an-Eye: SAFE NAVIGATION IN FOOTPATH FOR VISUALLY IMPAIRED USING COMPUTER VISION TECHNIQUES

BY

MD. AFIF AL MAMUN

ID: 162-15-7774

AND

IMAMUL KADIR

ID: 162-15-7775

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

Mr. Subroto Nag Pinku
Lecturer
Department of CSE
Daffodil International University

Co-Supervised By

Aniruddha Rakshit
Senior Lecturer
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

JUNE 2020

APPROVAL

This Project titled "**an-Eye: SAFE NAVIGATION IN FOOTPATH FOR VISUALLY IMPAIRED USING COMPUTER VISION TECHNIQUES**", submitted by Md. Afif Al Mamun and Imamul Kadir to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 9th July 2020.

BOARD OF EXAMINERS

Dr. Syed Akhter Hossain **Chairman**

Professor and Head

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

Dr. Fizar Ahmed **Internal Examiner**

Assistant Professor

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

Md. Tarek Habib **Internal Examiner**

Assistant Professor

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

Md. Tarek Habib **External Examiner**

Assistant Professor

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Daffodil International University

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Mr. Subroto Nag Pinku, Lecturer, Department of CSE,** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

Mr. Subroto Nag Pinku
Lecturer
Department of CSE
Daffodil International University

Co-Supervised by:

Aniruddha Rakshit

Aniruddha Rakshit
Senior Lecturer
Department of CSE
Daffodil International University

Submitted by:

Md. Afif Al Mamun

ID: 162-15-7774

Department of CSE

Daffodil International University

Imamul Kadir

Imamul Kadir

ID: 162-15-7775

Department of CSE

Daffodil International University

ACKNOWLEDGMENT

This thesis is the outcome of a year-long challenging journey, upon which many people have provided their support in many ways. These people made this journey easier and enjoyable for us. As this journey comes to an end, we'd like to express our gratitude towards them.

First, we express our heartiest thanks and gratefulness to Almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We would like to express our humblest gratitude to **Mr. Subroto Nag Pinku, Lecturer** for his constant support, guidance, and advice. If it was not because of him, this thesis would never be completed. We are fortunate to have him as our supervisor. He did not only played the role of our supervisor but also played the role of our guardian.

Our heartfelt appreciation goes to **Mr. Aniruddha Rakshit, Senior Lecturer** for his continuous support, time, and care. From the beginning of this research to this very end, he was always with us. We are deeply indebted to him.

We would like to express our gratefulness to **Dr. Syed Akhter Hossain, Professor, and Head of the Department of CSE.** We never had the chance to work with him directly but we always get encouraged and motivated by his endearment speeches.

We want to acknowledge our respect and love to our parents and family members for their constant support and care they've always provided us.

Our special acknowledgment goes to **Abed Siam, Jaki Al Mamun, Nazmus Shakib** for helping out in the data collection.

Finally, we must acknowledge with due respect to all the people out there who knowingly or unknowingly helped us in the successful completion of this work.

ABSTRACT

Navigating from one place to another has been a problematic task for the blind. In Bangladesh, the existing footpaths are mostly crowded or broken. Often, visually impaired people get hurt while walking in a footpath as they do not have anything but a stick to help them. Considering the problem scenario, we are proposing a smart solution to identify safe footpath and detect obstacles in a footpath. The system will also be capable of estimating the distance of the object as well as suggesting the safe pathway. To train the models we built a dataset of footpath images of Dhaka containing 3,000 hand-annotated RGB images for semantic segmentation and another dataset containing 500+ samples of real-world distances of reference objects w.r.t to their pixel coordinates in an image for distance estimation. We adopted and modified the U-Net architecture that is trained on our segmentation dataset which is capable of inference safe footpath with 96% accuracy with as low as 4.7 million parameters. The system utilizes YOLOv3 architecture for object detection and a polynomial regression based novel approach to estimate object distance. The distance measurement model obtains a score of 94%.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Introduction

Blindness has always been one of the most crucial health problems in Bangladesh. It can be caused by various reasons such as – accidental injuries, genetic issues, tumors, strokes, lazy eye, diabetic retinopathy, glaucoma, macular degenerations, cataracts, etc. The WHO published a report [1] on visual impairment prevalence in 2010 that estimated 285 million people worldwide are affected by visual impairment. Table 1.1.1 provides the geographical distribution of the problem.

Table 1.1.1: Regional Distribution Data of Visual Impairment. (Per 100 pop)

Region	Vision Loss	Blindness
Africa	2.5%	0.7%
America	2.6%	0.4%
Eastern Mediterranean	3.2%	0.8%
Europe	2.9%	0.3%
South-East Asia (India Excluded)	4.1%	0.7%
Western Pacific	2.8%	0.5%
China	5%	0.6%
India	4.6%	0.7%

The regional distribution shows that the Asian continent is suffering the most from visual impairment where the rate of vision loss maxed up to 5% and blindness to 0.8%. This data proves the severity of the problem is in South Asia.

According to a report [2] published in 2012 by the Directorate General of Health Services (DGHS), there are more than 750,000 people aged 30 years or more in Bangladesh are suffering from blindness. The report says around 5M people including children had refractive errors. It was also pointed out that the number of blind people may become double by the year 2020. Certainly, the number has increased so much since then. In a recent report [3] published in late 2018 by Seva Foundation, a non-profit organization from the US said that at the time of publishing their report there were around 871,000 blind people in our country which was roughly around 0.82% of the total population.

Navigating from one place to another has been very perplexing and challenging for visually impaired people. In most cases, blind or visually impaired people have almost nothing to assist them rather than a stick while they are moving. The scenario worsens when it comes down to our country as we have very narrow pathways. In this research, we propose a vision-based real-time system to assist visually impaired by identifying walkable footpath and detecting obstacles in the way along with an estimated distance of the identified object in the pathway by using computer vision techniques. The system can also guide its users in which way they should move to avoid obstacles.

One of the fundamental objectives of this research is to detect safe footpath. For this, it is monumental to identify the footpath boundary to the pixel level to perceive different knowledge like the borderlines, condition, or if there is any obstacle in the pathway. Hence, we used semantic segmentation to classify the footpath region from an image. Semantic segmentation is a technique that classifies an image to the pixel level. The technique is broadly discussed in later chapters. The segmentation task is carried out by U-Net [4], a convolutional neural network (CNN) architecture for semantic segmentation. U-Net was developed for rapid and precise segmentation for medical imaging. We tuned U-Net as though we can use it on our dataset of footpath images. For the detection of objects, we are using state of the art object detection technique YOLOv3 [5] with pre-trained weights trained on COCO [6] dataset along with our custom classes. YOLOv3 trained on COCO comes with mAp of 60.6. We used a regression-based model to estimate the distance of identified objects from their bounding box coordinates. Finally, for guidance, we used several image processing techniques. The In-depth explanation behind all these methodologies will be given in later chapters.

1.2 Motivation

Over 6 million people in Bangladesh need vision correction by spectacles and other means. Among them about 1.4 million children under the age of 15 are blind. Being human, we all need to move from one place to another no matter whether we are physically challenged or not. Compared to developed countries, we can easily deduce the complexity of moving around for a visually impaired in our country especially in the urban areas. The footpaths are occupied with hawkers and shops which left a little space to walk. They are narrow and often broken. Even normal people face problems while walking through the footpaths due to a huge amount of shops and hawkers. Also, there are uncovered manholes, pillars, trees, and often parked vehicles in those footpaths which leaves visually impaired peoples nearly impossible to walk on and pass by.

On the other hand, in recent years we see the enormous development in the field of artificial intelligence and computer vision. Advanced machine learning and deep learning algorithms are being applied almost in every sector. We can also observe this in the advancement of autonomous vehicles [7] technology. Very large-scale studies [8] are being undertaken to build self-driving cars although it's mostly for our luxury. By watching the evolving autonomous technologies, we asked ourselves what if we can use some key-concepts from autonomous vehicles and use it to minimize a social problem while keeping the cost minimal? What if we can build a system that may help the blind or visually impaired in their day-to-day movement?

Therefore, to assist peoples with vision-problem we decided to come up with an efficient but feasible solution and build an intelligent system that can guide them while walking through footpaths in real-time. The everyday struggle of visually challenged people and autonomous vehicle technology motivated us to do this research.

1.3 Rationale of the Study

The problem every blind person faces every day to walk from one place to another is unquestionable. The statistics shown in the Introduction chapter comprehensibly reveals the severity of the problem. As the population is rising so is the number of visually impaired. Navigation is one of the major problems they have to face every day. Several types of researches have been done following different approaches to find assistive solutions for the problem in different countries. Some of them are studied later in the background chapter. However, most of them are very much

costly and are mostly built for indoor use only. On the other hand, in the context of our country, there is no notable research that has been done to build an assistive navigation system that is socially and economically feasible even though its necessity is indubitable. Hence, we believe this is the first approach to come up with a solution in the problem domain by using computer vision and deep learning algorithms and the research is expected to be very much compelling and effective.

1.4 Research Questions

This research takes into account several questions regarding the topic and throughout the study, it tries to address them while keeping the efficiency and feasibility in consideration. While addressing each question it comparatively analyzes different ways to solve them and picks the best.

1. How to identify safe footpath? What existing methods can be used and which one is the most suitable?
2. What algorithm should be perfect for real-time obstacle detection in this scenario?
3. How to estimate identified obstacle distance without using external sensors/hardware?
4. How to generate suggestions for the safe pathway?

1.5 Expected Outcome

The expected outcome of this research is to a system for assisting navigation. Having said that, it specifically outputs the following:

1. An annotated footpath image dataset of Dhaka city footpaths for semantic segmentation.
2. Deducing a deep learning model for footpath identification after comparative analysis to similar algorithms in terms of computational cost and accuracy.
3. A low-cost unique object distance estimation model along with object detection.
4. An image processing model that can analyze images and tell the user in which way they should move to avoid obstacles.

1.6 Report Layout

The whole report is divided into several chapters by the contents where each chapter contains multitudinous sections to portray the whole research properly to the reader. Simplicity is maintained while writing every chapter. In chapter 1 an overview of the research is given. The chapter illustrates the purpose of the research, its rationale, the problems it intends to solve, etc.

Chapter 2 mainly focuses on the background of the study. Brief descriptions of each principle terminologies that are used in this research are given along with respective algorithms. The chapter draws an evolutionary flow of different terminologies and comparative analysis of the terms so that the reader may find out what could be the other options to solve a particular problem and why the used procedures are dominant over others. Reviews of related works are also given which includes their approaches, feasibility, and limitations.

The working plan, procedures, and implementation are broadly explained in Chapter 3. It is broken down into many sections and sub-sections to explain each of the procedures properly. The core section of the chapter includes data collection, preprocessing, and analysis procedures, training of the proposed models on the prepared dataset and finally combining the trained models altogether.

Chapter 4 briefly visualizes the experimental setup, results of the experiments, and discussion according to the analysis results. Several testing results of the final system are attached in this chapter to give a crystal idea of the overall research.

In chapter 5 we showed the socio-economic influence, sustainability plan, ethical aspects, and the impact of the research on the environment. The constraints and limitations of the system are also included in the chapter.

Finally, in the last chapter (Chapter 6), a complete summarized overview of the study is given to conclude the research. The future scope and plans based on the study are added later in the chapter.

CHAPTER 2

BACKGROUND

This chapter portrays the theoretical explanation of the terminologies used in this research. The chapter also focuses on related works about the topic that have been done previously. It depicts a comparative analysis of those work regarding this work in different criteria such as – performance, feasibility. Every terminology has been described unequivocally to keep the chapter unambiguous and the reason behind using it in this research.

