COP518 Robotics and Intelligent Systems Coursework

Members:
Mohammad Danish
Owen Leung
Ihsan Saygi
Ojeshwar Sharma
Anshul Singh
Kypros Tsolakis

Table of Contents

1. INTRODUCTION

In the realm of robotics, the ability to exhibit intelligent and adaptive behaviours is a cornerstone of modern robotic systems. This project delves into the fascinating world of robotic behaviours, drawing inspiration from Valentino Braitenberg's seminal work on synthetic psychology. Braitenberg's vehicles, conceptual models that demonstrate how simple mechanisms can create complex behaviours, serve as a foundation for this exploration. The project is divided into two distinct but interconnected parts, each focusing on different aspects of robotic behaviours:

Part A: Reactive Behaviours

The first part of the project is dedicated to demonstrating four key Braitenberg behaviours: aggressive, fear, love, and curiosity. Each of these behaviours represents a fundamental response pattern that can be observed in living organisms, and translating these into a robotic context presents an intriguing challenge.

- **Aggressive Behaviour**: Here, the robot will exhibit characteristics akin to aggression. It will actively approach and 'confront' specific stimuli, mimicking an offensive or assertive reaction. This could involve moving towards a designated object or target, emulating a form of attack or engagement.
- **Fear Behaviour**: In contrast to aggression, the fear behaviour will see the robot actively avoiding certain stimuli. This behaviour is crucial in demonstrating the flight response, where the robot increases its distance from perceived threats, effectively showcasing a survival instinct.
- **Love Behaviour**: This behaviour is characterized by the robot's attraction to certain stimuli, akin to affection or affinity in biological entities. The robot will move towards and follow a specific object, illustrating a form of bonding or attraction towards it.
- **Curious Behaviour**: Curiosity in robotics can be manifested as a combination of attraction and cautious investigation. The robot will approach a stimulus possibly orbit around a specific object but maintain a safe distance, displaying a behaviour that is explorative yet wary.

Part B: Following Behaviour

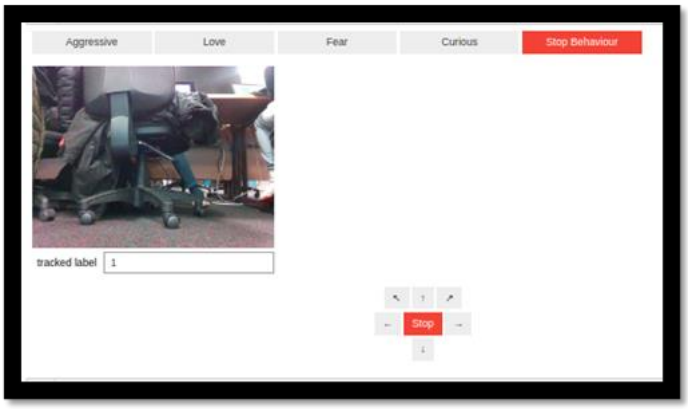
The second part of the project shifts focus to a more complex behaviour: following a person while maintaining a safe distance and avoiding collisions. This behaviour is more dynamic and requires a higher level of sensory integration and decision-making.

- **Person Following**: The robot is tasked with identifying and following a person. This involves complex processing of sensory data to recognize and track the movement of a human subject, adjusting its speed and direction accordingly with the aid of the RealSense Camera.
- **Collision Avoidance**: Integral to this task is the robot's ability to navigate safely. It must be able to detect and avoid obstacles, including other people, to prevent collisions. This involves real-time processing of environmental data and dynamic adjustment of its path.
- **Maintaining Safe Distance**: Alongside following and avoiding collisions, the robot must also maintain a safe and consistent distance from the person it is following. This requires careful calibration of sensors and control systems to ensure comfortable and safe interaction with humans.

In short, this project aims to bridge the gap between simple reactive behaviours and more complex, dynamic interactions in robotics. Through Part A, we explore foundational behavioural responses by Braitenberg, while Part B pushes the boundaries into more nuanced human-robot interactions. The successful implementation of these behaviours will not only demonstrate technical prowess but also provide valuable insights into the design of intelligent and responsive robotic systems.

2. PART A EXPLANATION

PART A CONTROLS



In Jupyter Notebook, we have implemented a GUI interface that allows users to toggle between different behaviours they want the robot to display. At the top, users can choose any of the four Braitenberg behaviours, or stop all behaviours by clicking on the corresponding button. Below is a screen displaying the captured frames from the camera, so that we have a vision of what the robot 'sees'. The tracked label can used to change the object class the robot will react to. Finally, there are seven buttons at the bottom so that we can manually control the movement of robot.

