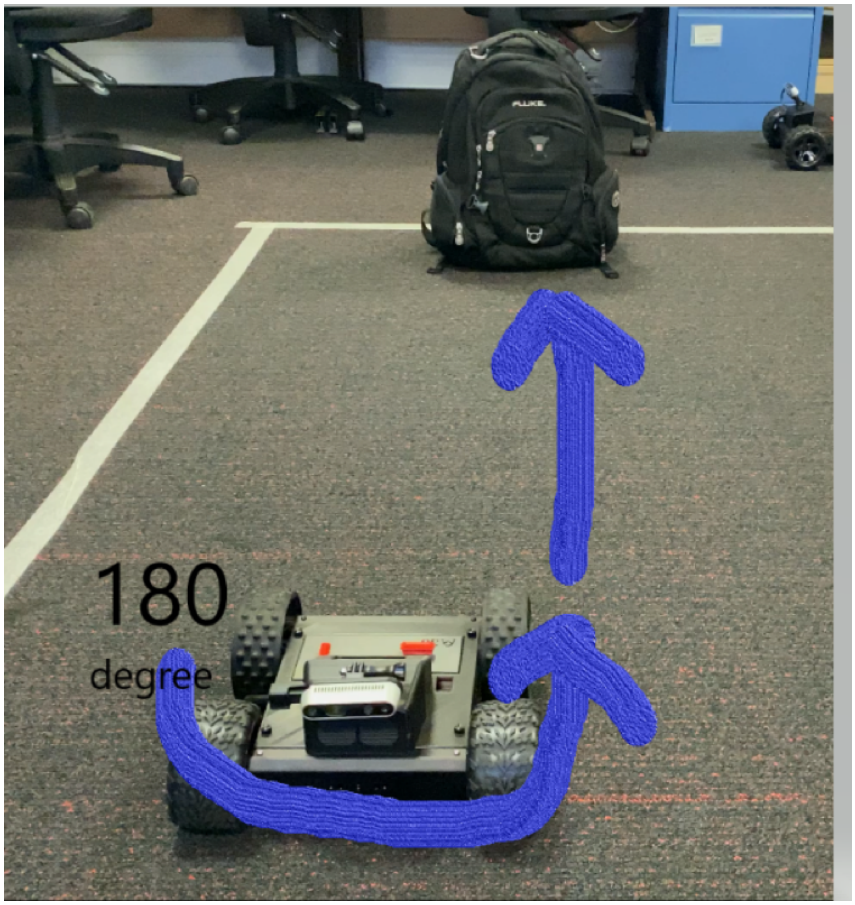

Figure 3.10: Testing of Facing Backward to the Object.

