University of Sheffield

Generate Walkable Area

Kai-Hsin Chen

Supervisor: Yoshi Gotoh

A report submitted in fulfilment of the requirements
for the degree of MSc in Advanced Computer Science

in the

Department of Computer Science

October 24, 2022

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Kai Hsin Chen

Signature: Kai Hsin Chen

Date: October 24, 2022

Abstract

Nowadays, visually compared people still need to use white canes or guide dogs to walk on street even with the advances in Artificial Intelligence technology, especially in computer vision. It takes lots of money to train a guide dog and sometimes the guide dogs kill its owner accidentally. Thus, to deal with this situation, this report aims to improve the visually impaired people's quality of life by adopting models concerning computer vision.

In the experiment, YOLO model, an object detection model, and IPM in `opencv` are used to achieve the goals. Walkable area is generated by YOLO model and transformed by IPM so that the images can be combined to create a map. All the steps are evaluated and discussed in the paper.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With the development of artificial intelligence (AI), everyday life of human beings becomes more and more convenient. For example, computer vision in AI is used in tumour detection to help doctors to diagnose cancer earlier and increase the survival rate of patients. Object detection in computer vision is used in advanced driver assistance systems (ADAS) to make people drive more easily and more safely. In recent years, more and more people focus on improving life of the disabled by developing AI applications. One of the applications is the Auditory AI Assistants, an AI application to help the deaf. To make the daily life of the visually impaired people better and more convenient, this report aims to create a walkable area map so that visually impaired people can possibly go out without white canes or guide dogs in the future.

1.2 Aims and Objectives

The final aim of this project is to create a walkable area map for virtually impaired people to make their everyday life more convenient. Nowadays, visually impaired people need either guide dogs or white canes when they go out and walk on streets. This may not be convenient and safe enough. Thus, with a walkable area map, some new technologies may appear to help visually impaired people. For example, GPS may be able to plan the routes that are free of obstacles with walkable area map. To achieve the final aim, the project is divided into several steps to accomplish several subaims:

- Primarily, train a human detection model that can predict the bounding boxes of the human beings precisely.
- Secondly, create algorithms that can mark walking people out of the predicted human beings.
- Thirdly, find the position of the feet of the humans that are marked as walking people.

- Finally, transform the output images to bird eye view so that the images can be combined to create a map.

To achieve the subaims above, the following are the objectives in this report:

- Learn about several object detection model and understand the algorithms in the model to detect the objects by reviewing some papers.
- Improve the existed model and make it perform better on the project.
- Changing dimensions of the images is difficult, reviewing some papers about inverse perspective transformation (IPM) is necessary.

1.3 Overview of the Report

This report consists of 5 main sections. The followings give a brief overview of each section in the report.

- Chapter 1: Introduction
  This chapter provides the reason to do the experiment and states the aims clearly.
- Chapter 2: Literature Survey
  This chapter provides synopsis of object detection models and gives an overview of IPM. Moreover, the creation of YOLO models are clearly explained.
- Chapter 3: Methodology
  This section explains the whole process to generate walkable area map, including data transformation, model training, human detection, etc. The algorithms to detect humans as well as the computations to transform the view of the images are also explained in this section.
- Chapter 4: Result
  This section shows the results. The first subsection gives the information of the evaluation metrics and then provides the performance of all the trained models as well as the original pre-trained YOLO v5 model. The second subsection shows the result of the walkable area. The third subsection demonstrates the result of image after IPM. The last subsection discuss the further research of the experiment.
- Chapter 5: Conclusion
  This chapter provides a brief conclusion of this project.

Chapter 2

Literature Survey

To generate the map of walkable area from videos, human detecting and tracking as well as inverse perspective mapping are two important methods and related works will be discussed in this section. Human detection and tracking consist of the parts of detection and tracking. Human detection is to locate all human beings in an image and human tracking is to combine all human detections to generate the paths of the human beings in a video.

2.1 Object Detection

2.1.1 Human Detection

Tracking humans provides tremendous information for many tasks such as accident avoidance and action recognition. Thus, many works related to human tracking are emerging since 1990s. Generally, human tracking can be achieved by two ways, which are tracking and detection. Both methods have advantages and issues. The tracking approach use tracking algorithms to predict the location of human based on the motion of the human. This method is fast but is hard to relocate a person once he or she is occluded, which means the person is once disappeared in the video. The detection approach can deal with this. However, detecting and locating occluded people are challenging and it is hard to distinguish two people in similar appearance and poses.

In 1997, a human tracking system using a multiclass statistical model of colour and shape was created by Wang et al [25]. This model generates a representation of human head and hands in two dimensions and performs well. Two decades later, in 2017, Camplani et al. reviews methods using depth and colour information to perform multiple human tracking, or MHT [3]. With the in depth data, some constraints can be added to human beings. For example, the size and the appearance of humans can be constrained. However, this approach also faces some challenges. For instance, the results of the depth sensors are not accurate enough while the sensors are far from the targets. Moreover, MHT can keep track of walking and standing people but it is hard to observe people who interact with others, bend down, or sit down then stand up.

In 2020, Kumar and Sukavanam proposed a model consist of three convolution neural network, or CNN, models to track human beings [9]. The input data sometimes plays an important role in solving problems. This model deals with a tough situation that the human is occluded partially or entirely in some frames in the video and it is a combination model using both approaches mentioned in the previous paragraph. In this model, the input data is a pair of frames, which are reference frame and current frame. With these two frames, two kinds of regions are created, the same areas and the different areas. Then, The first CNN model classifies the cells in each frame into background, boundary, and human whereas the second model further generates a displacement vector and a probability of moving, non-moving, and occluded in each cell in the reference frame, which is a three dimensional vector. After that, the third CNN model refines the displacement vector and presents the human regions from the first CNN model. Moreover, since this is a supervised learning method, the training data is crucial to direct the desired way of learning in a model. Kumar and Sukavanam create a synthetic data set in order to collect precise ground truth and control the variation in the data set, such as occlusion and displacement. The result of this model outperforms the other models at that time, including CTFWH and BHW, and is 13% better than the second best model.

2.1.2 RCNN models

Object detection models are generally classified into two categories, which are one stage object detection and multi-stage object detection models. This section will briefly explain multi-stage models, including R-CNN, fast R-CNN, and faster R-CNN.

R-CNN Model [6]

R-CNN model is proposed in 2014 by Girshick et al. in [6]. The model contains two stages: generating region proposals (bounding boxes) and classifying the generated region proposals as shown in Figure 2.1. In the first stage, the original R-CNN adopts selective search to generate region proposals. In this stage, the generated region proposals are of different sizes and cannot be fed into CNN model. Thus, the region proposals are also warped into squares with desired size. After the region proposals are created, convolutional neural networks (CNN) are applied to every region proposals to classify each region proposals into different classes including background. Moreover, the region proposals are also resized in the CNN. Finally, the results are predicted with the class probabilities and bounding boxes.

The testing time of R-CNN model is 13 seconds per image on Graphic Processing Unit (GPU) and 53 seconds per image on Central Processing Unit (CPU). R-CNN models are slow since each region proposal needs one forward pass.

Fast R-CNN Model [5]

Fast R-CNN is an improved object detection model comparing to R-CNN. A region of interest (RoI) pooling layer is added to extract a fixed-length vector from the output of

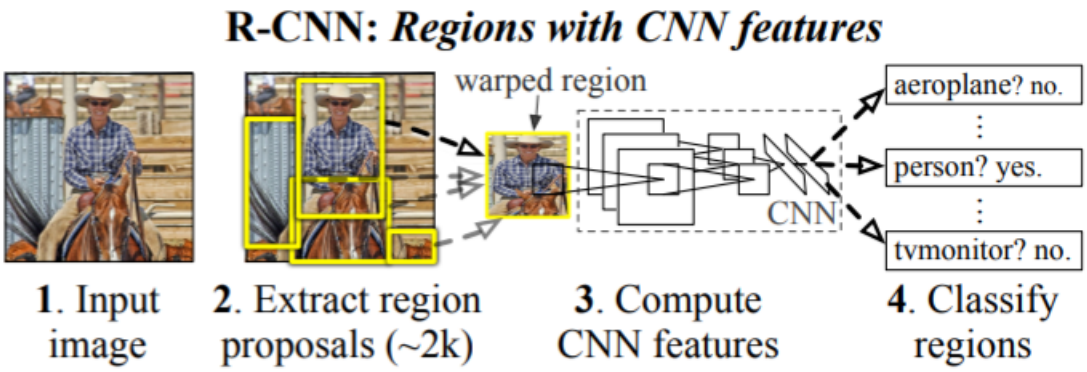

Figure 2.1: Overview of R-CNN pipeline [6]

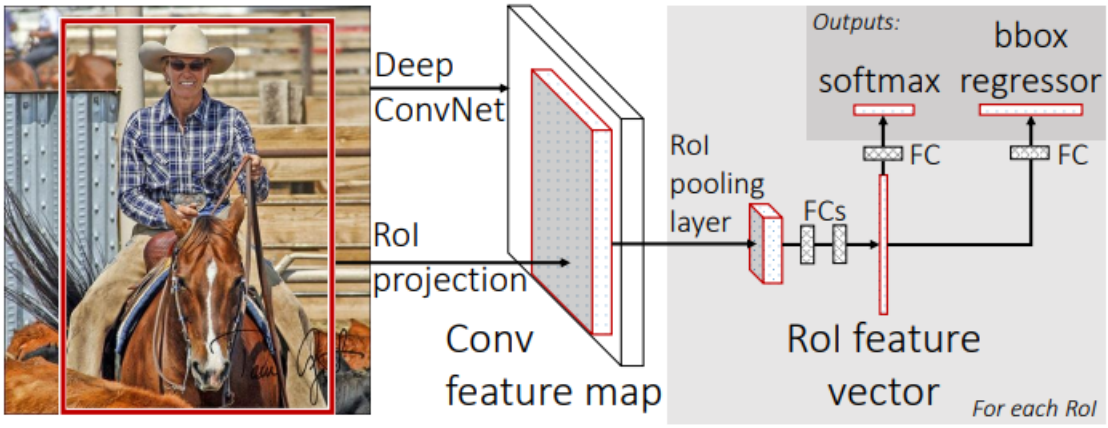

Figure 2.2: Overview of Fast R-CNN pipeline [5]

the CNN layers as shown in Figure 2.2. Instead of processing the region proposals, CNN layers process the image and create convolutional feature map in Fast R-CNN model. In RoI pooling layer, the window is divided into several gird cells and each grid cell is applied with max-pooling. With RoI pooling layer, Fast R-CNN model can simultaneously train numerous RoI samples and identify region of proposals. Thus it is faster than R-CNN model in training. Additionally, the loss function in Fast R-CNN combines classification loss and regression loss together while R-CNN model trains them separately. This reduce the training time as well. Furthermore, Fast R-CNN model choose to use L1 loss instead of L2 loss in regression. These all make Fast R-CNN a faster model comparing to R-CNN.

Faster R-CNN Model [20]

Both R-CNN and Fast R-CNN models choose to use selective search to generate region proposals. However, selective search is slow, resulting in the models being dragged down. Thus, Faster R-CNN uses Region Proposal Networks (RPN) as a solution.