2.1 LOVE BEHAVIOUR

2.1.1 Design Introduction

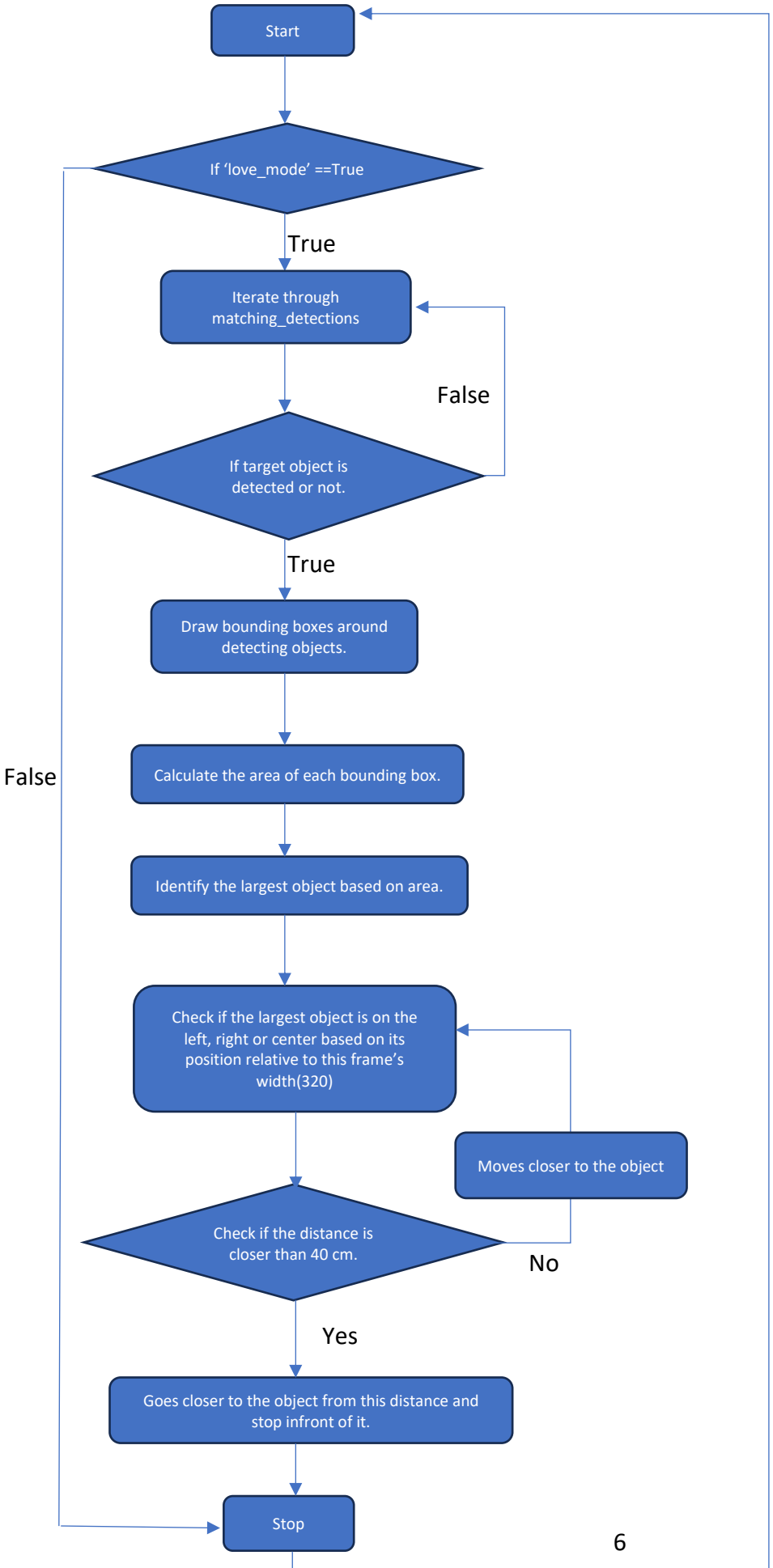
Among the different things that Braitenberg looked into, the "love" behaviour stands out for being emotionally rich. In this situation, the robot uses a RealSense camera to see and recognize the human or object it loves. When it identifies its beloved, the robot moves towards it and stops right in front.

The RealSense camera acts like the robot's 'eye,' helping it see and identify what makes it feel affection. This behaviour imitates how living things show love when they see something special. The camera guides the robot to activate its parts (like wheels or motors), making it move gracefully towards the loved one and stop smoothly in front.

In simple terms, the love behaviour in Braitenberg vehicles, and in the robot being studied, shows how basic machines can copy complex feelings like affection and connection. This idea helps us understand how robots can emotionally connect with their surroundings. Even with basic tools, the robot can genuinely and emotionally respond by purposefully stopping in front of the thing it loves.

2.1.2 Flowchart and Function Description

This flowchart should detail how the robot processes visual input, identifies objects, and decides on its movement.


Start
If 'love_mode' ==True
True
Iterate through matching_detections
False
If target object is detected or not.
True
Draw bounding boxes around detecting objects.
Calculate the area of each bounding box.
Identify the largest object based on area.
Check if the largest object is on the left, right or center based on its position relative to this frame's width(320)
Moves closer to the object
Check if the distance is closer than 40 cm.
No
Yes
Goes closer to the object from this distance and stop infront of it.
Stop
False

love_behaviour (matching_detections, image): The function takes two parameters:

- **matching_detections**: This is a list of objects detected in the robot's environment, each with details like location and size, often represented by bounding boxes.
- **image**: The current frame or image captured by the robot's camera, used for processing, and potentially visualizing the detections.

2. Global Variable:

love_mode: A global variable indicating whether the robot is in 'love mode.' This mode determines if the robot should exhibit love behaviour.

3. Detections and Area Calculation:

The function iterates over matching_detections to process each detected object. For each object, it calculates the bounding box and draws it on the image. It then calculates the area of each bounding box to find the largest object, as the robot's love response is likely triggered by the largest object in its view.

4. Decision-Making Based on Object Position:

The function analyses the position of the largest object (determined by its bounding box) relative to the robot's field of view. It divides the field of view into left and right sections, calculating the left area and right area, representing the areas of the largest object on the left and right sides of the image, respectively. These would be used to determine the direction of travel for the robot.

5. Love Response

- If no objects are detected or a certain condition is represented by **camera.warning_flag** is met, the robot stops.
- Otherwise, the robot performs a predefined 'love' movement. This means moving toward the detected object.

