Navigating Application For Visually Impaired User

By

I Nyoman Gde Gedar Marchiendo Pradnyana

ABSTRACT

Common navigating tools for visually impaired people such as walking stick, and a guide dogs are not effective enough to help people navigate to the destination. The idea of communicating information to the user is still not established, which can caused misinformation. With the development of computer vision technology, ob eject detection system can be created to assist the user to navigate to their destination.

This project attempt to create a mobile application to assist visually impaired people to navigate to their destination. One of the main functionality is to communicate information to the user about the obstacle in their walking path. It is designed to be easily used and easily accessed by user with mobile phone.

Contents

List of Figures

Chapter One

Introduction

1.1 Background

As a Human being we need our five senses to do our activity, in which includes the ability to see. Vision is crucial in every aspect of our live, from doing basic activity such as identifying an object, to a more complex one such as driving a car. That is why having a difficulty to see can reduce our productivity in our life. There are some cases where it affects people until it is hard for them to navigate to their destination, which force them to utilise service and tools to help them complete their activity. However, these tools and service still has drawbacks where it can reduce t he effectiveness of the user.

1.2 Aim

The current development of technology allow us to create a machine that are able to process vision, such as robots, drone, and self driving cars. With this ability, an Application that able to identify objects can be created, where it can capture an image and process them. With this technology, we can create a tool that act as a substitute for the user to see and navigate to their destination. With the information that the image capture, the application will able to deliver those information to the user.

The aim of this project is to create a tool for visually impaired people to navigate to their destination, by constructing a system that able to identify objects, and process the information to be delivered to the user as an instruction or alert. The system need to classified which object is considered as an obstacle to the user. In order to do so, the system need to detect the object location from the user and evaluate its position.

The system will be implemented inside an application where many user able to access it at the same time. Since the system need image type input, camera access is important in creating the application, thus the application will be built as a mobile application. By using the camera phone, the application will take a picture of the user walking path and send it to the system to be evaluated. The result of the evaluation will be a the information of obstacles in the user walking path.

1.3 Motivation

The most efficient way to help people with visual impairment is with the help of a person to guide them. The ability to identify obstacles and give clear instruction to the user is what they need. However, these services are not always available for the user. Thus, they tend to use tools that can give them this kind of help.

Usually, helping tools such as walking stick or using the help of a guide dog (Wiggett-Barnard and Steel, 2008) are used to assist people who has visual impairment to navigate. Both of these tools are used together to give the user more effective way to navigate, walking stick is used to identify objects in their surroundings, and guide dog guides the user to walk in a safer path. However, these tools still have drawbacks. Walking sticks unable to help the user to identify objects in the distance. Although guide dogs are trained to keep the user safe, it still has the possibility to lose focus and got distracted to their surroundings, which can potentially mislead the user. Furthermore, the availability of guide dog are less in some country and it is expensive to train them. In addition, both of these helping tools are not able to communicate with the user. Walking stick cannot tell the user about the obstacle position and the user can't fully understand what guide dog tries to tell them. Thus, the information and instruction sometimes is not clear.

Chapter Two

Literature Review

2.1 Object Detection

In the past decade the development of computer vision technology allow us to understand more on how machine able to see patterns in an image and identify those pattern to be objects, which brings us the topic of object detection. As stated in (Zou et al., 2023), The goal is to create a model that able to detect object with high accuracy and efficient speed. Many modern technology is using object detection as the foundation of their system such as auto pilot car, motion tracker camera, face recognition, etc. This can come to realisation because the development of object detection algorithm allow the system to detect object more accurately and faster speed. Compared to the early algorithm, the current algorithm is 100% faster and 100% more accurate (Zou et al., 2023). This improvement is supported with the invention of deep learning in the year 2012, where it implements the neural networks system which imitates the work of human brain. This method allow the model to be trained using more complex data, without having to do a lot of works in the data processing steps.

This project will use an existing object detection algorithm to construct the model for the system to identify and locate the obstacles in the image of the user walking path.

A high accuracy model is needed to determine objects location in the picture. Speed is also important, it should be able to match the user walking speed in order to give proper information the user. Late respond and inaccurate bounding box may result in a false information and can create a miscommunication between the user and the system.

There are many algorithm available and open source can be used in this project. Algorithm that is specialise in object detection such as YOLO (Jiang et al., 2022), EfficientDet (Tan, Pang, and Le, 2020), DETR (Carion et al., 2020), and CenterNet (Duan et al., 2019) are famous for their accuracy and speed. However, each of these algorithm has their own advantages and disadvantages. The most popular one is YOLO which is known for its speed but still yield a high accuracy result. With the popularity of the algorithm, many information such as tutorial and tools are available to help the training process and understand how to utilise it efficiently. After comparing and considering other algorithm, YOLO algorithm was chosen to construct the model.

2.2 Related Works

There exist works and research with a similar premises, but different approach and aim. Sun et al. (2019) create a dataset and model that evaluate dangerous obstacle in walking path for people to be more alert while using their phone while walking, the model able to calculate how the dangerous the obstacle is by identifying its shape and create an approximation of its size. (Pundlik et al., 2021) make an evaluation on how effective a chest mounted device for visually impaired to navigate inside their house. The proposed device has a similar functionality as the application in this project, however it utilises a video input to locate the object location and it will give signals to the user via vibration.

This project is heavily influenced by the usage of camera assisted parking system in a car and the usage of object detection in autopilot car. The concept of sensors in reverse

mode in a car is one of the inspiration on how the application evaluate the obstacle location. (Keall, Fildes, and Newstead, 2017) create an evaluation on how camera assisted vehicle able to reduce the number of accident that caused by driver while parking in reverse. An in-depth research on how path tracking algorithm is used in autopilot cars is discussed in (Rao and Yang, 2020), which also can be used in this project to guide the user based on the information in the captured picture.

2.3 Application Development

One of the aim in this project is to create a tool that is available for people with visually impaired to use easily without having to buy additional hardware equipment. Therefore the system need to be build in an environment that is easily accessible by many people. The best way to deliver such system is to create an application and deploy it to the internet. Additionally camera access also needed to capture images, therefore mobile application is the most suitable approach to develop this tool.

Framework that is available cross platform is preferable such as Flutter and React Native. Wu (2018) create a paper of comparison between these two frameworks. The difficulty and performance are aspects that need to be looked out for. It is stated that Flutter performs faster than React Native. However, the learning curve of React Native is easier than flutter. React native also provide a better documentation and libraries to develop and test the application. Expo platform (Zammetti and Zammetti, 2018) in React Native is one of the tools that allowed the user to easily setup a project and test it to production. Therefore, react native was chosen to develop the application in this project.

Chapter Three

Object Detection Model

This chapter will discuss about the theory and methodology used in the object detection model. The **Dataset** section will discuss about what dataset that suitable for the use case of the project, and the process to gather it. The **YOLO Algorithm** section will discuss about the theory and methodology used in the model architecture. The process and methodology to train the model will be discuss in the **Model Training** section. The last section, **Model Evaluation** will talk about the evaluation of the training result and the methods that were used to evaluate the model.

3.1 Dataset

Dataset for object detection have properties for training purposes. The main properties that need to exist for object detection training are bonding box coordinate and the class annotation of the object in the frame. Bounding box is a rectangle shaped box that wrapped around the object position in the picture frame and class annotation is the name of the object's class which it belongs to. Other properties such as segmentation and visual relationship also exist, however it is not needed for training the model in this project. The object classes are also important when choosing a dataset. This project need to detect ob-

stacles that exists in the walking path area or on the pavement. One of the issue is that obstacles are random, any type of objects can be considered as an obstacle if the object is blocking the walking path of the user. In addition to that location of the user also determine the shape and colour of the objects, for example, common obstacles in Birmingham are not the same as in Seoul.

One of the strategy is to use a large scale dataset to train the model, however it is challenging to find a dataset with such characteristic. A large scale dataset such as ImageNet (Russakovsky et al., 2015), COCO (Lin et al., 2014), and OpenImage (Krasin et al., 2017) don't have a comprehensive set of objects classes that can be considered as obstacle on the walking path. Most of the dataset that is suitable for this project are not publicly available to be used. For example, WideView Sun et al. (2019) created a dataset for pedestrian safety using a wide view camera angle, and SideGuide Park et al. (2020) created a dataset specific for visually impaired people complete with segmentation. SOID Ahmed and Yeasin (2017) also worth to mentioned, because it has a complete dataset of common obstacle on the sidewalk, however it is an image classification dataset thus it is not suitable for training an object detection model.

Because of the constraints, this project will limit the range of object classes that will be trained to the model. The dataset will only consist of common obstacles that can be found in the area around Selly Oak, Birmingham, United Kingdom. The most common obstacles in this area are: Waste Container, Furniture, Street Light, Car, Tree, Road blocks, Fire hydrant, Pothole, Electronic appliance. OpenImage dataset has a quite relevant object classes with this project requirement, although some classes such as pothole, road block, and electronic appliances are not available, it is still sufficient to train the model with the dataset.

Figure 3.1: Example of image data with bounding box and annotation

3.1.1 Training and Validation

The amount of training and validation data is crucial to the performance of the detection model. In this project the distribution of training and validation data are 80:20. In the case of training an object detection model, the more data to be trained the better the accuracy of the model. However, it is only true if the amount of outliers in the data is less. It is also worth to mention that the larger the training data the longer the training time depends on the environment and specification of the system that used to train. Therefore, a large amount of data will not always give the best result, since it takes longer to train and the accuracy is not guaranteed to increase. To find the correct amount of training data, experiments is needed. This project uses 300 training data and 60 validation data for each 7 classes: 1. Bicycle, 2. Car, 3. Fire Hydrant, 4. Furniture, 5. Street light, 6. Tree, 7. Waste Container. In total of 1200 data for training and 180 for validation.

OpenImages dataset provide bounding box and annotation properties to its data. Therefore, the user can immediately use the data to be trained. However, some configuration need to be made in order to use the dataset in YOLO. A label of the bounding boxes is needed for each image data in dataset, including the training and validation data. The OID toolkit (Vittorio, 2018) offers various features for working with the OpenImage dataset easily. One such feature enables users to easily download specific classes from the dataset with a specified quantity. Additionally, the toolkit includes a configuration for labelling data for the YOLO configuration. The label is a *.txt* format file with the name of the title of the data. The content of the file is lines of string containing the detected objects with its class and bounding box coordinate inside the picture frame. The format of the string will be:

$$object\_class\ center\_x\ center\_y\ width\ height. \tag{3.1}$$

object_ class will be a number of the class that the object assigned to, *center_ x* will be the center position of the x axis or the horizontal side of the object's bounding box, *center_ y* will be the center position of the y axis or the vertical side of the object's bounding box, *width* and *height* will be the width and height of the bounding box. The purpose of this labelling is because YOLO will train the model using different sizes of the image to increase the accuracy of the prediction, with this label format the bounding box will not change although the image size is different.

3.2 YOLO Algorithm

This project will use YOLO (You Only Look Once) version 4 as the object detection model. It is famous for its accuracy and speed, and it mainly used for real-time object detection. The YOLOv4 model architecture can be divided into 3 groups, backbone, neck, and head. This section will cover each groups in the model and the methods it used.