It is similar to Fast R-CNN that the input data of the convolutional networks is an image

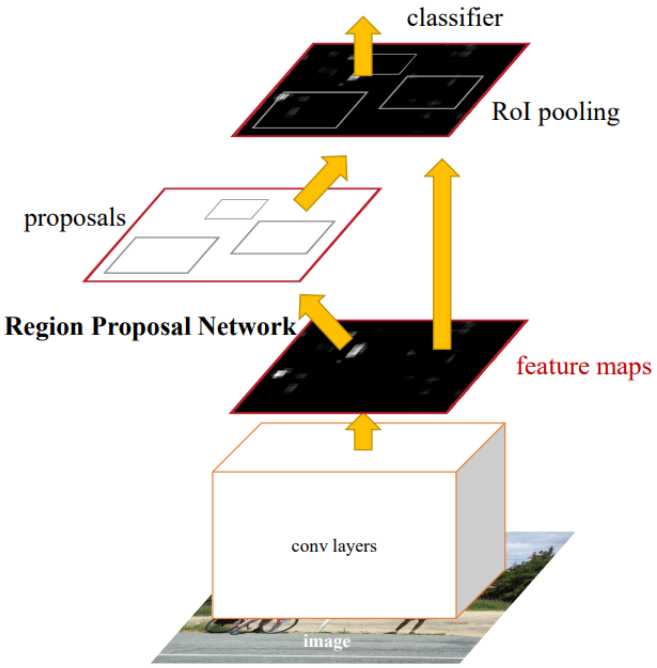

Figure 2.3: Overview of Faster R-CNN network [20]

and the convolutional networks generate a convolutional feature map. The difference between Fast R-CNN and Faster R-CNN is that the former uses selective search on the feature map to create region proposals and the latter create a separate model, RPN, as shown in Figure 2.3 to predict region proposals with the feature map, which is of a size $W \times H$.

RPN model predicts multiple anchor boxes, denoted as k as shown in Figure 2.4. Each anchor boxes consists of 4 coordinates predicted by regression layer and 2 probabilities of an object or not an object predicted by classification layer. All k anchor boxes have different sizes but share the same center. Thus, RPN model predicts $W \times H \times k$ anchor boxes on an image.

2.1.3 YOLO models

You Only Look Once (YOLO) models are first proposed in 2015 by Redmon et al. [17]. Different from the R-CNN models, YOLO models are one stage object detection models. The model predicts the bounding boxes and the class probabilites of each bounding box directly from the input images. This section will briefly discuss YOLO models from v1 to v5 and compare the difference among them.

YOLO v1

YOLO v1 model and Fast YOLO model are created in 2015 by Redmon et al. [17]. The former model consists of 24 convolutional layers and 2 fully connected layers, as shown in

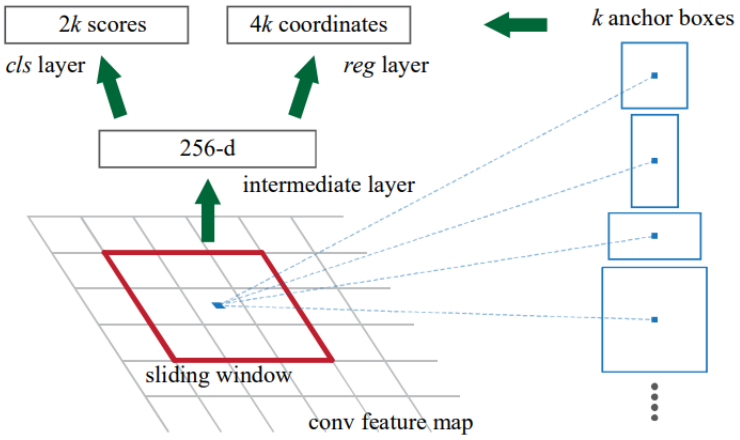

Figure 2.4: Overview of RPN [20]

Figure 2.5, while the latter contains only 9 convolutional layers. The convolutional layers aim to extract features of images while the fully connected layers predict the coordinates of bounding boxes and classification probabilities. Since YOLO models only use one neural networks, the speed of these two models can reach 155 and 45 frames per second respectively and can achieve real time object detection.

On the other hand, the accuracy of YOLO model lags behind the state of the art detection models because each grid cell in YOLO can only have one class.Thus, it is hard to detect objects presenting in groups, e.g. a group of humans. To deal with the problem, Redmon et al. combine Fast R-CNN model with YOLO. The reason they choose Fast R-CNN instead of other object detection models is that the distribution of errors of Fast R-CNN is very different from that of YOLO. Figure 2.6 and Table 2.1 show the types of error and meaning of each type. In Figure 2.6, the background error of Fast R-CNN is much larger than in YOLO and localization error of Fast R-CNN is much lower than in YOLO model. Thus, Redmon et al. use YOLO to reduce background errors in Fast R-CNN model by checking whether the bounding boxes that Fast R-CNN predicts is predicted to be a bounding box in YOLO. If so, this bounding box is less likely to be a background error. The combination of YOLO and Fast R-CNN increases mAP by 3.8% than using Faster R-CNN itself.

Type	Class	IOU
Correct	correct	$IOU \geq .5$
Loc (Localization)	correct	$.1 \leq IOU \leq .5$
Sim (Similar)	similar	$IOU \geq .1$
Other	wrong	$IOU \geq .1$
Background	any	$IOUl \leq 1$

Table 2.1: Error type

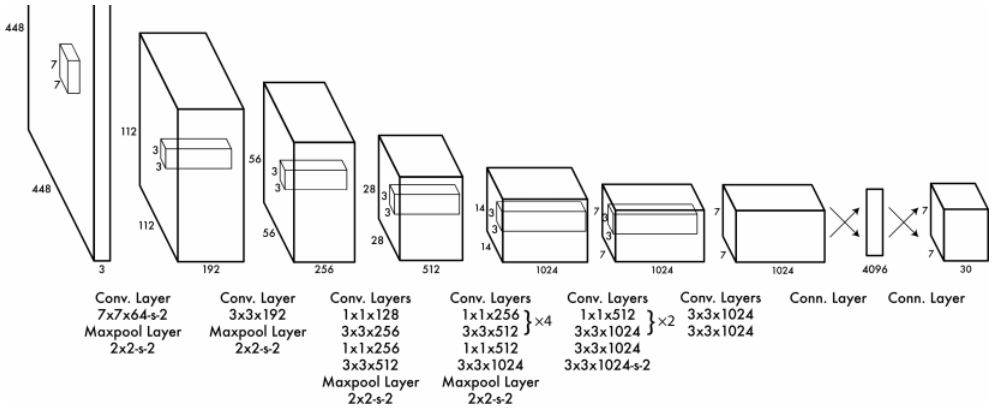

Figure 2.5: Architecture of YOLO model [17]

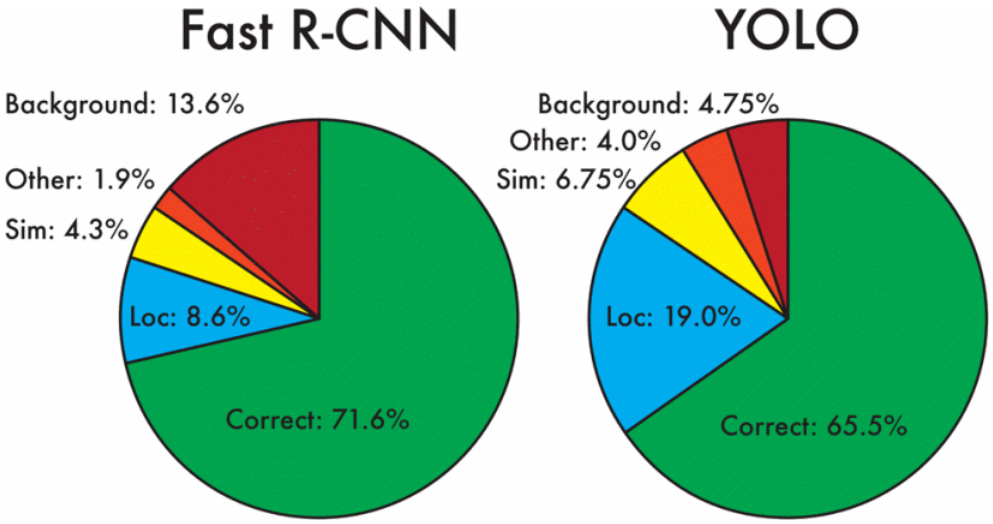

Figure 2.6: Error Analysis [17]

	YOLO								YOLO v2
batchnorm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	78.6

Table 2.2: YOLO to YOLO v2 [18]

YOLO v2

Comparing to state-of-the-art object detection models, YOLO has many shortcomings such as localization errors and low recall. To mitigate these weaknesses, Redmon et al. proposed another model called YOLO v2 in [18]. Batch normalization, high resolution classifier, etc. are added to the new model and will be discussed below.

Batch normalization makes neural networks faster and more stable by normalizing the data batch by batch and using these batches as input to train the model. Moreover, batch normalization can also avoid overfitting. Thus, Redmon et al. remove dropout from YOLO after adding batch normalization and the mAP is improved by 2%.

High resolution classifier is to train the model with higher resolution inputs. Then, the model can work better on high resolution data. In YOLO v2, the classification network is fine tuned on ImageNet using 448×448 resolution for 10 epochs and then the resulting network is fine tuned as well. With high resolution classifier, the mAP of YOLO v2 is increased by 4% comparing with YOLO. Fully connected layers in YOLO are also replaced by convolutional with anchor boxes. With anchor boxes, Yolo v2 predicts over a thousand boxes while YOLO only predicts 98. On one hand, using anchor boxes results in a small decrease in accuracy, which is 0.3%. On the other hand, model with anchor boxes get 7% higher recall comparing with YOLO. This means YOLO has room for improvement. This modified model with anchor boxes predicts objects with 13×13 feature map. While this is adequate for large objects, the model can hardly predict small object. Thus, Redmon et al. use a method called fine-grained features. A passthrough layer is added to the model to bring 26×26 feature map from previous layer. Then this layer stacks neighboring features to form a 13×13 feature map. In such way, the original features and features brought by passthrough layer can be concatenated. Fine-grained features improve the mAP a 1% increase.

To build a model appropriate for images with different sizes, multi-scale training is applied in YOLO v2. The size of input images is fixed in most of the object detection models. Instead, YOLO v2 randomly chooses the image size from multiples of 32 every 10 batches. This allows

YOLO v2 to predict well regardless of the size, or dimension, of the input. The larger the image size is, the slower the model runs and the more accurate the results are. Thus, users can choose the input size that works best with the tasks. Table 2.2 shows mAPs adding different methods to YOLO.

YOLO v3

In YOLO v3 model, Redmon et al. add a few convolutional layers into the feature extractor and generate a 3 dimensional array that includes information of bounding boxes, class predictions, and objectness [19]. Similar to YOLO v2, the feature map from earlier layers is used in YOLO v3 as well. The difference is in YOLO v3, the feature map is upsampled by 2 times and combined with other feature map from previous layers. Thus, more convolutional layers are needed to deal with this larger feature map and YOLO v3 results in a larger model than YOLO v2. As a result, the speed of YOLO v3 is slower but the accuracy is higher. In spite of the size of YOLO v3 model, it is still more efficient than other models with similar accuracy, such as ResNet 101 and ResNet 152.

YOLO v4

Bochkovskiy et al. create a new object detection model in 2020 [2]. Many parts in YOLO v4 are modified in order to get a more accurate and faster model. CSPDarknet 53 is used as backbone, PANet is chosen to be path-aggregation neck, and the head of YOLO v3 is continued to be used in YOLO v4. Bochkovskiy et al. additionally add spatial pyramid pooling (SPP) into YOLO v4. Moreover, lots of methods, such as Cross-GPU Batch Normalization, are modified or not used in YOLO v4 in order to allow others to reproduce the model on common GPU.