2.1.3 Test Procedure, Result and Analysis

Test No.	Description	Procedure	Results	Analysis
1]	Single Object Approach	Place a single beloved object in an open area. Activate the robot and observe its reaction.	Robot detects and approaches the object. Note reaction time and stopping distance.	Successful approach demonstrates basic love behavior.
2]	Various Object Interactions	Arrange multiple objects of different sizes. Run the robot. Observe responses to different objects.	Consistently approaches the beloved object. Shows uncertainty with multiple nearby objects.	Effective at prioritizing the beloved object. Struggles in crowded scenarios, suggesting a need for a more advanced decision-making algorithm.
3]	Dynamic Object Interaction	Introduce a moving object in an open area. Observe the robot's response to the moving object.	Attempts to approach the moving object. Less successful with faster objects.	Capable of responding to dynamic changes. Requires enhancements for swift-moving objects, possibly through real-time processing and predictive analysis.

2.1.4 Conclusion of Behaviour

The robot does a good job of showing love by going towards and stopping in front of the thing it likes. However, when faced with more complicated situations involving many or moving things, it struggles. This suggests the need for better thinking abilities to handle changing situations and emotions in a smarter way.

2.2 AGGRESSIVE BEHAVIOUR

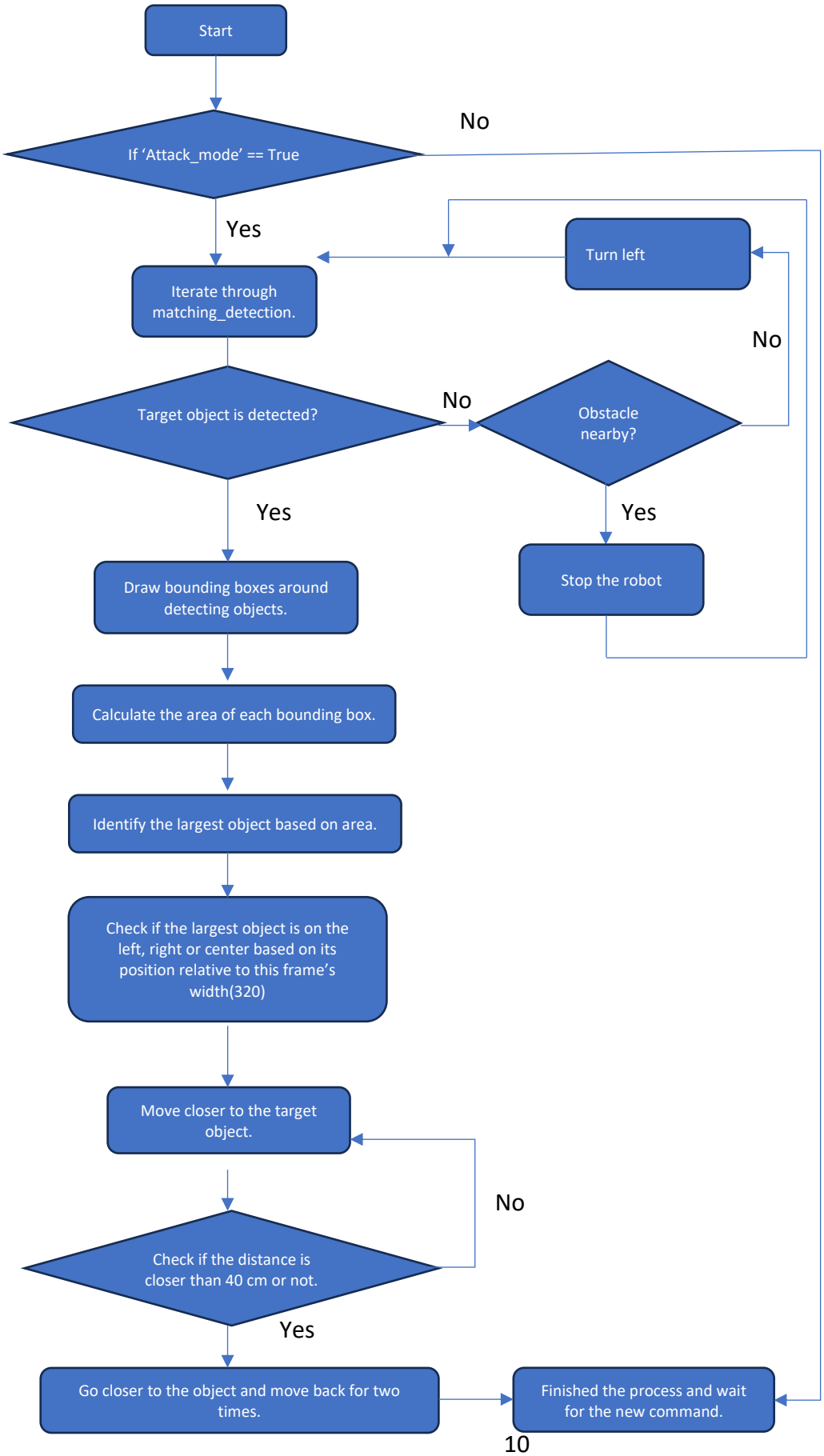
2.2.1 Design Introduction

For the aggressive behaviour, adhering to the principles of Braitenberg behaviours, our robot is designed to react in a way with a direct response to a stimulus without complex decision making. It will first 'look around' its surroundings by moving anti-clockwise in circle to search for the targeted type of object.

In our case, the target is set to be water bottles. Once a water bottle is in sight, the robot will immediately stop circling and start approaching it at an increased speed. When the robot is close enough to the bottle (~40 cm from it), an aggressive behaviour will be initiated. It will mimic an attack action by going backwards and forwards even more rapidly to show its aggressiveness. For safety, the original part of pushing down the bottle (an actual attack) was removed.

2.2.2 Flowchart and Function Description

The flowchart gives information about motor responses in aggressive behaviour mode.


Start
If 'Attack_mode' == True
No
Yes
Turn left
Iterate through matching_detection.
No
Target object is detected?
No
Obstacle nearby?
Yes
Yes
Draw bounding boxes around detecting objects.
Stop the robot
Calculate the area of each bounding box.
Identify the largest object based on area.
Check if the largest object is on the left, right or center based on its position relative to this frame's width(320)
Move closer to the target object.
No
Check if the distance is closer than 40 cm or not.
Yes
Go closer to the object and move back for two times.
Finished the process and wait for the new command.