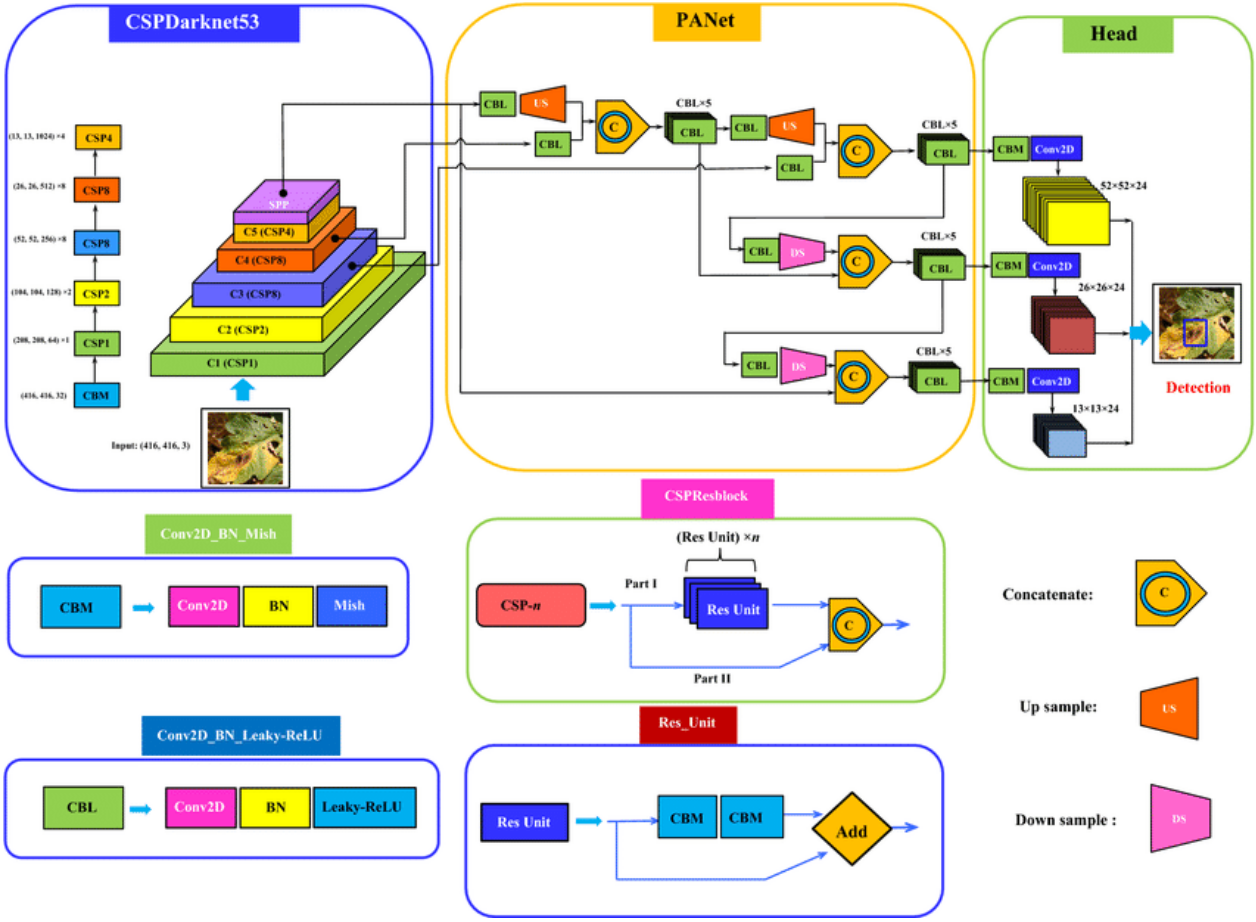


Figure 3.2: YOLOv4 architecture

3.2.1 Backbone

Cross-Stage Partial Network

In each version of YOLO, a new modification is applied to the network to increase the process effectivity and accuracy. In order to train the model to be more robust, Cross-Stage Partial Network is introduced in YOLOv4 as the backbone of the algorithm. A Modification of the previous network in YOLOv3, CSPN is used to improve the performance of the algorithm by reducing the computation by maintaining the accuracy. The approach will divide the network into multiple stages and each stage has their own specific object detection

task. The stages are connected and it allows information to flow between them, which enable the network to reuse the features from previous stage which allow the model to work faster. In addition to that, the division of the network allow the usage of multiple kernel size, which can improve the accuracy of the training. CSPDarknet-53 is an improved version of Darknet neural network which is used in YOLOv4. It is a convolutional network that is widely used in object detection and image classification

Spatial Pyramid Pooling

After the input image is processed in the CNNs, the output will be the feature map of the image. The feature map will then be fed into a Pooling layers where it will reduce the size of the feature maps. The reason pooling is applied to the feature map are dimensionality reduction, processing variants in input, better selection of the feature. In YOLOv4, SPP (Spatial Pyramid Pooling) is introduced to apply the pool operation to the feature map. It works by pooling features from multiple regions of the same feature map at different scales, which allows the network to handle inputs of various sizes. This is done by dividing the feature map into regions with pyramid shape, where each region corresponds to a different scale. The features are then pooled separately in each sub-region, and concatenated together to form the final feature vector. This results in faster inference and better accuracy. The output of SPP is fed into the fully connected layers, which predict the bounding boxes and class probabilities.

3.2.2 Neck

Path Aggregation Network

Instance segmentation is one of the stages in YOLOv4 object detection process. the purpose of this stage is for identifying, classifying and localizing various instances (objects) present in an image at the pixel level, and requires retention of the finest of the features present in the image. This enable the model to not only detect object but also segment them into individual instances, allowing for more precise localization of object images and also helps to differentiate between objects that are near to each other. By identifying and segmenting each individual instance of an object, YOLOv4 can provide more detailed information to downstream applications such as tracking and recognition.

3.2.3 Head

Convolutional Neural Networks

CNNs (Convolutional Neural Networks) layers is a deep learning algorithm that utilise Convolution to its layers. Convolution is a mathematical operation used to extract features of an image. To process an image using convolution, it is first converted into a matrix by assigning numerical values to its pixels. This matrix is then divided into smaller grids, each with dimensions of GxG. A filter or kernel, which is a small matrix of numbers, is applied to each grid using a dot product operation. The resulting value from the dot product is placed in the center of the grid and this process is repeated for each grid. The results from all of the dot products are combined to produce a feature map for the original image. By training this layers to a large dataset, it can learn to detect the features and characteristic of the dataset and identify them as a pattern of an object.

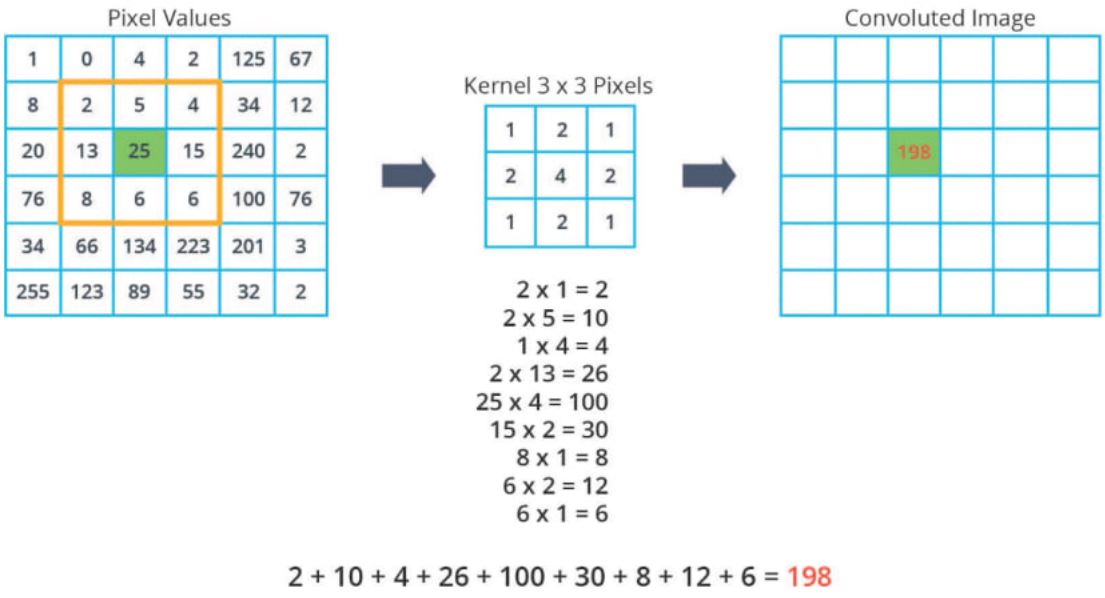


Figure 3.3: Example of convolution

Grid System

The difference between YOLO algorithm with traditional object detection algorithm is the utilisation of grid system to process the input image, instead of using the sliding window approach. In the traditional sliding window approach, there exist a window with a fix size that will slide across the image in an iterative manner. In each iteration the window will try to detect an object in its region. This approach can be computationally expensive if the dimension of the input image is large, because the larger the dimension the higher the iterative steps. In addition to that, the classifier (the prediction of the detected object) is implemented independently which can create a redundancy in detecting the object.

In YOLO, grid system is introduced where the image is divided into a grid of cells, and each cell is analyzed to determine the presence of an object using a probability value. A threshold is applied to filter out low-probability cells, and a Non-max suppression technique is used to merge overlapping bounding boxes to fit the object more accurately. Essentially,

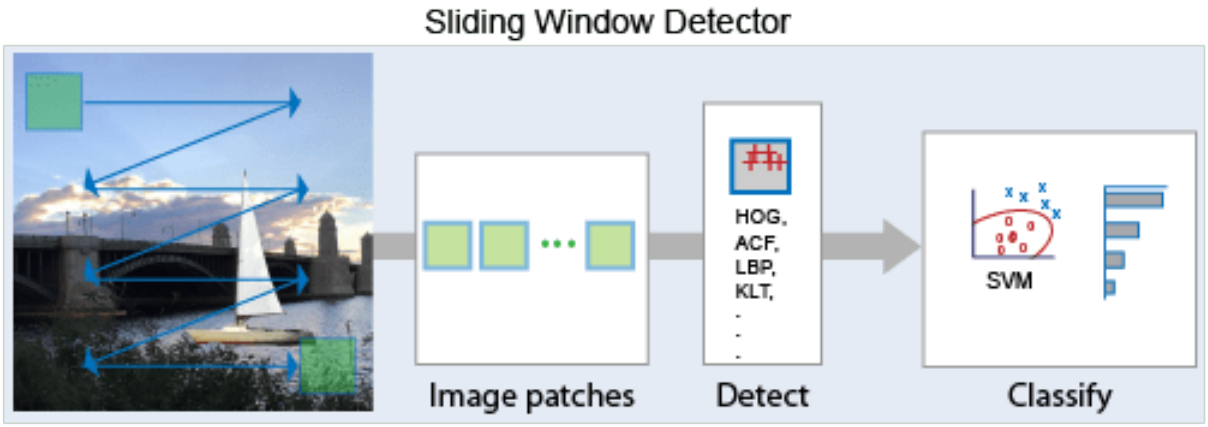


Figure 3.4: Representation of sliding window approach

Non-max suppression is a method to eliminate redundant bounding boxes and merge the remaining boxes to better fit the object in the image.