SPP is a module based on SPM, or spatial pyramid matching [8]. In order to fit CNN models, max-pooling process is used in SPP while bag of word procedure is used in SPM. SPP module is added into YOLO v4 because it can increase receptive field without any decrease in the speed of the model. Thus, SPP is added into YOLO v4 model.

In spite of SPP, many parts of detector and classifier is modified or added while training, e.g. Mosaic data augmentation, class label smoothing, etc. Table 2.3 and Table 2.4 show accuracies and APs after implementing different methods. The different features in Table 2.4 are:

- S: Eliminate Grid Sensibility
- M: Mosaic Data Augmentation
- IT: IoU Threshold
- GA: Genetic Algorithms
- LS: Class Label Smoothing

- CBN: Cross Mini-batch Normalization
- CA: Cosine Annealing Scheduler
- DM: Dynamic Mini-batch size
- OA: Optimized Anchors

With these experiments and the results using different backbones and pre-trained weights, Bochkovskiy et al. create YOLO v4 with the best results. Moreover, the results are similar no matter what the size of mini-batch is. This means that this model can work on common graphic processors and an excellent detector can be obtained.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.9%	94.0%
✓							77.2%	94.0%
	✓						78.0%	94.3%
		✓					78.1%	94.5%
			✓				77.5%	93.8%
				✓			78.1%	94.4%
					✓		64.5%	86.0%
						✓	78.9%	94.5%
	✓	✓		✓			78.5%	94.8%
	✓	✓		✓		✓	79.8%	95.2%

Table 2.3: classifier accuracy on CSPResNeXt-50 [2]

2.2 Inverse Perspective Mapping

Inverse Perspective Mapping (IPM) is a method to transform a two dimensional image into three dimensional space and then map it into two dimensional image again from a bird eye view [13]. The objects in images and videos are 2 dimensional while they are in fact stereoscopic. For example, two parallel straights lines in 3 dimensional space may intersect while putting them into two dimensional space. From Figure 2.7, the three lane markers are not parallel and would intersect outside of the image. However, these three white lines are parallel in 3 dimensional space. This causes some problems to AI in computer vision field. For instance, in road detection, perspective view causes inefficiency and the road detection models are then too slow for real-time detection. Thus, Oliveira et al. apply IPM to the input data to transform images from perspective view to images from bird eye view [15]. However, there are still constraints on this method. The application and constraints will be discussed in this section.

In [1], Bertozzi et al. state that each pixel contributes different amount of information in an image because the distance between objects and the camera varies. Thus, when processing images, perspective effect should be considered. However, this is difficult for SIMD machines.

S	M	IT	GA	LS	CBN	CA	DM	OA	loss	AP	AP_{50}	AP_{75}
									MSE	38.0%	60.0%	40.8%
✓									MSE	37.7%	59.9%	40.5%
	✓								MSE	39.1%	61.8%	42.0%
		✓							MSE	36.9%	59.7%	39.4%
			✓						MSE	38.9%	61.7%	41.9%
				✓					MSE	33.0%	55.4%	35.4%
					✓				MSE	38.4%	60.7%	41.3%
						✓			MSE	38.7%	60.7%	41.9%
							✓		MSE	35.3%	57.2%	38.0%
✓									GIoU	39.4%	59.4%	42.5%
✓									DIoU	39.1%	58.8%	42.1%
✓									CIoU	39.6%	59.2%	42.6%
✓	✓	✓	✓						CIoU	41.5%	64.0%	44.8%
	✓		✓					✓	CIoU	36.1%	56.5%	38.4%
✓	✓	✓	✓					✓	MSE	40.3%	64.0%	43.1%
✓	✓	✓	✓					✓	GIoU	42.4%	64.4%	45.9%
✓	✓	✓	✓					✓	CIoU	42.4%	64.4%	45.9%

Table 2.4: Detector training accuracy on CSPResNeXt50-PANet-SPP, 512x512 [2]

As a result, IPM is applied to the input data to remove the perspective effects by transforming the images to bird eye view. Even though this is a good method, Bertozzi et al. still point out a constraint in IPM, which is the surface in the image should be flat. Oliveira et al. further state that it should be an obstacle free scene and that the transformation from the camera to the scene should preserve the distance between objects, which means that the position of the camera should remain still [15]. If there is an obstacle, the results after IMP will be like the book in Figure 2.8.

In spite of these constraints, IPM is still applied to lots of tasks, such as lane mark detection and Advanced Driver Assistance System(ADAS). In [10], Lin and Lien use IPM to preprocess the input data. Then, used the data to perform lane detection and road marker detection. With IPM, the road marker would be transformed into the actual scale and thus reduce the difficulty of identifying the road markers. In [16], Prakash et al. use one of the constraints of IPM combined with some other methods such as Selective Edge Filtering(SEF) to create obstacle detection model. The constraint is that there should be no obstacles in the scene to perform a perspective transformation without distortions. Prakash et al. use several consecutive frames to find the pattern of the obstacles based on distortions and perform obstacle detection.

2.3 Similar Works

This section will discuss some similar works that aim to improve the life of the visually impaired people and review some papers that aim to conduct walking human tracking.

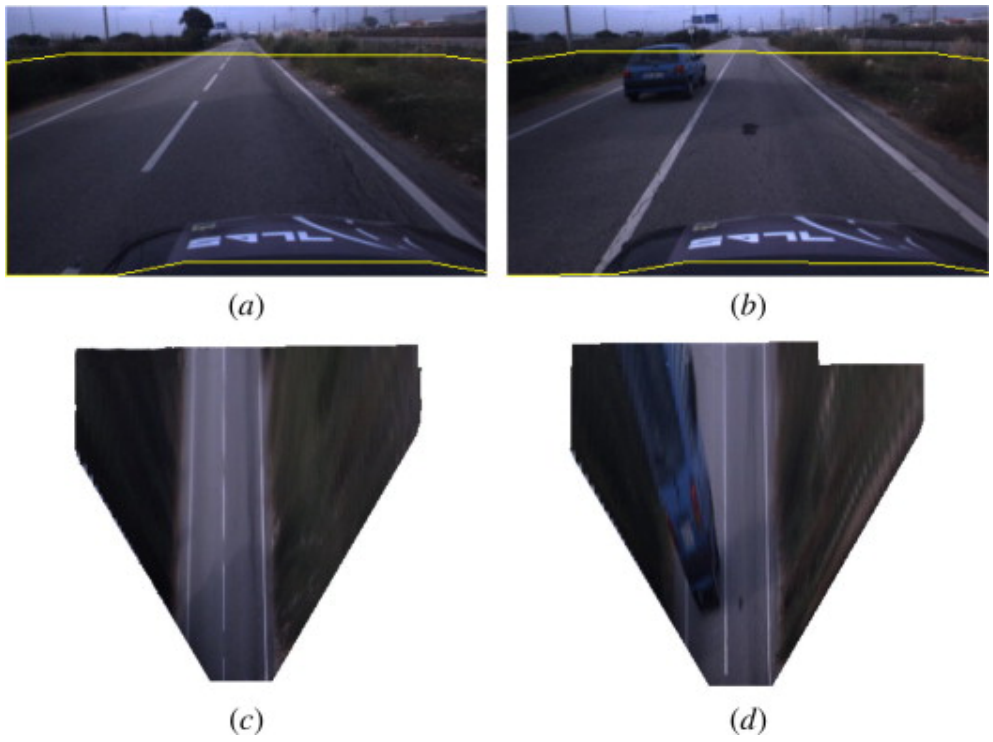

Figure 2.7: Parallel lines in images [15]

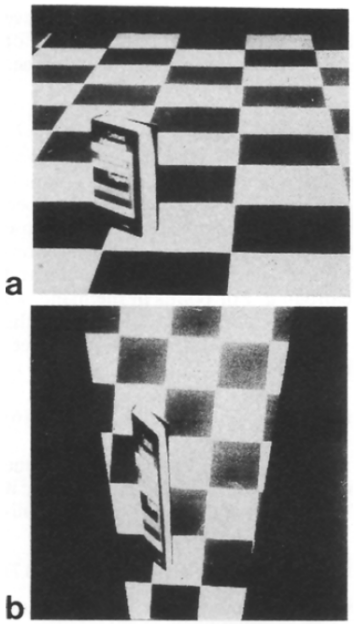

Figure 2.8: (*a*) shows a book in original image and (*b*) shows the book in the image after inverse perspective mapping [13]

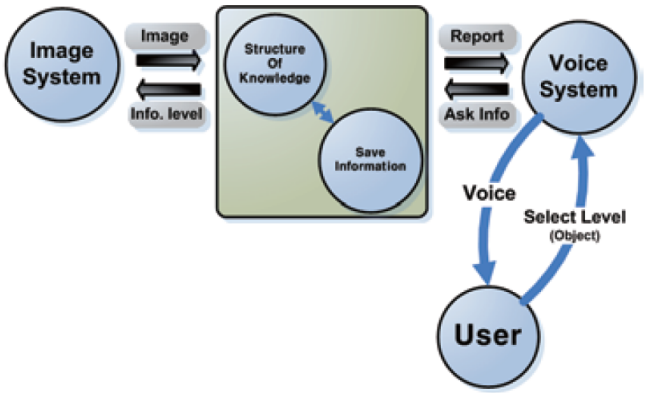

Figure 2.9: Overview of the multimodel in [26]

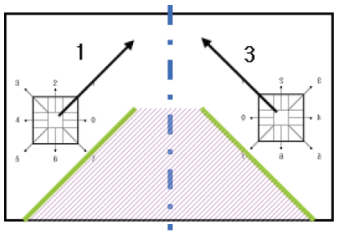

Figure 2.10: Line extraction based on 8 chain-code

2.3.1 Walking Assistance System

Yu et al. proposed a walking assistance system based on a multimodal information transformation technique in 2009 and the overview of the model is shown in Figure 2.9 [26]. The model processes images that are obtained by a charge-coupled device (CCD) camera. First, the model generates the walking area using line extraction. Secondly, the model detects obstacles in the detected walking area by Gabor-Filter. Thirdly, the location of the detected obstacles are calculated based on the position of vanishing points of the obstacles. Last, the information gained from the previous steps is transformed to voice information and the visually impaired people can know which direction they are facing and where the obstacles are.

In the first step, an input image is divided into left and right halves. Then, 8 direction chain-code is used to find the line in the images. To detect the walking area, the lines that have direction 1 in the left half and that have direction 3 in the right half will be extracted as shown in Figure 2.10. The boundary of walking area can be extracted using these lines.

In the second step, Gabor-Filter is used to detect obstacles that are vertical to the ground and is represented in Equation 2.1 and Equation 2.2 where k_x and k_y indicate spatial frequency and σ is the scale of Gabor-Filter formula.

$$G(x,y) = cos(k_x x + k_y y) exp(\frac{-(x^2+y^2)}{2\sigma^2}) \tag{2.1}$$

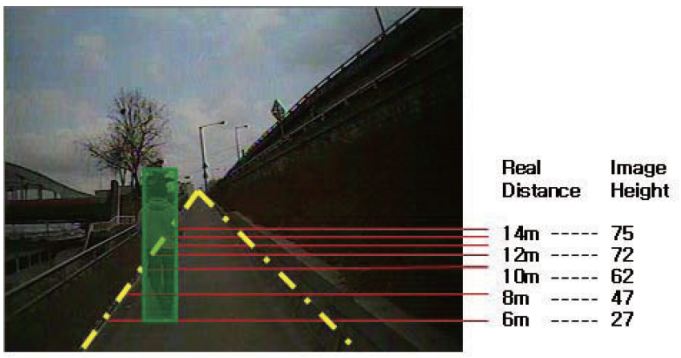

Figure 2.11: Relationship between Real Distance and Image Height

$$G(x,y) = sin(k_x x + k_y y)exp(\frac{-(x^2+y^2)}{2\sigma^2}) \tag{2.2}$$

In step 3, the location of the vanishing points is used to decide the sight direction and to calculate the obstacle location. In Figure 2.11, the vanishing point is in the middle of the image and the relationship between real distance and image height is calculated.