Here's a detailed description of the function and its components:

1. **Function Definition:**

aggressive_behaviour (matching_detections, image): This function takes two parameters:

- **matching_detections**: This is likely a list of objects that have been detected in the robot's environment. Each object in this list would typically include details such as the location and size of the object, often represented by bounding boxes.
- **image**: This is the current frame or image captured by the robot's camera. It's used for processing and possibly visualizing the detections.

2. Global Variable:

attack_mode: A global variable that indicates whether the robot is in 'attack mode'. This mode determines if the robot should exhibit aggressive behaviour and 'attack' an object.

3. Detections and area calculation:

The function iterates over **matching_detections** to process each detected object. For each object, it calculates the bounding box and draws it on the **image**. It then calculates the area of each bounding box to find the largest object, as the robot's aggressive response is likely triggered by the largest object in its view.

4. Decision Making Based on Object Position:

The function analyses the position of the largest object (determined by its bounding box) relative to the robot's field of view. It divides the field of view into left and right sections and calculates **left area** and **right area**, which represent the areas of the largest object on the left and right sides of the image, respectively.

5. Aggressive Response:

- If there is no matching detections from the robot's camera (I.e. **len(matching_detections) == 0**), the robot will move in anti-clockwise circle using **robot.left(0.4)**, which means keep moving left at 40% of full power, to search for any targeted object type (bottles) in its proximity.
- If no bottles are detected, the robot will not stop its circular motion. Only if there is an obstacle within 40cm from the robot (which will set **camera.warning_flag** to 1), the robot will pause its motion until the obstacle moves away.
- If there is any bottle nearby (i.e. **len(matching_detections) > 0**), it will check whether it is within 40cm from the robot using **camera.warning_flag**

- When **camera.warning_flag** is equal to 1, it signifies that the detected object is in close proximity to the robot. The robot will then 'attack' the object by first stepping back, then sprint towards the object to show its aggressiveness.
- Otherwise, the robot will continue moving towards the object until it is close enough. The direction of travel is determined by comparing the offset of the detected object from centre of the captured frame.

2.2.3 Test Procedure, Result and Analysis

Test No.	Description	Procedure	Results	Analysis
1]	Detection mode test	Place no bottles near the robot. The robot will move in an anti-clockwise circle to search for bottles. Next, stand within 40cm from the robot. The robot will stop in front of you. Move away and the robot sets off again.	The robot will always move in circle indefinitely if no targeted objects are detected unless there is an obstacle in front. Once an obstacle is clear, it will continue its search.	The robot can be set to detect different object types; It does a good job avoiding collision during search. However, it would not move away from the obstacle by itself.
2]	Object detection test	Place a bottle near the robot, the robot stops circular motion when it detects the bottle, then speeds up and approaches the bottle	The bottle is detected by the camera and will start moving towards the bottle as expected. Occasionally, it may lose detection of the bottle and goes back to detection mode briefly.	Object size and type affects sensitivity of detection. Even though, our robot can still accurately move towards the target.
3]	Attack motion test	The robot performs an attack by first going backwards, then accelerating towards the bottle, and stops right before the bottle.	The robot can always demonstrate an aggressive motion towards the bottle without knocking down.	Initiating the attack in close range to the object ensures accuracy.

2.2.4 Conclusion of Behaviour

To conclude, our robot exhibits its aggressive behaviour in a direct and intuitive way. It can search for stimuli in its surroundings, then approach and attack the object very successfully. There is also obstacle avoidance during search to pause the robot if it is blocked. Even when object recognition loses detection sometimes due to small object size, our robot can always carry on its aggressive behaviour and finish the ‘attack’. An enhancement can be made in the future so that the robot can move around in search for targets.

2.3 CURIOUS BEHAVIOUR

2.3.1 Design Introduction

Among the various behaviors that Braitenberg explored, the "curious" behavior is particularly notable for its simplicity and effectiveness. In this case, it is described how the robot uses visual inputs (labels) with using RealSense camera to detect objects and then respond with movements that simulate a curious response, i.e., The robot observes these objects by actively steering close. The robot changes its position five times to observe the object from five different angles.

This curious behavior can be found in the natural environment. For many animals taking some information from objects is quite important to be alive. Animals use their sense like seeing, smelling, touching, tasting, and hearing to understand what is the object whether it is an enemy, friend, food, or something that it needs to run away.

In the project robot, the depth sensor and camera are used to observe objects. For observing the purpose of objects, the robot is designed to see objects from different angles. After detecting a new object, the robot reaches the object until 40 cm left to take information from different angles.

2.3.2 Flowchart and Function Description

Here's a detailed description of the function and its components:

1. Function Definition:

Curious_behaviour(matching_detections, image): The function takes two parameters:

- matching_detections: This is likely a list of objects that have been detected in the robot's environment. Each object in this list would typically include details such as the location and size of the object, often represented by bounding boxes.
- image: This is the current frame or image captured by the robot's camera. It's used for processing and possibly visualizing the detections.

