## 2. Human Following Task

### 2.1. Detection Selection

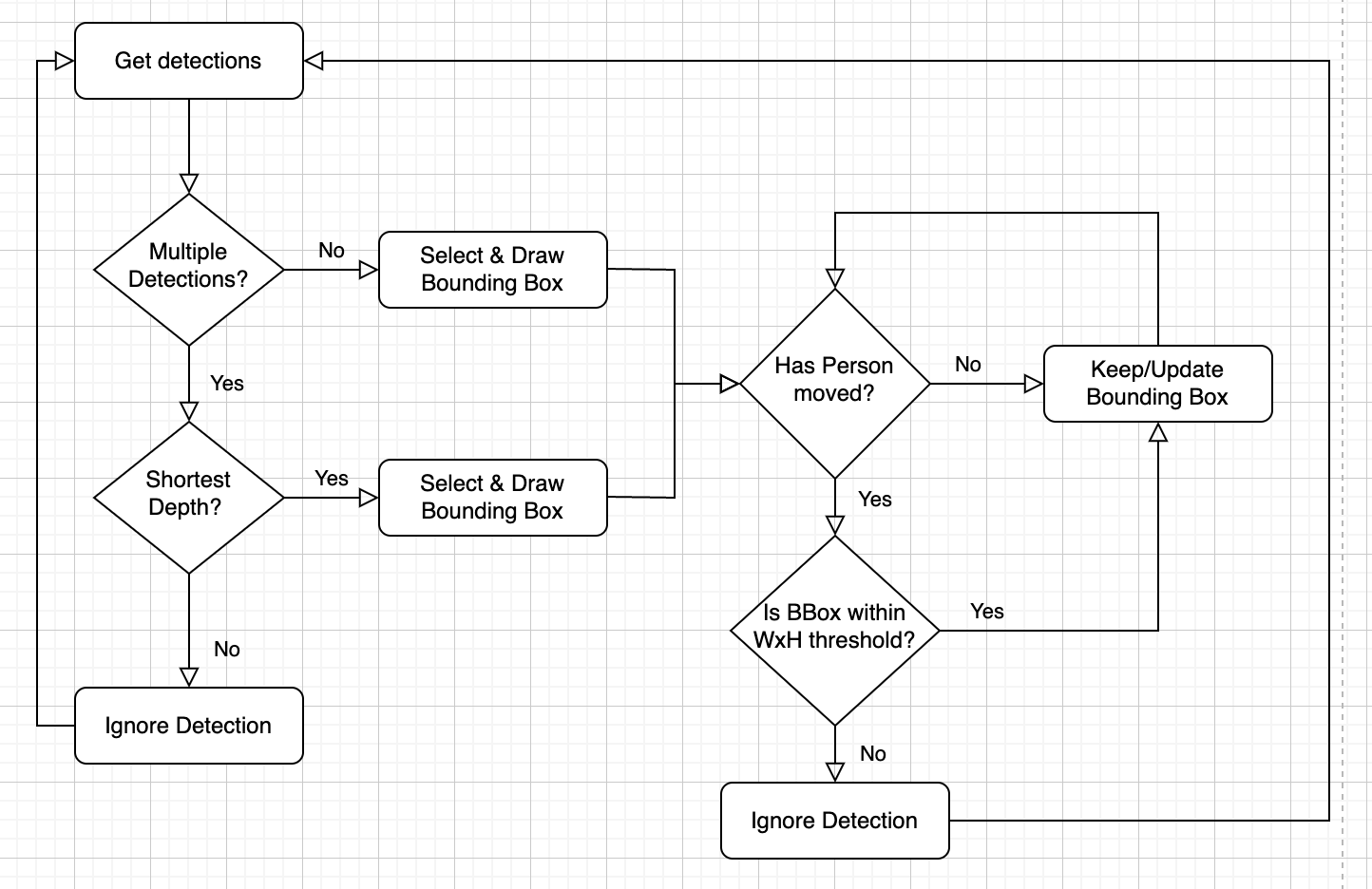
#### 2.1.1. Design

Each Human ‘target’ is defined using the pre-trained model as discussed later, the result of these definitions are captured in an image using a bounding box. This bounding box is crucial for the design and implementation for the given task, as it relays information that the target has moved resulting in the robot to respond.

