Loughborough University

# Robotics and Intelligent Systems

## Group 3

Anirudh Singh Sisodia (F215611)

Darshan Ravichandiran (F216970)

Godlove Otoo (F219655)

Rajdeep Sengupta (F217897)

Sabarivelan Ganesan (F218923)

Sindhuja Arulmani (F218893)

Gowtham Kesavan (F219073)

# Table of Contents

# Specification and Analysis

## Task Description

This project uses mobile robots built with the latest AI (Artificial Intelligence) and sensing technology. The tasks involve implementing reactive and following behaviors on a mobile robot.

**Reactive behaviors**

The Robot demonstrates the following behaviors:

- Exploration:

Robot searches for the colors in the environment by moving forward right and scanning the whole area.

- Aggression:

On seeing red in the environment at a certain distance, the robot becomes aggressive by moving backward and forward.

- Fear:

On seeing yellow in the environment at a certain distance, the robot becomes fearful by moving backward in a zigzag manner.

- Love:

On seeing blue in the environment at a certain distance, the robot becomes happy and moves forward towards it.

- Curious:

On seeing green in the environment at a certain distance, the robot becomes curious and spins in front of it.

**Following behavior**

The Robot follows a person and keeps a safe distance, avoiding collisions with people and other objects.

## Robot Description

The robot has the latest Realsense camera from Intel, a Nano board from Nvidia that supports image processing and deep learning etc., proximity sensors, IMU, motors and driving circuits. The colors and depth image from an RGBD sensor (Intel Realsense D435i) is accessed.

## Environment Description

Environment used for Part A:

- The execution of reactive part A was done in MSc laboratory (N109), where we used certain colored objects for triggering the emotional response. A barricade was created to restrict the playground area of the robot.
- Within this environment we made use of a yellow card identity holder, a can with a green lid, a red bag, and blue shoes.
- Since this was a partially observable environment there were certain uncontrollable factors such as the lighting in the room.

Two environments were used for Part B:

- The first environment was a section of the MSc laboratory (N109). Chairs were used as the environment's walls and school bags as obstacles. The obstacles were stationary and there was no knowledge of the environment's layout.
- The second environment was the corridor of the MSc laboratory. Two different obstacles were used in this environment; a bag and a fire

extinguisher (of varying width). The obstacles were also stationary in this case, and the environment's layout was not known in advance.

# Design

## Reactive Behavior

- **Flowchart**

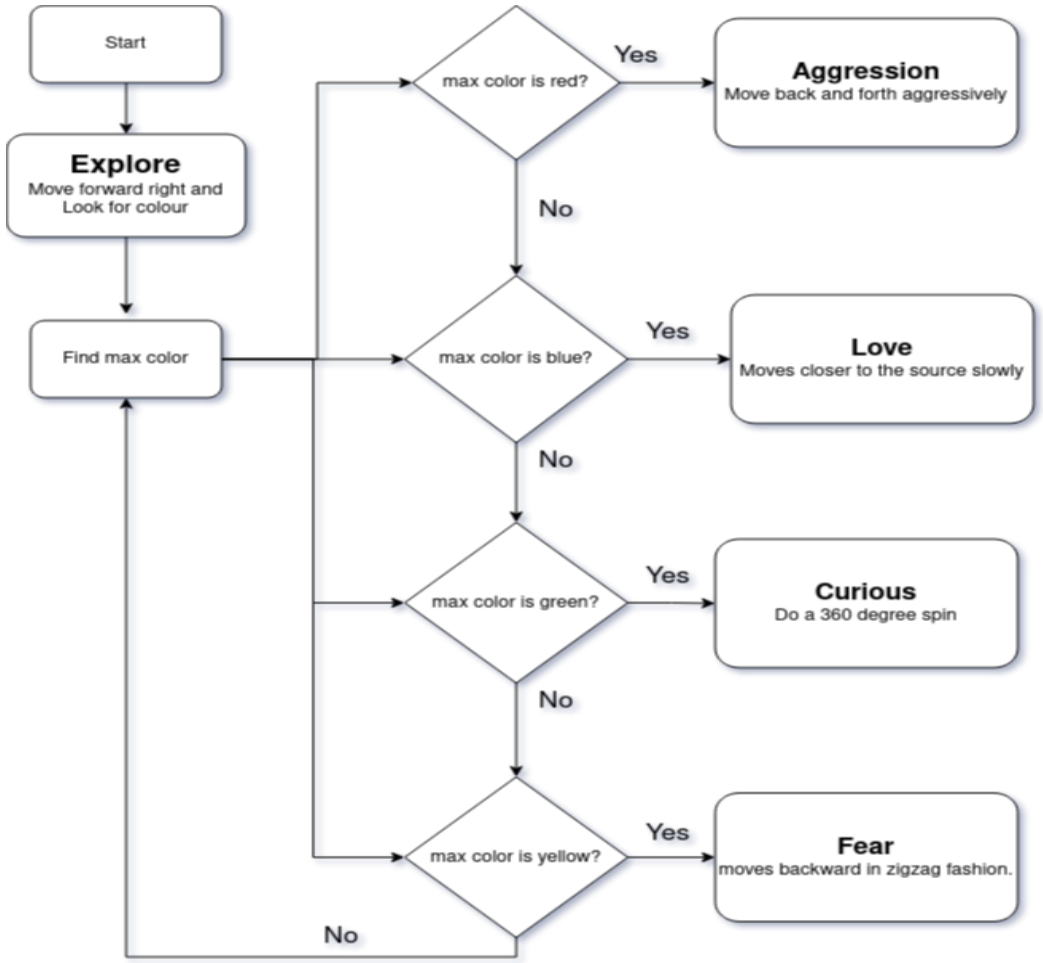



Figure 1: Flowchart for Reactive behavior

The above flowchart explains how the robot exhibits certain behaviors namely aggression, love, fear, and curiosity. The robot initially starts by exploring (by moving forward right) the environment for specific colors which represent the mentioned emotions.