2. Global Variable:

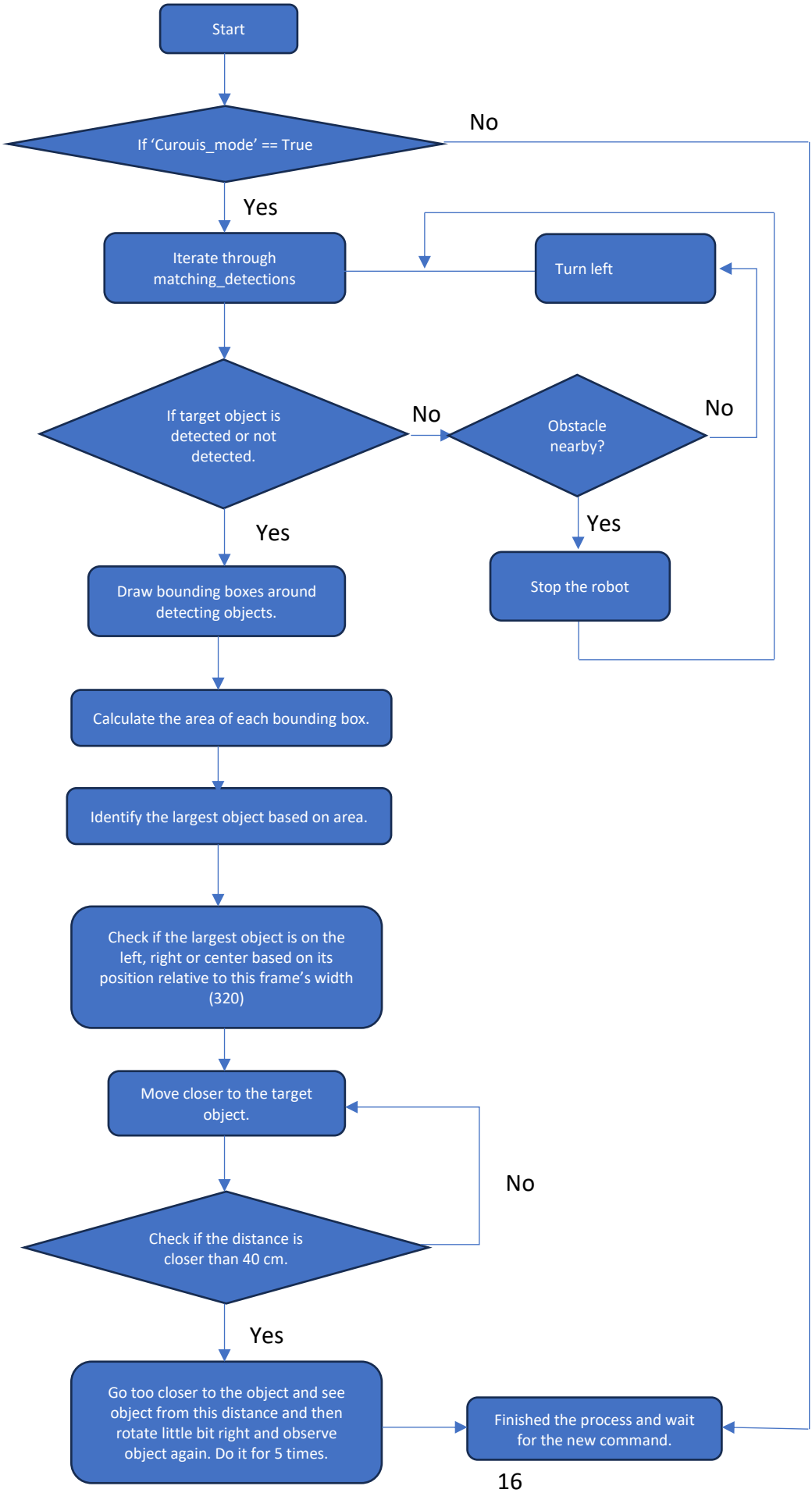
Curious_mode: A global variable that indicates whether the robot is in 'Curious mode'. This mode determines if the robot should exhibit curious behaviour.

3. Detections and area calculation: The function iterates over matching_detections to process each detected object. For each object, it calculates the bounding box and draws it on the image. It then calculates the area of each bounding box to find the largest object, as the robot's fear response is likely triggered by the largest object in its view.
4. Decision Making Based on Object Position: The function analyses the position of the largest object (determined by its bounding box) relative to the robot's field of view. It divides the field of view into left and right sections and calculates the left area and right area, which represent the areas of the largest object on the left and right sides of the image, respectively.

5. Curious Response:

- If no objects are detected or a certain condition is represented by the camera.warning_flag is met, the robot turns around itself to find an object.
- If the robot detects an object robot moves through to the object until a 40 cm distance is left. After it reaches object robot changes its position in 5 different ways to observe object from different perspectives.

The flowchart gives information about motor responses in curious behaviour mode.


Start
If 'Curouis_mode' == True
No
Yes
Iterate through matching_detections
Turn left
If target object is detected or not detected.
No
Obstacle nearby?
No
Yes
Yes
Stop the robot
Draw bounding boxes around detecting objects.
Calculate the area of each bounding box.
Identify the largest object based on area.
Check if the largest object is on the left, right or center based on its position relative to this frame's width (320)
Move closer to the target object.
No
Check if the distance is closer than 40 cm.
Yes
Go too closer to the object and see object from this distance and then rotate little bit right and observe object again. Do it for 5 times.
Finished the process and wait for the new command.

2.3.3 Test Procedure, Result and Analysis

Test No.	Description	Procedure	Results	Analysis
1]	Different object detection test	Bottle(object) is tried to detect with the camera and curios mode is applied.	The bottle is detected with a camera and curious mode is applied successfully.	Different objects can be used to observe with curious mode.
2]	Distance test	Different objects (human and bottle) are tried to detect from 2m 5m and 10 meters distances	Big objects (humans) are detected from all distances. Bottle can be detected from 2 meters precisely but can not be detected from other distances precisely.	Object size is a matter to detect when different distance tests are applied.
3]	Angle rotation test	The robot has to observe the object from 5 different positions. To do this it needs to change his position each time.	Robots mostly change their position with the perfect angle, but sometimes tires slip so the correct angle is missed.	The floor and the tire type are important to move with a perfect angle.