The main problem regarding target detection is the process of selecting the initial target. A solution to this issue would be to implement the robot’s depth camera. This method includes choosing the ‘target’ that has the shortest (average depth inside the bounding box).…

After this problem, the next issue is determining when the detected ‘target’ has moved either to the left, right, towards or away from the robot. As this information will allow the robot to move in its required direction. This could firstly be done by defining a threshold which determines whether the next detection is the same target or another Human. This threshold would take into account the width, height and depth of the bounding box. Then examine whether the same target has moved by calculating whether the new detection is within the stored ‘target’ threshold.

The below Flow Chart describes the process of how each detection is defined and the results are given by drawing/ignoring or updating the bounding box for use in the movement section of the robot.



#### 2.1.2. Functions

As touched upon above, each detection is defined using the pre-trained convolutional neural network model MobileNet. MobileNet is used because of its low latency processing which is perfect for use in an external device such as a robot.

The pretrained model takes an image as input along with an integer label which represents the type of object that the model should detect. In this case, a person would have the label ‘1’, and after processing would return a dictionary of arrays with both; positional axis of the detected person and the confidence that the detection is a person. These positional axis are then translated using the height and the width of the whole image for an easier representation when designing and implementing bounding boxes for each detection.

Initially each bounding box is drawn using the cv2 library’s draw() function but for the specified problem, we need only the selected target’s bounding box for further processing. However, during some testing, the most common error occurs when the bounding boxes are trying to be defined when there are no initial detections. By employing an initial boolean flag that triggers when a detection is found was the fix to this issue, and when this flag is true the main loop proceeds to be executed.

Using the bounding box as discussed above, most of the information inside the bounding box is not usable, i.e. either there is too much leeway on empty space (area that isn’t human). Hence, by resizing the bounding box to be more central around the human, gives us the opportunity to implement the average depth value of the “zoomed in” boundaries.

The resizing of the bounding box was done geometrically by scaling each bounding box coordinate by a factor of 0.5 (50% decrease). This can be seen in the following lines;

**for** det **in** matching\_detections:

new\_bbox **=** det['bbox']

**for** i **in** range (len(new\_bbox)):

new\_bbox[i] **=** 0.5**\***new\_bbox[i]

new\_bbox[0] **=** new\_bbox[0] **+** 0.5**\***(new\_bbox[0] **+** new\_bbox[2])

new\_bbox[1] **=** new\_bbox[1] **+** 0.5**\***(new\_bbox[1] **+** new\_bbox[3])

Further on from this scaling in the same “for det in matching\_detections” loop, includes a conditional. This conditional compares each other detection with any new detections “new\_bbox” to see whether the bounding box is within the width and height threshold (WxH Threshold). As discussed above, these lines of code stop any other people who are walking past to be the selected ‘target’.

After the scaled bounding box and selected target are defined. Using the depth camera’s depth values allowed us to determine how far away each detection was. Which is then used for processing the robot’s forward and backward movement.

#### 2.1.3. Testing

( Only What happens in the Image NOT MOVEMENT )

- What happens with the bounding box when run?

- What happens when there is more than one person behind it?

- What happens when the selected target moves left/right?