By using the grid system, YOLO algorithm able to increase the performance of its detection. It is because grid system allow the algorithm to use single neural network to perform both object detection and classification task. Furthermore, it is more efficient on detecting object with different scales and location than using sliding window approach.

Anchor Boxes

In the first version of the algorithm, the grid system assign prefix bounding box to each cells. This meant that the prediction of the object is not flexible since the bounding box size is fixed. Resulting in low accuracy to detect object that were significantly larger or smaller than the default bounding box.

YOLO version 2 offers a modification to the default bounding box using anchor boxes applied to each cells. It has the same concept as the bounding box, however instead only using one box, multiple anchor boxes is applied to a single cells each with different dimension. The idea is to find the best fitting box to the detected object in a single cell, allowing the

model to accurately detect object that has different aspect ratio and sizes. In order to find the best fitting bounding box, Non-max-suppression method is applied to filter out the boxes.

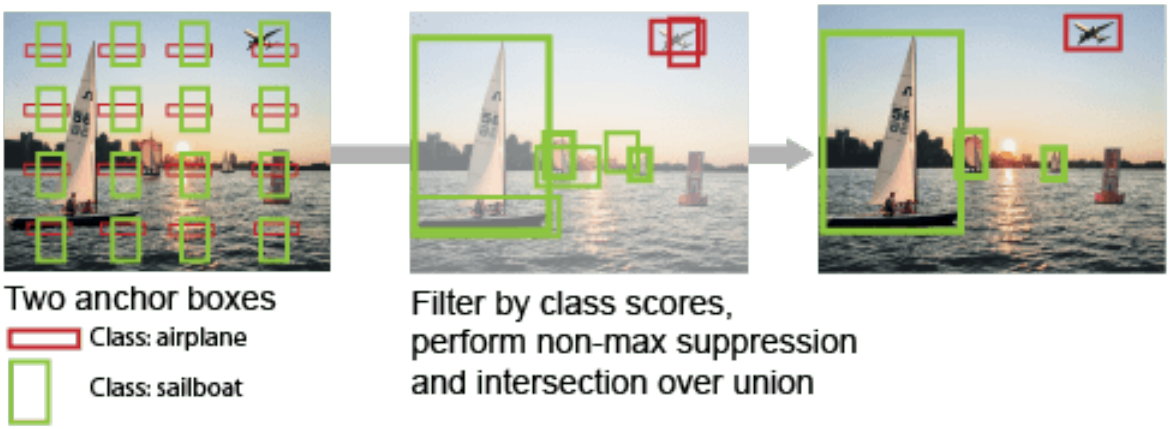


Figure 3.5: Representation of Anchor boxes in grid system

Non-maximum-suppression

As the model tried to find the best bounding box, it will try to generate many candidate boxes in the process. Thus it will need to filter out the candidate boxes to find the best fitted one. The first filter will filter out the candidate boxes that has the probability score lower than the given threshold. The issue is when there exists boxes that pass the probability threshold, however it overlaps with the others. In this case, NMS (Non-maximum-suppression) algorithm is applied to find the best fitted box for the detected object. The algorithm will apply score to the overlapping boxes and filter out the box that has lower score than the threshold. It will iterate the same process until the last bounding box with the highest score is found. The score that applied to the overlapping bounding box is the IoU score.

System Specification	
CPU	AMD Ryzen 7
RAM	8GB
GPU	NVIDIA RTX 2070
OS	Windows

Table 3.1: System specification for model training

Intersection Over Union

IoU (Intersection Over Union) is a metrics to calculate the union of two overlapping bounding box. The purpose of IoU score is to give a quantitative measure of the overlapping area between two boxes, the bigger the area the higher the IoU score. The selected box will be the one with higher IoU score than the given threshold.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.2}$$

3.3 Model Training

This project use (Bochkovskiy, n.d.) repository to train YOLOv4 model. The repository provides tools to work with the output of the bounding box, and it also provides tools to train the model with GPU. This section will discuss the prerequisites and configuration to train the model. The specification of the system to train the model is listed in the table below

3.3.1 Dependencies

Since this project trains the model in personal computer, the environment of the system needs to be configured. As default, the code will be trained using CPU, however it will take a longer time to train using it. Therefore, GPU will be used to train the model. It is also worth to mentioned that the code itself needs a couple library, which is not present in the system and needs to be installed. Here are the list of dependencies that need to setup before training the model:

- Git

  is used to clone the repository of Bochkovskiy (n.d.) where the source code of the training model located. Cloning a repository is the same as locally copy the source code from github to the local machine.

- CUDA

  CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model developed by NVIDIA. It is used to accelerate the computation process by offloading some of the computations to the GPU instead of relying solely on the CPU.

- CMake

  Before building the program, the dependencies need to be configured first. CMake is a common tools in windows to generate build files and dependencies, using the setup that the user provides.

- MSVC

  The source code mainly uses C to develop the model. Since the system operating system use Windows, it is required to install and setup MSVC (Microsoft Visual C++) to compile and build the program.

- OpenCV

  In YOLOv4, OpenCV is used for various tasks such as reading the input images and videos, pre-processing the images, applying post-processing techniques such as drawing bounding boxes and labels on the detected objects, and displaying the results. OpenCV provides support for hardware acceleration, which can significantly speed up the processing time of the YOLOv4 algorithm on systems that have compatible hardware.

- cuDNN

  CUDA Deep Neural Network can significantly speed up the computation of convolutional layers, which are a critical component of the YOLOv4 model. By leveraging the parallel processing power of GPUs, cuDNN can perform the large number of convolutions required by the YOLOv4 model much faster than a CPU-based implementation.

- Python

  Python is also used in the source code, therefore the system need to installed Python version 3 in order to use the model.

3.3.2 Training

The model were trained using scaled YOLOv4 with configuration from Bochkovskiy (n.d.) repository. The repository provides a configuration template file to train custom object using YOLOv4 model. However, since the model going to be trained with custom object some configuration still need to be made. The configuration that were made are:

The repository also recommend to use the pre-trained weights in order to increase the accuracy of the trained model. Pre-trained weights are used as an initialization of the neural network's weights with values that have been learned from another dataset. These weights are typically learned from a large dataset, such as ImageNet, and then transferred to the

Batch	64
Subdivision	32
Max Batches	14000
Width & Height	416
YOLO layer classes	7
Convolution layer filters	36

Table 3.2: Configuration for model training

object detection model to aid in the training process. The idea is that these weights have already learned general features and patterns from the large dataset, and can therefore help the object detection model to learn features specific to the object classes in the training data more quickly and accurately. The pretrained weights can also help to prevent overfitting of the model to the training data.

3.4 Model Evaluation

The model training spend approximately 24 hours, and the result of the model accuracy were measured using two metrics, Mean Average Precision (MAP) and Average Loss. The result chart can be seen in Figure 3.6. The blue line is assigned to **Average-Loss**, and the red line is assigned to **Mean Average Precision**. This section will discuss about the metrics Average loss and Mean Average Precision and conclusion about the model evaluation result.

3.4.1 Average Loss

Average Loss is one of the metric to measure how well the model predict an object. It measures the difference between the true prediction from the training data and the given output from the model. The loss is calculated using a loss function, which is the combination of binary cross-entropy and sum of squared error.

3.4.2 Mean Average Precision

Mean Average Precision is a popular metrics used in measuring the accuracy of an object detection model. It is a combination of Confusion Matrix method with IoU method. The metrics compute the accuracy by taking the average precision (AP) values of each classes of object image in the category. The AP value for each class is calculated by computing the precision and recall at different thresholds, and then calculating the area under the precision-recall curve (PR curve) for that class. Once all of the Average Precision value is computed for each class, the mAP score will be computed by taking the average of the AP across all of the classes.

3.4.3 Model Conclusion

The final score for the average loss is 1.55 and 52.9% for the mAP. Based on the measurements, it can be concluded that the accuracy of the model is relatively low. There are two reasons for this:

1. Insufficient Training Data

   Each of the object classes has their own complexity. Complexity can be determined from the variety of the shapes and colour. There are three classes in the dataset that have a high complexity rate, which are: **Car, Furniture, and Tree**. In the dataset,

the object Car is not restricted to only a particular type of car. It can be vary from sedan, pickup, to a toy car. Same as the car class, Tree class also has a high variations to its object. In OpenImage dataset, the Furniture class has sub-classes such as Table, Chair, and Sofa. Therefore it is considered as a high complexity class.

2. Dependencies and Configuration error

   The training process is carried out several times with different configurations. This is because during the training, the model may fail due to insufficient memory caused by configuration and dependency errors. One of the dependency issues is the installation of OpenCV, which does not support CUDA, thereby slowing down the training process.