The image captured by the camera is segmented based on colors, and the reaction will be based on the intensity of the color that has the highest frequency.

Upon capturing the colors in the picture frame, the robot calculates the dominant color in that picture by using a custom hash-based algorithm. This algorithm simply captures the intensity of the colors and returns the color with the maximum intensity.

For example, if at a certain point of time the picture frame has red as a dominant color then the robot will behave aggressively by moving backward and forward for a certain period. After which it starts exploring for new color.

Similarly, if the robot finds blue as the dominant color, then it will show behaviors of love which are represented by slowly moving towards the object and stopping there for a few moments. Also, if yellow is seen as the predominant color, then it will act in fear by moving backward in a zig zag fashion.

Moreover, if the robot sees green as the dominant color, then it will do a 360-degree spin.

- **State Diagram:**

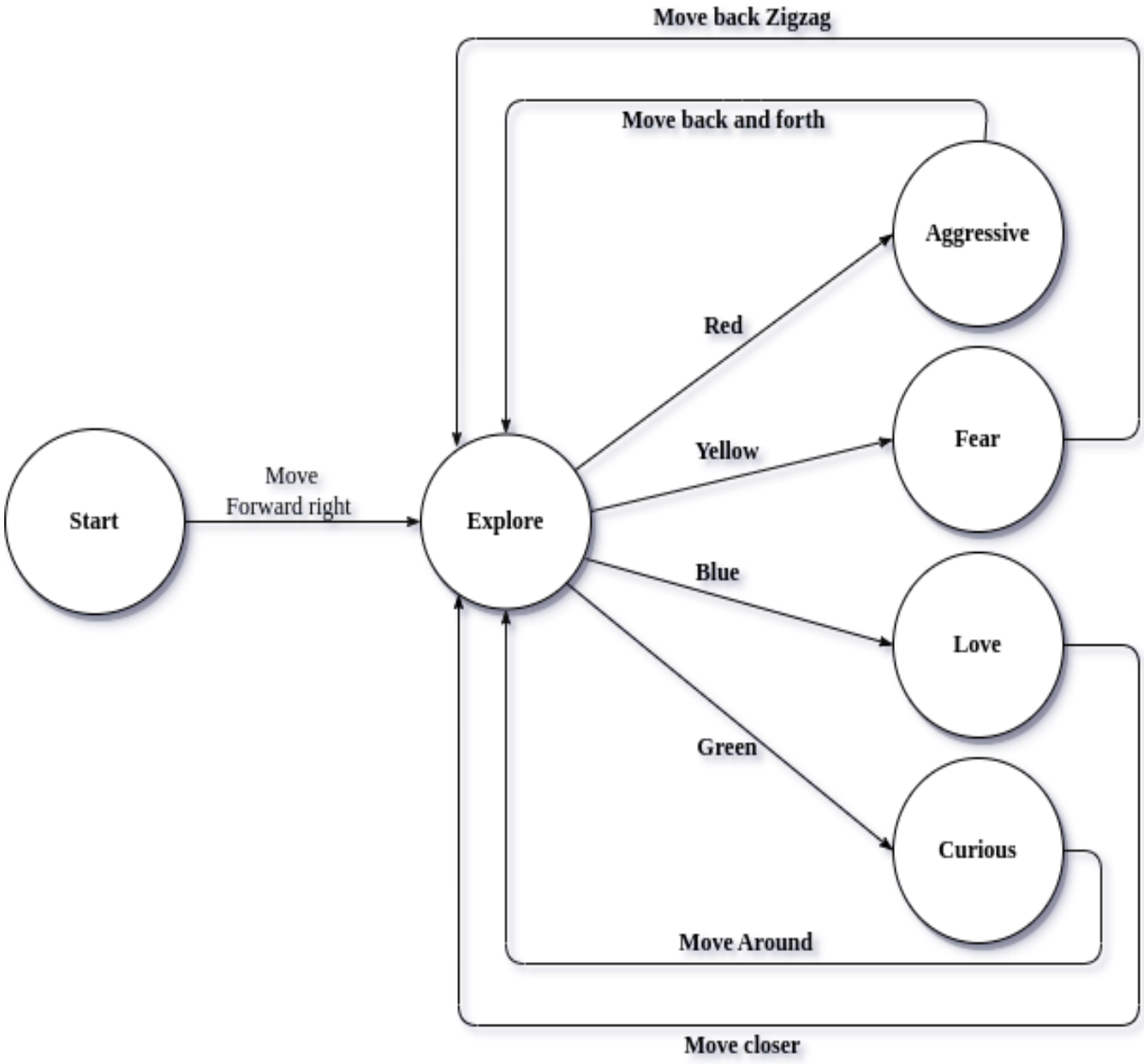



Figure 2: State diagram for reactive behavior

| q | σ | $\delta(q, \sigma)$ |
|---|---|---|
| **Start** | Move Forward Right | Explore |
| **Explore** | Red | Aggressive |
| **Explore** | Yellow | Fear |
| **Explore** | Blue | Love |
| **Explore** | Green | Curious |
| **Aggressive** | Move back and forth | Explore |
| **Fear** | Move back zigzag | Explore |
| **Love** | Move Around | Explore |
| **Curious** | Move closer | Explore |