2.3.4 Conclusion of Behaviour

In conclusion, the robot has the ability to observe new objects by going close to them and looking this object from different perspectives. For this behaviour, the robot can work with very different objects, but the size and the distance of the object have significant importance to complete this task.

2.4 FEAR BEHAVIOUR

2.4.1 Design Introduction

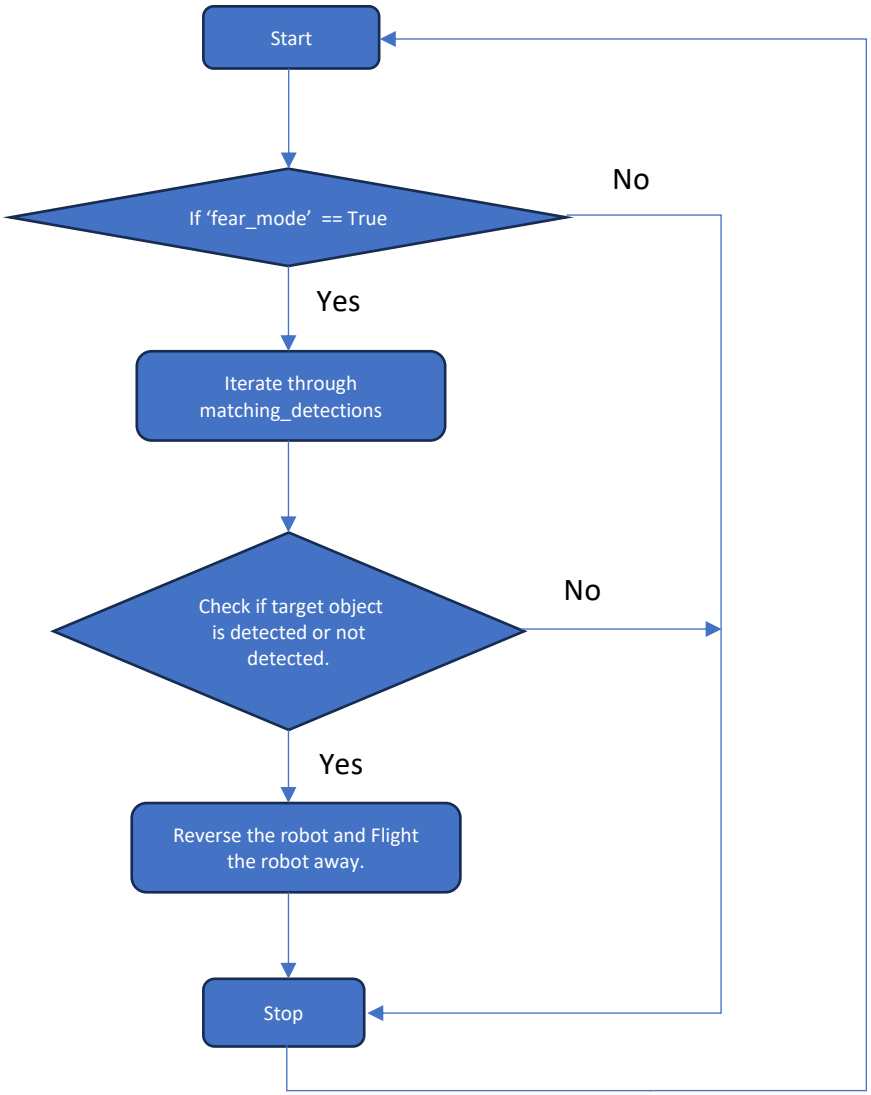
Among the various behaviours that Braitenberg explored, the "fear" behaviour is particularly notable for its simplicity and effectiveness. In this case, it is described how the robot uses visual inputs (labels) with using RealSense camera to detect objects and then respond with movements that simulate a 'fear' response, i.e., the robot avoids these objects by actively steering away from the perceived threat and then moving in the opposite direction to create distance, effectively simulating a 'flight' response.

This behaviour is akin to an instinctive reaction found in living organisms, where the presence of a specific stimulus triggers a flight response. The camera serves as the robot's 'eye,' allowing it to perceive its surroundings. The response to this perception is an immediate activation of the actuators (such as wheels or motors) to create distance from the perceived threat in a swift manner. In summary, the fear behaviour in Braitenberg vehicles, and by extension in the robot you're studying, is a prime example of how simple mechanical systems can mimic complex biological behaviours.

This principle is not only fundamental in understanding the basics of robotic intelligence but also serves as a stepping stone towards developing more sophisticated autonomous systems capable of navigating and interacting with their environment in an intelligent manner. The simplicity of this design belies its effectiveness; even with basic sensors and motors, the robot can exhibit a convincingly life-like response to its environment.

2.4.2 Flowchart and Function Description

This flowchart should detail how the robot processes visual input, identifies objects, and decides on its movement.



Here's a detailed description of the function and its components:

1. **Function Definition:**

fear_behaviour(matching_detections, image): The function takes two parameters:

- **matching_detections**: This is likely a list of objects that have been detected in the robot's environment. Each object in this list would typically include details such as the location and size of the object, often represented by bounding boxes.
- **image**: This is the current frame or image captured by the robot's camera. It's used for processing and possibly visualizing the detections.

2. **Global Variable:**

fear_mode: A global variable that indicates whether the robot is in 'fear mode'. This mode determines if the robot should exhibit the fear behaviour.

3. **Detections and area calculation:**

The function iterates over **matching_detections** to process detected objects. For each object, it calculates the bounding box and draws it on the image. However, unlike the previous approach, it does not calculate the area of each bounding box or determine the largest object. The mere presence of any object in the robot's field of view is sufficient to trigger the fear response.