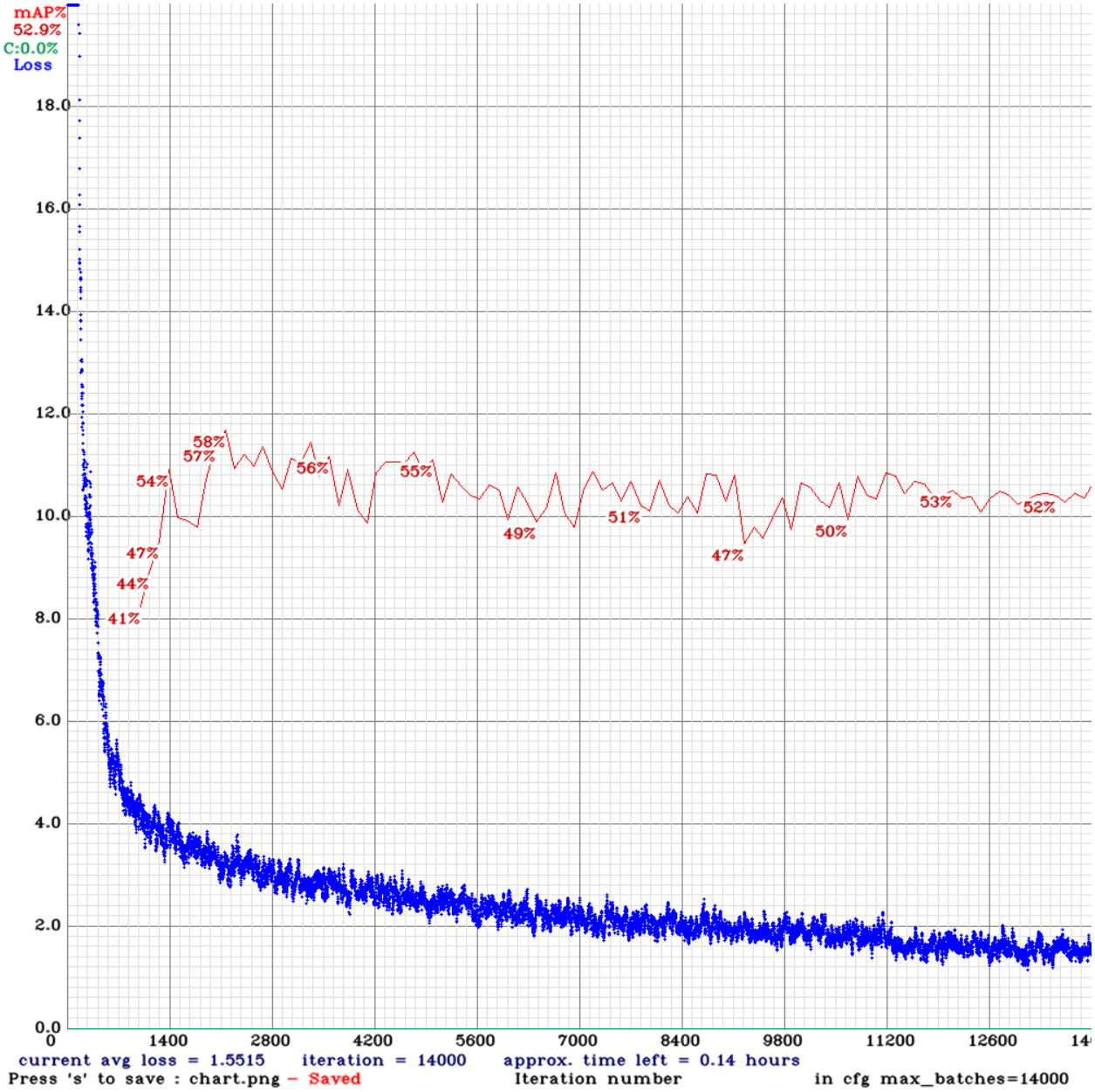


Figure 3.6: YOLOv4 Training Evaluation Chart

Chapter Four

Walking Path

Walking Path is one of the important aspect of this project. It is an area which act as the users walking path while navigating to their destination. It is important to create a walking path that is suitable for the user, because it can change the behaviour of how the user walk while navigating. The structure of the walking path is influenced from the sensors in the reverse parking camera implemented in vehicle. However, the reverse parking camera is static and only focus on the safety of the back of the vehicle (Keall, Fildes, and Newstead, 2017). The walking path in this project need to be flexible and adaptable to the user perspective, which is important to ensure the users safety. Sun et al. (2019) discuss about the obstacle in the sidewalk from the perspective of a camera phone. Using a wide lens camera, the system able to recognise which object that is classified as a dangerous object in the user walking path. However it only rely on the object shape and size, and it doesn't have the ability to tell the information about the obstacle's location. With the ideas from related works, this section will discuss the thought process on attempting to construct the walking path.

4.1 Functionality Requirements

Functionality requirements are the features that need to be implemented to the walking path, in order to be usable by the user. These requirements need to be satisfied in order to give an accurate information to ensure the users safety. The requirements are:

- **The walking path can give alerts to the user**

  An alert is a notification that provides information about an obstacle detected in the path of the user. There are two types of alerts: **dangerous** alert and **potentially dangerous** alert. **Dangerous** alert are sent to the user if the objects are blocking the walking path, with the aim providing the user information about the obstacle so that the user able to avoid it safely. **Potentially dangerous** alert are sent if there are no dangerous obstacle and the object is located in the potentially dangerous zone of the walking path area. The purpose is to give information about the user surroundings so that user able to knowledge the object that surrounds them ahead.

- **The walking path able to give information about the obstacle position**

  The information about the obstacle position is crucial in an alert notification. By using the obstacle position, user able to identify a safer path to navigate.

- **The walking path will prioritise the nearest obstacle**

  When the model process the image input, it will try to detect every object regardless of their position. However, the nearest detected object in the picture frame should be evaluated first. Therefore, the walking path need to have a distance threshold while doing evaluation for the detected obstacles.

- **The walking path able to adapt to the user perspective.**

  The implementation of the walking path system will use the image input from the user perspective taken from their phone attached to their chest. Since each person have different body proportions, it is important to create a walking path that adaptable to

the user specification. The aims is to create an accurate walking path that user can rely on.

4.2 Limitation

Commonly the pedestrian's walking path form a similar form to the sidewalk, which means, the system able to identify the sidewalk in the input picture frame. However, unlike the road which have line boundary printed on, sidewalk commonly doesn't have a similar kind of boundary. In addition to that, pedestrian cannot always relies on sidewalk, in some occasion they need to rely on other walking platform that has different textures such as: walking on dirt path, grass, etc. With these concerns, creating another detection model to detect sidewalk will take a lot of time, since it is mandatory to find or create a new dataset. However, the result of the model itself is not guaranteed to increase the effectivity of the system. Therefore limitations are introduced to the walking path to make it more general. **The walking path will only available for walking in a straight path**, this limitation will ensure the walking path is more general and works in any type of environment. However, the drawbacks of this limitation is that the system will not able to identify path that has curve to it.

4.3 Construction

This section will discuss about the process of constructing the walking path with the guidance of the functional requirements and limitation. The section will be divided into 2 parts: Properties where the properties of the walking path are defined, experiment section will discuss about the experiments including mistakes and changes, and the last one is conclusion.

4.3.1 Properties

As stated in the limitation and the functional requirements, the walking path will form an area of a straight line. The size of the area itself should be adaptable with the user body, which can be measured with the distance between the users shoulder. Which means that the user need to give another input consisting of their measurement to initialise the walking path. This will become a hassle to the user and such activity require another person to complete. Therefore, the size of area will take the average of shoulder width of the pedestrian. According to Fryar, Gu, and Ogden (2012) the average shoulder width for men is 41.1 cm and for women is 36.7. Therefore the size of the walking path area will be 50 cm.

From 4.1 the area will be divided into two, which are dangerous area and potentially dangerous area. The potentially dangerous area will be located in the outer side and the dangerous area will be in the inner side of the walking path. These two area will be separated with lines that act as a boundary. The width of the walking path area is 50 cm, which means the potentially dangerous area will be located outside of those range adding each 10 cm on the left and right side of the area.

As required in 4.1, a distance boundary in the walking path will be created, where it will function as a threshold to evaluate the object position in the picture frame. This threshold will act as the minimum distance for an object to be evaluated to the walking path boundaries. If the object is further that threshold then the object position will be ignored, otherwise it will be evaluated. The threshold of the walking path will be 200 cm from the user, to ensure the user able to process the information before making a contact with the obstacle.

To satisfy the 4.1 requirement, the walking path area will be divided into three parts: Left, Right, and Center. To divide the area, a boundary line is created in the center of the walking path that divide the walking path area into two areas. Furthermore, the location of

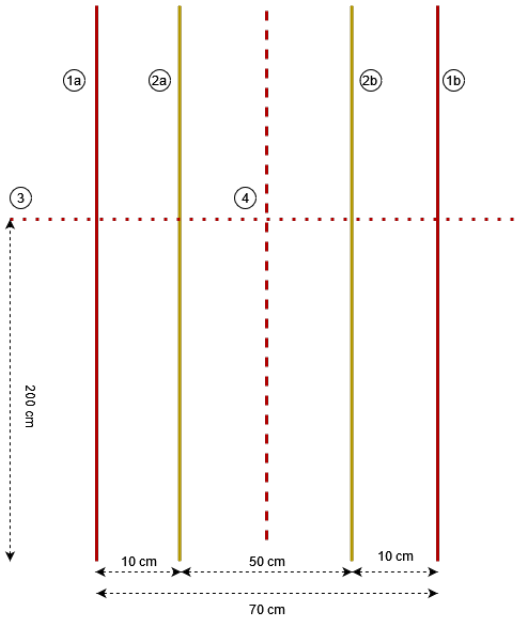
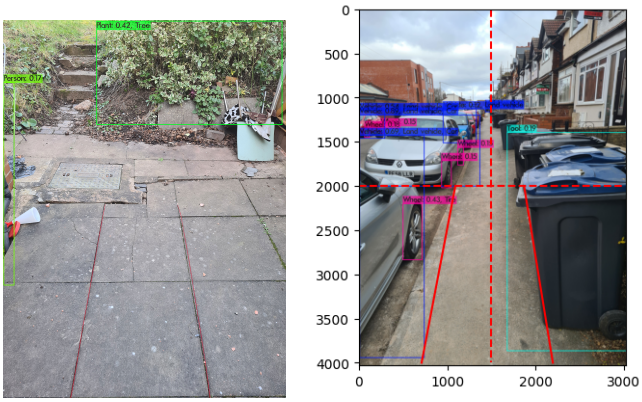


Figure 4.1: Walking path viewed from the top

the detected obstacle will be classified into three types: Left of the user, Right of the user, and in the center.

The representation of the walking path can be seen in **Figure 4.1**, the visualisation is represented in aerial view to clearly described the property of the walking path. The properties will be labeled with numbers in the circle. **Property 1** will be the boundary line to evaluate potentially dangerous obstacle, where 1a is the left boundary and 1b is the right boundary. **Property 2** will be the boundary line to evaluate dangerous obstacle, where 2a is the left boundary and 2b is the right boundary. **Property 3** will be the distance threshold to evaluate obstacle position.**Property 4** will be the boundary line to divide the left and right section of the walking path.



(a) (b)

Figure 4.2: Attempt to recreate the walking path by putting two strings on the ground and capture the image in the perspective of the user

4.3.2 Experiments

Projecting the walking path to user perspective

The visualisation of the walking path were described in 4.1, however the walking path is still constructed from the aerial perspective. The walking path need to be constructed from the user perspective, in order to evaluate the objects that are located in the user walking path. To be able to capture the users perspective, the position of the camera need to be adjust. Since the system will be implemented in a mobile application, the mobile phone camera will be used to capture the perspective of the users. One of the approach is to attach the mobile phone to the users chest using a strap while the camera facing the way which the user go. To project the walking path to the perspective of the user, an experiment is conducted by attempting to recreate walking path with a string attached to the ground with the same property.

By using the image reference from Figure 4.2a, the linear function of the boundary line can be created. *Geogebra* provides tools to easily plot a point and create a line between

two points. However, the system will utilise python in order to process coordinate from the output of the object detection model. Therefore, after obtaining the approximate coordinate from Geogebra, the boundary lines are plotted and the linear function is determined in pyplot, using the transferred coordinate. Figure 4.2b is the result of applying the projected walking path to an example result image.

Line's behaviour in image

Although the position of the projection is accurately placed according to the string in the image, that doesn't mean the walking path is correct according to the users perspective. It is because, the behaviour of a straight line in a picture frame doesn't applied to the projection. When there exist a straight line in a picture, the behaviour of the line will curved towards a point in the picture. This point is called as the *Vanishing Point*.

According to Andersen (2007) A vanishing point is a point on the image where parallel lines in 3D space appear to converge when projected onto the 2D plane. This is known as one-point perspective when the parallel lines are perpendicular to the picture plane. The vanishing point in one-point perspective corresponds to the point of view, or "eye point," from which the image should be viewed to ensure accurate perspective. Therefore, the projection of the walking path will create a vanishing point considering the fact that the boundary lines are parallel to each other and have the same direction.

By using the mobile phone camera attached to the users chest, the position of the camera will vary for each user depending of their body proportion. This will affect the position of the camera because the height and angle between the camera and the ground will create a different vanishing point in the picture. Furthermore, if the vanishing point in the picture changes then the shape of the walking path will also change. One of the approach to answer this problem, is to create a system that able to detect vanishing point in an image.