2.1 Preliminaries/Terminologies

The system as a whole is a combination of several tasks e.g. safe footpath segmentation, object detection, distance measurement, and navigation guide. Each of them was accomplished using different methodologies, different background theory applies to them. Therefore, to illustrate each section properly they are explained in different sections.

2.1.1 Footpath Segmentation

One of the very core purposes of this research was to identify and segment the footpath region from an image. By nature, it is an image processing task and one may think of different ways to accomplish it. In classical computer vision, the segmentation of an image can be done in various ways. Some techniques are explained below in brief along with their sustainability to solve the problem being discussed.

1. Edge Detection: Edge detection is a popular method of segmentation by which sharp changes of pixel intensity are detected and object edges are identified and can be separated from the background. The basic idea of edge detection techniques is to identify discontinuities in an image. One of the most popular methods to detect edge is using the Canny edge detector [9]. This edge detection technique follows several step-by-step processes like noise filtering with Gaussian filter, gradient calculation, non-maxima pixel suppression in edges, thresholding, and hysteresis, etc. The result of applying Canny edge detection can be found in Figure 2.1.1.1.

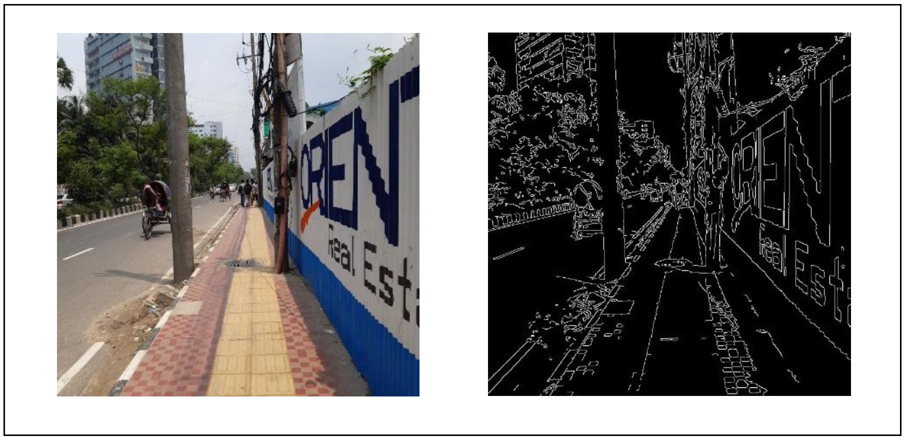


Figure 2.1.1.1: Canny edge detector

The Canny edge detector is a quite efficient technique to detect edges in a photo. It was initially applied to our dataset to find out the footpath boundary in an image. However, even though this technique was able to detect footpath edges on some ideal cases, it fails whenever the scene gets complicated therefore Canny edge wasn’t a preferable solution for this problem. Figure 2.1.1.2 illustrates two cases of the algorithm in the dataset. In the figure, the first sample shows a successful identification of the footpath boundary and the second one shows a failure. Therefore, it was concluded that Canny edge detection wouldn’t be a practical solution to the problem.

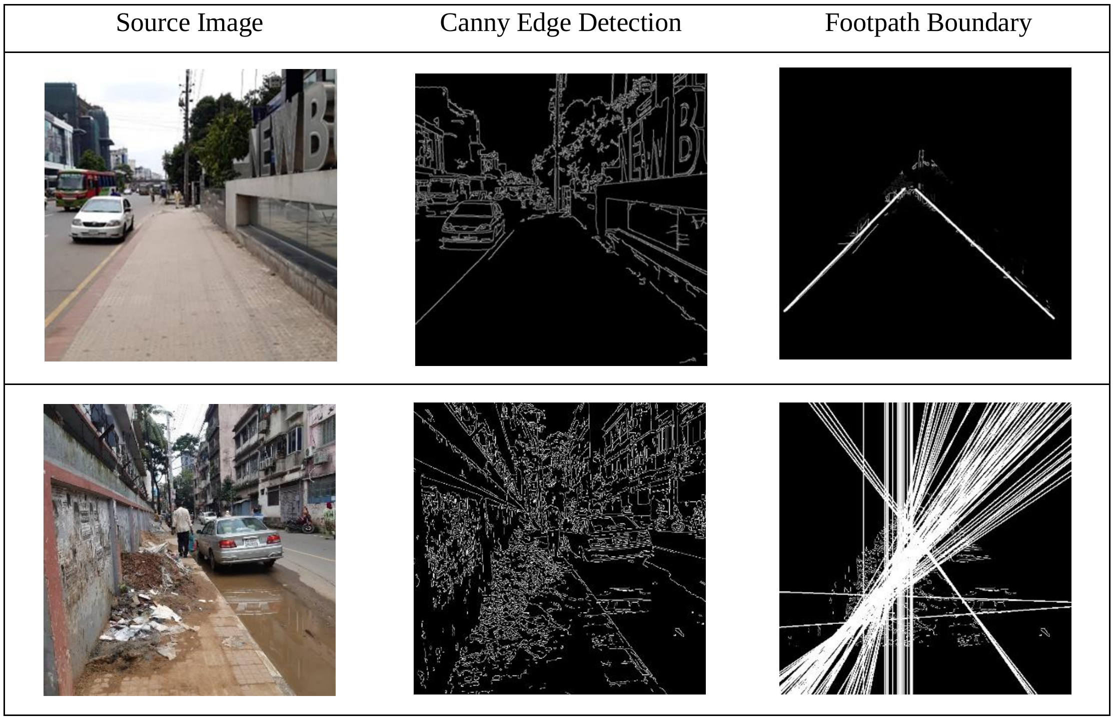


Figure 2.1.1.2: Observation of the Canny edge detector in the dataset

2. Thresholding: Image Thresholding [10] is a simple and straightforward approach to create binary images from grayscale images. It can be done in several ways but at its simplest form for a digital grayscaled image if a pixel intensity in the image is lower than some thresholding constant it is replaced by a black pixel otherwise it is replaced by a white pixel or in digital form the pixel values are replaced by 0 and 255 respectively for a black and white pixel. Therefore, for a grayscaled image of shape *(m, n)*, if pixel intensity is denoted by $I_{(p,\,q)}$ where $1 \leq p \leq m$ & $1 \leq q \leq n$ and thresholding constant is T_c,

   $$\begin{gathered} If\ I_{(p,\,q)} < T_c\ then\ I_{(p,\,q)} = 0 \\ Otherwise,\ I_{(p,\,q)} = 255 \end{gathered} \tag{2.1.1.1}$$

   Figure 2.1.1.3 illustrates the result of image thresholding.

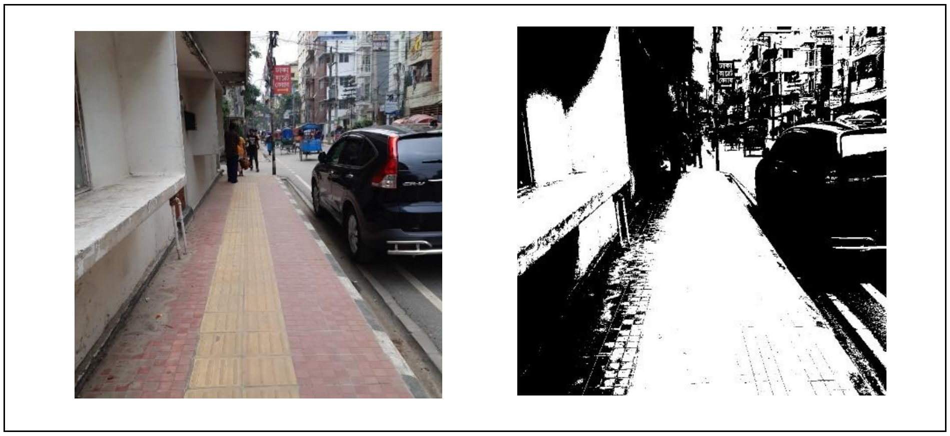
   

   Figure 2.1.1.3: Image Thresholding.

   However, thresholding techniques need to be applied to grayscaled images. To apply thresholding in an RGB image, the image first needs to be converted to a grayscale image. It turns out as our dataset contains footpath images that are not uniform in color, so the conventional thresholding technique wouldn't be that much effective here.

3. Feature-Based Clustering: In plain words, clustering is the grouping of similar types of objects. It is an unsupervised algorithm that tries to group homogeneous pixels that belong to a particular region and divergent to other regions of an image. One

of the most popular clustering technique is K-Means Clustering [11]. The algorithm classifies objects based on their attributes into K number of clusters or groups hence the name K-Means. Figure 2.2.2.3 visualizes the operation of the K-Means algorithm on a sample from the dataset when K = 3.

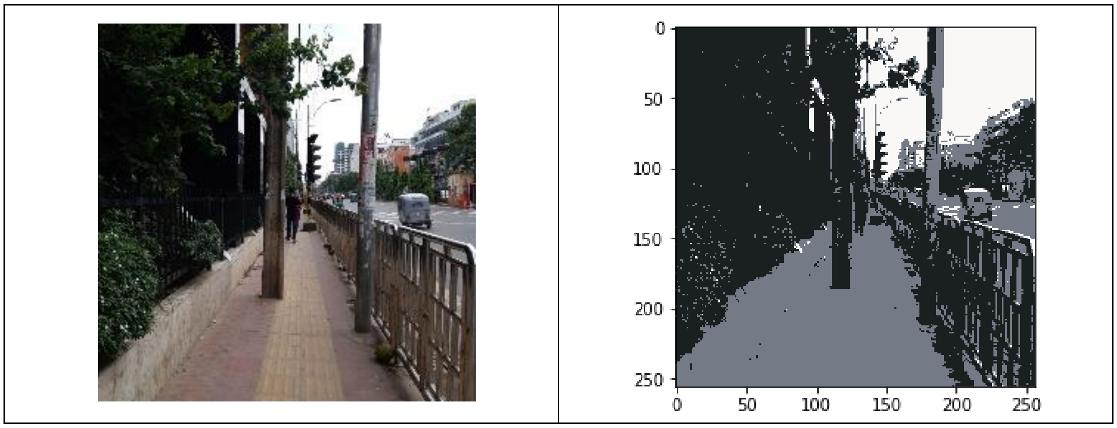


Figure 2.1.1.4: K-Means clustering

However, K-Means clustering comes with some disadvantages as well. In this algorithm, it is hard to predict the value of K and it has to be set manually. When the value of K is low, the problem can be alleviated by a few iterations with different values. But as the value of K increases determining it becomes complicated and time-consuming. There is also a known problem of outliers in this algorithm. Hence, this algorithm does not provide a sustainable solution.

4. Deep learning-based Semantic Segmentation: Deep Neural Networks (DNN) [12] have proved their robustness and proficiency in different classification tasks. The use of deep neural networks in the field of computer vision has rapidly increased. We can notice the use of DNNs in classification, object detection, semantic segmentation, etc. In a classification task, the purpose of a model is just to tell if a certain object exists or not in an image. Convolutional Neural Networks (CNN) are popular for image classification. The background of CNN is described in the following section. For an object detection model, the model draws rectangular bounding boxes around identified objects, and then it tries to figure out which known object is inside the box. As object detection is in the scope of the research, it is also discussed later. However, this procedure doesn’t provide the shape of the identified object. But as per requirement, the precise location of the footpath in an

image is required to identify walkable footpath, and the best way to do that would be semantic segmentation. Semantic segmentation is an image classification procedure that associates each pixel of an image to its respective class label. It can be portrayed as a classification task that is done at the pixel level. Semantic segmentation has been a popular technique for autonomous vehicles, robotic applications, medical imaging, etc. As it is a state-of-the-art technique for image segmentation and its robustness, accuracy, feasibility surpasses other techniques semantic segmentation is the best way to go for footpath detection. The model that has been followed is U-Net. The intuition behind both is covered in the following chapter.