Table 1: State table for Reactive behavior

- **Behavioral Diagram**

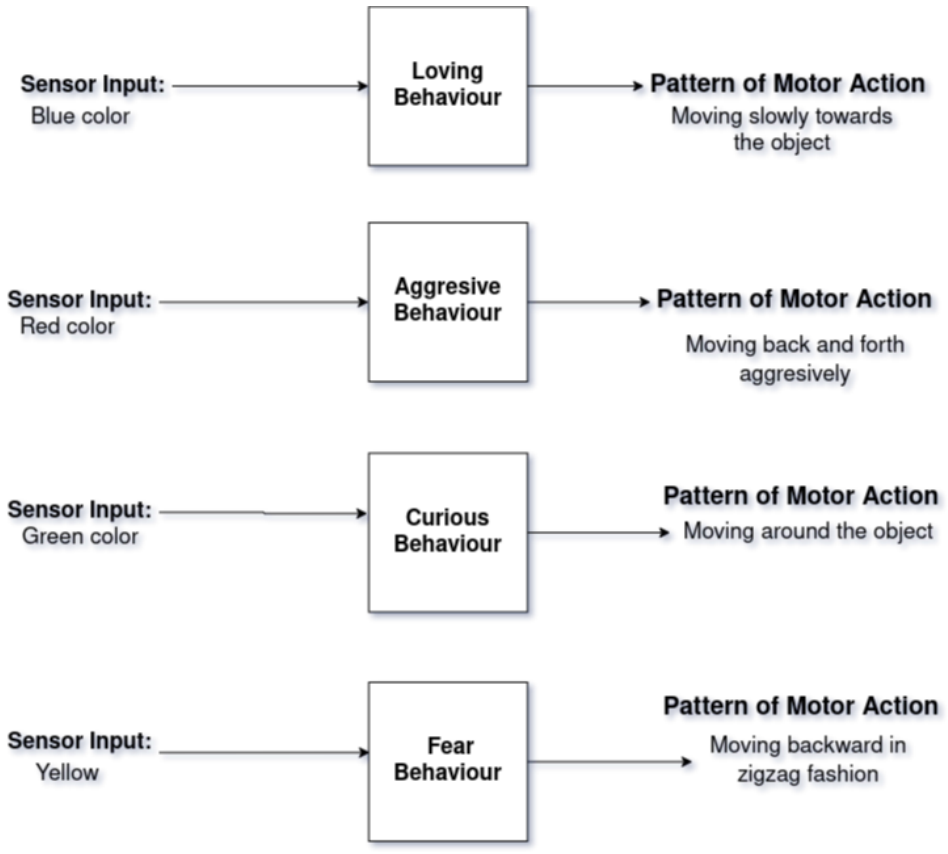



Figure 3: The above diagram represents the individual behavior in greater detail.