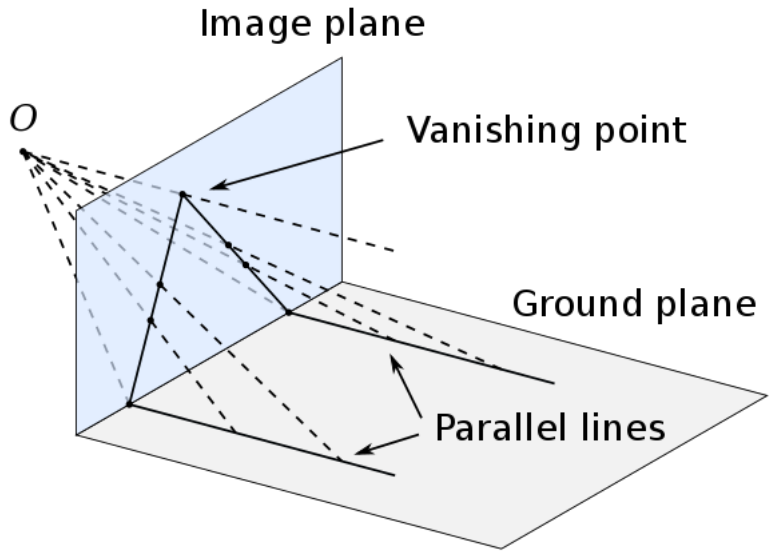


Figure 4.3: Representation of one-point perspective by Mgunyho (n.d.)

With the popularity of the autopilot car technology, many researchers attempted to construct a model that able to detect vanishing point by processing input image. Mgunyho (n.d.) create a model that able to detect vanishing point using Locally-Adaptive-Soft-Voting (LASV). Another attempt also conducted by Chang, Zhao, and Itti (2018), which created a deep learning model to find vanishing point based on google streeet view images. These example attempts have a relatively high accuracy in detecting the vanishing point in the image, but the dataset that they used is originated based on road dataset for vehicle. A similar attempts also conducted for sidewalk database (Jeon et al., 2014), however the accuracy of the model is still low and the resources is not open for public. With the time constraint, and low resources for the dataset, this project will not create a model that able to detect vanishing point in the image. Instead, the projection of the walking path will be based on one particular image in Figure 4.4, and the users need to adjust the camera based on the walking path provided by the system.

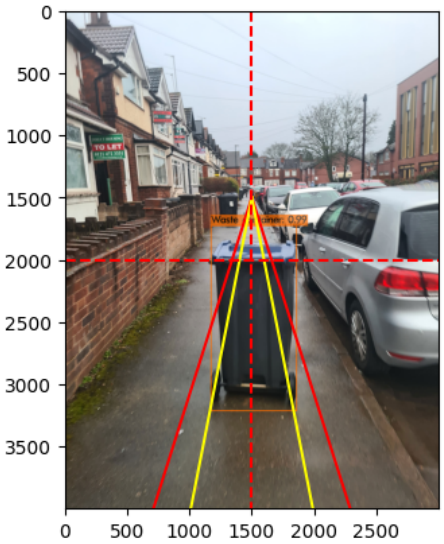


Figure 4.4: Projection of the walking path based on the vanishing point in this particular image

Precision in evaluating the obstacle bounding box

In figure 4.2b, it appears that the obstacle's bounding box on the right side is violating the potentially dangerous boundary line in a significant amount of area. However, the original size of the obstacle doesn't appear as large as the bounding box. In some cases, this could happened because of the low-accuracy of the model. In other case, it is because the direction of the bounding box is not parallel to the direction of the boundary line. This can create a large part of the bounding box violating the boundary line. Therefore, the bounding box needs to be adjusted so its direction will be parallel to the boundary line.

The direction of the bounding box will always be perpendicular to the x-axis, thus one of the approach is to rotate the image in the amount of angle between the boundary line and the x-axis. The rotation will make the boundary line to be perpendicular to the x-axis and the bounding box direction will be the same as the boundary line. This approach will divide the image into three sections: Left, Center, and Right, where each section will be fed into the model. The drawback of this approach is the increasing processing time to evaluate

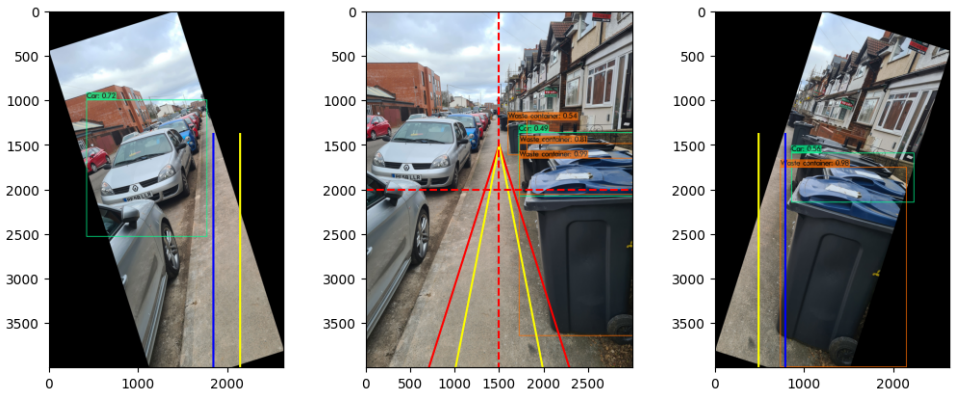
(a) Before rotation (b) After rotation

Figure 4.5: Direction of bounding box affect the evaluation accuracy

the obstacle. The tradeoff is still valuable, because the increase of evaluation accuracy is high, and the difference of processing time is low. This is because the model able to detect one input image in the range of 0.5 to 1 second.

4.3.3 Conclusion

After doing research and attempting to construct the walking path, the final solution of the walking path can be seen in Figure 4.6. When the system received image input, it will be divided into three section: Left, Center, and Right. Left and Right sections will be rotated to match the boundary line direction using OpenCV library in Python. After rotating the image, it will then be fed into the model for detection which will give result of bounding boxes coordinate. The coordinate of these bounding boxes will then be evaluated with the boundary lines exists in each section. In every sections, the bounding box will only



(a) Left Section (b) Center Section (c) Right Section

Figure 4.6: Representation of the solution to the walking path

be evaluated if its y-position is less than the distance threshold line which is represented with the red dotted line. Left and Right section will evaluate the coordinate using the blue line (potentially dangerous line) and yellow line (dangerous line). The center section will evaluate based on the red doted line on the center to detect if the obstacle located on the center of the walking path. Because of time constraint and lack of data to train the a vanishing point detection model, the walking path will not be able to adapt to the user perspective, which made the user to manually configure the camera position to match the given walking path.

4.4 Coordinate Processing

The solution of the walking path have the coordinate for each line that can be used to evaluate the obstacle position. The evaluation will be conducted using python and mainly *pyplot* library. The main process of the coordinate processing can be seen in Algorithm 1. The result from the model detection will be a list consist of three lists from the detected bounding boxes in Left, Center, and Right image section. The code will iterate each of this list and plug it into evaluation function and the evaluation result will be processed through a function to give a string contains the information of obstacle in the image.

Algorithm 1 Main Process

1: *potentially_dangerous* ← [] ▷ List to store potentially dangerous obstacle location
2: *dangerous* ← [] ▷ List to store dangerous obstacle location
3: *bbox_lists* ← *result_from_model*
4: **for** *i* **in** *bbox_lists* **do**
5: **evaluate_left**(*i*[*left*], *dangerous*, *potentially_dangerous*)
6: **evaluate_right**(*i*[*right*], *dangerous*, *potentially_dangerous*)
7: **evaluate_center**(*i*[*center*], *dangerous*, *potentially_dangerous*)
8: **end for**
9: *ans* ← **parse_answer**(*potentially_ dangerous*, *dangerous*)
10: **return** *ans*

The evaluation function has the responsibility to evaluate the obstacle and store the position of the obstacle. For example in Algoirthm 2, **danger_left** function in line 3 will take the rightmost coordinate of the bounding box and evaluate the position according to the danger line on the left of the walking path by returning a *True* boolean if the **rightmost position** > **danger line coordinate**, and *False* otherwise. The same procedure also conducted to evaluate the potentially dangerous line using **p_danger_left** in line 4. Similar approach also used to evaluate the list of bounding box for the right side image. To evaluate the center, the function need two location parameters, the leftmost and rightmost coordinate. If the condition **leftmost** < **center line** < **rightmost** was satisfied then the function will return *True*, and *False* otherwise. Furthermore, line 5 and 9 in Algorithm 2 will check the distance of the obstacle, if the condition is not satisfied then the obstacle will be skipped and not be stored to the location storage.

To store the position of the obstacle, two type of list were passed through the evaluation function. In Algorithm 1 line 1 and 2, *potentially_ dangerous* and *dangerous* were declared to store the obstacle location. In Algorithm 2 line 6 and 10, the function will store the location of the obstacle by appending the position to the location storage. The approach

to store the location is by prioritising the dangerous obstacle after the potentially dangerous one. Therefore, if there is one dangerous obstacle on the corresponding side, the function will immediately return the position and ignoring other objects. The difference of handling potentially dangerous obstacle is that it will keep iterating the coordinate list until it is finish or there exist a dangerous obstacle in the list.

Next process is to parse the answer using the location storage in Algorithm 2. The function will check if there a dangerous obstacle detected, if it does then it will only process the dangerous obstacle and immediately return the information. To differentiate the output between dangerous, and potentially dangerous information, different strings is assigned to each type. For dangerous obstacle the information string will be *"Caution! dangerous obstacles on your list of position sides"* assigned to the ans variable in line 7. For potentially dangerous obstacle the information string will be *"There are obstacles on your list of position sides"*.

Algorithm 2 Evaluate Left Function

```
1: function EVALUATE_LEFT(dangerous, potentially_dangerous, coordinate_list)
2:     for i in list do
3:         danger_state ← danger_left(i.right)
4:         p_danger_state ← p_danger_left(i.right)
5:         if danger_state & i.distance ≤ distance_threshold then
6:             dangerous.append("left")
7:             return
8:         end if
9:         if p_danger_state & i.distance ≤ distance_threshold then
10:            potentially_dangerous.append("left")
11:        end if
12:    end for
13: end function
```

Algorithm 3 Answer Parsing

```
1: function PARSE_ANSWER(potentially_dangerous, dangerous)
2:     ans ← ""
3:     danger_ans = "Caution! dangerous obstacles on your"
4:     p_danger_ans = "There are obstacles on your "
5:
6:     if dangerous.length > 0 then
7:         ans ← danger_ans + dangerous.join()
8:     else
9:         ans ← p_danger_ans + potentially_dangerous.join()
10:    end if
11:    return ans
12: end function
```

Chapter Five

Application Design and Engineering

This chapter will discuss about the construction of the mobile application by utilising the object detection model and walking path system. To construct the application, this project will use the Software Development Life Cycle as the guidelines, where the steps to develop the application consist of Defining requirements, Designing, and Testing.

5.1 Requirements

Requirements is a description of what the application must do or the features that need to be implemented in order for the user able to use it, thus the purpose of the application can be achieved by satisfying the necessary requirements. In this section the requirements will be divided into two types, Functional Requirements, and Non-Functional Requirements.

5.1.1 Functional Requirements

Functional requirements are statements of how the application should function towards inputs. These requirements are labelled according to their priority. Label **M** is

necessary to implement with a high priority, which is essential to implement in order for the application to work. label **S** is not necessary with a medium priority, where it is should be implemented to the application. Label **C** is optional with low-priority, the implementation of this requirements are not prioritise and can be ignored without affecting the performance of the application. The functional requirements will be divided into two parts, Capture and Sending Information and Obstacle Detection. This division is based on the main functionality of the application itself.