In the last step, Yu et al. use a multimodal transformation to transform the the acquired image information to voice information. They created some sentences with blanks to be filled out by the information as shown in Table 2.5 where *blank* represents the blanks.

Table 2.5: Template Sentences

Types	Sentences
Obstacle	There is a obstacle *blank* o'clock direction There is a obstacle far from here about *blank* m
User	You are walking on the *blank* side on the sidewalk You are seeing the *blank* of the sidewalk area

With this walking assistance system, visually impaired people can use their ears instead of their eyes to figure out where to walk on the streets. This report aims to do achieve a similar task but with different approach and to improve the everyday life of visually impaired people.

2.3.2 Moving Human Detection

In 2019, Wen et al. proposed a model to detect moving humans in [24]. This model does not use any human detection models. Instead, it is based on a classification algorithm of second re-projection error. The input of this model is provided by a visual sensor and with these images, the transformation matrices between frame and frame are calculated to detect

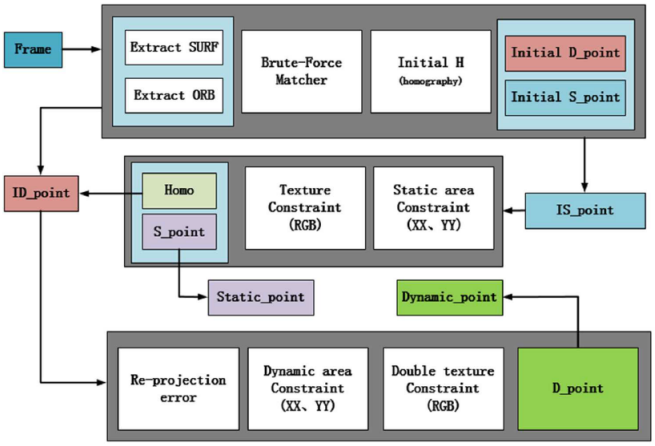

Figure 2.12: Structure of the moving human detection model [24]

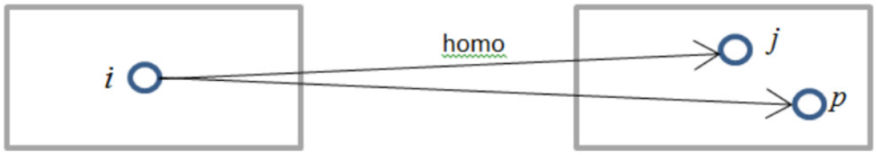

Figure 2.13: Projection point and corresponding point

the dynamic feature points and static feature points. The model can be divided into three parts as shown in Figure 2.12 and will be briefly discussed in the following paragraphs.

The first part is initialisation. The features of the images are extracted in this part. Two feature extraction algorithms can be chosen. One is Oriented FAST and rotated BRIEF (ORB), which works fast and can reach real-time requirement, the other is Speeded Up Robust Features (SURF), which extracts many feature points but work slower than ORB. After feature extraction, the corresponding feature points between two consecutive frames are matched by Brute Force method. With the matching result, homography matrix can be calculated by Equation 2.3, where P_1 and P_2 stand for the projection points and corresponding points respectively, using random sample consensus (RANSAC) since the matching results contains some mismatched points.

$$P_2 = H \cdot P_1 \tag{2.3}$$

Then, the distance between the projected feature points on $(n-1)^{th}$ frame and the corresponding feature points on n^{th} frame is calculated as shown in Figure 2.13, where j is the projection point of i and p is the corresponding point of i. If the distance is larger than the Mahalanobis distance between the projection points and the corresponding points, the point may be either a dynamic point or an uncertain point. If the distance is smaller than the Mahalanobis distance, it may be a static point. To conclude, the first part divides the feature points into the static group and the dynamic or uncertain group.

Before moving on to dynamic and static point detection, two kinds of constraints, texture constraint and area constraint, are discussed first in this paragraph in order to perform walking human detection. Texture constraint is to calculate the difference of an area between two adjacent frames. The original formula is shown in Equation 2.4 where $I_k(x, y)$ and $I_{k-1}(x, y)$ are the corresponding points and the projection points respectively. If the interframe difference of a point is larger than the threshold T, it is a dynamic point.

$$d_{k-1,k}(x,y) = \begin{cases} 1, \text{if } |I_k(x,y) - I_{k-1}(x,y)| > T \\ 0, \quad otherwise \end{cases} \tag{2.4}$$

However, because of the noise in the background such as illumination changes, many static feature points will be marked as dynamic. Thus, Wen et al. further proposed an formula using three channels (RGB) to perform texture constraint as Equations 2.5 and 2.6, where α, β, and γ are the weights of the three channels respectively.

$$d_n(x,y) = \alpha(|R_n(x,y) - R_{n-1}(x,y)|) + \beta(|G_n(x,y) - G_{n-1}(x,y)|) + \gamma(|B_n(x,y) - B_{n-1}(x,y)|) \tag{2.5}$$

$$\alpha + \beta + \gamma = 1 \tag{2.6}$$

Area constraint do the similar thing as texture constraint. Instead of points, area constraint uses areas to reduce mismatches. The equation is shown in 2.7

$$f(P_n, P_{n-1}) = \begin{cases} \text{Correct correspondence, if} |u_n - u_{n-1}| < \theta U_{max} \text{ and } |v_n - v_{n-1}| < \eta V_{max} \\ \text{Error correspondence, otherwist} \end{cases} \tag{2.7}$$

where $P_n(u_n, v_n)$ and $P_{n-1}(u_{n-1}, v_{n-1})$ stand for the corresponding points between two consecutive frames while U_{max} and V_{max} are maximum difference and θ and η are coefficients. With these two constraints, static points can be detect more accurately and mismatch points can be reduced.

The second part is to detect static points. With the two constraints above, lots of mismatch points can be reduced. Thus, the results of static points after initialization and the two constraints are accurate enough. After the static points are predicted, these static points are used to generate homography matrix again in order to predict dynamic points.

The last part is to detect dynamic point. One of the characteristics of the dynamic points is that the projection point and the corresponding point would have a large difference because the texture of the background and the texture of the object are different. To obtain the dynamic points, the second texture constraint is designed as Equation 2.8 and Equation 2.9 using re-projection error.

$$dd_n = \alpha(|i(x,y) - j(x,y)|) + \beta(|i(x,y) - j(x,y)|) + \gamma(|i(x,y) - j(x,y)|) \tag{2.8}$$

$$f(x,y) = \begin{cases} \text{Dynamic point, if } d_n < T \text{ and } dd_n > T \\ \text{Error correspondence, otherwise} \end{cases} \tag{2.9}$$

This classification model to detect walking human works well indoor but is not evaluated outdoors.

Chapter 3

Methodology

To conduct the experiments, some technologies and some programs are required and are listed below. First, yolo v5 model is required to train a custom human detection model to detect humans in the videos. Second, labelImg is used to annotate the selected images in order to evaluate model performance [22]. Inverse Perspective Mapping(IPM) function in `Opencv` is used to transform the images to the ones of bird eye view.

3.1 Data Selection

Training data is a critical factor in building models and should be selected properly to make the model fit the tasks. Sometimes, changing training data can even improve the shortcomings of a model. For example, in [9], Kumar and Sukavanam use images from different datasets such as UCF-101 and Hollywood2 to create synthetic dataset. The synthetic dataset includes people with different poses, images under various illumination, and even some people are occluded in the images. This helps create an astonishing human detection model. As a result, dataset CrowdHuman [21] is selected as the training data in order to build a model that can predict humans whether they are occluded or not.

CrowdHuman is a dataset created by Shao et al. in order to evaluate human detect models' performance in human in crowds [21]. CrowdHuman database contains about 25,000 images of humans in crowds, including 15,000 for training, 4,370 for validating, and 5,000 for testing. The images are well annotated. Human visible region bounding box, head bounding box, and human full body bounding box are all marked. Moreover, whether a person is occluded or not is also annotated. The average amount of humans in an image is 23 per image and there are about 470 thousands people in total.

Among the three kinds of bounding box, head bounding box and human full body bounding box are chosen to train the model because of the reasons below. First, the model should be able to detect the humans that are occluded so the human full body bounding boxes are chosen. Second, humans with only heads visible should be detected as well. The position of human feet is generally six to seven times of the length of head below human heads. Thus, with the head bounding box, the position of human feet can be predicted. The

human visible region bounding box is useless because people in close circuit television(cctv) are usually occluded. Thus, if a human detection model is unable to predict positions of human once occluded, it is useless in most cases.

30 videos from Videvo.net are chosen to be the test set. This is because many videos from Videvo.net have the same features as cctv. First, the position of the camera should remain still all the time. Second, some of the videos are taken in the day while some of them are taken at night. Moreover, the chosen videos contain a wide variety of the density of humans in order to figure out the performance of the model under different situations. The length of the videos varies from 5 seconds to 24 seconds. The resolution of the videos are also different to see how the model performs under blurred videos and clear videos. Last, the distance between the camera and the targets differs as well. Thus, if the trained model works well on all of the videos, the model can be applied to most of the other videos.

3.2 Data Transformation

Data transformation is performed in order to change the original data into an appropriate style to fit yolo v5 model.

The initial data is in one odgt file. Each line in the file is a JSON format and includes the whole annotations in an image. The annotations include three types of bounding boxes and occlusion etc. The position of each bounding box is denoted by four numbers, which are the x and y coordinates of the left top point of the bounding box, the width of the bounding box, and the height of the bounding box.