## Following Behavior

- **Flowchart:**

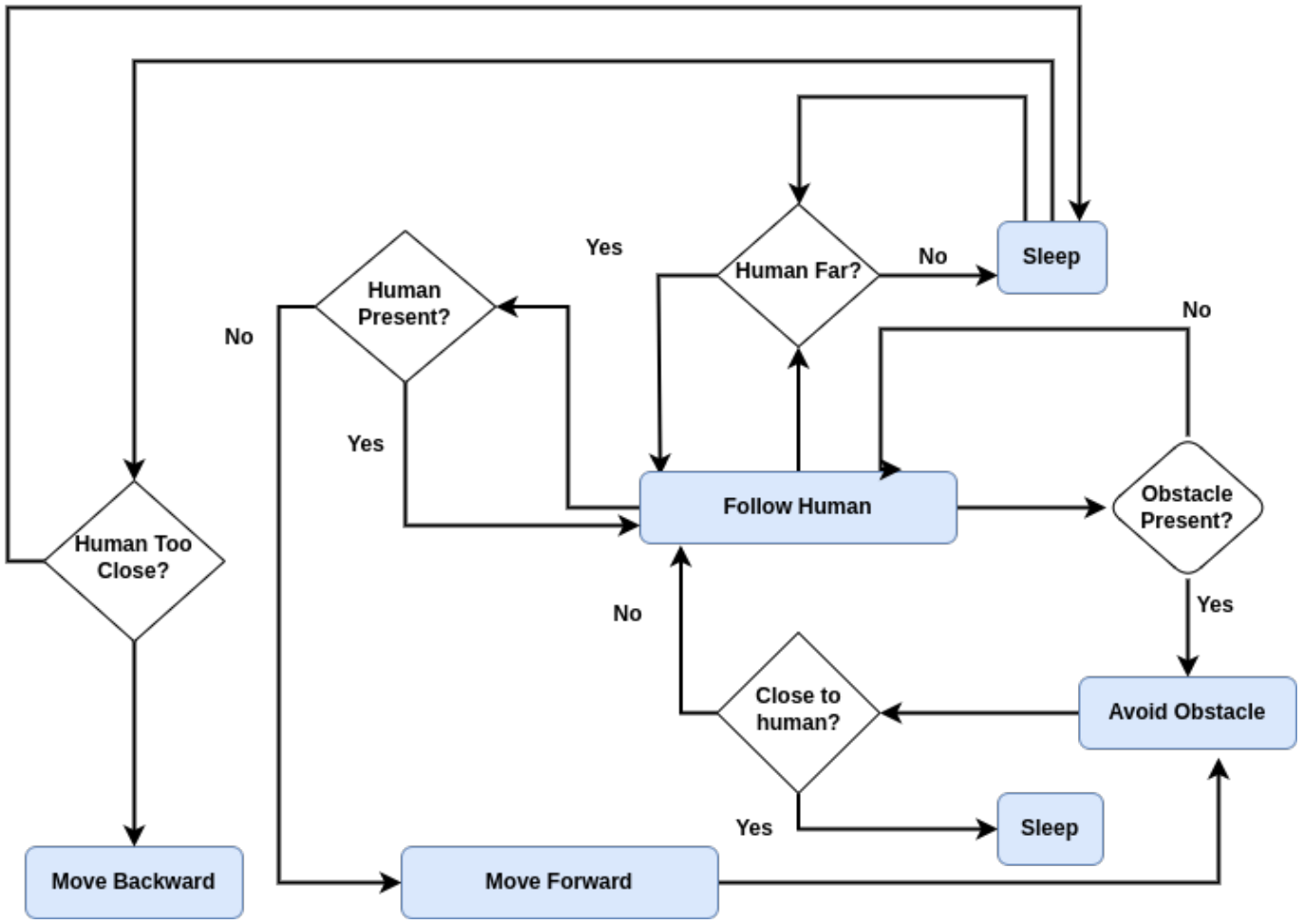



Figure 4: Flowchart for following behavior

This flowchart explains how the robot behaves in different scenarios. The robot has 3 main behaviours Following Human, sleep and Avoiding Obstacle when this robot is moving around in a closed space follows a human if it sees one far from it with the help of Realsense camera **d435i** camera input, Sleeps when it reaches close enough to a human and does all this while avoiding any obstacle that comes in the way. Below is the table that represents the whole

flowchart in a compact table. This shows how the robot will behave when the human is far and close as well as if any obstacle is close to the robot.

Below table explains different flows

| Input | Behavior |
|---|---|
| Human Far? | Follow Human |
| Close To Human? | Sleep |
| Obstacle Close? | Avoid Obstacle |
| Human Too Close? | Move Backward |
| Human Present? | Mov Forward |

Table 2 :Table for various behavior

The code gets real-time input from the camera to decide whether there is any human or obstacle in front of it and uses the depth input to find the distance of the robot from that human or obstacle based on which it decides which behavior to call. This is done in an infinite loop until the robot keeps on receiving the camera input. If the robot detects a human, it tries to follow that human until it reaches a safe distance and then sleeps near the human by calling the sleep behavior. In the case of the human moving again, the robot will chase the human by calling the follow human behavior. In the case of a human coming too close to the robot that is not safe for the robot it will call move backward behavior to maintain a safe distance from the human.

While it is following humans if the robot faces any obstacle on the way, then it calls avoid obstacle behavior and maneuvers to avoid the obstacle and stays on course to quickly follow the human again. Below we can see different states a robot can be in and all the actions that if taken will change the state of the robot according to the state, action transition model represented by the state diagram and the table.