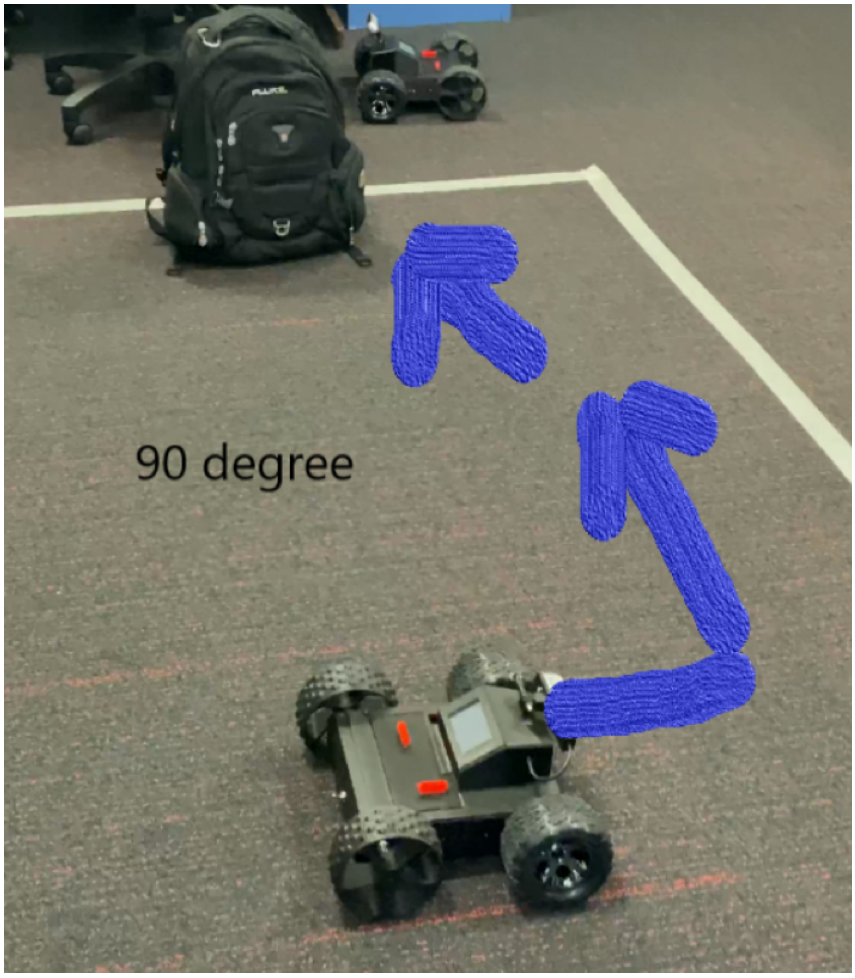

Figure 3.11: Testing of Facing Sideways to the Object.

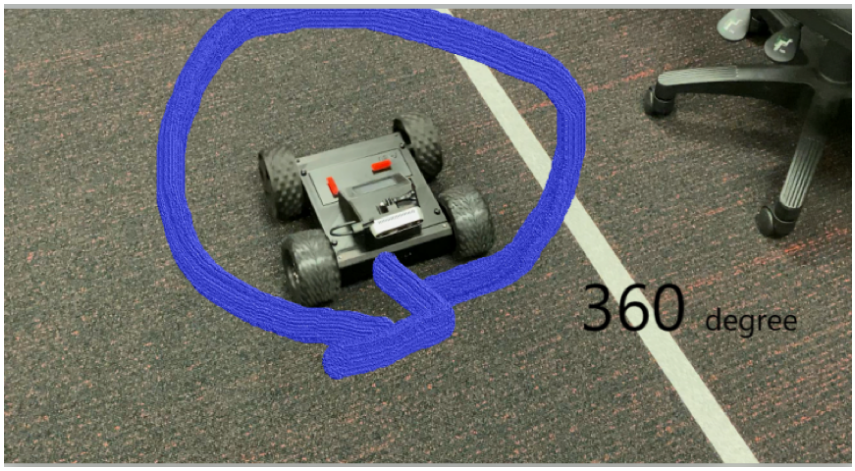

Figure 3.12: Testing of No Object in the Environment.

3.4.7 No Object in the Environment

In this situation, we do not put anything around the robot.

From this capture shown in Figure 3.12, we can find the outcome the robot is turning in place for 360 degrees again and again.

Chapter 4

Following Behaviours

In this chapter, we describe the design, implementation and testing of a mobile robot with following behaviours. The mobile robot follows a designated person, while keeping a safe distance. When the designate person is not on the scene, the robot should wander around to look for the person.

4.1 Assumptions

Several assumptions are made in our design of following behaviours.

1. There is no obstacle between the person and the robot.
2. When more than one person are on the scene, the robot is given instruction on which person to follow. That is, a person is designated to be followed.
3. The designated person is dressed differently to other persons on the scene such that the robot can tell the difference.

4.2 System Design

4.2.1 Flow Chart

A flow chart of the mobile robot is shown in Figure 4.1. When the robot starts and receives the first image from the camera, it locates all persons on the image using an object detection model [1, 2]. If no person is found from the camera image, the robot activates "Wander Behaviour." If one or more persons are located, for each person located, a bounding box and a confidence score are given by the model.