The input format of YOLO v5 is that one image in 'images' folder corresponds to one txt file in 'labels' folder. Each txt file includes all bounding boxes in the corresponding image. The position of each bounding box is marked by 4 numbers as well, which are the x coordinate and y coordinate of the center of the bounding boxes, and the normalized widths and heights. The computation of these numbers are shown in Equation 3.1, Equation 3.2, Equation 3.3, and Equation 3.4. The subscripts 'old' indicates values in CrowdHuman dataset while subscripts 'new' indicates the transformed formats.

$$x_{new} = \frac{x_{old} + width_{old}}{2 * width_{image}} \tag{3.1}$$

$$y_{new} = \frac{y_{old} + height_{old}}{2 * height_{image}} \tag{3.2}$$

$$width_{new} = \frac{width_{old}}{width_{image}} \tag{3.3}$$

$$height_{new} = \frac{height_{old}}{height_{image}} \tag{3.4}$$

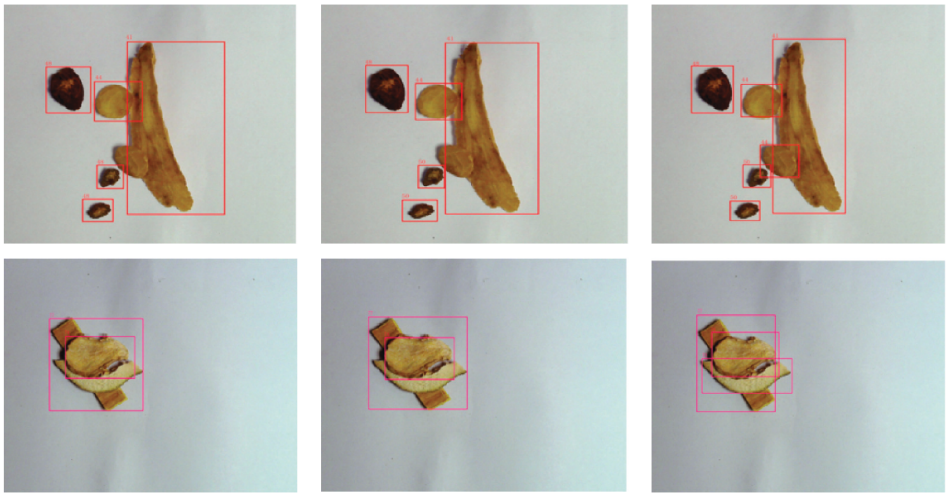

Figure 3.1: Object detection results of Faster RCNN, SSD, and YOLO v5. (a) object detection result of Faster RCNN (b) object detection result of SSD (c) object detection result of YOLO v5 [11]

3.3 Model Selection

There are lots of object detection models created for different purposes, such as convolutional neural network(CNN), Region-based convolutional neural network(R-CNN), YOLO, etc. Each model has its own advantages and shortcomings. For example, Faster R-CNN model has a high precision on object detection but is not efficient enough to perform real time detection because the testing time is about 0.22 seconds per image [12]. As a result, Faster R-CNN model cannot be used in advanced driver assistance system(ADAS) since the testing time is too long. Thus, choosing a proper model that fits the task is important.

YOLO v5 model is selected to generate the walking area because of the reasons below. First of all, YOLO looks at the whole image instead of a small region. This means the background can also contribute to the detection. It may help to detect humans that some parts of their body are occluded in the images. Second, the testing time for YOLO is short. The speed can amazingly reach 111 frames per second(FPS) on graphic processing unit(GPU) [23]. The amount of the training images and the validating images are large, which is 15,000 and 4,370 images respectively, and the model is used for detecting humans in videos, which means there are lots of frames to be predicted, so an efficient model is preferred. Last but not least, YOLO v5 detect a objects in a crowd better than other models such as Faster RCNN and SSD [11]. As shown in Figure 3.1, Faster RCNN and SSD miss to detect one of the objects while YOLO v5 detect all the targets. In the videos from cctv, lots of humans are occluded or in crowds. For example, the video in Time Square or Tokyo train station do contain lots of people and the boundary between the people are not clear. Thus, using YOLO v5 is more appropriate than any other object detection models..

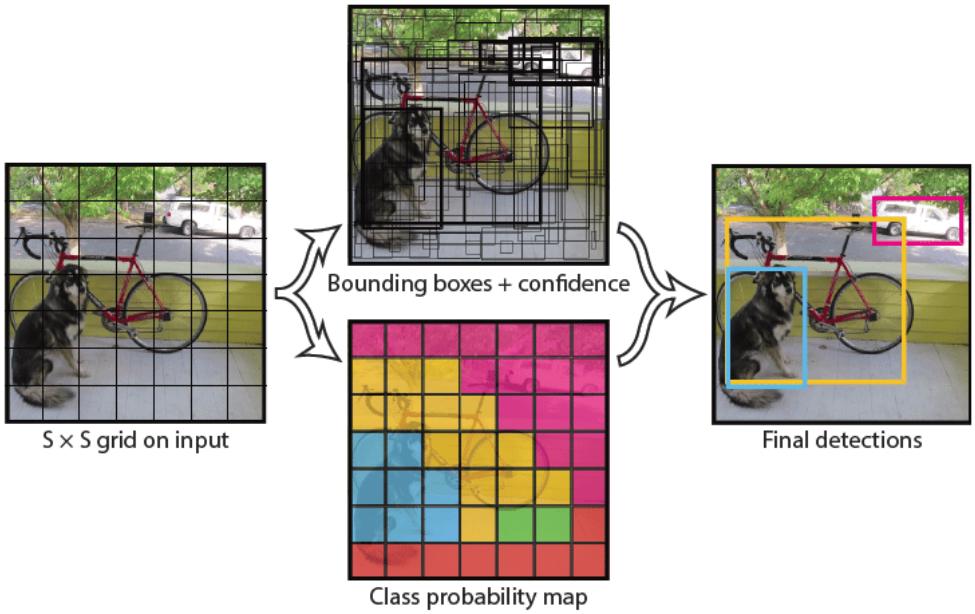

Figure 3.2: How YOLO v5 works

On the other hand, the Faster R-CNN models have better performance on detecting small targets than YOLO because the faster R-CNN models have 9 anchor boxes in 1 grid while YOLO only generate 2 anchor boxes in a grid [7]. However, small object detection is not the main purpose in generating walkable area since the humans with small bounding boxes are the one that are far from the camera. The IPM transformation has its own distance limit constraint. Thus, the humans that are too far from the cctv are unnecessary.

3.4 YOLO v5 Platform

YOLO v5 is launched in June 2020 by Ultralytics. YOLO v5 is a one stage object detection model that can detect up to 80 kinds of objects.

YOLO first divides an input image into N grids and the dimension of each grid is $S \times S$ as shown in Figure 3.2. Each grid is in charge of predicting and localizing the objects within the grid. To carry out this task, each grid predicts B bounding boxes whose centers are in the grid and predicts the class of the objects as well as the prediction score (probability) that the object is in the grid.

However, this generates some problems. First, it is hard to detect objects in a group since each grid is only responsible for one object. Second, it generates multiple bounding boxes for one object since different grid may predict the same object with different bounding boxes. In fact, the first problem does not affect the experiment a lot because the experiment focuses on predicting the position of the feet. If the people are in groups, their feet are highly possible to be in close positions. As for the second problem, YOLO v5 use the Non Maximal

Suppression (NMS) to solve it. In NMS algorithm, YOLO takes the bounding boxes with the largest prediction score of each decision first, and then suppress the bounding boxes that have the largest IoU with the selected bounding boxes. NMS is repeated till the end.

3.5 Model Training(to be finised)

The original YOLO v5 model can hardly detect humans than only appear in few pixels and people whose body is occluded. It may be because YOLO v5 model is built to detect up to 80 kinds of objects, including 'person', 'bird', 'bicycle' etc. In this essay, the only target is human being. Thus, to improve the performance on human detection and generate the walkable area, a new model based on YOLO v5 is trained using CrowdHuman dataset. The experimental models are trained on High Performance Center(HPC) using 16 central processing units(CPUs). Belows are a list of the hyperparameters and a list of configurations and the explanation of them.

Configurations:

- Img: Img is the size of square images. The longer side of the image is resized to 640 and the shorter side of the image is padded to 640 with grey color while remaining the same ratio between both sides.
- Cfg: Cfg is the architecture of the model. The 4 available choices in the descending order by complexity are yolov5x.yaml, yolov5l.yaml, yolov5m.yaml, yolov5s.yaml. In spite of these 4 built models, a yaml file can also be defined to fit the purposes.
- Batch: The batch size, which is defined as the number of samples used to update weights for one time.
- Epochs: The number of epochs, which is defined as the number of times of complete pass of the whole training set.
- Data: A yaml file containing the information of the training set and validating set. For example, the path of the data, the number of the class, and the name of the classes.
- weights: This is the pre-trained weights. The model will be further trained based on this weights.

Hyperparameters:

- Warm up epochs: The number of epochs that use warm up learning rate to update weights in the beginning. The warm up learning rate is always larger than the learning rate to update weights faster in the beginning.
- Learning rate: The learning rate is the scale to update the weights in one iteration.
- Batch: the amount of training samples in one iteration.

- epoch: The amount of times that the whole training set is used in training step.

Some model are trained using different hyperparameters in order to choose the best model as shown in Table 3.1.

epochs	10	20	30
warmup epochs	3	4	5
learning rate	0.01	0.005	

Table 3.1: Table of values of each hyperparameter.

3.6 Human Detection

This paragraph will discuss the way the chosen model works, including how the model detects objects and how the model distinguishes the humans from the background.

The model is based on regression and it can predict the location as well as the class of several objects simultaneously. Leaky ReLU activation function is used in the hidden layers while the sigmoid activation function is used in the final layer as shown in Equation 3.5 and Equation 3.6.

$$Leaky\ ReLU = max(0.1 * x, x) \tag{3.5}$$

$$Sigmoid : \sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.6}$$

Moreover, to make the model learn the four numbers, which are the x and y coordinates of the center of the bounding boxes and the widths and the heights of the bounding boxes, and the class simultaneously, the loss function of the model is designed as Equation 3.7[4].

$$\begin{aligned} Box\ Loss = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} M_{ij}^{object} [(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2] \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} M_{ij}^{object} [(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2] \\ + \sum_{i=0}^{s^2} \sum_{j=0}^{B} M_{ij}^{object} (C_i - \hat{C_i})^2 \\ + \lambda_{noobject} \sum_{i=0}^{s^2} \sum_{j=0}^{B} M_{ij}^{noobject} (C_i - \hat{C_i})^2 \\ + \sum_{i=0}^{S^2} M_i^{object} \sum_{c \in nClasses} (p_i(c) - \hat{p_i}(c))^2 \end{aligned} \tag{3.7}$$

In the formula, S is the number of the grid cells in the image, B is the bounding boxes that a grid cell predicts, and c is the class that a grid cell predicts. i means the i^{th} grid cell while j represents the j^{th} bounding box in the grid cell. x and y indicate x and y coordinates of the center of the anchor boxes while w and h are the width and the height of the anchor boxes. C gives the confidence score of each anchor box and $p_i(c)$ gives the confidence probability. M_{ij}^{object} is 1 if the cell contains an object and 0 otherwise while M_{ij}^{object} is 1 if the cell contains no object and 0 otherwise. Last, λ_{coord} and $\lambda_{noobject}$ represent the penalties of localization errors and recognition errors during training.

The loss function can be divided into 5 parts as 5 lines in Equation 3.7. The first line shows the penalty of incorrect prediction of the position of the center of the bounding boxes. The second line is the penalty of imprecise width and height of the predicted bounding boxes. The square root in the second term means that the small bounding boxes are penalized more than the large bounding boxes with the same amount of error. For example, if the size of a bounding box is 100×100 but predicted as 121×121, the loss is 2. However, if the size of a bounding box is 64×64 and predicted as 85×85, the loss is 2.44. These two terms can be summed as the penalty of localization error. The third and the fourth terms are like cross entropy loss function because if there is an object in the cell, the fourth term would be 0 while if there is no object in the cell, the third term would be zero. Moreover, the $\hat{C}_i$ in the equation is the Intersection over Union(IoU) of the predicted bounding boxes with the ground truth and C_i is the confidence score of the anchor boxes. The way to compute IoU is shown in Equation 3.8

$$IoU = \frac{B_{predict} \bigcap B_{true}}{B_{predict} \bigcup B_{true}} \tag{3.8}$$

where $B_{predict}$ is the predicted bounding box and B_{true} is the ground truth bounding box. These two terms aim to reduce the difference between confidence score and IoU. Last, the fifth term in the equation shows the classification loss.

After training, the model can detect humans in the image. The way the model chooses the bounding boxes for each human being is to compute all the confidence score of the anchor boxes and plot the bounding boxes whose confidence score is larger than the threshold.