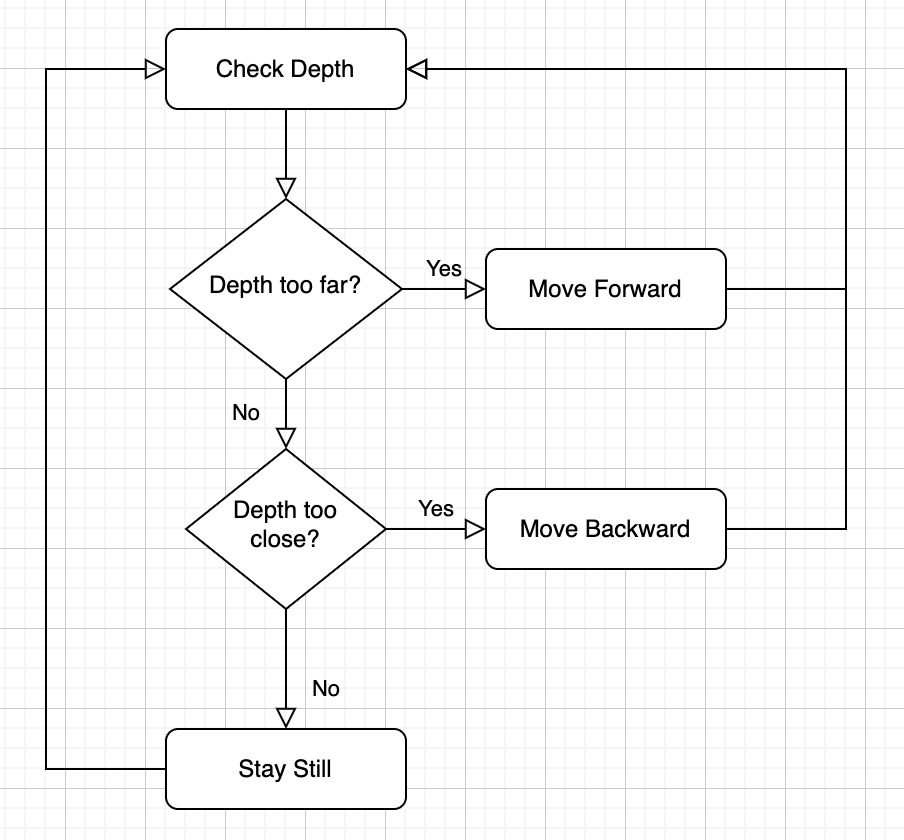
(Bounding Box Screenshots)

### 2.2. Robot Movement

#### 2.2.1. Design

The specifications main priority regarding the robot’s movement is that the robot follows the target at a ‘safe distance’, due to the vagueness in this we had decided that the robot’s forward and backward movement would stabilise around a metre away from the target. Where, the robot being too close would result in its backward movement and vice versa.

Using the flowchart below, we can see how the robot processes whether the target is too close or too far. The starting box “Check Depth”, takes the centre point of the target and compares this value with the “Too Far” and “Too Close” variables which will be discussed later. Then as seen with both of the decision boxes below, if any of these are true the robot will move forward or backwards depending on whether it is too far or too close to the target respectively.