2.1.2 Semantic Segmentation

Before starting with semantic segmentation, it is required to get basic intuition of CNN as CNN is the basic building block of a segmentation model.

Convolutional Neural Network (CNN/ConvNet): Convolutional Neural Network is a neural network architecture that mainly deals with data similar to the grid-like topology such as image data, time-series data where image data can be illustrated as a 2D grid and time-series data as a 1D grid [13]. The reason behind using CNNs over regular deep neural networks for image data is primarily complexity. An image digitally is a matrix where each cell of the matrix is storing pixel values. Therefore, for an image of shape 32 × 32, the input layer needs to be of size 1024 to hold all the pixel values let alone hidden layers. For an image of resolution 256 × 256, the input layer needs 65,536 neurons. It is observed how the complexity increases for regular networks. However, CNN solves this problem as they do not require the whole image as input rather the first convolutional layer is only connected to a certain number of pixels in their receptive field. In Figure 2.1.2.1 a convolutional neural network is visualized. Different layers

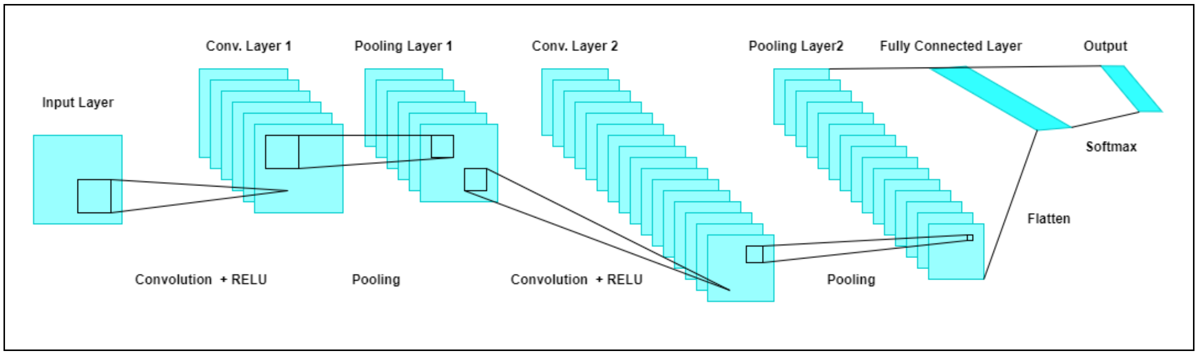


Figure 2.1.2.1: A convolutional neural network structure

CNN predominantly consisted of 3 types of layers. Namely –

1. Convolutional Layers
2. Pooling Layers
3. Fully Connected Layers

Semantic Segmentation: As in the preceding section, it is said that semantic segmentation is straightforwardly a classification task at the pixel level where every pixel in an image gets classified to a certain label. Figure 2.1.2.2 visualizes the semantic segmentation process in an image.

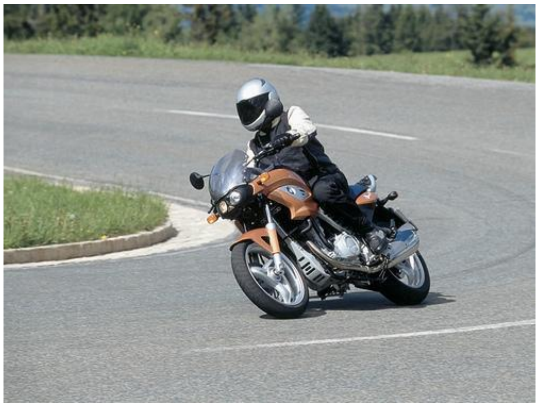
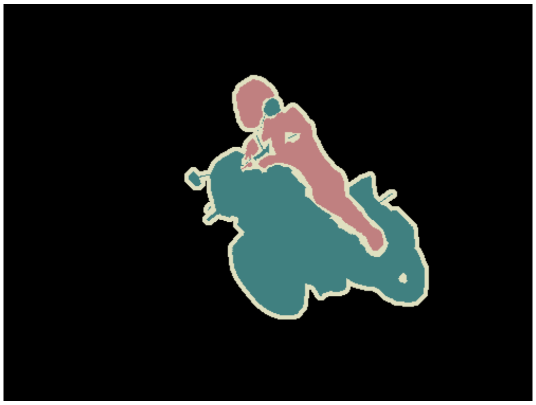
Original Image	Semantic Segmentation
	

Figure 2.1.2.2: Semantic segmentation

In the figure, it is visualized that two classes bike and person are classified and segmented but other pixels in the image are turned black as they are out of observation.

Over the past few years, it has been one of the most popular methods to classify images at the pixel level and mostly done in collaboration with deep learning. Some of the state-of-the-art networks for semantic segmentation are Fully Convolutional Network (FCN) [14], PSPNet [15], LinkNet [16], DeepLab [17], U-Net, etc. Among them, U-Net was chosen for the base model as one of the main objectives of this research is to keep the computation cost minimal. Hence after several customization and minimization of them, a modified version of U-Net with comparatively few parameters was proposed for the footpath segmentation task.

U-Net: U-Net is a state of the art model for semantic segmentation and has been quite successful in terms of medical imaging. This architecture follows an encoder-decoder model and can be illustrated as "U" shape hence it was named like that. Figure 2.1.2.3 shows the architecture of the original U-Net.

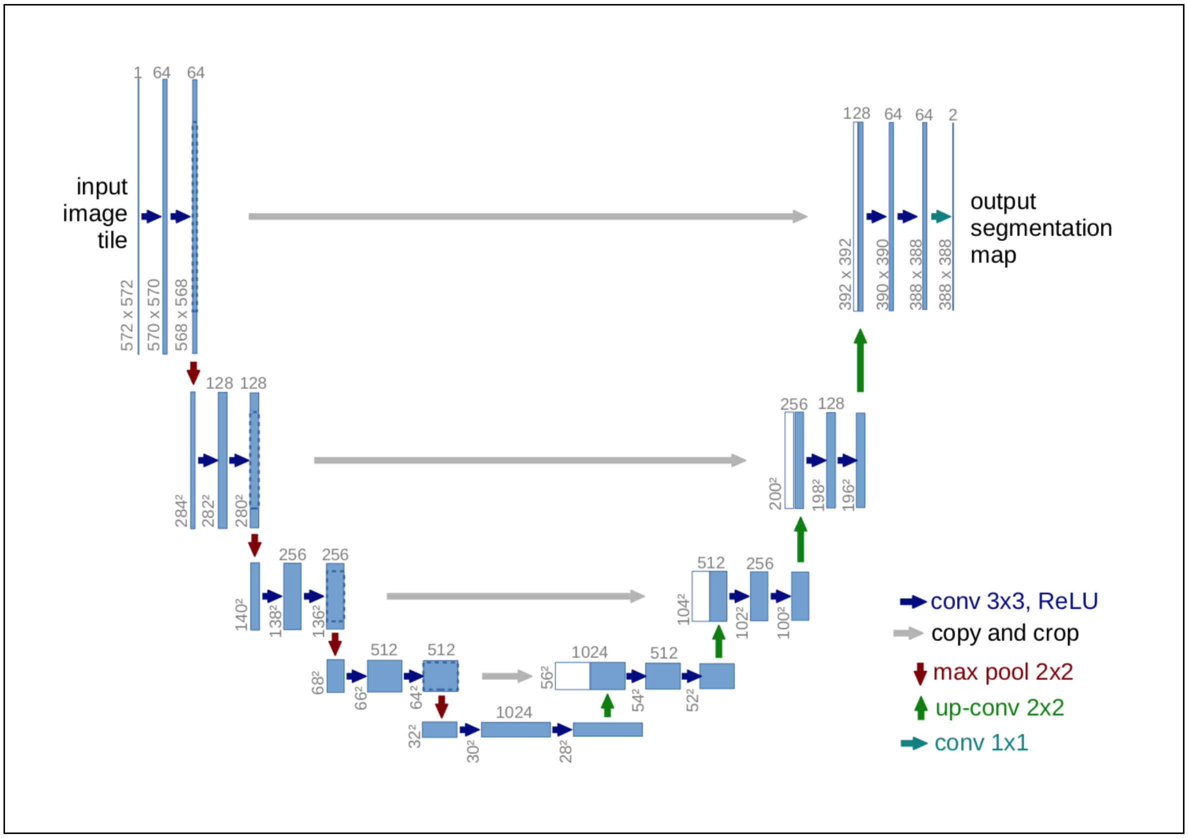


Figure 2.1.2.3: U-Net architecture

The network can be divided into two parts, one is the contracting path that downsamples an input image to high-level features by the convolution process followed by the expanding path that concatenates the outputs from the contracting paths and upsamples the image to its original size along with generating pixel-wise mask. This architecture was adapted and customized as per requirements to build a model while keeping the parameters count and loss as low as possible but increasing accuracy as high as possible. Two important concepts upsampling and downsampling are at the core of this architecture. They're discussed here:

Downsampling: Downsampling is done by the pooling layers in a convolutional neural network. These layers are used to shrink the shape of an input image to reduce computation complexity. Traditional techniques of downsampling like Max Pooling & Average Pooling ensure translational equivariance. Figure 2.1.2.4 illustrates max and average pooling operation where we can observe that like convolutional layers each neuron in a pooling layer is connected to only its receptive field in the previous layer. For max pooling, the maximum value is taken from the receptive field and for average pooling the mean value from the receptive field.

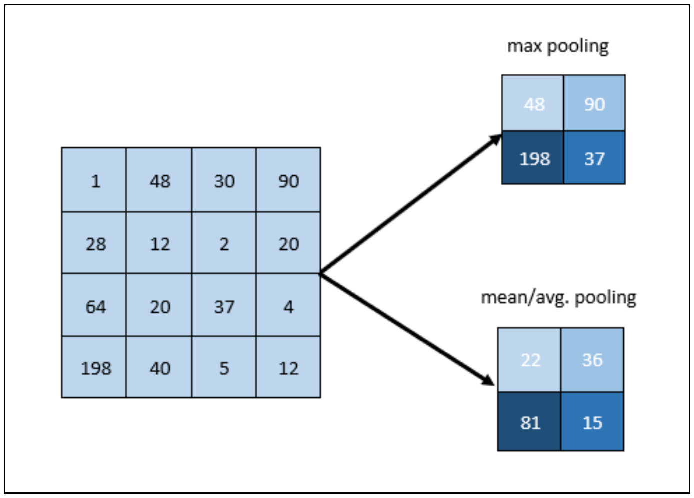


Figure 2.1.2.4: Max pooling & average pooling

Upsampling: By downsampling an image, a model learns what is in the image by reducing the feature maps. For a classification task, it is sufficient as it will only output a class label. Whether a segmentation model needs to output an image that is analogous to the input image shape with all the pixels classified. To mark an object location the necessity of upsampling comes in. It can be defined as the opposite task of downsampling. There are several methods of upsampling such as transposed convolution, unpooling, nearest-neighbor interpolation, etc. Figure 2.1.2.5 visualizes the nearest-neighbor upsampling technique in action.