3.7 Walking Human Detection

After training the model, humans can be detected from videos. The next step is to separate humans into two groups, which are 'walking' and 'others', because the people that are walking are the main targets to find out the walkable area. The sitting people do not provide any information of the walkable area since they do not move. Then, the position of their feet may not be without obstacles. Instead, people tend to avoid running into barriers so the area of the feet of walking people is the walkable area. Thus, the map of walkable area can be generated by collecting the spaces of the feet of walking people so detecting walking humans then plays a critical role in this experiment.

To divide the detected human beings into two groups, one of the characteristics of video is used. Video consists of several consecutive frames, which means that a frame is related to the previous frames in some ways. For example, the bounding boxes of standing and sitting people tend to remain the same in several consecutive frames and the bounding boxes of walking people change from frame to frame. Thus, each frame from the 10^{th} frame to the last frame is compared to the previous 10 frames firstly. If a bounding box of a person remains in the same position with the same size in these 10 consecutive frames, this bounding box is marked as the 'other' group. This rule can also avoid using the false positive bounding boxes to generate the walkable map since most of the false positives, such as traffic lights or billboards, do not move. Second, if a bounding box appears in the same position with the same size in more than 60% of the frames, this bounding box is also marked as the group 'other' because this person may not be detected in some frames if he or she is occluded or hidden by other people for a short period of time. Third, if the bounding box shares the same bottom line with the image, this person will be marked as 'other' as well. This is because it is likely that the bounding box of that person does not include the feet of the person. It is more likely that only the upper body of the person is shown in the image. Last, all the bounding boxes that are not marked as 'other' are in 'walking' group.

3.8 Walkable Area Creation

To generate a map of walkable area, the locations of the feet of 'walking' group are important. It is because people tend to avoid obstacles while walking. However, the YOLO model is not trained to detect human feet in this experiment. The YOLO model can only detect the human full body bounding boxes. Thus, the bottom lines of the human full body bounding boxes are considered as the position of their feet in this experiment. The way to create walkable area map is to collect all the bounding boxes that is marked as 'walking' in the all of the images from a video and select one of the images that has relatively less human to be plotted the walkable area on. With all the bounding boxes of the people and an image, the function `line` in `OpenCV` is used to plot all the bottom lines of the bounding boxes on the image. Finally, the walkable area of the scene is created.

3.9 Transform Image to Bird Eye View

The images with walkable area should be combined to create a walkable area map. To combine different images, the view of all the images should be transformed to the bird eye view, which is the view from the top. Moreover, the scales of the images should be unified to assure that the same object has the same size in different images. To transform the view of the images, Inverse Perspective Mapping(IPM) in `OpenCV` is used in this section.

IPM is a mathematical method to conduct homography transformation by marking 4 points in the original image and corresponding them to 4 new points in the result image manually. With the 4 points, a transformation matrix can be created and the whole image

can be transformed to an image from bird eye view by this matrix. The mathematical way to compute the transformation matrix is shown in following steps.

- Step 1: Calculate λ, μ, and τ in Equation 3.9, where x_1, x_2, x_3, and x_4 correspond to the x coordinates of the 4 points in the source image(original image), while y_1, y_2, y_3, and y_4 correspond to the y coordinates of the 4 points in the source image.

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \lambda \\ \mu \\ \tau \end{pmatrix} = \begin{pmatrix} x_4 \\ y_4 \\ 1 \end{pmatrix} \tag{3.9}$$

- Step 2: Create a matrix A that will map point (1, 0, 0) to a multiple of point (x_1, y_1, 1), point (0, 1, 0) to a multiple of point (x_2, y_2, 1), point (0, 0, 1) to a multiple of point (x_3, y_3, 1), and point (1, 1, 1) to a multiple of point (x_4, y_4, 1), as shown in Equation 3.10

$$A = \begin{pmatrix} \lambda \cdot x_1 & \mu \cdot x_2 & \tau \cdot x_3 \\ \lambda \cdot y_1 & \mu \cdot y_2 & \tau \cdot y_3 \\ \lambda & \mu & \tau \end{pmatrix} \tag{3.10}$$

- Step 3: Create matrix B by repeating the same steps as step 1 and step 2 using the coordinates of 4 points from the destination image, which is the result image after IPM.

- Step 4: Create matrix A^{-1}, which is the inverse matrix of A as shown in Equation 3.11, where $det(A)$ represents determinant of matrix A and $adj(A)$ represents adjugate matrix of matrix A.

$$A^{-1} = \frac{1}{det(A)}(adj(A)) \tag{3.11}$$

- Step 5: Calculate a combined matrix C in Equation 3.12.

$$C = B \cdot A^{-1} \tag{3.12}$$

- Step 6: Map the points from the source image to the destination points by matrix C by Equation 3.13, where (x, y) is the coordinate of location of the point in source image and (x', y') is the corresponding coordinates of location in destination image.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = C \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.13}$$

- Step 7: Project the points to the plane $z = 1$ as shown in Equation 3.14 and Equation 3.15.

$$x'' = \frac{x'}{z'} \tag{3.14}$$

$$y'' = \frac{y'}{z'} \tag{3.15}$$

The reasons of each step will be explained in this paragraph. In step 1, the dimension is expanded from 2 dimensional coordinates to 3 dimensional coordinates by adding 1 as the coordinate as z-axis. In step 2, matrix A is created to map 4 basis vectors, which are (1, 0, 0), (0, 1, 0), (0, 0, 1), and (1, 1, 1), to the selected 4 points in the source image. In step 3, matrix B is created to map the 4 basis vectors to the chosen 4 points in the destination image. In step 4, the inverse matrix of A can map from the source image to the basis vectors. It is because matrix A maps from basis vectors to the source image and the inverse matrix can do the same thing in the opposite direction. In step 5, matrix C can map points from source image to destination image. It is because the points is first transformed by matrix A^{-1} and then transformed by matrix B. Thus, the points are first mapped from the source image to the basis vectors and then from the basis vectors to the destination points. In step 6, the points are mapped by matrix C. Last, in step 7, the dimension of the points are transformed from 3 dimensional coordinates to 2 dimensional coordinates by dividing z coordinate to make it 1. These are the whole steps in IPM and can be simply achieved in `Opencv`.

The following steps below are taken to perform IPM transformation in `Opencv` in `Python`. First, 4 points are selected from the original image. Usually, 4 points that form a square are selected because it is easier to assigned them to a new image. Second, mark 4 points in the new image and each corresponds to one of the 4 points in the original image. For example, in Figure 3.3, the four blue points are marked in the original images, which would form a square in the new image and each point corresponds to a corner of the square. Third, with these 8 points(4 in the original image and 4 in the new image), the perspective transformation matrix can be calculated by the function `getPerspectiveTransform` in `OpenCV`. Finally, the new image can be created with the perspective transformation matrix by the function `warpPerspective` in `OpenCV`. Then, the image of the bird eye view is created and the next steps are to uniform the scale of the images and combine them together to create a walkable area map.

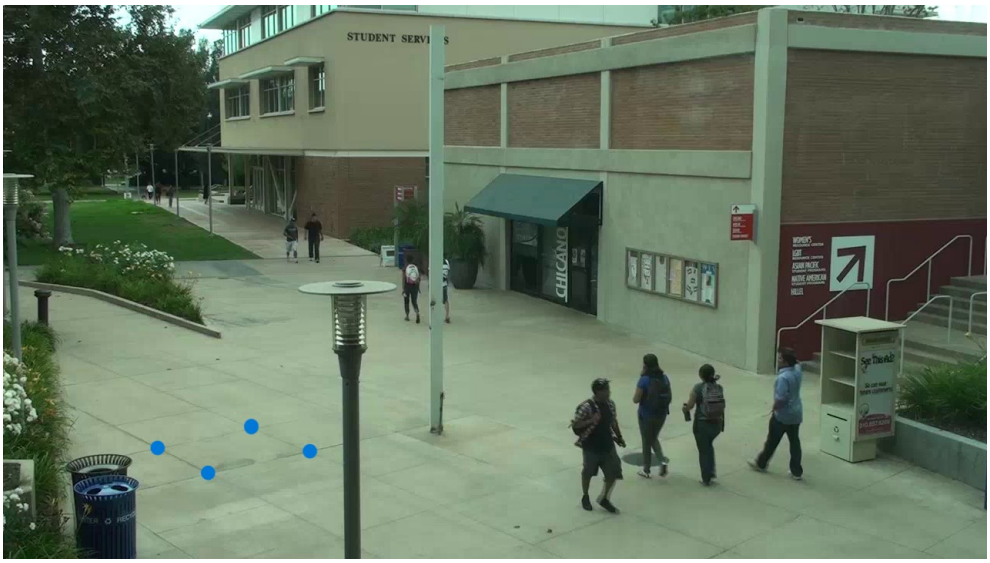

Figure 3.3: Original Image of Inverse Perspective Mapping

Chapter 4

Result

This paragraph will discuss the results of each steps of the experiment and give a simple explanation of the results. The results include the performance of the human detection model, images of walkable area, and images of bird eye view.

4.1 Human Detection Models

This section will discuss the results of different human detection models. First, some metrics using to evaluate the model will be explained first. Second, the results between the pre-trained YOLO v5 model and the YOLO v5 model trained on CrowdHuman dataset will be compared. Last, the models with different hyperparameters will be compared by some metrics in a table.

4.1.1 Evaluation Metrics

Several metrics are commonly used to evaluate the performance of the model and the best model can be chosen based on these metrics. Most of the metrics are calculated based on 4 outcomes of the model, which are true positive(TP), true negative(TN), false positive(FP), and false negative(FN). These 4 outcomes are explained below.

- TP: an outcome that the model correctly predicts the positive class, e.g. the model predicts human and there is a human in the image.
- TN: an outcome that the model correctly predicts the negative class, e.g. the model predicts no human and the image contains no human.
- FP: an outcome that the model incorrectly predicts the positive class, e.g. the model predicts there is a human but there is no human in the image.
- FN: and outcome that the model incorrectly predicts the negative class, e.g. the model predicts there is no human but there is at least one human in the image.

With these 4 outcomes, several metrics can be computed and the models can be evaluated by the metrics. The followings are some metrics that are used to evaluate the models in this experiment.

- Precision: computed by Equation 4.1

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

- Recall: computed by Equation 4.2

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

- F1 score: computed by Equation 4.3

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.3}$$

- Average precision(AP): computed by Equation 4.4 [14]

$$AP = \frac{1}{N}\Sigma_{i=1}^{N}\frac{1}{M}\Sigma_{j=1}^{M}Presicion_i(Recall_j) \tag{4.4}$$

Precision is the ratio of the total amount of the correct positive prediction over the amount of true positive predictions. Recall is the the ratio of the correct positive prediction over the total amount of positive samples. For example, if a image contains 100 humans and the model correctly predicts 70 humans (TP) while incorrectly predicts 20 humans (FP), the precision is 7 over 9 and the recall is 0.7. Neither precision or recalls tells the whole story. Sometimes, an excellent precision comes with an awful recall while sometimes an excellent recall comes with a terrible precision. Thus, to achieve a balance between precision and recall, F1 score is created. F1 score is computed by recall and precision and both of the metrics contribute equally. Thus, F1 score is an excellent way to evaluate model performance when both precision and recall are important.

Average precision reveals whether the model can correctly predict positive samples without predicting too many negative samples as positive or not. Average precision is also known as Area Under the Precision-Recall Curve or Area Under Curve. This metric also measures the trade-off between recall and precision.