Capture and Sending Information

1. **M** - Automatically capture users perspective using camera
2. **M** - Process string result using text-to-speech
3. **M** - User able to see the walking path in the application
4. **S** - User able to change the specification of the information

Obstacle Detection

1. **M** - Able to detect obstacle in the input image according to the training data
2. **M** - Evaluate object position in the image
3. **M** - Able to process information about dangerous obstacles
4. **M** - Able to process information about potentially dangerous obstacles
5. **S** - Able to process the image using users specification

5.1.2 Non-Functional Requirements

Non-functional requirements, are the attributes that describe how well the application perform. Non-functional requirements will focus with performance, reliability, security, usability, maintainability, and other aspects of the application. These requirements are:

1. Fast Input Processing

   In order to maintain the correctness of the information, the system need to process the image input as fast as the walking speed of the user. The faster the duration the better.

2. Accurate detection

   Precision of the bounding box need to be accurate for evaluating the obstacle position. This will affect the correctness of the information.

3. Accessibility for visually impaired user

   This is one of the crucial aspect in the application development. The user of the application will targeted to visually impaired people, therefore the accessibility of the application is important to ensure it is accessible for the corresponding user.

5.2 Design

In the design process the functionality of the structure and flow of the application will be defined. The requirements from the previous section will be the guide to designing the application. The process of designing the application consist of designing the use case, flow diagram, architecture, and UI/UX.

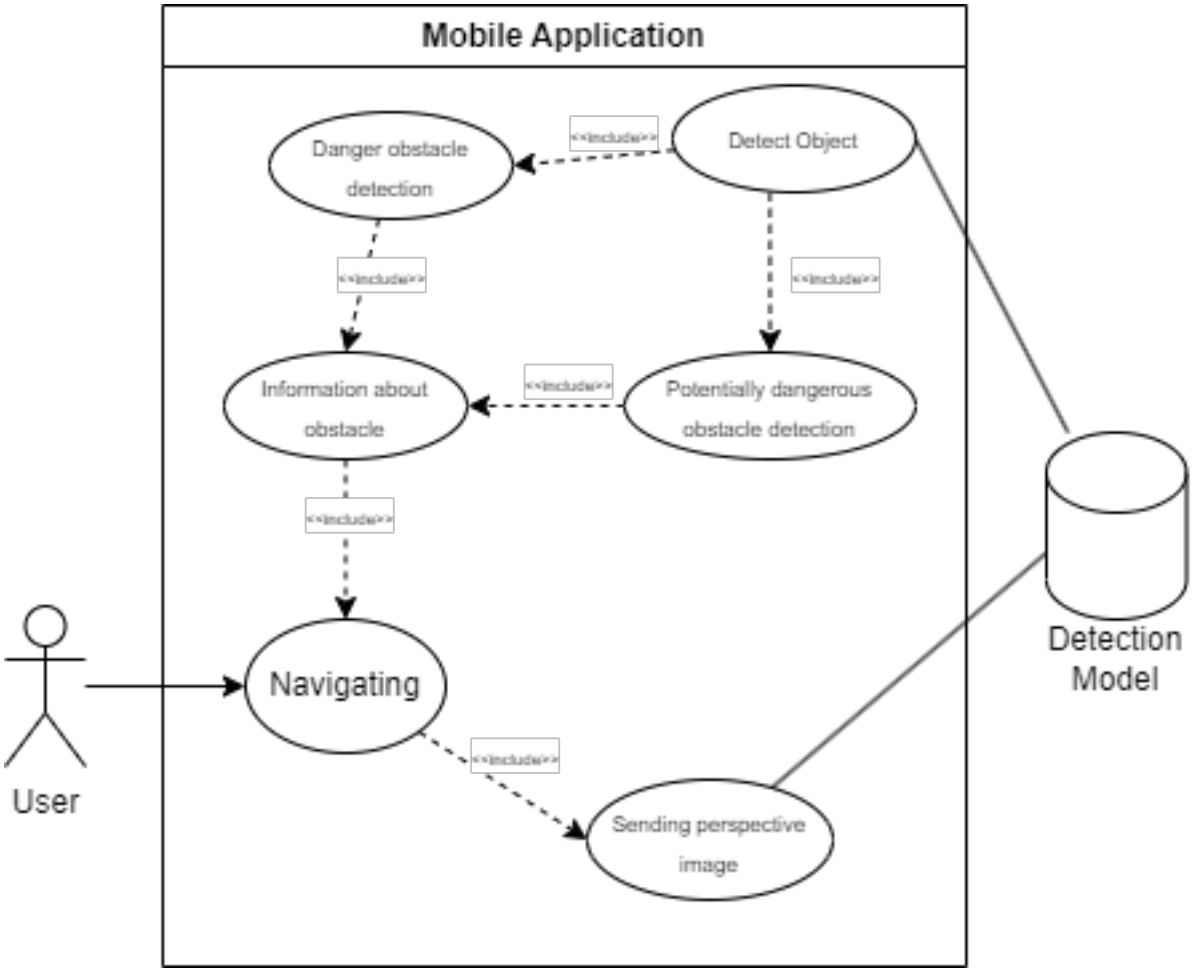


Figure 5.1: Use case diagram

5.2.1 Use Case

Use case are used to describe the interaction in the system using the perspective of the actors. Actors are subjects that can interact with the system, in this case the actors are user and detection model. The use case will define the functionality that are present in the system, and user able to interact with it. This application will only have one use case which is when the user navigate to their destination using the application. The interaction is represented within the use case diagram in figure 5.1.

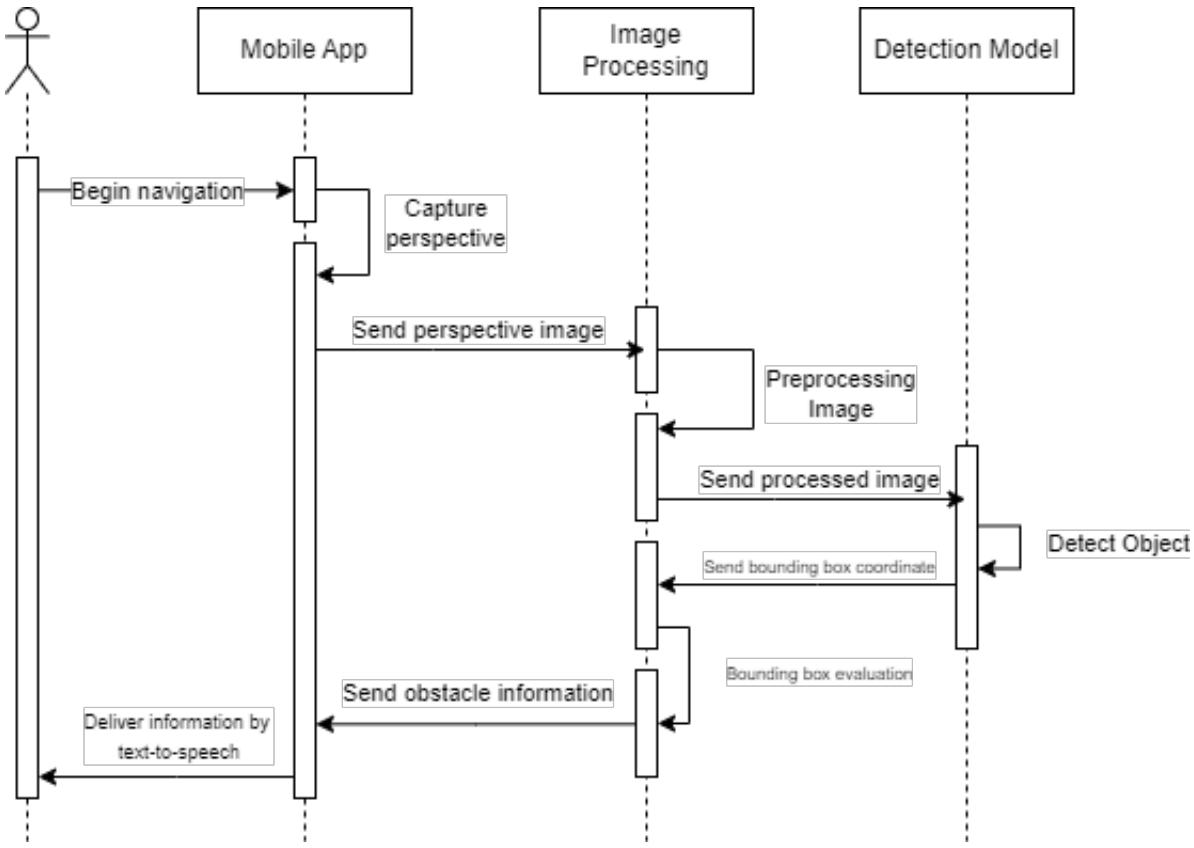


Figure 5.2: Sequence Diagram of navigating process

5.2.2 Sequence Diagram

Sequence diagram provides a visual representation of how functions communicate with each other over time, showing the sequence of messages or interactions between functions in a chronological order. The sequence diagram is represented in figure 5.2, where it describe the sequence of a loop in navigating process. The idea is that the navigation process will loop continuously, until the user stop the process.

5.2.3 Architecture

The architecture of the project can be separated into two main section, back-end and front-end. Back-end section will responsible to handle server-side functionality such as

processing input image and doing object detection. The front-end will responsible to handle client-side functionality including capturing image, input processing, and output processing. The two main section, will communicating through API with HTTP request.

Back-end

The back-end of the application is mainly built using Python and its library. Python were chosen, because most of the image processing steps are utilising python library such as Pyplot, OpenCV, and Numpy. Furthermore, the connection between front-end and back-end is established using FastAPI web framework where it provides an easy solution to construct a RESTful APIs using Python.

The backend also responsible to communicate the input images through the detection model. The model itself is an executable file which can be accessed through shell command. Python os library provide function to execute shell script which will invoke the model file to be executed. The script will include the path to the input image and the parameter that the user specify from the front-end.

Front-end

As previously discussed in literature review, the front-end section will use React Native Expo. React Native application is mainly constructed using JSX, HTML, and CSS. Expo helps to build a React Native application by providing an easy access to the mobile phone so developers able to test the application in a real environment easily. Expo also provide tools to utilise the phone camera and media library and other UI/UX tools such as Favicon to give variation to the application display.

One of the responsibility of front-end is to handle input from the user. The input itself consist of strings and image file. The image file is the perspective image of the user and

the strings will be the configuration of the information that the user desire. Expo camera library will capture the perspective image and form data will be used in order to send the input through a fetch function

API

As specified in the back-end section, the application will utilise REST API using FastAPI library from Python. It utilise standard HTTP methods to communicate and exchange information between front-end and back-end. The HTTP methods that were used in this application is only POST method, because the functionality of the application can be satisfied by using only POST method. In the current development of the application, the version of the API is already in version 2, the details of the request are:

- **api/v1/uploadfile**

  Early version of obstacle detection from the server. The features only limited to detect potentially dangerous obstacle.

- **api/v2/uploadfile**

  Current version of uploading input image. New features allows user to specify configuration to the detection model, and dangerous obstacle are now evaluated from the walking path.

Component Diagram

Component diagram are used to describe the component that exist inside the system architecture. The components are represent with a rectangle and a title, which are connected with the other components using a line. The relationship has two attributes, the one who provides (describe with circle) and the one who receives (describe with half open circle).