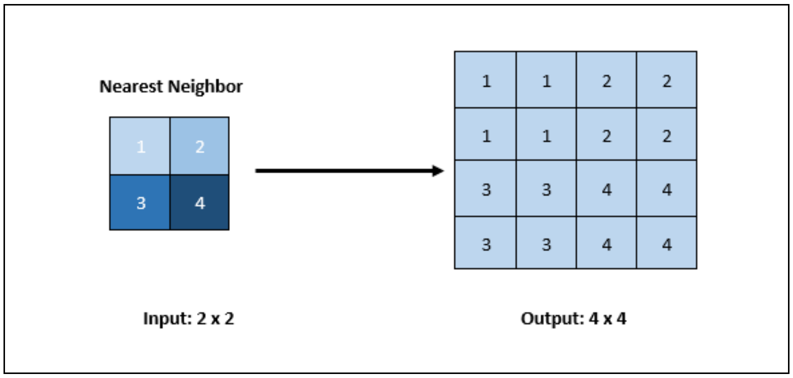


Figure 2.1.2.5: Nearest-Neighbor upsampling

Transposed Convolution: Transposed convolution is one of the preferred techniques for upsampling. Theoretically, it is the opposite of convolution and generally fabricates an output feature map whose spatial dimension is greater than of its input feature map. However, a transposed convolution does not restore original input values as in deconvolution but only the spatial dimension.

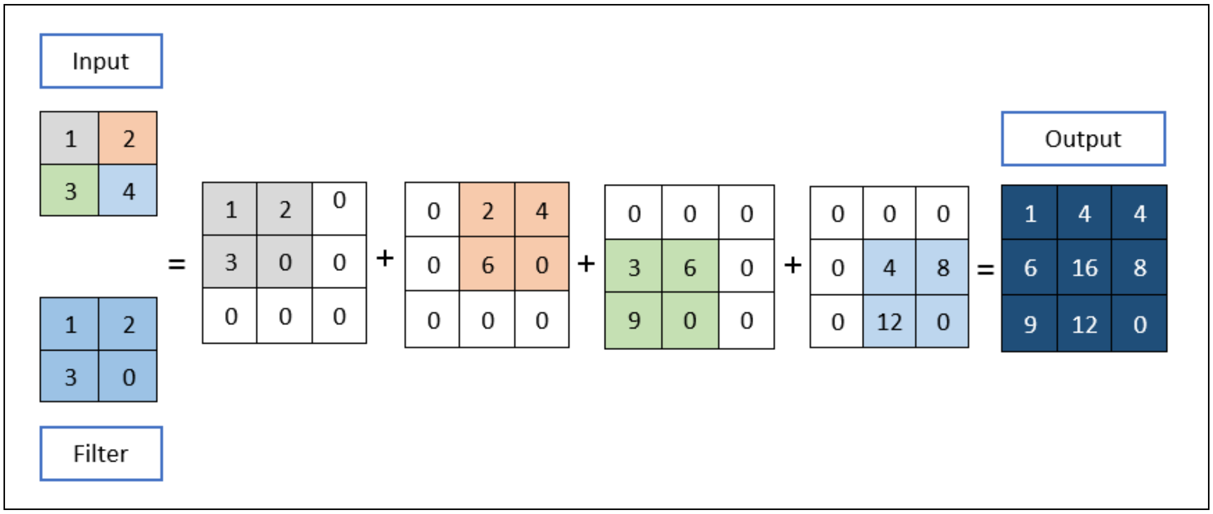


Figure 2.1.2.6: Transposed convolution operation

Figure 2.1.2.6 shows transposed convolution of a 2 × 2 input matrix by a 2 × 2 filter or kernel. Here each color in the intermediate matrix responds to the result of the transposed convolution on the same colored cell in the input matrix. Finally, the output is obtained by adding up the intermediate matrices.

2.1.3 Object Detection

Detecting obstacles in the pathway is another objective of this research. Object detection is needed to warn the user about their surroundings and avoid anything that may collide into them.

Object detection is pretty much similar to semantic segmentation as they both classifies and localize objects in an image except in semantic segmentation pixel-level classification takes place and in object detection, a bounding box or bounding box coordinates around the classified object is returned. Figure 2.1.3.6 visualizes classification, object detection, and semantic segmentation operation in the same image.

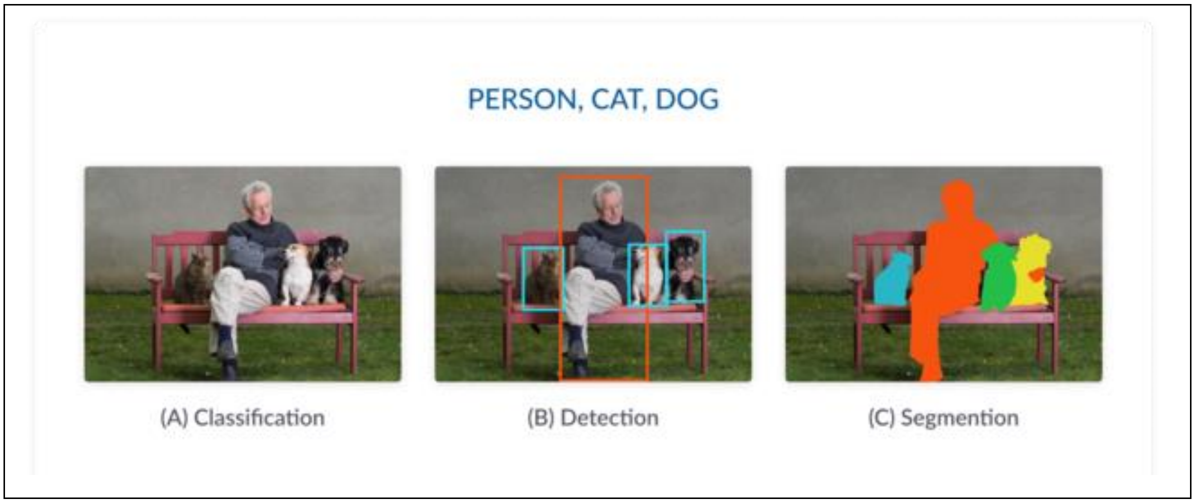


Figure 2.1.3.6: Classification, object detection & segmentation

Several popular object detection methods are Fast R-CNN [18], Faster R-CNN [19], SSD [20], YOLOv3, etc. For this research YOLOv3 (successor of YOLO & YOLOv2) is used as it is most suitable in this scenario. The main reason for using it here is its speed. It outperforms other object detection methods to a great extent and is perfect for real-time object detection when smaller objects are not of great concern. A comparative analysis of different object detection techniques is given in Table 2.3.2 to illustrate it better. A brief description of YOLOv3 is given here.

YOLOv3: It is a one-step object detection model that directly predicts bounding box coordinates (x, y, w, h where x = coordinate of the x-axis, y = coordinate of the y-axis, w = width of the bounding box, h = height of the bounding box), class labels and confidences of every prediction. This architecture is a coalition of regression and classification. Rather than considering the whole image at once, this architecture partitions the input image into

M × M sized grid. Inside every grid, n number of bounding boxes are considered where the network returns coordinates and the class probability of every box. If any class probability that is more than the threshold value is taken for final detection. Figure 2.1.3.7 illustrates the one-step object detection process of YOLOv3.

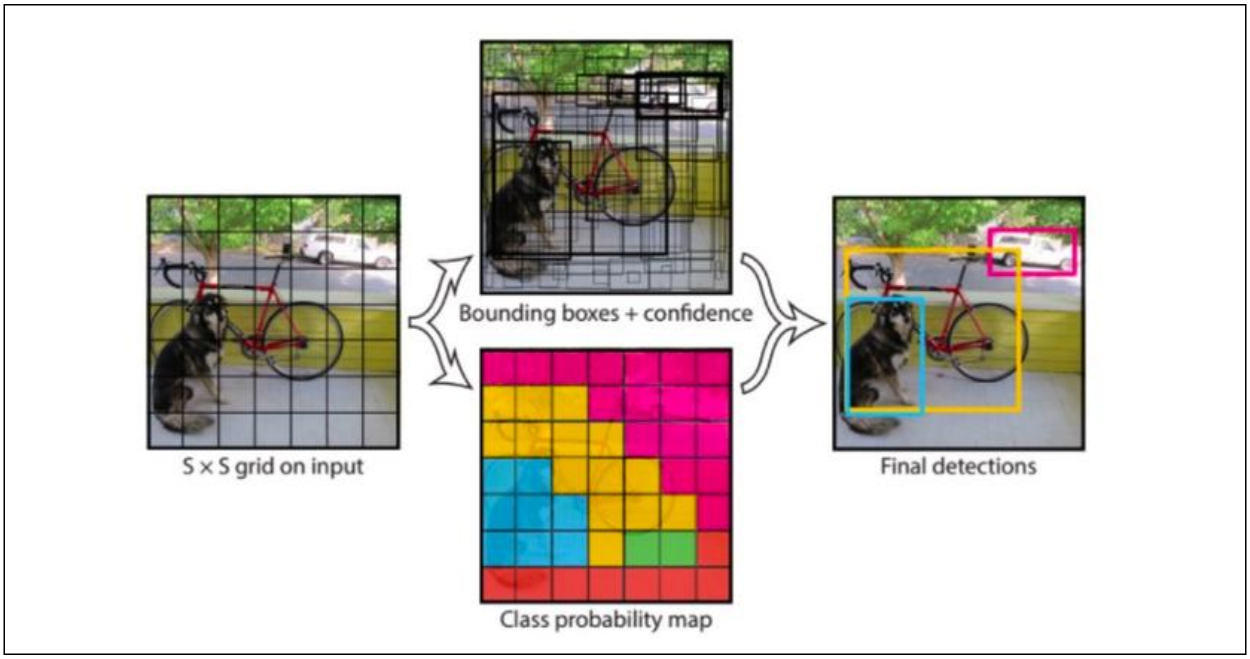


Figure 2.1.3.7: YOLOv3 object detection process

A few metrics are used for the performance evaluation of object detection models. Some of them are –

Intersection over Union (IoU): IoU is the proportion of the overlapped region and the total region of the predicted region and the ground truth value. The output always ranges from 0 to 1, the higher the better. Mathematically, it can be described as –

$$IoU = \frac{A \cap B}{A \cup B} \qquad (2.1.2.1)$$

Where, $A =$ Ground Truth value

$B =$ Predicted value

Confidence Score: Confidence score refers to the probability of a bounding box containing an object.