4.1.2 YOLO v5 Model

The original YOLO v5 model can detect 80 different classes, including humans. However, the performance of the original YOLO v5 model on human still has room for improvement. Thus, a model based on YOLO v5 is trained on CrowdHuman dataset to achieve a better performance.

(a) (b)

Figure 4.1: A comparison between two YOLO v5 models. (a) shows the result of the pre-trained YOLO v5 model. (b) shows the result of the YOLO v5 model trained on CrowdHuman dataset.

The results of the original YOLO v5 model and the YOLO v5 model trained on CrowdHuman dataset are shown in Figure 4.1. The figure shows that the original YOLO v5 model performs well on the people near the camera. However, the people that are far from the camera can hardly be detected. Moreover, once a human is occluded, the bounding box of him or her do not contains the whole human body. This will cause problems in the step of creating walkable area since the bottom of the bounding box has a great chance not being the position of the feet of people. On the other hand, the YOLO v5 model trained on CrowdHuman dataset performs well on the people far from the camera. Moreover, the bounding boxes of the occluded people contain the full body of them. However, there is still some shortage. The model sometimes cannot detect humans in crowds. It sometimes only predicts one bounding box for a crowd of human.

The reason that makes YOLO v5 model perform better on occluded humans is because many training images contains occluded humans in CrowdHuman dataset. Thus, the model learned to find out the bounding box of the full body instead of only the parts that are visible. Furthermore, the training set of the model is large, which consists of 15000 images. The huge training set makes the model to detect humans that only appear in several pixels. The training set also contains crowds of humans. Thus, the model performs better than the original YOLO v5 model in detecting crowds of humans. However, it cannot separate the humans in crowds perfectly because a grid cell in the image is only responsible for one object but this shortage does not influence the performance of the model. It is because the bottom line of the bounding box often includes the positions of feet of most of the humans in the crowd. To conclude, the model trained on CrowdHuman dataset is specifically trained for this experiment so it performs better than the original model in the experiment.

4.1.3 Tuning Hyperparameters

After the training data is chosen, the next step is to train the models with different hyperparameters, such as warmup epochs and learning rate. Table 4.1 shows the performance

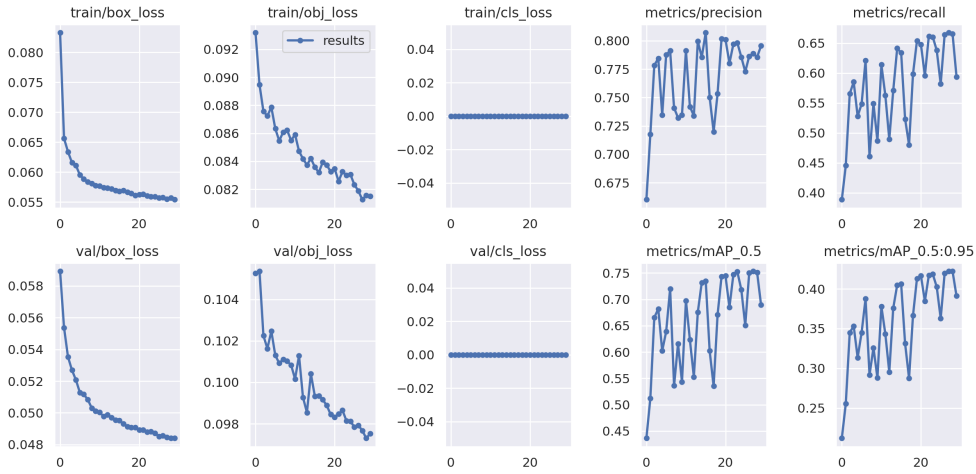

Figure 4.2: Training result of the second model

of the models trained with different hyperparameters.

Model	batch	epoch	w epoch	lr	recall	precision	mAP 0.5	mAP 0.5:0.95
1	16	10	3	0.01	0.6425	0.7856	0.7364	0.4064
2	16	20	3	0.01	0.7998	0.6597	0.7525	0.4195
3	16	10	4	0.01	0.7826	0.6418	0.7338	0.4062
4	16	30	5	0.005	0.5939	0.7955	0.6893	0.3911

Table 4.1: Comparison of models with different hyperparameters

In this task, precision is important because it is correctly predicted positive samples divided by all samples that are predicted positive. If the precision is low, it means that the model predicts a lot of bounding boxes contains no human beings. This leads to incorrect walkable area map. The recall is the ratio of correctly predicted positive samples to all positive samples. Thus, the low recall results in predicting only a little humans in the image and plotting only some part of walkable area map. This can be solved by using longer videos. As a result, in this report, precision is chosen to evaluate the models. Thus, the model 4 is chosen to perform human detection and to generate walkable area.

Figure 4.2 shows the loss curve, recall curve, precision curve, and mAP curve of training set and validation set. The box loss is the regression loss of the bounding boxes and it is calculated by Mean Squared Error as shown in Equation 4.5 where $\hat{B}_i$ is the predicted bounding boxes, B_i is the ground truth bounding boxes, and n is the amount of the bounding boxes. The object loss, which is calculated by the confidence of the presence of an object, and classification loss are cross entropy loss as shown in Equation 4.6 where N is the amount of the bounding boxes, y_n is the true value, x_n is the predicted value, and σ is the sigmoid

function which is shown in Equation 3.6.

$$MSE = \frac{\Sigma(B_i - \hat{B}_i)^2}{n} \tag{4.5}$$

$$Loss = \frac{1}{N}\sum_{n=1}^{N}(y_n \cdot log\sigma(x_n) + (1 - y_n) \cdot log(1 - \sigma(x_n))) \tag{4.6}$$

Training loss and validation loss curves show that the model is not overfitting nor underfitting. The classification loss is always 0 because the model only predict one class, human.

4.2 Walkable Area

This section will compare the performance of the model with constraints to the model with no constraints as well as the predicted walkable area to the ground truth. Precision, recall, and IoU are used to evaluate the performance of the walkable area generation in 30 different videos.

Table 4.2 shows the performance of the model with constraints and without constraints in 30 different videos as well as the average performance. As the table shows, the average precision of model with constraints is slightly better than the model without constraints. However, the average recall and IoU of the model with constraints are considerably lower than the model without constraints. This is because the first and second constraints set in Section 3.7 are so strict that only few bounding boxes are deleted by these constraints. Most of the removed bounding boxes are deleted based on the third constraint. As a result, if the bottom 1% of the scene in a video is walkable area, the IoU and recall of the walkable area of that video would be higher in the model without constraints than in the model with constraints, e.g. HS046 as shown in Figure 4.3a where the blue area is the ground truth walkable area and Figure 4.3b where the red area is the prediction removed by constraints. The table indicates that the constraints do not improve the performance of the model greatly and sometimes even make the model perform worse than the model with no constraints. However, this does not mean that the constraints are unnecessary. It only shows that the constraints need to be modified.

Table 4.2 also shows that the model performs best in the video “London Streets 1” which is shown in Figure 4.4. This is because the only thing that may affect the performance of the model in this video is the humans occluded by other humans. As shown in Figure 4.5, the humans in crowds or the humans occluded by others share the same bounding box. As mentioned the Section 4.1.2, the problem that YOLO v5 model cannot detect the humans in crowd separately does not affect the performance a lot. The precision of the walkable area is amazingly high. On the other hand, the recall and the IoU is low. This will be improved if the length of the videos becomes longer, which means there are more people in the videos so that more walkable area are predicted and the recall and IoU will be higher. The model performs worst in the video “London Buses” as shown in Figure 4.6. This is because this

(a) (b)

Figure 4.3: Walkable area comparison. (a) shows the ground truth walkable area. (b) compares the result generated by model with constraints and model without constraints.

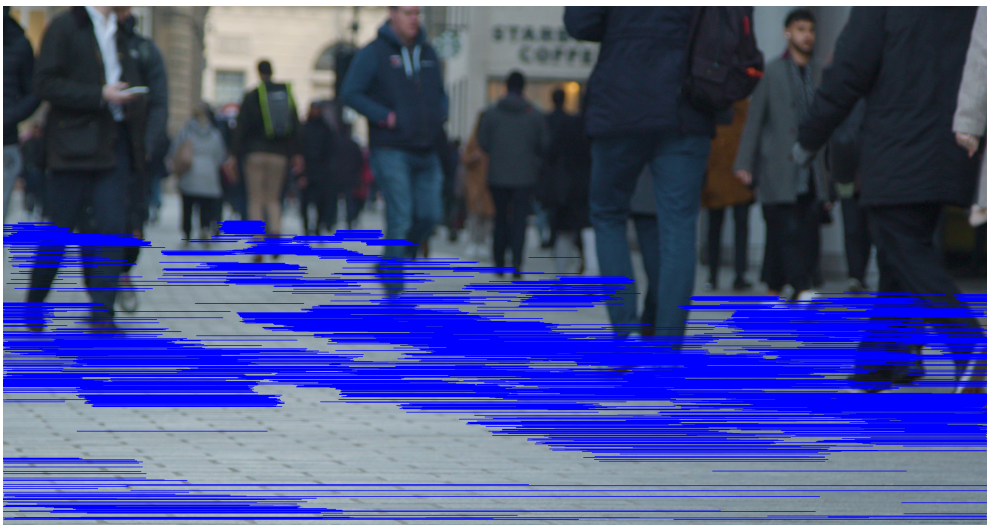

Figure 4.4: Predicted walkable area in video 'London Streets 1'

video contains humans on the buses and these people are also detected. Moreover, they are divided into 'walking' group as the buses are moving. Thus, some false positive areas are generated. This shows that the human detection model is strong as it can detect people with some parts occluded on the bus. However, these people are not the targets in this experiment. Thus, some more constraints are needed to remove these predictions.

4.3 IPM

This section will show the result after IPM and briefly discuss the result final results.