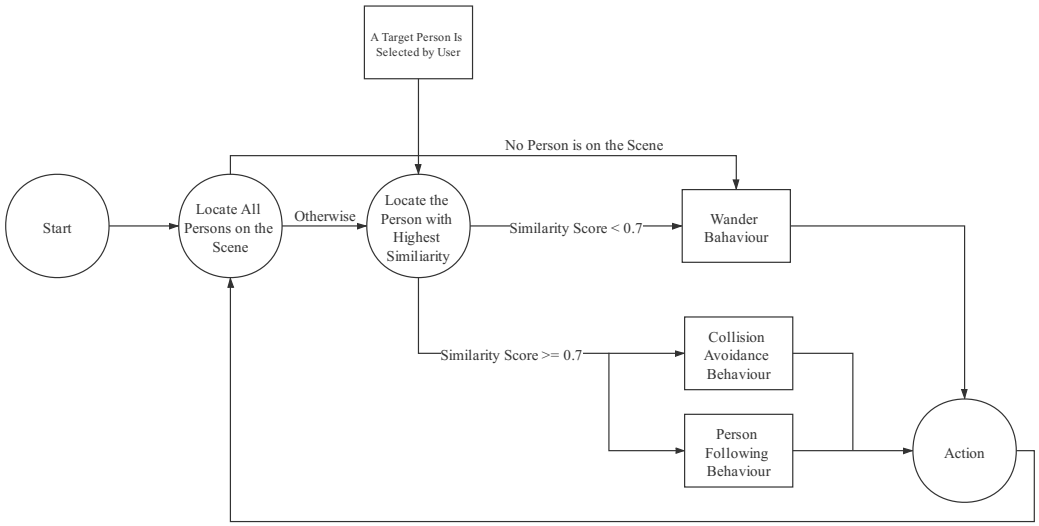

Figure 4.1: Flow Chart of the Mobile Robot with Following Behaviours.

Each bounding box is resized and compared against the target image for a similarity score using the euclidean distance. The person with the highest similarity score is selected. During the comparing stage, the user may also change the target image to any bounding box of located person. By default, the target image is a blank image.

If the highest similarity score is lower than 0.7, "Wandering Behaviour" is activated. Otherwise, both "Collision Avoidance Behaviour" and "Person Following Behaviour" are activated. However, the output action from "Collision Avoidance Behaviour" has higher priority than "Person Following Behaviour".

Once the robot finishes the action and receives a new update from the camera, it will repeat from locating all persons on the scene again.

4.2.2 Wandering

Our initial design of Wandering behaviour is to keep moving forward and turn left when an obstacle is found in front of the robot. However, we observe that such behaviour leads to frequent dead ends and bumping due to the size of the robot and the complexity of our environment. The initial Wandering Behaviour is replaced with self-rotating as shown in Figure 4.2. That way, the robot will keep rotating in the same direction until a person is located. It will not move into the dead end or bump into the obstacle.

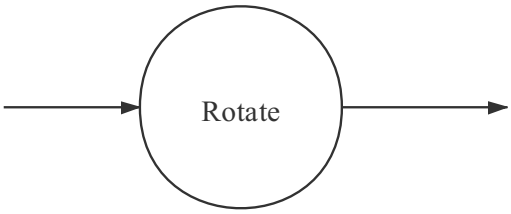

Figure 4.2: Wandering Behaviour of the Mobile Robot.