4. **Simplified Decision Making:**

Instead of analyzing the position and size of the largest object, the function now simply checks if any object is detected. If one or more objects are detected, the robot immediately initiates an avoidance maneuver. There's no division of the field of view into left and right sections, and no calculation of areas for different sides of the image. The decision to avoid is made solely based on the presence of any object, regardless of its specific characteristics or location in the robot's visual field.

5. **Fear Response:**

- If no objects are detected or a certain condition represented by **camera.warning_flag** is met, the robot stops.
- Otherwise, the robot performs a predefined 'fear' maneuver. This typically involves moving in a direction away from the detected object. In your provided snippet, the robot turns left and then moves forward, indicating an avoidance strategy.

2.4.3 Test Procedure, Result and Analysis

Test No.	Description	Procedure	Results	Analysis
1]	Single Object Avoidance	Placing a single large object in an open area. Activate the robot and observe the reaction.	Robot detects and avoids the object. Noted reaction time and distance that it is moving in a direction away from the detected object i.e., turns left and then goes forward.	Successful avoidance demonstrates basic fear behaviour. Indicating the robot's effectiveness in dynamic environments and its potential applicability in various practical scenarios.
2]	Multiple Object Scenarios	Arrange multiple objects of varying sizes. Run the robot. Note responses to different objects.	Consistently avoids larger objects. Shows confusion with multiple close objects.	Effective at prioritizing larger objects. Struggles in dense environments, suggesting a need for a more complex decision-making algorithm.
3]	Dynamic Object Avoidance	Introduce a moving object in an open area. Observe the robot's response to the moving object.	Attempts to avoid the moving object. Less successful with faster objects.	Capable of responding to dynamic changes. Needs improvements for fast-moving objects, possibly in real-time processing and predictive analysis.

2.4.4 Conclusion

The robot effectively demonstrates basic fear behaviour by avoiding large, stationary objects, but it struggles with complex scenarios involving multiple or moving obstacles,

indicating a need for advanced processing and decision-making capabilities for more dynamic environments.

3. PART B EXPLANATION

3.1 Design Introduction

The objective of the robot is follow a person and keeping a safe distance so to avoid collision from the person and avoiding the obstacles in doing so. To detect the human, we are using ssdmobilenet_v2_coco engine which is a pre-trained object detection model based on the Single Shot Multibox Detector (SSD) framework and MobileNetV2 architecture. Robot is mounted with a camera which is taking picture and image processed by using RealSense library to capture the color of 640*480 pixels of image at 30 frames per second. Robot is also mounted with a vision sensor which gives the depth image indicating the distance between the robot and object in its direct vision. After taking the picture converting the image to their RGB using Open CV library and giving the process image to the coco engine to detect all the human in the frame and drawing a blue rectangle indicating the detected humans. The Robot will follow the human based upon the position of detected human. As there can be many detected humans in the image, robot will follow the human with the largest area. Depending upon the position of the human based upon its area in relation to the centre X axis of the image robot will take appropriate movement. If the detected human area is on the right side from the centre of image, then robot will take right and vice versa, on other hand if detected human area lies on the centre X axis it will move forwards towards the detected human. Using this mechanism, the robot will try to follow the human by making appropriate movements.

3.2 Function Description

Here's a detailed description of the function and its components:

Camera (SingletonConfigurable):

This class gives the blueprint of all camera related activities. The class is inherited by the **Camera** Class which will called during initialization. The **SingletonConfigurable** class from the **traitlets.config.configurable** module is a base class in the traitlets library that extends **Configurable** to enforce the singleton pattern. Once its instance is created, we can start capturing the rgb and depth images. This class consist of following function once its initialized.

Start function:

This will start to get frame data on a parallel thread by making class thread variable to be true and capturing the images using **_capture_frames** function.

Stop function:

This will stop the thread to stop running by making the class thread variable to be false.

_capture_frames function:

This function will run on a thread till the class thread variable is not false, thus continuously capturing depth and RGB images, converting them into numpy array respectively. To get only the central area of the vision sensor, the depth image numpy array got manipulated by making pixel value above 190 and below 290 along the Y axis to be 0 and similarly less than 160 and more than 480 along x axis. By calculating the depth minimum value we can get the distance between the object and our robot. This function makes the class warning flag to be true whenever the minimum distance taken from above is less than 400mm.

ObjectDetector:

This Class will load the coco engine pretrained model when we creates its instance. This will enable to give the images to the created instance to detect all the object in the **image.**

Processing:

This function got called when there is any change in the color_value variable which was taken from the camera instance created from the Camera class object. This function takes the images, resize it to 300*300 which is passed to model and gets the detected objects. The detected object is compared with the desired label i.e. in our case its human which will be passed to human_following function for further processing.

human_following(matching_detections,image):

The function takes two parameters:

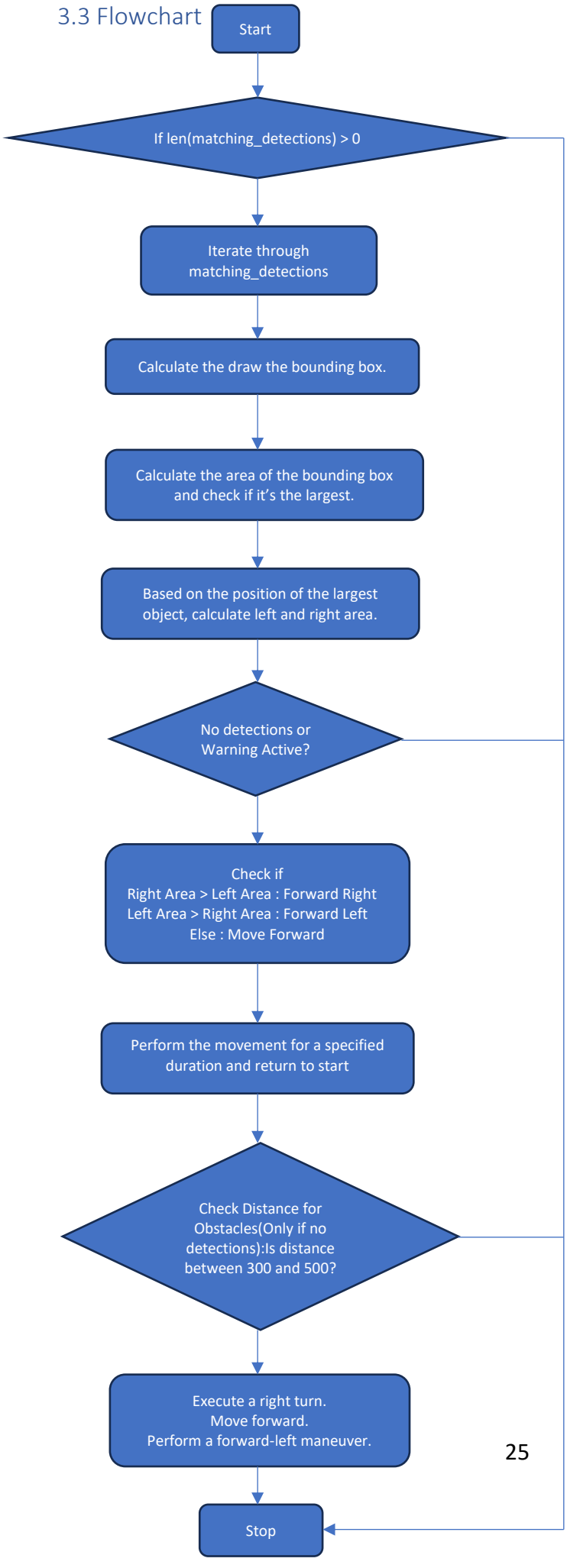
The detect object in our case human, will be passed to this function which will gives the detected human coordinates in the images. It iterates through the detected bounding boxes, calculates their areas, and keeps track of the largest bounding box. Calculating the area of the bounding box we can follow the object with the largest area and make the movement of the robot based upon the detected area position in the image from the center of x axis. If the detected largest areas is on the right of center

of x axis then robot will move right and vices versa. If the detected human are lies along on the center of x axis, the robot will move forward. The robot will follow the human and stops when the distance between the robot and human is less than 400mm which sets by the capture frame function running on parallel thread.

Obstacle Avoidance:

To follow the human and avoid any obstacles in between the robot will try to follow the human, if the robot comes very close to the obstacles and it will be able to detect the human is such case robot will follow the reactive behaviour. Based upon the pre determined distance from the obstacles robot will swiftly carry around the obstacles till it detect any human. Once the human is detected, robot will continue to follow the human based upon the previous logic.

3.3 Flowchart



Test No.	Description	Procedure	Results	Analysis
1]	Human Following in a Controlled Environment	Place the robot in a room with a single person walking at a moderate pace. Observe the robot's ability to follow the person while maintaining a consistent distance.	The robot successfully follows the person, adjusting its speed to match the person's movement. Maintains a steady distance throughout the test.	The robot demonstrates effective human following capabilities in a controlled environment. Suggests good performance of the detection and tracking algorithms under ideal conditions.
2]	Human Following in a Crowded Environment	Test the robot in a busy environment with multiple people moving in different directions. Observe how the robot identifies and continues to follow the designated person.	The robot occasionally loses track of the person or confuses them with others. Some hesitancy or incorrect movements noted in crowded situations.	The robot shows challenges in differentiating the target person in a crowded environment. Indicates a need for more advanced algorithms for target recognition and tracking in complex scenarios.

3]	Collision Avoidance with Static and Dynamic Obstacles	Place both static (like furniture) and dynamic (like a rolling ball) obstacles in the robot's path while following a person. Observe the robot's ability to avoid collisions while maintaining its following behaviour.	Successfully navigates around static obstacles without losing track of the person. Struggles with sudden movements of dynamic obstacles, leading to occasional stops or incorrect maneuvers.	The robot effectively avoids static obstacles while following a human, but its response to dynamic obstacles needs improvement. Suggests the need for faster sensor processing and more predictive movement algorithms for dynamic obstacle avoidance.

3.4 Conclusions

To sum it up, our smart robot is like a high-tech detective. It uses a super-smart camera to spot people, avoid crashing into things, and follow someone without bumping into them. It's a great example of how cool technology can be used to make robots do clever things in the real world!

4. GENERAL CONCLUSION

The exploration and implementation of various robotic behaviours, inspired by Braitenberg's theoretical vehicles, represent a significant step forward in understanding and developing intelligent autonomous systems. This project successfully demonstrated a range of reactive behaviours - aggression, fear, love, and curiosity - as well as a complex human following behaviour with collision avoidance.

Key Achievements:

- Demonstration of Reactive Behaviours: The project effectively showcased how simple sensor-actuator mechanisms could be employed to simulate complex behaviours observed in living organisms. Each of the behaviours - aggressive, fearful, loving, and curious - was distinctly and successfully exhibited by the robot, offering profound insights into the connection between sensory input and motor responses.

- Complex Following Behaviour: Perhaps the most challenging aspect of the project was the implementation of a sophisticated following behaviour. The robot's ability to track a human subject while navigating around obstacles and maintaining a safe distance illustrated a higher level of autonomy and intelligence, pointing towards the future of interactive and adaptive robotics.

- Challenges and Learning: The project presented various challenges, particularly in complex scenarios involving multiple or dynamic stimuli. These challenges were invaluable learning opportunities, highlighting areas such as sensor fusion, real-time data processing, and algorithm optimization that are critical for advancing robotic capabilities.

END OF REPORT