Figure 4.7 shows the final output of the experiment. The image is complicated because it contains lots of obstacles such as cars and humans. IPM transforms the image to the bird eye view where the z-axis is 0. Thus, the obstacles are deformed in the final images. Moreover,

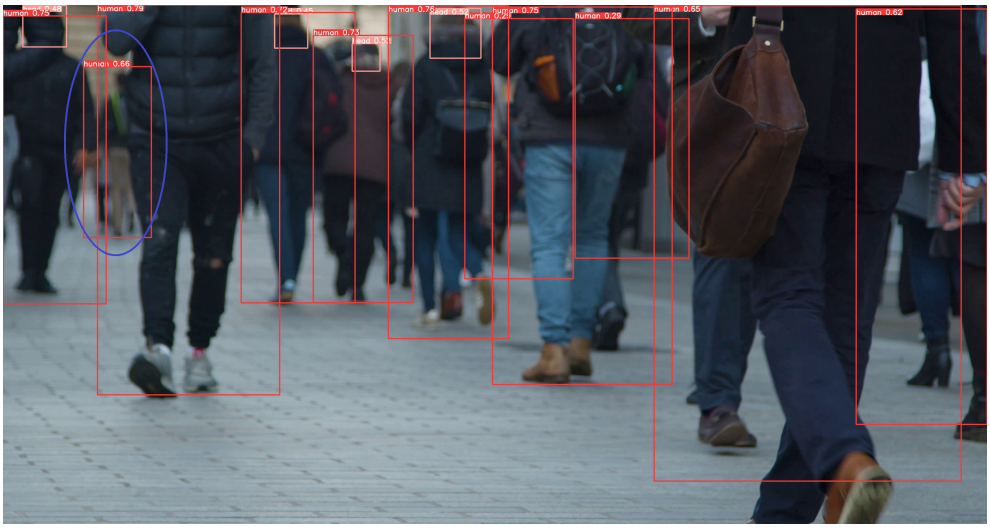

Figure 4.5: Human detection result of London Street 1

Figure 4.6: Predicted walkable area in video 'London Buses'

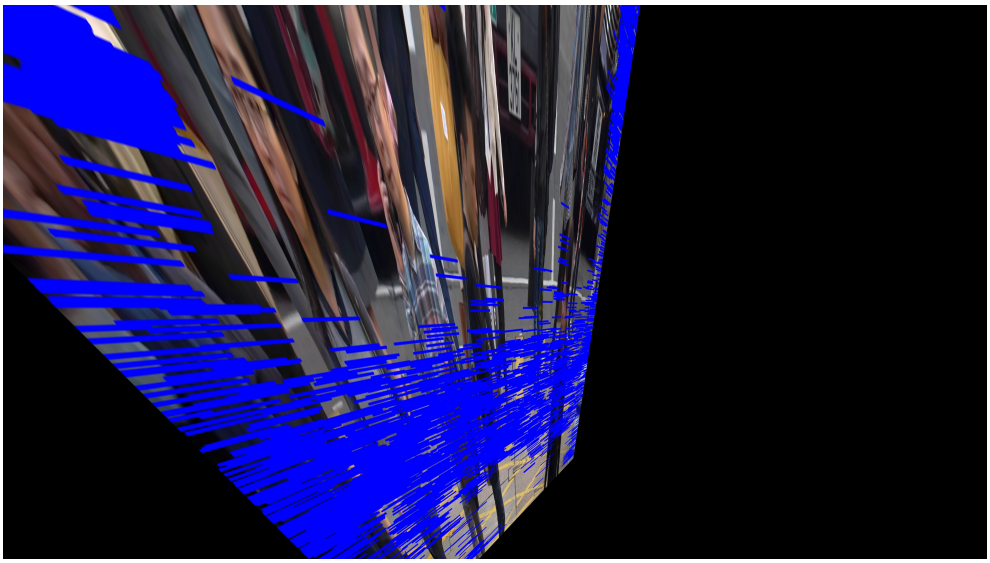

Figure 4.7: Final result after IPM in video 'KowloStreets'

the 8 points, 4 in the original image and 4 in the transformed image, are chosen manually. As a result, the result of IPM may not be the precise bird eye view because the 4 points in the transformed image can hardly be accurately chosen.

4.4 Further Research

This section will discuss the methods the author would try to improve the performance of the model in future works.

First, using the bottom line of the bounding boxes as the position of the feet of humans is rough. The way to find the feet in future work is to combine this model to another YOLO model trained to predict feet. After the first model predicts the bounding boxes of humans, the second model uses these bounding boxes as input and predicts bounding boxes of the feet in the bounding boxes of humans. This may predict humans feet more accurately because the first model helps avoid the second model to generate false positives in the background.

Secondly, the constraints are too strict to remove the humans that do not move. Thus, the constraints will be designed to divide the bounding boxes that only move a little bit to the 'non-walking' group. Moreover, the current constraints regard people on vehicles, such as buses, as walking people, which leads to false positives in walkable area. To avoid this, the YOLO model should be able to detect vehicles as well. Then, some constraints can be designed to avoid using the humans on vehicles. For example, if the IoU of a bounding box of a human and a bounding box of a vehicle is greater than a threshold, the bounding box of the human must be deleted. Furthermore, jaywalking is common in some countries and the generated walkable area may be unsafe. This is also a problem to be solved in the future.

Thirdly, IPM is needed to combine all generated walkable area into a map. Thus, the

way to generate the images from bird eye view precisely is required. In this experiment, the transformation can only be performed manually. The way to do it automatically will be further designed in future works.

Lastly, after the accurate walkable area map is created, the model to transform the image information to voice one can be created to let the visually impaired people 'hear' their surroundings. This can probably allow them to walk on streets without any other helps.

	with constraint			without constraint		
Video	IoU	precision	recall	IoU	precision	recall
London Streets1	36.73%	99.96%	36.73%	38.28%	99.91%	38.3%
Oxford Shoppers1	31.33%	99.28%	31.4%	33.11%	99.32%	33.19%
Oxford Shoppers2	47.78%	97.28%	48.42%	50.20%	97.41%	50.88%
ClipVidevo	50.51%	91.09%	53.13%	53.29%	90.71%	56.36%
GardenHill	17.02%	60.00%	19.2%	16.99%	59.62%	19.20%
General public	55.61%	83.75%	62.34%	59.23%	82.25%	67.92%
HS046	35.09%	94.50%	35.82%	44.31%	95.59%	45.23%
Jakarta	39.18%	90.05%	40.96%	40.13%	90.26%	41.94%
KowloStreets	43.74%	72.23%	52.58%	45.35%	72.95%	54.52%
Leicester Square at Night	30.71%	94.0%	31.32%	31.21%	93.15%	31.94%
London Streets2	38.46%	99.96%	38.47%	39.23%	98.36%	39.49%
London Buses	25.44%	42.58%	38.73%	26.46%	43.03%	40.74%
London OxfordCircus	24.71%	73.22%	27.16%	24.71%	73.22%	27.16%
London WestminsterBridge	53.98%	70.44%	69.79%	58.41%	70.51%	77.29%
Melbourne Federation Square2	39.2%	64.51%	49.97%	38.90%	59.39%	52.99%
Melbourne Federation Square1	31.71%	91.87%	32.62%	32.58%	92.07%	33.52%
NYC RainStreet	30.7%	97.49%	30.95%	32.93%	97.40%	33.22%
Piccadilly Circus Night	65.37%	95.89%	67.25%	71.55%	96.23%	73.62%
Timelapses	54.98%	95.30%	56.51%	56.72%	95.22%	58.39%
Tokyo Commuters	72.45%	98.57%	73.21%	74.22%	98.61%	75.01%
Tokyo CommutersTimelapse1	57.74%	98.94%	58.10%	60.37%	98.99%	60.74%
Tokyo CommutersTimelapse2	56.22%	97.77%	56.95%	59.05%	97.87%	59.81%
Tokyo PedestrianCrossing	62.07%	95.41%	63.99%	70.32%	95.92%	72.48%
Tokyo ShinjukuStation	51.24%	90.23%	54.25%	54.98%	90.83%	58.21%
Tokyo Timelapse2	68.61%	93.48%	72.05%	71.90%	93.76%	75.51%
Tokyo TrainStation	70.96%	99.78%	71.07%	73.28%	99.79%	73.40%
Toronto IceRink6	77.35%	98.98%	77.97%	81.34%	99.03%	81.99%
guangzhou	64.55%	95.86%	66.4%	66.59%	95.43%	68.78%
mixkit	83.01%	98.00%	84.45%	84.31%	98.03%	85.77%
moskva	57.23%	98.45%	57.75%	58.47%	98.48%	59.00%
Average	49.12%	89.30%	51.98%	51.61%	89.11%	54.89%

Table 4.2: Comparison between walkable area with constraint and without constraints in 30 different videos

Chapter 5

Conclusions

In summary, the performance of the model is acceptable but there is still room for improvement. The human detection model works well in detecting the full body of the human even if he or she is occluded in this experiment. However, the constraints applied to figure out the walking people do not work as expected. As a result, the constraints need to be modified and some new constraints need to be added in future work. The result of IPM transformation can be improved in the future works. To conclude, this experiment is acceptable but still need some more improvement to be applied in the real life.

Bibliography

[1] Bertozz, M., Broggi, A., and Fascioli, A. Stereo inverse perspective mapping: theory and applications. *Image and vision computing 16*, 8 (1998), 585–590.

[2] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection.

[3] Camplani, M., Paiement, A., Mirmehdi, M., Damen, D., Hannuna, S., Burghardt, T., and Tao, L. Multiple human tracking in rgb-depth data: a survey. *IET computer vision 11*, 4 (2017), 265–285.

[4] Gilani, L., Tahir, S. F., Rasheed, U., Saqib, H., Hassan, M., and Alquhayz, H. Fruits and vegetables freshness categorization using deep learning. *Computers, Materials and Continua 71* (01 2022), 5083–5098.

[5] Girshick, R. Fast r-cnn. In *2015 IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV)* (NEW YORK, 2015), vol. 2015 of *IEEE International Conference on Computer Vision*, IEEE, IEEE, pp. 1440–1448.

[6] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation.

[7] Han, C., Gao, G., and Zhang, Y. Real-time small traffic sign detection with revised faster-rcnn. *Multimedia tools and applications 78*, 10 (2018), 13263–13278.

[8] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence 37*, 9 (2015), 1904–1916.

[9] Kumar, N., and Sukavanam, N. An improved cnn framework for detecting and tracking human body in unconstraint environment. *Knowledge-based systems 193* (2020), 105198.

[10] Lin, C.-Y., and Lian, F.-L. System integration of sensor-fusion localization tasks using vision-based driving lane detection and road-marker recognition. *IEEE systems journal 14*, 3 (2020), 4523–4534.

[11] Lv, B., Wu, L., Huangfu, T., He, J., Chen, W., and Tan, L. Traditional chinese medicine recognition based on target detection. *Evidence-based complementary and alternative medicine 2022* (2022).

[12] Ma, S., Huang, Y., Che, X., and Gu, R. Faster rcnn-based detection of cervical spinal cord injury and disc degeneration. *Journal of applied clinical medical physics 21*, 9 (2020), 235–243.

[13] Mallot, H. A., Bülthoff, H., Little, J., and Bohrer, S. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics 64*, 3 (1991), 177–185.

[14] Mohan, H. M., Rao, P. V., Kumara, H. C. S., and Manasa, S. Non-invasive technique for real-time myocardial infarction detection using faster r-cnn. *Multimedia tools and applications 80*, 17 (2021), 26939–26967.

[15] Oliveira, M., Santos, V., and Sappa, A. D. Multimodal inverse perspective mapping. *Information fusion 24* (2015), 108–121.

[16] Prakash, C. D., Akhbari, F., and Karam, L. J. Robust obstacle detection for advanced driver assistance systems using distortions of inverse perspective mapping of a monocular camera. *Robotics and autonomous systems 114* (2019), 172–186.

[17] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection.

[18] Redmon, J., and Farhadi, A. Yolo9000: Better, faster, stronger.

[19] Redmon, J., and Farhadi, A. Yolov3: An incremental improvement.

[20] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence 39*, 6 (2017), 1137–1149.

[21] Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., and Sun, J. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123* (2018).

[22] Tzutalin. Labelimg. Free Software: MIT License, 2015.

[23] Wang, Z., Jin, L., Wang, S., and Xu, H. Apple stem/calyx real-time recognition using yolo-v5 algorithm for fruit automatic loading system. *Postharvest biology and technology 185* (2022), 111808.

[24] Wen, S., Wang, S., Zhang, Z., Zhang, X., and Zhang, D. Walking human detection using stereo camera based on feature classification algorithm of second re-projection error. *Frontiers in neurorobotics 13* (2019), 105–105.

[25] Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. Pfinder: real-time tracking of the human body. *IEEE transactions on pattern analysis and machine intelligence 19*, 7 (1997), 780–785.

[26] Yu, J., Han, Y., and Hahn, H. Walking assistance system for sight impaired people based on a multimodal information transformation technique. *Journal of Institute of Control, Robotics and Systems 15*, 5 (2009), 465–472.