- **State Diagram**

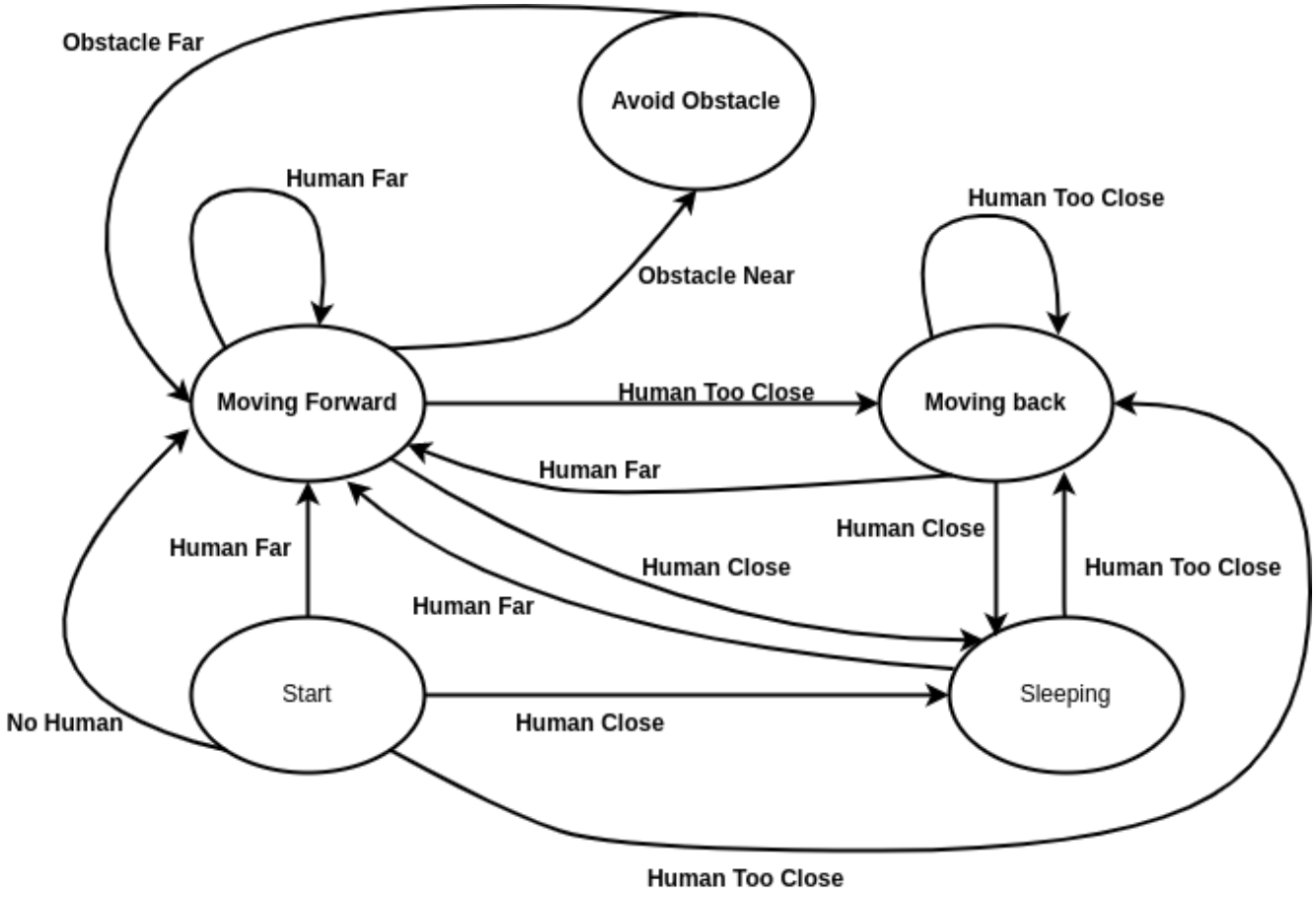



Figure 5: State diagram for following behavior

| $q$ | $input$ | $f(q, input)$ |
|---|---|---|
| **Start** | No Human | Moving Forward |
| **Start** | Human Close | Sleeping |
| **Start** | Human Far | Moving Forward |
| **Start** | Human Too Close | Moving Backward |
| **Moving Forward** | Human Far | Moving Forward |
| **Moving Forward** | Human Close | Sleeping |
| **Moving Forward** | Human Too Close | Moving Backward |
| **Moving Forward** | Obstacle Near | Avoid Obstacle |
| **Moving Backward** | Human Too Close | Moving Backward |
| **Moving Backward** | Human Far | Moving Forward |
| **Moving Backward** | Human Close | Sleeping |
| **Sleeping** | Human Far | Moving Forward |
| **Sleeping** | Human Too Close | Moving Backward |
| **Sleeping** | Human Close | Sleeping |
| **Avoid Obstacle** | Obstacle Far | Moving Forward |

Table 3: State Table for following behavior

# Implementation & unit testing

Description of classes and program functions

The classes and functions used in this project have two main references:

- NVIDIA's jetbot project website: https://github.com/NVIDIA-AI-IOT/jetbot [1]
- COP518 Practicals

**ObjectDetector class**: This class initializes the object detection model using the TensorFlow-TensorRT SDK and executes it. The model used in this project is MobileNet-V2 (a pretrained convolutional neural network that can classify images into about a thousand classes)

**Camera class**: The camera class initializes the RGBD sensors by setting configurations such as the resolutions for the camera and depth sensors. It also starts the thread for capturing color and depth images.

- **_capture_frames()**: This is a method in the Camera class that captures real-time feed from the RGB and the depth sensors. After obtaining feed from both sensors, it converts the data captured into a NumPy array and stores it in a variable for further computation.
- **start()** method starts a thread to control the operation of the _capture_frames() method if no thread is running yet while the **stop()** method ends the operation of the _capture_frames() method.

**Robot class**: The robot class initializes the robot by connecting to the drivers that control the robot's motors. It also contains functions for general movement of the robot.

**Part A (Reactive behavior):**

The process function after capturing the image and finding the color with highest frequency creates a lightweight thread for executing the robot operations.

**operations()**:

Input -> Camera class, maximum color (Red/Green/Blue/Yellow), Robot class

Returns: None

This function sets the speed of the motors and performs the required operation as per the color it detects.

**Part B (Following behavior):**

**detection_center() function**:

Input -> detection: output from model

Returns -> x and y coordinates of centre

This function computes the center of a detection using four points from the detection's bounding box. It returns a point that indicates the center of the bounding box.

**norm()**:

Input -> vec: x and y coordinates

Returns -> length of vector

This function computes the Euclidean distance of points given as input.

**closest_detection()**:

Input -> detections: a list of detections

Returns -> the closest detection

This function uses the distances computed with the norm() function to return the object with the least distance (i.e., The closest object).

**processing()**:

Input -> change: the color image variable

This function processes camera input to follow humans and avoid obstacles.

- If there is no human present, the robot moves forward until it gets too close to an object then it turns left.
- If there is a human:
    - The robot follows the closest human while maintaining a safe distance.

- The robot detects obstacles while following the human and avoids the closest obstacle by maneuvering its way around it.

## Unit Testing

### Part A

| Test Cases | Expected Results | Actual Results |
|---|---|---|
| Case 1:<br>No colors detected | The robot moves in a forward right fashion looking for color. | Robot moves forward right as expected. |
| Case 2:<br>Detects red | The robot moves back and forth for a certain period | Robot moves back and forth when red is detected as expected. |
| Case 3:<br>Detects blue | The robot moves towards the object slowly and pauses there for a certain period. | Robot moves towards blue objects slowly and stops as expected. |
| Case 4:<br>Detects yellow | The robot moves backward in zigzag fashion. | Robot moves backward in a zigzag fashion as expected. |
| Case 5:<br>Detects green | The robot rotates 360 degrees. | Robot rotates 360 degrees as expected. |

Table 5: Test cases for Reactive behavior

**Part B**

| Test Cases | Expected Results | Actual Results |
|---|---|---|
| Case 1:<br>No Human Present, Object Present | The robot will move forward and turn left if there is any obstacle till it finds a human | Robot moves |
| Case 2:<br>Human Present, Object Present | The robot tries to follow a human by avoiding any obstacles in its way and stops if it is too close to the human | Robot follows humans and avoids any obstacles as expected. |
| Case 3:<br>Human Present, No Object Present | The robot will follow a human and stops before if it is too close to the human | Robot follows humans and stops at safe distance as expected. |

Table 6: Test cases for following behavior

# System Testing

## Reactive behavior:

When the robot is allowed to move around freely on its own without any restrictions or boundaries, it shows a variety of emotions. The following elaborate the cases where the robot is allowed to move independently. (Note: Due to the issue of lighting, the robot does not see solid colors, but it sees a spectrum of colors)

- Case 1: In an environment where green is dominant it tends to show curiosity by rotating 360 degrees. It tends to repeat this behavior frequently.
- Case 2: In an environment where red is dominant it tends to show aggressiveness by moving back and forth.
- Case 3: In an environment where yellow is dominant it becomes fearful of its surroundings and tends to move backwards in a zigzag fashion.