Precision: Precision refers to the ratio of true positive encounters and the sum of true positive and false positives. It signifies how accurately the model was capable to predict. It can be calculated by Equation 3.2.5.2.

$$Precision = \frac{TP}{TP + FP} \qquad (2.1.2.2)$$

Average Precision (AP) is a metric commonly used to evaluate the performance of an object detector. Even though the precision-recall curve can be used for this task, it can get complicated when the curves intersect. However, it can be described as the area under the interpolated precision-recall curve. It is given by –

$$Avg.Precision = \sum_{i=1}^{n-1} (r_{i+1} - r_i) P_{interpol}(r_{i+1}) \qquad (2.1.2.3)$$

$$Here\ r_1, r_{2,} \ldots, r_n\ are\ recalls\ where\ precisions\ are\ first\ interpolated$$

AP is only calculated over one class. Therefore, it is important to calculate the mean AP over all classes as most object detection algorithms have multiple classes. To compute Mean Average Precision (mAP) for N_c classes we use –

$$mAP = \frac{1}{N_c} \sum_{i=1}^{N_c} AP(i) \qquad (2.1.2.4)$$

Recall: Recall implies the proportion of the true positives and the ground truth value, which is the summation of true positives and false negatives. This metric symbolizes if the model could predict all the objects in it or not. Recall can be computed by –

$$Recall = \frac{TP}{TP + FN} \qquad (2.1.2.5)$$

2.1.4 Obstacle Distance Measurement

To measure the distance of an object by only processing images multiple techniques [21] can be followed. Such as –

1. **Stereovision:** The method utilizes two different camera sensors, where mostly one of them is a depth sensor. The depth sensor is used to measure the depth or the distance of the identified object from the camera. As it has a dedicated sensor to measure the depth it is highly accurate. But the method requires extra hardware (i.e. extra camera sensor, depth sensor, etc.) to accomplish its task. It is also complicated to set up different sensors altogether as their exists hardware compatibility and make them work. On the other hand, this procedure needs to process multiple images at once to estimate the distance of the subject which increases complexity. In the case of our proposal as we don't prefer external hardware to keep the cost minimal, therefore, stereo vision is not a preferable technique.
2. **Time of Flight Camera:** In this technique, the distance of an object is measured by the time taken of a light ray to transmit and reflected from the subject to the camera sensor. However, this biggest con of this method is that it is complex to separate the reflected signals as it contains so many constraints and a set of parameters like the property of the surface of the object, distance between the camera and the object, reflected light intensity, etc. This method is also highly affected by the surrounding environment. Thus, all these constraints make this approach very unsuitable for this research.
3. **Monovision:** Monovision generally refers to the use of a single camera to estimate the distance of an object from the camera. It is highly used on robotic vision-based systems where the distance doesn't have to be too precise rather a faster, workable solution will do. The charms of monovision techniques are that they do not require any additional hardware, calibration is easy, comparatively low computation is required for these techniques. With a little sacrifice of accuracy, good utilization can be obtained by using monovision techniques. In a previous study [22], we notice how keeping some parameters like focal length, camera position, camera angle fixed object distance can be measured by only digital image processing without any stereoscopic lens. By considering all the use-cases, we came up with a

regression-based monovision approach to estimate the distance. It is described shortly in the following section.

2.1.5 Regression-Based Distance Measurement

Regression analysis refers to some statistical procedures to draw a relationship between one dependent variable and one or more independent variables. Regression can be of many forms such as Linear Regression [23], Polynomial Regression [23], Logistic Regression [24], etc. These methods are rapidly used for prediction and forecasting tasks. The general form of most of them are given in Equation 2.1.5.1

$$Y = \varphi(X, \beta) + \epsilon \qquad (2.1.5.1)$$

Where, $Y =$ Independent variable

$X =$ Dependent variable(s)

$\varphi =$ Regression function

$\epsilon =$ Error

The concept behind using a regression model to predict distance for this case is very straightforward. We are mapping the real-world object distance into pixel distance in an image. However, it is necessary to find out a proper regression model that fits properly into the data points. The proper model for the scenario can be selected if the data points are analyzed properly. In this context, Polynomial Regression tends to work very well and fits well into the data points. In Section 3.2.2, we closely look into the dataset and discuss why the method is dominant over other regression-based models. To train the model, we used a dataset keeping several parameters fixed. It is elaborately discussed in Section 3.4.1.7. If we consider the bounding boxes as the output of the object detection model Figure 2.1.5.1 we get vertical coordinate X, horizontal coordinate Y, object height h, and width w and we keep some parameters like image resolution, the focal length of the camera, camera position, viewing angle from ground constant and consider every detected object will be touching the ground then we notice a very close relation of the real world distance and the pixel distance in the image.

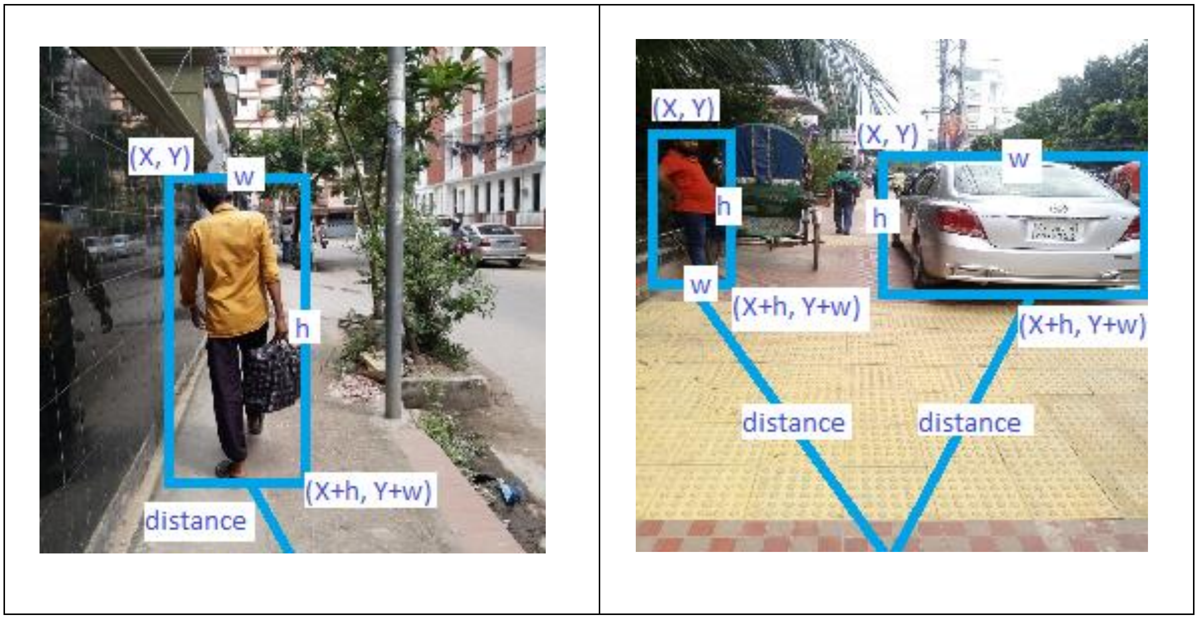


Figure 2.1.5.1: Sample images showing the pixel distance of the identified object

As in this research, only the objects detected in footpaths are under consideration it is safe to assume that the lower part of an object will always be touching the ground. Hence, we just need to calculate the pixel distance of center coordinates $(X_c = \frac{X+(X+h)}{2}, Y_c = Y + w)$ and simply map it to the real-world distance. To predict the distance, we built a dataset as well to train the regression model. The dataset has a structure given in Table 2.1.5.1.

Table 2.1.5.1: Object Detection Dataset Structure.

Dependent Variables, X		**Independent Variable, Y**
X_c	Y_c	Real-World Distance

Therefore, for each object detected on the footpath we just need to input these parameters (X_c, Y_c) returned from the object detection model to the distance model to get an estimation. The methodology along with the constraints are described in Section 3.4.1.7.

2.2 Related Works

In recent trends, significant development of computer vision and artificial intelligence can be noticed. Rapid use of deep neural networks can be observed in classification tasks to advanced vehicle technology. Whether it was inconceivable to think a car without a driver, nowadays we are seeing such artificial intelligence-driven automobiles in newspapers, TV channels, science magazines, etc. Soon enough we will start to see them in the highways. The idea of autonomous vehicles has only become possible by providing machines the ability to 'see'. Enabling computers to perceive an image or a scene made this unimaginable thing possible. Perceiving knowledge about the environment can be a lot useful in different manners. The power of vision enables a system to extract a lot of information about its surrounding. This ideology is also being used for building assistive tools for navigation to help visually impaired people. So far, several approaches have been taken using miscellaneous technologies to develop such systems.

A. Dionisi and et al. (2012) [25] used Radio Frequency Identification (RFID) transponders for localization of things. RFID utilizes radio waves to transmit data from a tag. A reader receives these signals and can help its user to make decisions. This technique works better in daylight and may help its user for indoor movement or finding things or retrieving distance information of objects that are previously tagged. But this procedure comes with a complexity of manually tagging things beforehand they can be localized. The existing database also needs to be updated as newer items are being tagged. But this type of system can't assist in outdoor navigation as a person may encounter an infinite number of things outside and it is impossible to tag them all.

In his work [26], Pradeep and et al. (2010) proposed a stereo-vision based method for the real-time navigation system for the visually impaired. His proposed method consisted of head-mounted sensors to get an observation of surroundings. The system was pretty robust as it used advanced computer vision techniques like Simultaneous localization and mapping (SLAM) for keeping location track of its user simultaneously and extracting information from an unknown environment. This system is well known as the first complete wearable system which used head-mounted sensors and provided tactile feedback. This system required special sensors and organization to work.

Lee Y.H. and Medioni G. (2015) introduced a novel system [27] using RGBD cameras for assisting the navigation of visually impaired people. RGBD sensors are mainly used for 3D mapping, localization or to build navigation systems. These sensors provide depth information along with the 3 color channels (Red, Green, and Blue) hence the name RGBD. The proposed system is composed of a smartphone User Interface (UI) to communicate with the device using audio and haptic feedback, an RGBD camera to get information about the surrounding with depth, a real-time navigation algorithm which performs Six Degree of Freedom (6-DOF) featured odometry in collaboration with the RGBD camera. This specific system was mainly developed for indoor environments and worked pretty well. While being tested it improved mobility of blindfolded users by 58% than the users who used white canes.

Intelligent walking sticks [28, 29] also have become a common trend for developing navigation systems for visually impaired people. In "An Implementation of an Intelligent Assistance System for Visually Impaired/Blind People" [28] by Chen and et al. (2019) proposed an intelligent system containing a smart walking stick, a wearable glass, mobile-based application and, an online platform. When the smart glass is worn and the smart stick is held by the user, the has the potentiality to detect obstacles and their distance. The system is capable of transmitting information to the online platform when the user falls.

Another intelligent stick for assisting blinds was introduced by Pruthvi and et al. (2019) In their work, authors introduced an embedded system which is based on Raspberry Pi (A single-board computer) connected to a camera for visual perception and ultrasonic sensor for distance measurement. For object detection, they used YOLO for faster results. The device captures the surrounding environment and retrieves information from it. Information on detected objects or any other warnings is relayed to the user by the connected earphones.

Kanwal and et al. (2015) in their research [30] presented a navigation system using camera and infrared sensors. The system detects objects by identifying corners in input images from a camera. Kinect's infrared sensors are used to estimate the depth value of the obstacles. The proposed system also recommends the safe path and tells the user when to stop or start moving to avoid obstacles in the pathway.

In this research, we propose a system that minimizes prior limitations and finds an advanced solution in the problem domain. This proposal contributes to the following sectors –

1. Building a semantic segmentation dataset for Bangladeshi footpaths.
2. Minimizes U-Net architecture to make it more efficient to use with lower parameters.
3. Finds a novel and efficient regression-based solution for object distance estimation from the camera.
4. Finally, and most importantly it shows a way to minimize the navigation problems for visually impaired.

2.3 Comparative Analysis

In this section, we will explore and analyze different algorithms. For the sake of simplicity and a better portrayal, different algorithms are grouped by their objective and are described in a tabular manner separately so that the reader can easily get an overall intuition of the models. Table 2.3.1 provides a comparative analysis of different image segmentation approaches.