The components were wrapped with the main sections defined in the system architecture. Representation of the component diagram can be seen in Figure 5.3

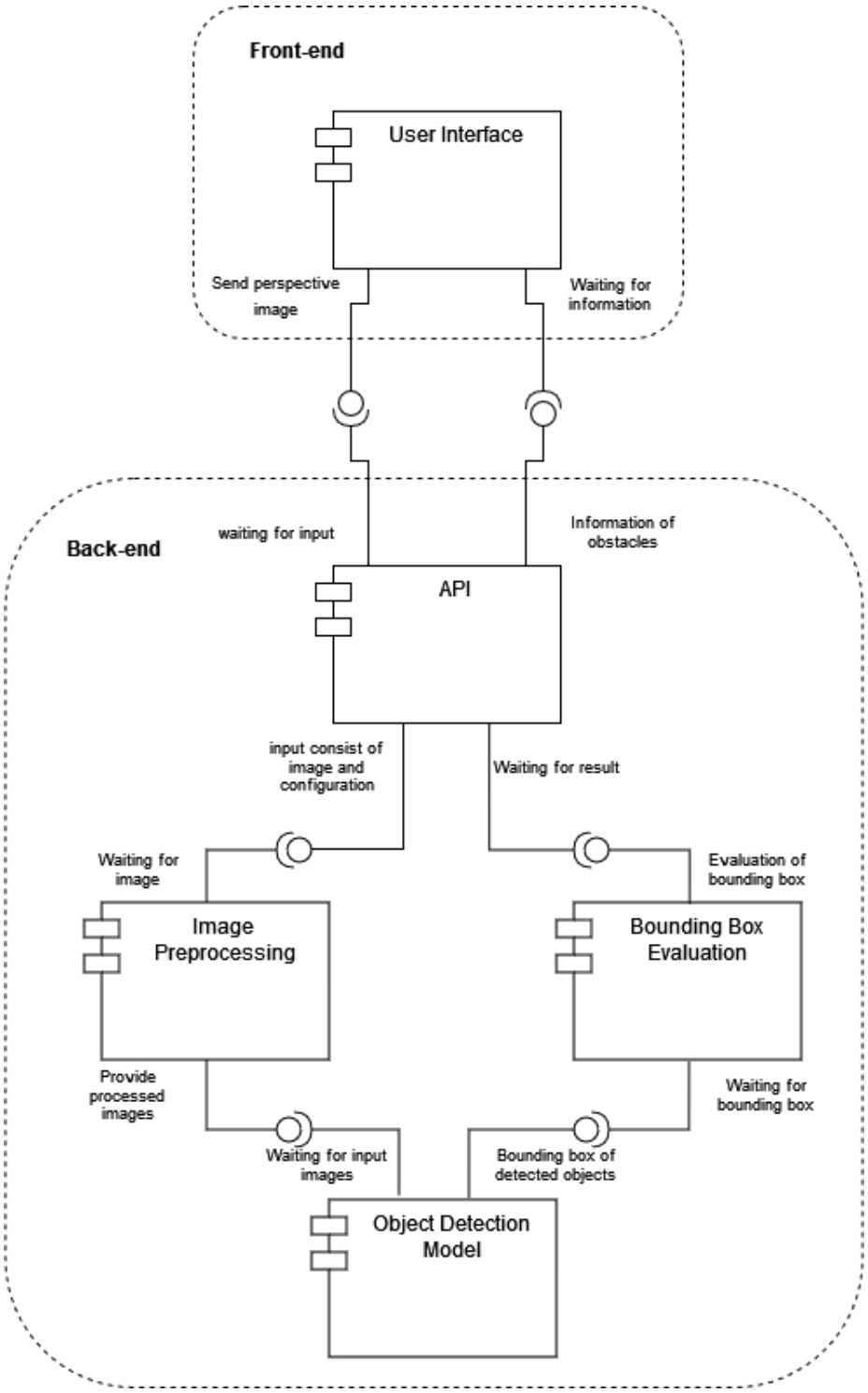


Figure 5.3: Component Diagram

5.2.4 User Interface and User Experience

User interface (UI) will be the main platform for the user to interact with the application. Events such as giving input and processing the output from the system will be provided in the UI. One thing to highlight in designing the UI is accessibility feature to support user with visual ability. Chiti and Leporini (2012) create a research to evaluate the behaviour of blind people while interacting with mobile application, this evaluation will be one of the guide to construct the accessibility feature on the application. As mentioned in the front-end of the system architecture, the UI will be implemented using React Native Expo with JSX, HTML, and CSS.

The prototype design can be seen in Figure 5.4, and Figure 5.6. The implementation of the prototype can be seen in Figure 5.5 and Figure 5.7. To satisfy the accessibility constrain of the app some design choice were made, these choices are:

1. Contrast Colour

   For user to easily differentiate each component in the interface.

2. Light background

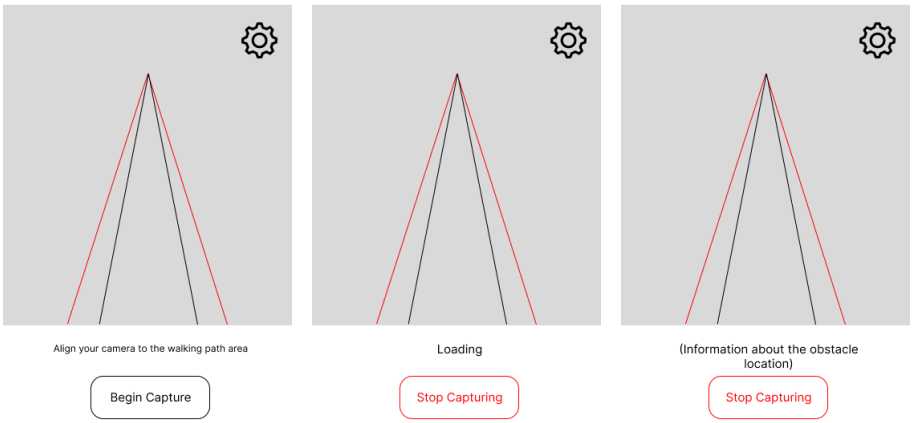
   Because the application will be used mainly outside, a bright background colour will help to clarify the interface.

3. Medium size font

   To assist user that have difficulty to read small size font

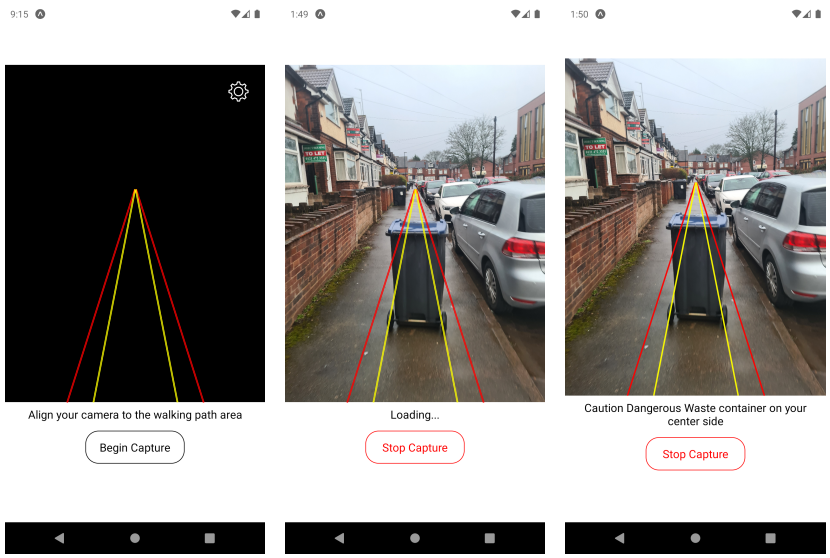
4. Title in every button

   Most operating system in mobile phone allow the user to activate built-in accessibility feature where the system able to read aloud the context of the phone. By giving title to each button the built-in accessibility feature able to read a loud the context of that the user touch.



(a) Initial state (b) Loading state (c) Display state

Figure 5.4: Home page prototype presented with 3 states

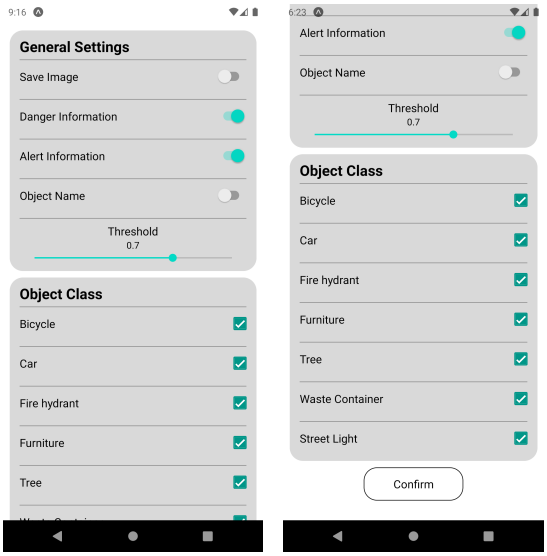


(a) Initial state (b) Loading state (c) Display state

Figure 5.5: Home page implementation with 3 states

(a) (b)

Figure 5.6: Setting page prototype



(a) (b)

Figure 5.7: Setting page implementation

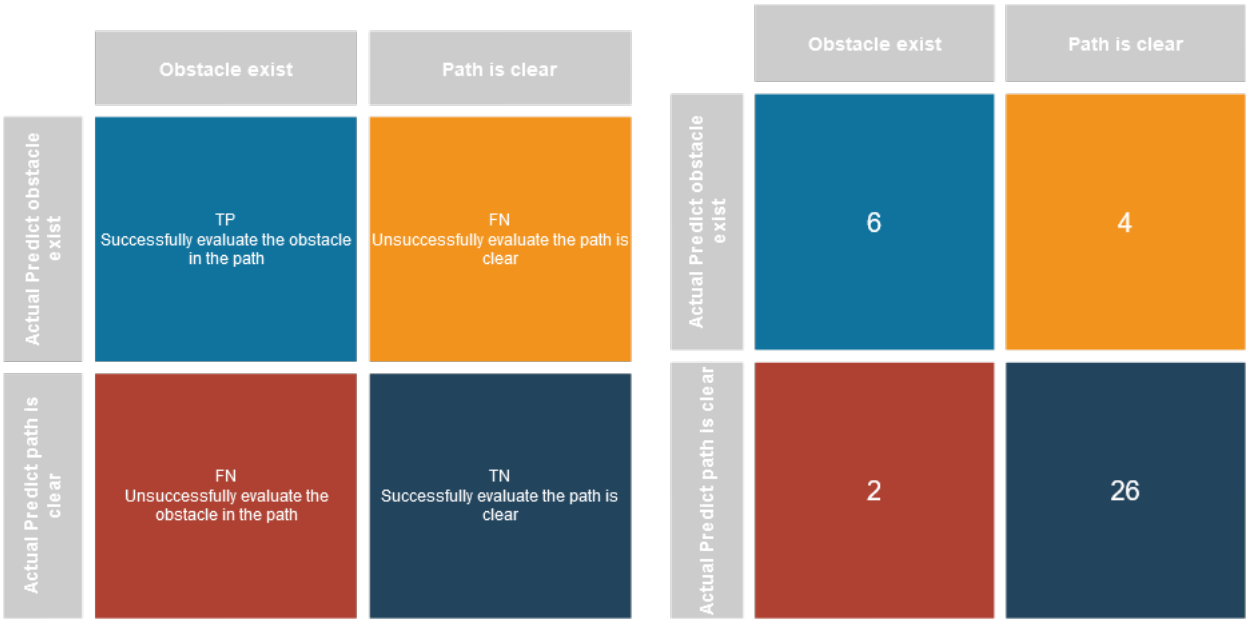


Figure 5.8: Confusion Matrix

5.3 Evaluation and Testing

This section will discuss about evaluation and testing of the application. The purpose of this section is to evaluate how well the application work according to the requirements defined in the previous section. Evaluation process will evaluate how well the application work by measuring the output from the system. Testing process will check the application functionality by testing its features and usability.