- Case 4: In an environment where blue is dominant the robot indicates signs of love by moving towards the target and halts in front of it.

### Following behavior:

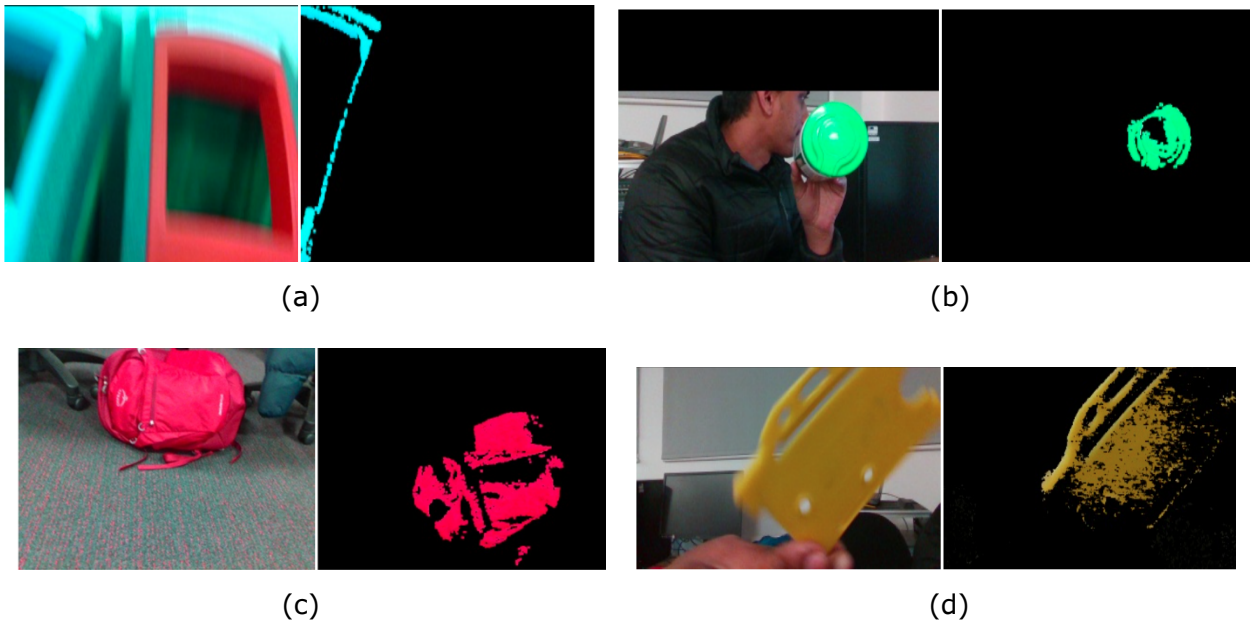
The integration of following human behavior and object avoidance is explained in Test case 1 of the Result and Analysis section.

## Result and Analysis

### Reactive behavior

The input is read from the camera and a region of interest from the image is taken and filtered for colors such as red, green, red, and yellow. The filters detect the colors in a specified range.

Figure 6 shows the result for color detection. The frequency of colors from a given frame of image is calculated. Based on the color that has maximum frequency, the action of the robot is decided.



(a) (b)

(c) (d)

Figure 6: Color Detection (a) Blue color, (b) Green color, (b)Red color (d) Yellow color

When the frame has high frequency of red color, the robot shows aggression towards the object. High frequency of yellow color in the environment makes the robot fearful, and the robot moves away from the object.

With a higher frequency of green color, the robot becomes curious and spins. When the environment has more blue color the robot loves the color and moves forward towards it. If the robot does not find the mentioned colors in the frame it explores and goes around for colors.

Environmental factors such as lighting play a significant role in this lab experiment as there are different shades of red, green, yellow, and blue. Depending on the intensity of the light in the room a shade of blue can seem to be either light blue or dark blue. Therefore, the light of the surrounding environment must be considered while performing reactive behaviors.

## Following Behavior

### Analysis of Issues Encountered

**Lag in capturing frames**

We noticed a lag at certain times when we ran the code. Further investigation showed that the lag resulted from two main sources. The first source was multiple if statements with computationally expensive logic. To fix this, we restructured the code to reduce the number of if statements. The second reason for the lag was the time.sleep() function. To resolve this, we decided to use for loops at certain points as a replacement.

**Backward Right Tire Motor Issue**

Our robot had an issue in the human following without obstacle test case in environment 2 the back right tire was malfunctioning and sometimes where it got stuck or started rotating too much compared to other motors due to which the maneuvering was not perfect where it turned left, and the robot was terribly slow allover, but it was fine once it got fixed and we tested again in the same environment and the same situation.