Table 2.3.1: Comparative analysis of segmentation models

Approach	Features	Limitations	Evaluation	Ref
Edge Detection	It is a very simple and fast algorithm to detect object edges. The algorithm is computationally very efficient and straight-forward.	Only works well when the object edges are sharp. The algorithm hardly works in images having complex scenes. The algorithm highly depends on the pixel intensity in an image.	Not suitable for this research.	[9]

Feature Based Clustering	The algorithm groups image pixels by characteristics. It can be used when features are known. Comparatively simple to implement and easy to use when set of features is small.	Complexity proportionally increases with features. The algorithm doesn’t work well when features are unknown.	Unsuitable for the problem domain.	[11]
FCN	It only consists of convolutional layers. It can input a random sized image to produce an output of a fixed size. It uses deconvolution to upsample feature maps.	The algorithm works better than classical computer vision techniques. However, it is comparatively slow for real time use. The FPS of the algorithm is relatively slower.	Better solution needed for real time use.	[14]
U-Net	A segmentation model that follows encoder-decoder model and was developed for medical image segmentation. It requires very less data and is efficient to run on resource constraint devices. It is also useful to be used in real-time segmentation sceneris.	The model performs pretty much better than most of the previously developed techniques. However, data should be annotated properlyfor a good result.	A good solution to real-time footpath segmentation.	[4]

There are several methods for object detection some of them are mentioned in Section 2.1.3. Table 2.3.2 yields a comparison study between different object-detection models.

Table 2.3.2: Comparison between object detection models.

Model	Overview	Evaluation	Ref
Fast R-CNN	In this model a single-stage training updates inner layers. Every feature map produces a fixed layer feature vector for every region proposal by an ROI pooling layer and then fed into fully connected layers.	This model achieved 66% mAP on PASCAL VOC 2012. However, it is not suitable for real-time use.	[18]
Faster R-CNN	The model is composed of a Deep ConvNet (DCN) for region proposal and a detector (Fast R-CNN) that uses proposals from the prior ConvNet.	Faster R-CNN ahieved 73.2% mAP at 7 FPS which is pretty good but not enough.	[19]
SSD	SSD is a very fast object detector that uses only one network to detect objects in an image unlike Region Proposal Networks (RPN). It is relatively simple This algorithm aggregates predictions from different feature maps which allows it to detect objects of various sizes.	On PASCAL VOC 2007 SSD gained 74.3% mAP for images with 300×300 resolution. Simillarly, for 512×512 images it achieved 76.9% mAP.	[20]
YOLO	The YOLO algorithm is very quick and precise to detect objects. At the time of writing this paper, it has 3 versions namely YOLOv1/v2/v3. It divides an input image in multiple grids and uses a single neural network to predict objects. It's almost 1000x and 100x faster than R-CNN and Faster R-CNN respectively. It is comparatively lightweight that makes it enable to use in low processing powered devices.	On COCO dataset YOLO achieved 28.2 mAP in 22ms where SSD obtained 28 mAP in 61ms. Even though the model sometimes struggles to detect objects of tiny shape, its computation cost, speed, and accuracy make it highly feasible to develop a real-time object detection system. The most remarkable thing about YOLO is that it can also work very efficiently with little or no external GPU support.	[5]

The technical report of YOLOv3 also provides a comparison chart of performance between different models. It is adopted here in Table 2.3.3 to give the reader a better idea about the algorithm.

Table 2.3.3: YOLO comparison chart.

Dataset	Model	mAP	Time (milliseconds)
COCO	SSD 321	28	61
	DSSD 321	28	85
	R-FCN	29.9	85
	SSD 513	31.2	125
	DSSD 513	33.2	156
	FPN FRCN	36.2	172
	RetinaNet 50 (500)	32.5	73
	RetinaNet 101 (500)	34.4	90
	RetinaNet 101 (800)	37.8	198
	YOLOv3 320	28.2	22
	YOLOv3 416	31	29
	YOLOv3 608	33	51

2.4 Scope of the Problem

The research is intended for figuring out the problems a blind person may face while walking in the footpath and coming up with an intelligent solution that may help them while walking around. In this study, it is conjectured that building such a real-time system is complicated when the resource is very limited. However, the current study may not solve the problem domain perfectly but it proposes a very efficient system to aid the visually impaired that could be of great social impact and it enables a new pathway to build similar systems for the problem domain.

2.5 Challenges

Every study faces many challenges in its lifecycle. This one is not an exception either. While conducting this research in each stage we faced many challenges that had to be dealt with very cautious and strategically. Some notable challenges that were faced during this study are as follows –

1. Problem Formulation: Formulating problem domain is the primary but very crucial step to conduct any research. It was a very thought-provoking task to find out the topic.

2. Data Preparation: Preparing dataset is the very first and probably the most important step to prepare any machine learning/deep learning algorithm as the better the data, the better the result. For this research, we needed different kinds of footpath images but there was no available dataset. Therefore, we built the first annotated footpath dataset for Dhaka city and annotated it. Another dataset was also prepared for distance measurement. The dataset was diversified enough to recognize several types of footpaths.
3. Selection of the Best Suited Method: As this research lies under the field of computer vision, each problem it intends to solve can be solved in different ways. Nevertheless, the pivotal task is to find the best-suited method after analyzing other approaches that are may be available. Hence, for each problem, we studied several methods and picked the one with the best performance. Methods from classical computer vision to state-of-the-art deep learning architectures are analyzed before side-by-side in terms of performance and feasibility before final selection.
4. Model Training: As the research highly involves deep neural networks, it requires a lot of costly computational resources to train it properly. We had to figure out the right platform to train and evaluate different models.

CHAPTER 3

RESEARCH METHODOLOGY

This chapter portrays a comprehensive and thorough description of the study. As the research serves multiple purposes each methodology is explained separately for the convenience of the reader. To solve a problem in this research each of the methods was chosen after a comparison study between several methods considering their accuracy, reliability, and feasibility. Previous works are taken into account as well as their limitation. The whole workflow was divided into 4 different where each stage contains multiple tasks to be done hierarchically. This segmentation made the whole task easier to carry out. Figure 3.1.1 illustrates the overall workflow of the project.

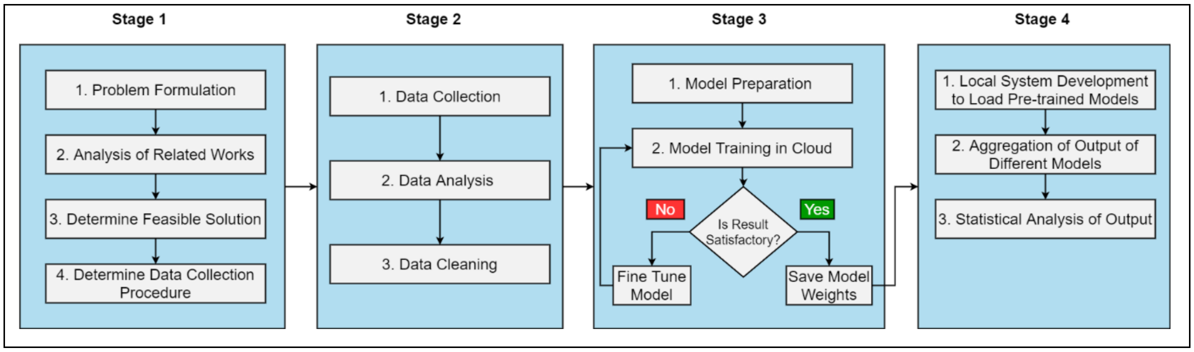

Stage 1
1. Problem Formulation
2. Analysis of Related Works
3. Determine Feasible Solution
4. Determine Data Collection Procedure
Stage 2
1. Data Collection
2. Data Analysis
3. Data Cleaning
Stage 3
1. Model Preparation
2. Model Training in Cloud
Is Result Satisfactory?
No
Yes
Fine Tune Model
Save Model Weights
Stage 4
1. Local System Development to Load Pre-trained Models
2. Aggregation of Output of Different Models
3. Statistical Analysis of Output

Figure 3.1: The methodological workflow of the research.

Throughout this whole paper, each stage was explained elaborately for a better assessment of the reader. Stage 1 which is mostly correlated to background study and finding a feasible solution was described in Chapter 2. In the rest of this chapter, stage 2 to 4 will be discussed explicitly.

3.1 Research Subject and Instrumentation

The sole purpose of this research is to develop a system that may assist visually impaired people to navigate with a bit of ease. This research covers several objectives like building a dataset for footpath images of Dhaka, proposing a deep neural network segmentation model for footpath identification, object detection, estimating detected objects distance from the camera which is in the pathway, and a guide to avoiding obstacles safely. One of the principal goals of this research is to fulfill each objective only by image processing rather than using additional hardware as economic feasibility is a great concern because we want this system to be accessible by anyone who needs it regardless of their pecuniary condition. However, it would be also worth mentioning that the system may also come useful for robotic vision as it is capable of identifying safe footpath regions from images along with its other objectives. The proposed system is highly dependent on different computer vision and image processing techniques. Therefore, the datasets we've built or used here mainly consist of images or numeric data extracted from images. To collect the data, different cameras and a fixed-length tripod were used. Figure 3.1.1 (a) visualizes a mobile phone mounted on a tripod which was used to capture photos to build the datasets and (b) illustrates the measures taken to index the real-world distances of reference objects.

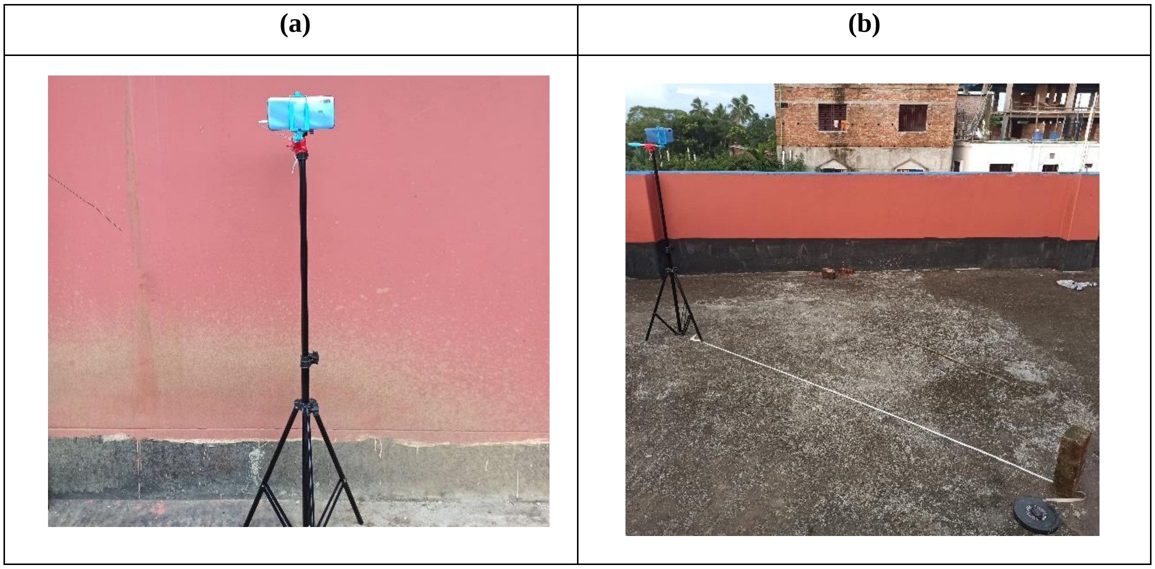

(a) **(b)**

Figure 3.1.1: Data Collection Instruments

3.2 Data Collection Procedures

At the time of writing this paper, there is no known dataset with Bangladeshi footpath images that can be used for semantic segmentation. Therefore, building a dataset with footpath images of Dhaka was the only option ahead. Multiple mobile cameras were used to capture footpath images in different areas of Dhaka city. Image samples were mainly collected from Shukrabad, Jigatola, Dhanmondi R/A, Dhanmondi-27/32, Muhammadpur, Mirpur-1/2/6/7/10/11/12, Mirpur DOHS, Rupnagar, Shyamoli, and Farmgate area. The data collection phase started in July 2019 and ended in early September of 2019. To keep the data symmetric in the dataset the photo resolution was kept 1:1 for most of the photos. The preferred resolution was either 720 × 720 (0.52 MP) or 1088 × 1088 (1.18 MP). However, all of them were converted into 256 × 256 later. This phenomenon is elaborately explained in section 3.2.2. On the other hand, for distance estimation, we utilized a regression-based approach. Hence another dataset was also prepared. This dataset contains model objects pixel co-ordinates and real-life distance in numeric format. However, each of the numeric data was extracted from a reference digital image. The reference images are also taken from the same camera keeping resolution 720 × 720 and later converted to 256 × 256 as the segmentation dataset. The procedure is shortly described in Section 3.2.2.