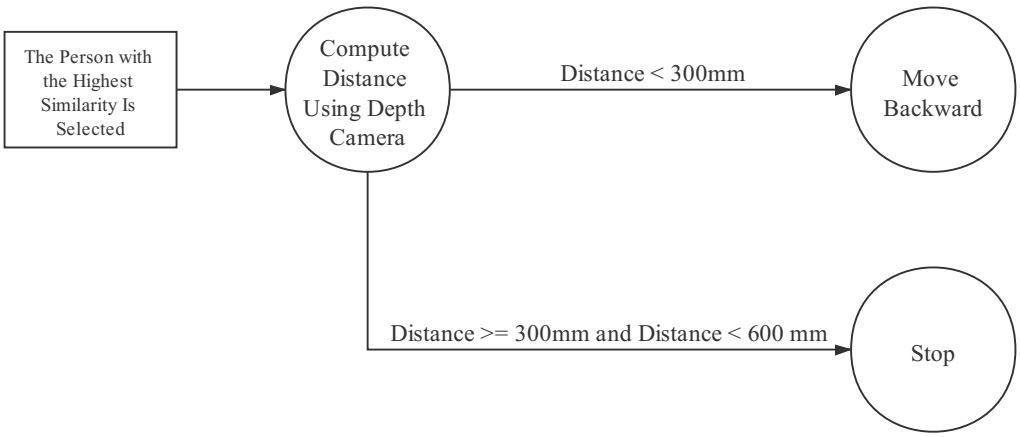

Figure 4.3: Collision Avoidance Behaviour of the Mobile Robot.

4.2.3 Collision Avoidance

Collision Avoidance Behaviour prevents robot from bumping into the designated person and always has higher priority than Person Following Behaviour. Figure 4.3 describes the behaviour in details. The distance between the robot and the designated person is computed using the camera's depth information. If the distance is smaller than 300 mm, the robot moves backward to avoid bumping into the person. If the distance is equal or larger than 300 mm and smaller than 600 mm, the robot stops moving. Therefore, the robot always keeps a safe distance between 300 and 600 mm from the person when it stops moving.

4.2.4 Person Following

Person Following Behaviour keeps tracking the designated person and moving the robot to the person. Figure 4.4 describes the behaviour in details. The target image is first replaced with the new bounding box if the similarity between them is larger then 0.9. This action

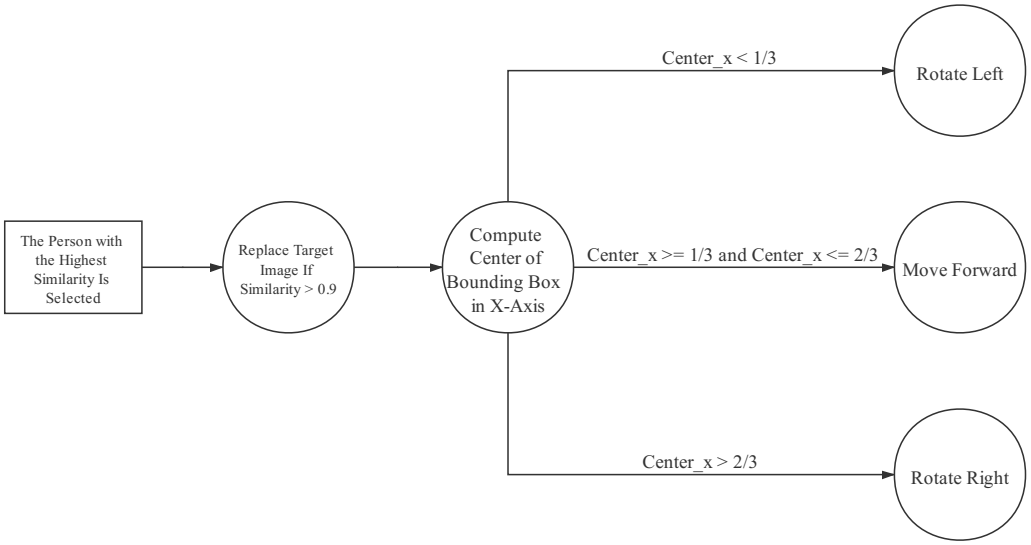

Figure 4.4: Person Following Behaviour of the Mobile Robot.

keeps the robot updated with the information of the target. It prevents the robot from decreasing similarity score over time and losing the target in the end.

The center of the bounding box in x-axis is later computed to determine whether the person is on the left, center or right. If the person is on the left side, the robot rotates to the left and vice versa. If the person is in the center area, the robot moves forward.

4.3 Implementation

The following behaviours are implemented in Python Programming Language and Jupyter Notebook environment. The implementation uses the source code from Tutorial 5 as the framework and improves upon that. Several important adjustments are made and described as follows.