Analysis of Working as Expected Results:

**Test 1 Obstacle Avoidance with Human Following**

***Case 1 Environment 1:***

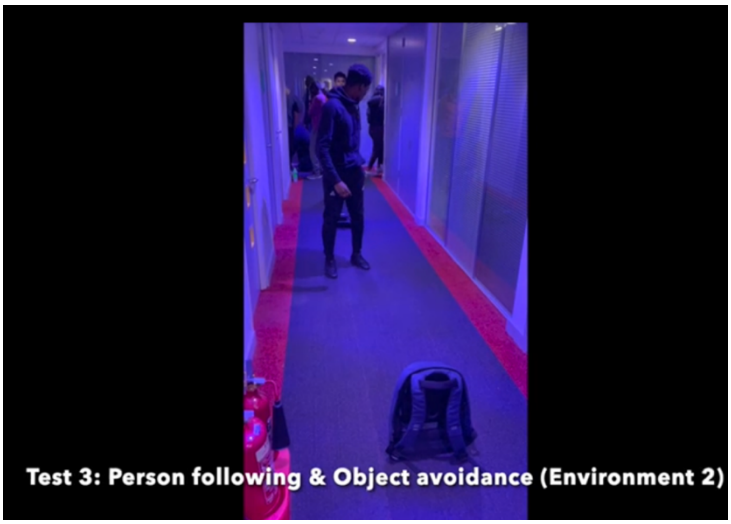

A bag was used in this test as an obstacle to see if the robot avoids the obstacle and starts following the human again by maneuvering and retracking the original path as it was doing to follow the human before encountering the obstacle.

**Result: Success**

Robots always go forward when there is no human and turn left where there is an obstacle but in this test case as there was a human in front of the bag it

identifies the human and starts following. Here, Robot identified the obstacle also and found the center of the obstacle and by using depth input avoided it and started changing its direction away from the obstacle and then changed direction back to follow the human behind the obstacle.

**Case 2 Environment 2:**

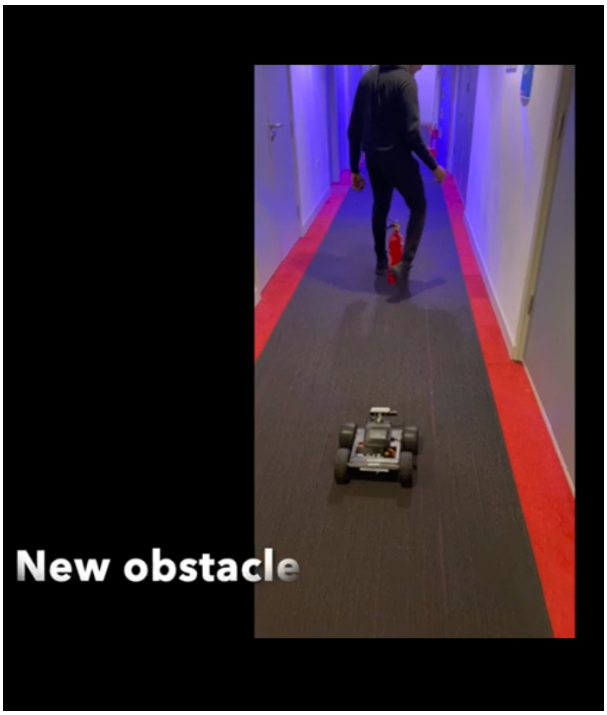



Here my teammate was walking in the hall and the robot was following him and after that my teammate stepped over a bag (obstacle) to see if the robot avoids that and continues to follow him.

**Result: Success**

It took some time for the robot to understand that there was an obstacle in front of it when my teammate stepped over the bag but eventually it executed that maneuvering routine just like the first test and avoided the wall after some struggle.

***Case 3 Environment 2 but with different obstacle:***



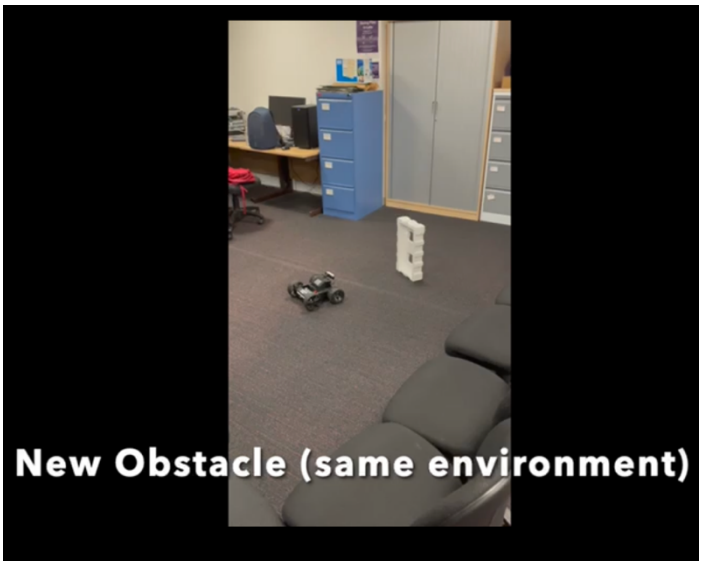

This time we did the same thing and changed our obstacle from a bag to a fire extinguisher.