3.2.1 Semantic Segmentation Data Collection & Preparation

We combined the whole data collection & preparation for the segmentation process in a few steps. The workflow can be visualized by Figure 3.2.1.1.

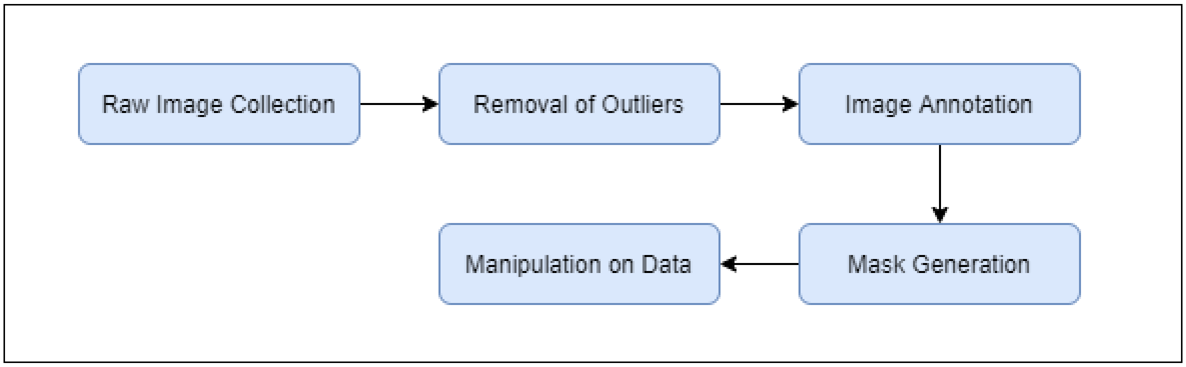


Figure 3.2.1.1: Segmentation data collection & preparation workflow

Each of the steps illustrated in the figure is shortly described in the following paragraphs.

3.2.1.1 Raw Image Collection

As per the requirement of an image database, the very first step is to collect images of the working domain. It should be mentioned that the footpaths of Dhaka city are very asymmetric by looks, condition, or even in width. That being the case it was pretty much unrealistic to capture photos of only one area and leave others behind as the model may work better on the dataset but it will not provide any good result or probably it will fail in the real-world scenario. So, to make the model robust and pragmatic several places of Dhaka city were selected and photos were taken accordingly to make the dataset versatile. The versatility in the dataset also helps the model to be tolerant of different cases. A vast number of samples were needed to make the model work better. Around 3,200 photos were taken from different places in Dhaka which contained different types of footpaths.

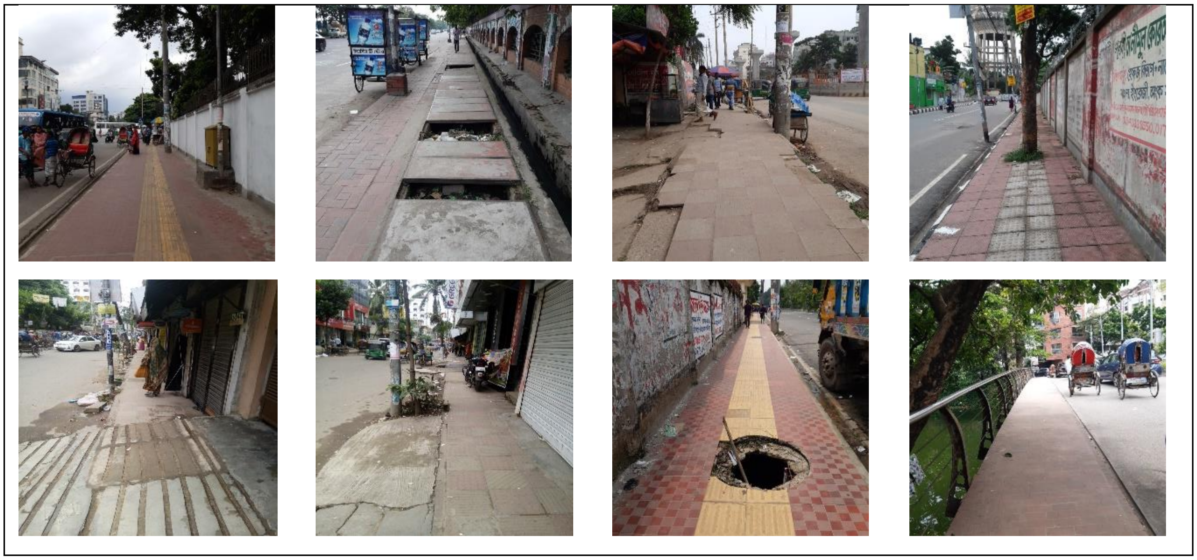


Figure-3.2.1.1.1: Different image samples from the dataset.

From Figure 3.2.1.1.1, it is observable that footpaths in the selected region are very diversified. They vary in color, shape, building materials, and conditions as well. Hence, it is a very challenging task to make a system to recognize all these variants and identify walkable pathways. Classical computer vision techniques would fail for a variegated dataset like this as symmetricity of data is very important for those techniques to work. Having all these criteria in mind, it was only feasible to devise a deep learning model as recently they have shown their expertise in image classification or localization tasks like this.

3.2.1.2 Image Cleaning

Data cleaning refers to the process where inaccurate, faulty, or irrelevant data get removed or corrected. In this case, even though cleaning raw images is very trivial there as each photo was taken very carefully and there is very little that could be done after detecting faulty data but it is always beneficial to have a glimpse at the whole dataset if possible. Such is advised as for training a neural network model as faulty, incorrect, or inconsistent data may lead a network to wrong prediction which is very much unintended. Therefore, the whole dataset was revisited thoroughly and 3,000 of 3,200 images were taken. For cleaning purpose following steps were taken:

1. **Removing outlier images:** Sometimes unexpected data can be found in a dataset that doesn't belong to the data domain. Those data are referred to as outliers. There is an exigent need for removing outliers to retrieve the best result from the proposed model. Hence the whole dataset was manually checked for such images or outliers. In the dataset, some random images were detected and removed.
2. **Removing analogous observations:** Some images in the dataset were too analogous to another sample as if they were duplicate of the other. Such samples may trigger overfitting in the model. Hence, it was necessary to clean such instances. The dataset was checked meticulously as much as possible and if two samples with this much similarity were found, one of them was removed from the dataset.
3. **Reshaping Images:** Different devices with different resolutions were used to capture all the images. Hence, they were asymmetric in shape. It is indispensable that all the images in the same batch should have the same shape while training a convolutional neural network model. Therefore, keeping the resource available for training and the feasibility in mind, resolution of 256 × 256 was determined and all the instances were resized into this shape before further processing.

3.2.1.3 Image Annotation

Image annotation is the procedure of assigning labels to an image. In other words, assigning metadata to image data where metadata reveals some useful information about the image. To make a computer perceive images or extract information from it at first it is necessary to provide it such labeled or annotated images are given to it so that it would learn eventually. The more annotated data is provided, the more a machine can learn thus becoming artificially intelligent. To put it simply, image annotation is a way of telling a computer what is in an image, or which part of the image is what. There are different annotation techniques as well as tools that are available for annotating images. Some of the most popular methods are bounding box annotation, polygon annotation, semantic segmentation, etc. In this case, as the proposed system contains a semantic segmentation model, obviously a similar annotation process is required to carry out the task.

As the objective of this research is to recognize safe footpath artificially, it is required to annotate the footpath region from the images so that computers may learn which portion of an image is a safely walkable footpath and which part is not. Each of the samples in the dataset was hand-annotated by the researchers. The metadata of each image was saved in JSON files for later use which contain the region of footpath in an image. For annotation Labelme [31], a graphical user interface tool was used to manually label images. The operation of annotating was carried out very carefully and rigorously to keep the error as minimal as possible.

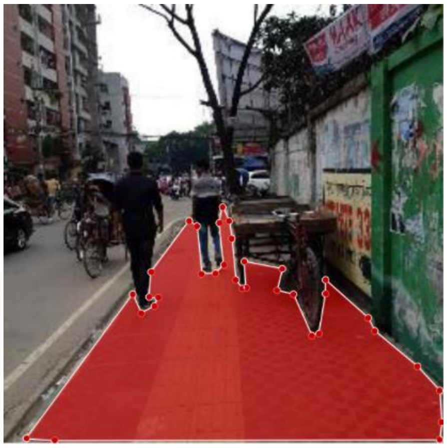
Original Image	**Annotated Image**
	

Figure-3.2.1.3.1: Image before and after annotation.

Figure-3.2.1.3.1 visualizes an instance from the dataset where an image is shown before and after hand labeling.

3.2.1.4 Mask Generation

Image masking refers to the task of making some portion of an image disjunctive to the rest of the image. One way of image masking is setting the pixel values of the unnecessary region to null, which is cutting off the region of the image that is not going to be used for the observation or experiment. It is done so that the system may learn which pixels or region of an image contains useful data and which region contains none. Pixel wise masking provides a granular understanding of the image to the system.

In this research, masking is done in a way so that the segmentation model has the insight of ground truth value with which it can compare its prediction. Each image in the dataset has a ground truth or mask image related to them. These masked images are generated from the annotation metadata (JSON files) generated in the prior data preparation process. Masked images are of similar shape that is (256 × 256) and grayscaled. Each pixel is either black (minimum intensity of 0) if they are outside of the footpath region in an image or white (maximum intensity of 255) if they are inside the footpath region. A python script was used to read annotated JSON files from the dataset and generate corresponding masks of the image. Figure 3.2.1.4.1 should provide an intuition about the masking images.