4.3.1 Computing Similarity Scores

After all persons on the scene are located using a object detection model, a similarity score between bounding box of each person (variable named `obj`) and the target image (variable named `target`) is computed. Both `obj` and `target` are resized to a resolution of 300-by-300 using `cv2.resize` function. Both `obj` and `target` are converted to HSV color space using `cv2.cvtColor` function and later normalised through being divided by 255. The similarity score between `obj` and `target` is defined $1 - distance_{euclidean}(obj, target)$.

4.3.2 Computing Distance

In Collision Avoidance Behaviour, the distance between the designated person and the robot is computed. Applying the bounding box of the person to the depth image, a matrix of depths inside the bounding area is retrieved. The distance between the person and the robot is simply the minimum non-zero value in the matrix.

4.4 Test & Analysis

Two testing scenarios are performed and analyzed in this section.

4.4.1 Single Person Following

In this scenario, only one person is presented in the environment. The robot is expected to follow the person around the room.

Figure 4.5 shows an video capture of our testing scenario. As the person walks around in the room, the robot keeps following in a safe distance and never loses track of the person.

4.4.2 Single Person Following under Attempted Hijacking

In this scenario, two persons are presented in the environment. One person is chosen as the designated person to follow. The other person makes several attempts to hijack the robot by stepping in front of the designated person and then walking away. The robot is expected to always follow the designated person and not follow the other person.

Figure 4.6 shows an video capture of our testing scenario. The camera image is shown on the left, the bounding box the person in the center and the target image on the right. The robot distinguishes between two persons by computing the similarity score between the bounding box and the target image.

As the designated person walks around in the room, the robot keeps following in a safe distance and never loses track of the person. The other person makes several attempts to hijack the robot. However, the robot distinguishes between the two persons and does not follow the other person.

Figure 4.5: Testing of Single Person Following.

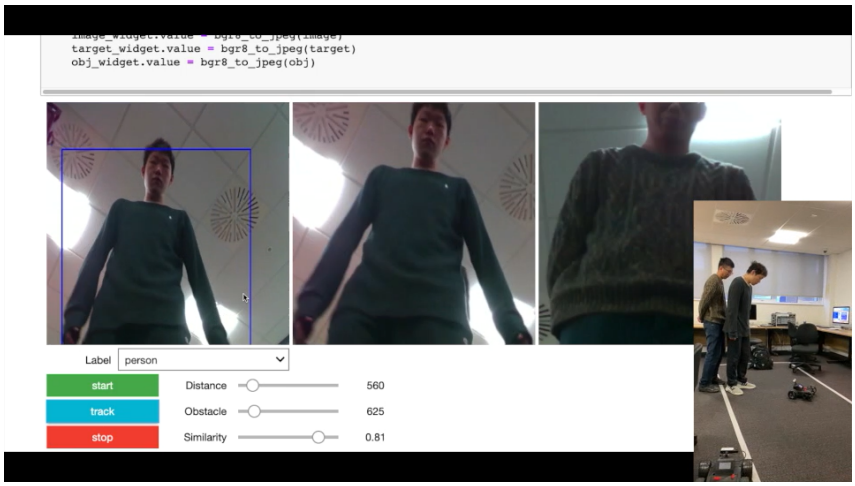

Figure 4.6: Testing of Single Person Following under Attempted Hijacking.

Chapter 5

Conclusions

In this coursework, both Reactive Behaviours and Following Behaviours of mobile robot are designed and implemented. The robot with Reactive Behaviours mimics Braitenberg behaviors, more specifically love, aggression, fear and curiosity. The robot first wanders around the environment to locate the object of interest (backpack in this case). Once the object is located, a Braitenberg behavior is performed by the robot.

The robot with Following Behaviours keeps following the designated person on the scene while keeping a safe distance. The robot first wanders around the environment to locate the person of interest. Once the designated person is located, the robot moves toward the person while maintaining a safe distance. In the scenario of possible hijacking by another person, the robot distinguishes between the two persons and only follows the person with the highest similarity score.

Testing procedures are proposed and performed on both tasks. The testing results show that our design and implementation is correct and robust.

References

[1] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot Multibox Detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.