**Result: Success**

Our robot successfully executed the avoidance maneuver I.e., first to change the direction away from the obstacle and trace back the path to human it was following before avoiding the obstacle and continued to follow the human

**Test 2 Obstacle avoidance, No Human:**

**Case 1 Environment 1**

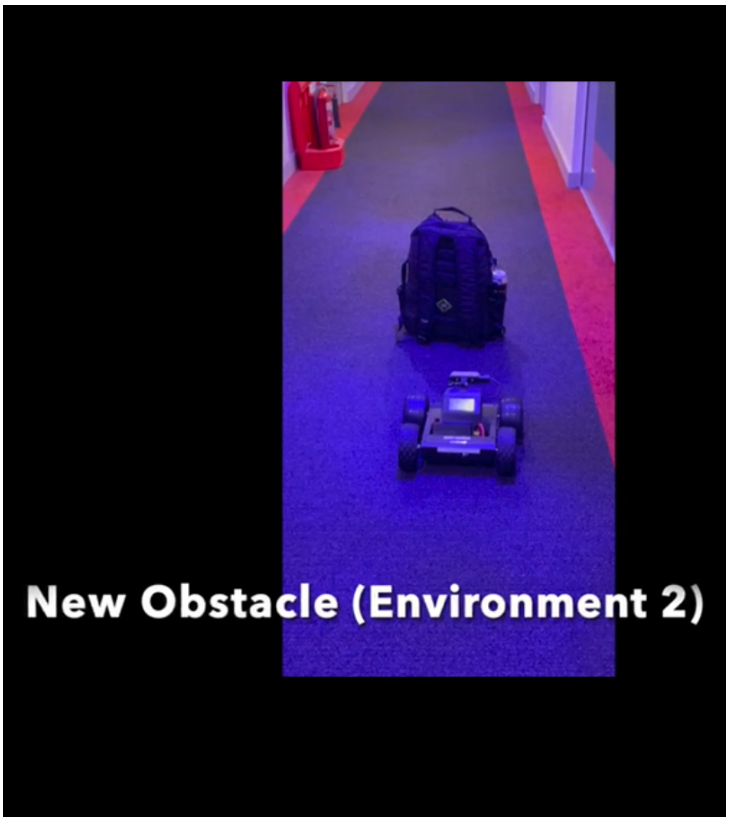



This time we used a polystyrene piece as an obstacle in the same environment without any humans and tried to see if robot avoided that obstacle and then goes forward. After avoiding the first obstacle it goes forward where it encounters another obstacle i.e., a wall and then it turns left there to avoid it and looks for humans.

**Result: Success**

Robot identified the obstacles again and avoided it by changing the direction to go left and then always to go forward when there is no human. It encountered a wall after that, so it turned left again to look for humans and so on.

**Case 2 Environment 2:**




This time we changed the obstacle to a bag and tried to test in a different environment to see how it avoids the obstacle this time.

**Result: Avoided the obstacle but not as expected (Emergent behavior)**

Our robot avoided the obstacle successfully but this time instead of always going left in case of obstacle and no human it chose to execute the maneuver to trace back the path when chasing a human and the reason we found was that the model was showing the bag as a human, so it executed that maneuver. So, after avoiding the obstacle it started looking for a human and as there was no human in front it decided to just go forward.

**Test 3 Human Following, No obstacle:**

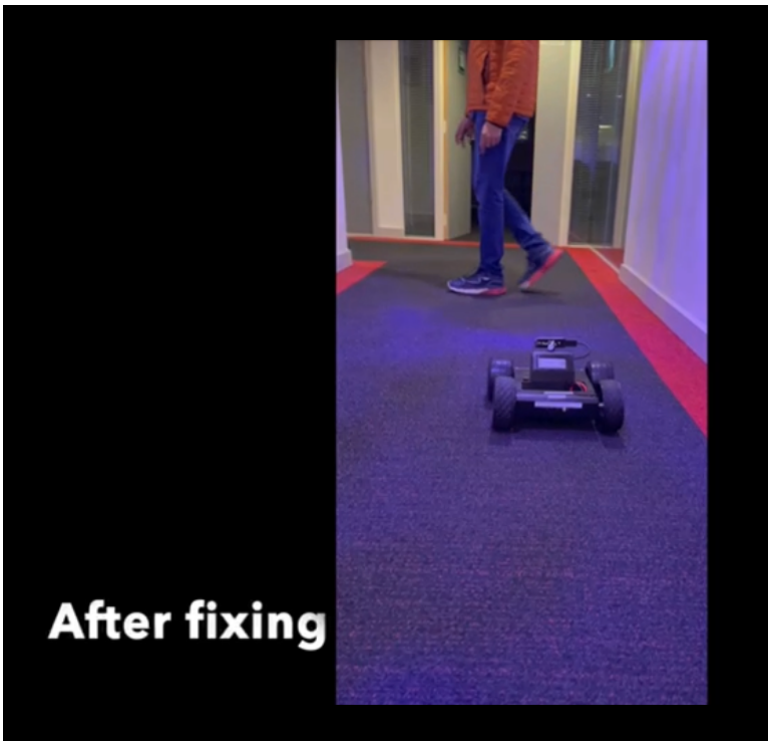
**Case 1 environment 2:**


We tried to see if the robot follows the human when the human is moving through a hall that has a turn in it we wanted to see if it follows the human

**Result: Success but very slow**

It successfully followed the human along the path of the hallway and took a turn also successfully, but it was very slow as there was a problem with the robot as its backward right Tyre was malfunctioning because of the motor. And we had to take the robot to fix that motor.

**Case 2 after fixing the Tyre issue:**




After fixing it we tried the same thing with a different human and tried some difficult quick maneuvers as the robot was fixed now.

**Result: Success**

It successfully followed the human and turned according to the path of the hallway as the human was going.

## Conclusion

From the above experiments conducted we conclude that, in terms of reactive and following behavior, the robot performed better in a controlled environment. However, the robot's performance needs to be assessed in a real-time environment.

# References

[1] NVIDIA's jetbot project website: https://github.com/NVIDIA-AI-IOT/jetbot