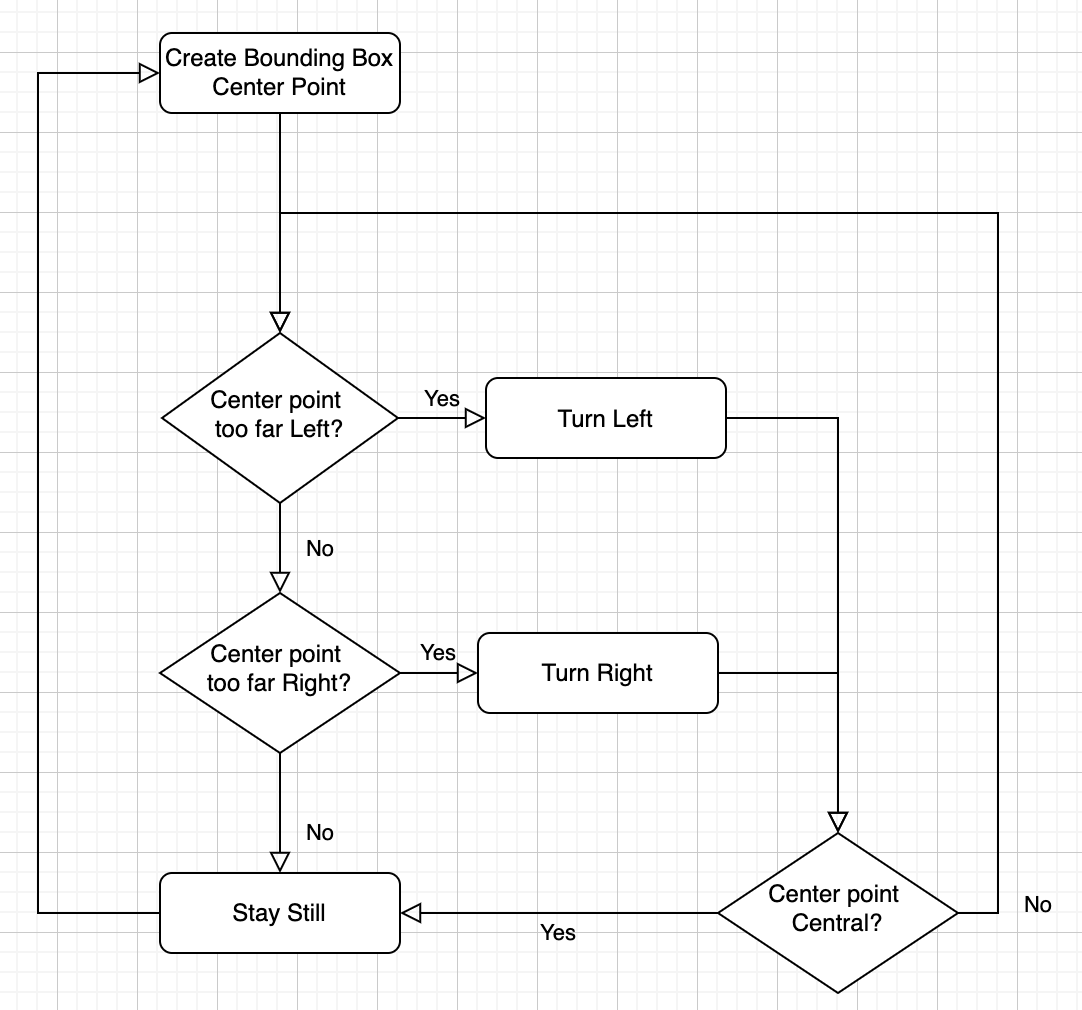


- Right/Left Movement (Turning)

- Very Far away? (maybe improvement in 2.3.3)

- etc?

Turning Flowchart:



#### 2.2.2. Functions

Turning function:

**if** ((bbox[0] **+** bbox[2]) **\*** 0.5) **>** (width**\***0.7):

robot**.**right(0.5)

**elif** ((bbox[0] **+** bbox[2]) **\*** 0.5) **<** (width**\***0.3):

robot**.**left(0.5)

Forward/Backward Function:

**if** 690 < depth < 710:

robot**.**stop()

**elif** depth > 710:

robot**.**forward(0.4)

**elif** depth < 690:

robot.backward(0.4)

- Depth / RGB Camera (Forward/Backward)

- WxH Threshold (Left/Right)

#### 2.2.3. Testing

xxx

- What happens when it finds its target?

- What happens when the bounding box moves left/right?

- What happens when the target’s depth inc/dec?

- How does the robot respond to multiple targets?

### 2.3. Human Following Test

#### 2.3.1. Test variables

xxx

(Screenshots/Pictures)

#### 2.3.2. Conclusion

xxx

- Talk about how both the detections and robot movement work together, i.e.

- What if the target moves backwards and to the side…?

#### 2.3.3. Improvements

xxx

- Rate of forward/backward movement?

- Slight Flickering in BBox?

- any more?