5.3.1 Evaluation

Evaluation matrix was used to evaluate the information output from the application. The evaluation were conducted by giving 38 input images taken using camera by walking along the sidewalk and see if the output were correct based on the condition of the image.

Figure 5.8 represent the confusion matrix of the evaluation. Based on the matrix we can calculate the accuracy, precision, recall, and F1 score.

- **Accuracy = 0.84**

  Represents the number of correctly classified data instances over the total number of data instances.

$$Accuracy = \frac{TP}{TP + TN + FP + FN} \tag{5.1}$$

- **Precision = 0.75**

  Measures the accuracy of positive predictions made by the application.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

- **Recall = 0.6**

  Measures the ability of the application to correctly identify all the positive instances.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

- **F1 Score = 0.67**

  The metric which takes into account both precision and recall

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.4}$$

5.3.2 Testing

Two types of testing will be conducted to test the application. First is integration testing and second is usability testing. Integration testing will test the functionality of the application. Usability testing will test the performance of the application by testing it on user.

Integration Testing

Integration testing will test every functionality of the app and will evaluate the output based on the expected output. Each function that are tested should satisfied the requirements defined in the requirement section. The result of integration testing can be seen in table 5.1

User Testing

User testing is a test to find how the application usability satisfy the user. The testing will be conducted twice from 2 different person. The developer will only give a brief description on the purpose of the application, and the tester will be allowed to use the application without getting any guidance from the developer. This approach will allow the developer to understand how well the application able to define itself, and how easy it is for user to navigate trough the UI. The testing process also let the user test the application using the accessibility mode from the phone. The testing was conducted using Samsung Galaxy S20 with limitations of not being able to strap the phone to the tester chest. These are the results of the testing from the feedback summary of the tester:

- **Tester 1** able to understand and navigate through the application without requiring any guidance from the developer. One major concern from them, is that the duration of image processing is too long and can give false information about the obstacle in front of them. Furthermore, the application give different results while capturing the same obstacle, which means the accuracy of the application is still low.
- **Tester 2** has the same problem as Tester 1, where the duration and accuracy decrease the quality of the application significantly. Furthermore, they were having trouble to configure the walking path since there are low information about how to use it. The UI is simple and easy to navigate, but still can have improvement by giving tool-tips for the configuration menu.

Table 5.1: Integration testing result

Description	**Expected**	**Result**
Start and stop the process according to user	The application will loop the process until user stops it	Pass
Detect obstacle	The application able to detect obstacle from the perspective image	Pass
Detect obstacle position	The application able to identify obstacle position to the user	Pass
Detected obstacle violating the danger line	The application gives information about dangerous obstacle	Pass
Detected obstacle violating the potentially dangerous line	The application gives information about potentially dangerous obstacle	Pass
User change the general settings	The application will give result based on the configuration	Pass
User change the object class configuration	The system will give result based on the configuration	Pass
Information delivered to the user	The application will deliver the information using text-to-speech approach	Pass
User able to see the walking path	A walking path is present in the user interface	Pass
User able to navigate through the app using accessibility mode	Each buttons are assigned and configure to be accessible using accessibility mode	Pass

Chapter Six

Conclusion

The project able to finish the application by constructing the object detection model and evaluating the obstacle position. The information of the obstacle can be delivered as the target specified, and communication between user with the application is understandable. However, it doesn't satisfy the performance qualification that is crucial to the usability aspect of the application. The application duration to process input is excessively time-consuming and it is considered as impractical to use. The accuracy of the detection model is still low and can be improved. Lastly, the walking path system is still not adaptable to the user, which affect the correctness of the evaluation.

6.1 Future Works

Based on the conclusion there are works need to be done to improve the quality of the application. Those works are:

- Improving the processing duration

  The system should be able to process in real time by evaluating video input instead of using image. This can be done by constructing the model into the mobile system,

therefore there are no need of communication between client and server.

- Improving the model accuracy

  As discussed in the object detection model section, the model low accuracy is caused by low training dataset, therefore an improvement can be made by training the model with better dataset.

- Construct a Vanishing Point detection for walking path

  To construct a more accurate walking path, a vanishing point detection is important. It enables the system to accurately construct the walking path based on user perspective.

- Adding instruction to the UI

  Based on feedback from the testing, user still have difficulty to use the application because of low information about the application. Improvements can be made by creating a tutorial video or toolkit on how the application works.

References

Ahmed, Farruk and Mohammed Yeasin (2017). "Optimization and evaluation of deep architectures for ambient awareness on a sidewalk". In: *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, pp. 2692–2697.

Andersen, Kristi (2007). *Geometry of an Art*. Springer.

Bochkovskiy, Alexey (n.d.). *Alexeyab/Darknet: Yolov4 / scaled-yolov4 / yolo - neural networks for object detection (windows and linux version of darknet)*. URL: https://github.com/AlexeyAB/darknet.

Carion, Nicolas et al. (2020). "End-to-End Object Detection with Transformers". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, pp. 213–229. ISBN: 978-3-030-58452-8.

Chang, Chin-Kai, Jiaping Zhao, and Laurent Itti (2018). "DeepVP: Deep Learning for Vanishing Point Detection on 1 Million Street View Images". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4496–4503. DOI: 10.1109/ICRA.2018.8460499.

Chiti, Sarah and Barbara Leporini (2012). "Accessibility of Android-Based Mobile Devices: A Prototype to Investigate Interaction with Blind Users". In: *Computers Helping People with Special Needs*. Ed. by Klaus Miesenberger et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 607–614. ISBN: 978-3-642-31534-3.

Duan, Kaiwen et al. (Oct. 2019). "CenterNet: Keypoint Triplets for Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Fryar, Cheryl D, Qiuping Gu, and Cynthia L Ogden (Oct. 2012). "Anthropometric reference data for children and adults; United States, 2007-2010". In.

Jeon et al. (2014). "Estimating the vanishing point of a sidewalk". In: URL: http://hdl.handle.net/11025/26436.

Jiang, Peiyuan et al. (2022). "A Review of Yolo Algorithm Developments". In: *Procedia Computer Science* 199. The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 2021): Developing Global Digital Economy after COVID-19, pp. 1066–1073. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.01.135. URL: https://www.sciencedirect.com/science/article/pii/S1877050922001363.

Keall, M.D., B. Fildes, and S. Newstead (2017). "Real-world evaluation of the effectiveness of reversing camera and parking sensor technologies in preventing backover pedestrian injuries". In: *Accident Analysis Prevention* 99, pp. 39–43. ISSN: 0001-4575. DOI: https://doi.org/10.1016/j.aap.2016.11.007. URL: https://www.sciencedirect.com/science/article/pii/S0001457516303992.

Kong, Hui, Jean-Yves Audibert, and Jean Ponce (2009). "Vanishing point detection for road detection". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 96–103. DOI: 10.1109/CVPR.2009.5206787.

Krasin, Ivan et al. (2017). "OpenImages: A public dataset for large-scale multi-label and multi-class image classification." In: *Dataset available from https://github.com/openimages*.

Lin, Tsung-Yi et al. (2014). "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, pp. 740–755. ISBN: 978-3-319-10602-1.

Mgunyho, original drawing by Niharikamaheshwari (n.d.). *Vectorized version of Vanishing Point*. URL: https://commons.wikimedia.org/w/index.php?curid=81515012.

Park, Kibaek et al. (2020). "SideGuide:A Large-scale Sidewalk Dataset for Guiding Impaired People". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10022–10029. DOI: 10.1109/IROS45743.2020.9340734.

Pundlik, Shrinivas et al. (Sept. 2021). "Home-Use Evaluation of a Wearable Collision Warning Device for Individuals With Severe Vision Impairments: A Randomized Clinical Trial". In: *JAMA Ophthalmology* 139.9, pp. 998–1005. ISSN: 2168-6165. DOI: 10.1001/jamaophthalmol.2021.2624. eprint: https://jamanetwork.com/journals/jamaophthalmology/articlepdf/2782065/jamaophthalmology_pundlik_2021_oi_210041_1631134762.56061.pdf. URL: https://doi.org/10.1001/jamaophthalmol.2021.2624.

Rao, Yutai and Fan Yang (2020). "Research on Path Tracking Algorithm of Autopilot Vehicle Based on Image Processing". In: *International Journal of Pattern Recognition and Artificial Intelligence* 34.05, p. 2054013. DOI: 10.1142/S0218001420540130. eprint: https://doi.org/10.1142/S0218001420540130. URL: https://doi.org/10.1142/S0218001420540130.

Russakovsky, Olga et al. (2015). *ImageNet Large Scale Visual Recognition Challenge*. arXiv: 1409.0575 [cs.CV].

Sun, Chang et al. (2019). "Wide-View Sidewalk Dataset Based Pedestrian Safety Application". In: *IEEE Access* 7, pp. 151399–151408. DOI: 10.1109/ACCESS.2019.2947165.

Tan, Mingxing, Ruoming Pang, and Quoc V. Le (June 2020). "EfficientDet: Scalable and Efficient Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Vittorio, Angelo (2018). *Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset*. https://github.com/EscVM/OIDv4_ToolKit.

Wiggett-Barnard, Cindy and Henry Steel (2008). "The experience of owning a guide dog". In: *Disability and Rehabilitation* 30.14. PMID: 18953747, pp. 1014–1026. DOI: 10.1080/09638280701466517. eprint: https://doi.org/10.1080/09638280701466517. URL: https://doi.org/10.1080/09638280701466517.

Wu, Wenhao (2018). "React Native vs Flutter, Cross-platforms mobile application frameworks". In.

Zammetti, Frank and Frank Zammetti (2018). "React native: a gentle introduction". In: *Practical React Native: Build Two Full Projects and One Full Game using React Native*, pp. 1–32.

Zou, Zhengxia et al. (2023). "Object Detection in 20 Years: A Survey". In: *Proceedings of the IEEE* 111.3, pp. 257–276. DOI: 10.1109/JPROC.2023.3238524.

Websites consulted

- Wikipedia – https://www.wikipedia.org/
- Google Scholar – https://scholar.google.com/
- Matlab – https://www.mathworks.com/
- Towards Data Science – https://towardsdatascience.com/
- Medium – https://medium.com/
- Kaggle – https://www.kaggle.com/
- v7 Labs – https://www.v7labs.com/blog/yolo-object-detection
- Pjreddie – https://pjreddie.com/
- Figma – https://www.figma.com/
- Stackoverflow – https://stackoverflow.com/
- Expo – https://expo.dev/
- FastAPI – https://fastapi.tiangolo.com/

- Matplotlib – https://matplotlib.org/
- OpenCV – https://opencv.org/