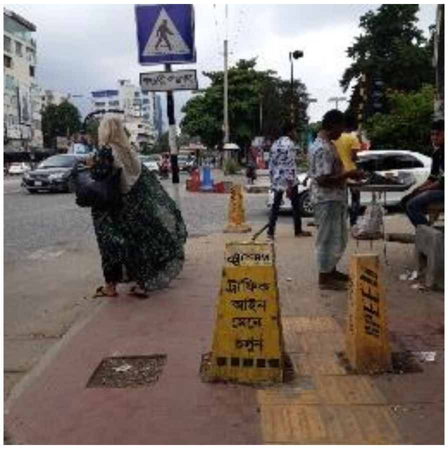
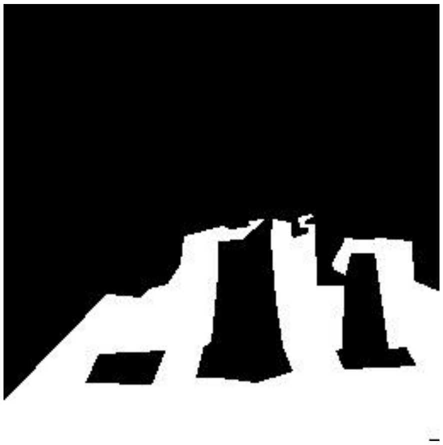
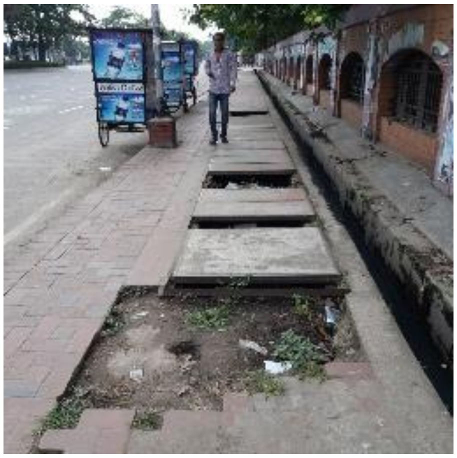
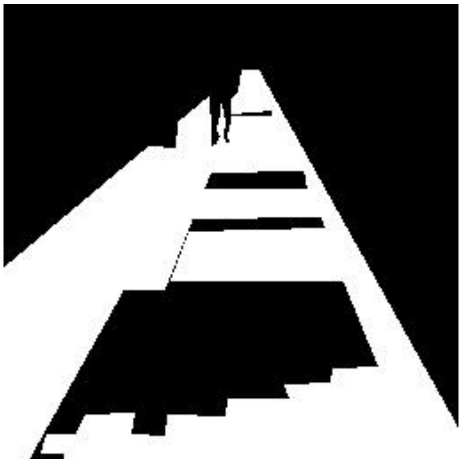
Original Image	**Masked Image**
	
	

Figure-3.2.1.4.1: Images before and after masking.

These mask images will help in training as they will let the model know that white regions have maximum weight in contrast the black has the minimum.

3.2.1.5 Image Manipulation

It is often found that manipulating dataset images may improve a neural network model in various ways. A lot of tests have been done on our model to improve its performance and accuracy. Different image manipulation techniques like rotation, sharpening, blurring, etc. have been applied on the dataset and we noticed the model tends to work better in both training and validation if all the images in the dataset are trained along with their blurred mimics at different factor.

Image Blurring: Image blurring is a technique to smoothing images so that the edges in an image are less observed. For blurring an image a kernel or filter is used which is also known as a low-pass filter. What this filter does is, it lets low-frequency pixels to go through and blocks high-frequency pixels. Here frequency simply alludes to the alteration of pixel values. In digital images, around object edges, pixel values vary to a great extent thus forms high-frequency. However, as much as we can block this high-frequency the more blur an image gets.

To create blur images of the original images in the dataset different kernels (K_N) of shape (N, N) were used. Equation 3.2.1.5.1 explains a general form of the kernel used –

$$K_N = \frac{1}{N \times N}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad (3.2.1.5.1)$$

Where, $2 \leq N < 6$

Equation 3.2.5.1 shows that as the number of N increases the image gets more blurred. The value of N was randomly picked to transform an image. These blurred images were included in the dataset before training. Point to be noted that for the blurred images mask of the original images were used as the ground truth value. Figure 3.2.5.1 shows several samples from the dataset along with their blurred copies and masks. Different level of blurring of data samples is observed in the given figure.

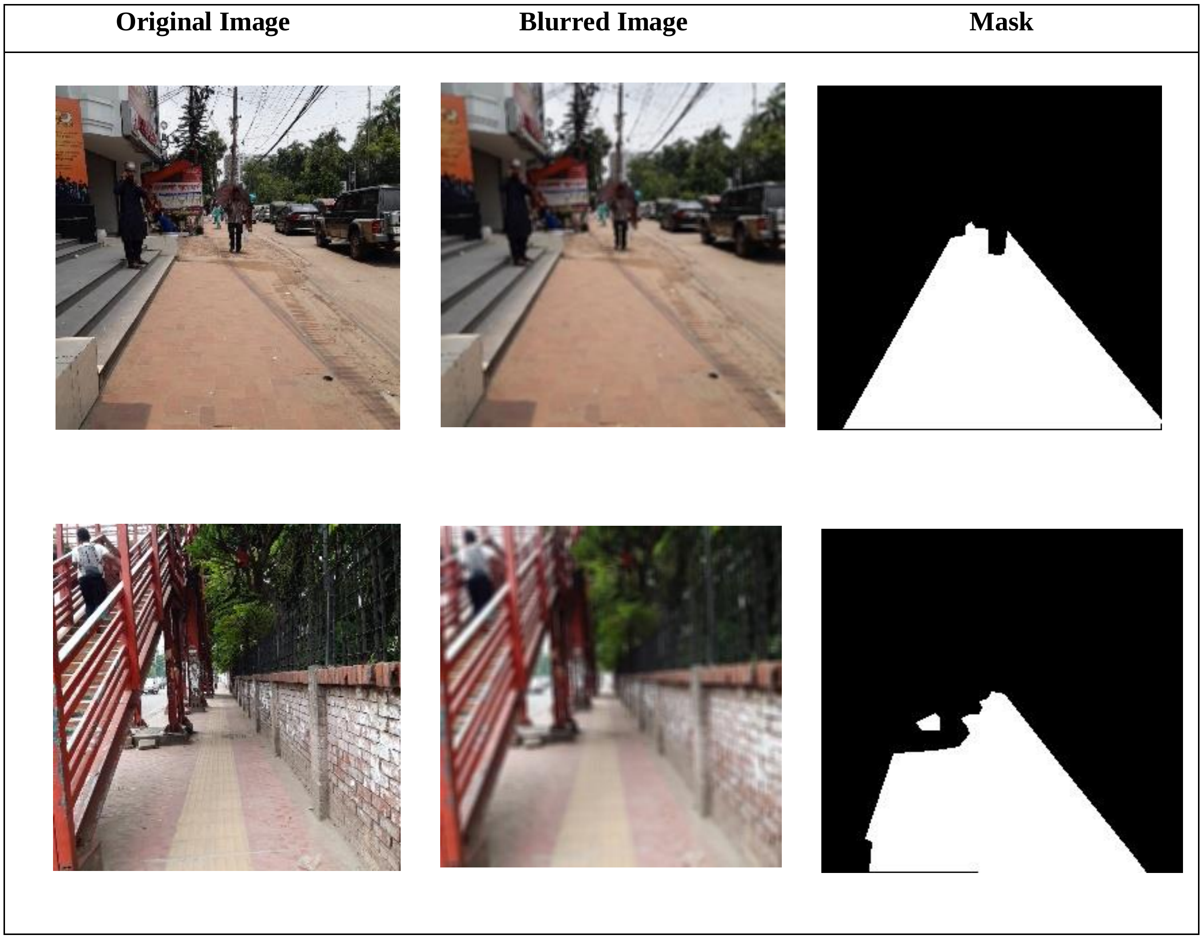

Original Image | Blurred Image | Mask

Figure-3.2.1.5.1: Original image, blurred version, and corresponding mask.

3.2.2 Distance Estimation Data Collection & Preparation

Compared to the preparation of the segmentation dataset, it was easy to work on the distance estimation dataset as it doesn't require images from different places. The workflow of the whole procedure is given in Figure 3.2.2.1.

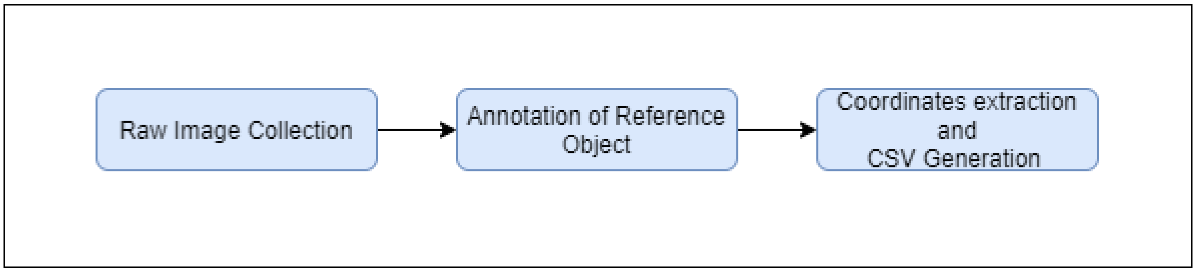

Raw Image Collection
Annotation of Reference Object
Coordinates extraction and CSV Generation

Figure 3.2.2.1: Workflow of distance estimation dataset preparation.

We shall explore these steps in brief in the following section as some of the procedures are already introduced in Section 3.2.1.

3.2.2.1 Raw Image Collection

This step involves capturing images of reference objects. We followed a simple method for this. Setting the camera on a tripod at the height of 50 inches from the ground, images of the reference object were taken. Parallelly, the real distance of the object was indexed. Keeping the camera fixed in the same position we changed the position of the reference object as many times as necessary and continued the procedure. Thus, completes the raw image collection step. Figure 3.2.2.1.1 illustrates a reference object while being captured -

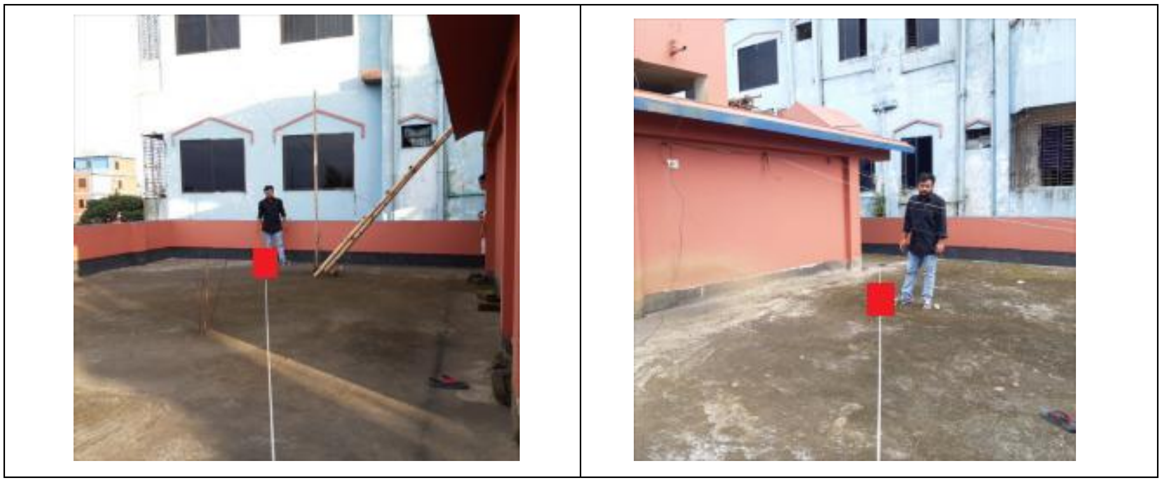


Figure 3.2.2.1.1: Raw image collection of the reference object

Figure 3.2.2.1.1 illustrates a few samples from the dataset where the reference object can be seen in different positions of the image frame. The red rectangles in both photos are symbolizing reference objects. Both of the rectangles are created graphically over the actual object for better illustration.

3.2.2.2 Annotation of Reference Objects

To find the pixel distance, the coordinates of the reference object is needed to be found. The best way to figure that is by annotating the object. Hence, similar to the footpath annotation technique we used manual hand labeling to annotate images. This dataset is simpler as the reference object is only a rectangle-shaped object and each image has only one object in it. Figure 3.2.2.2.1 visualizes an annotated sample.

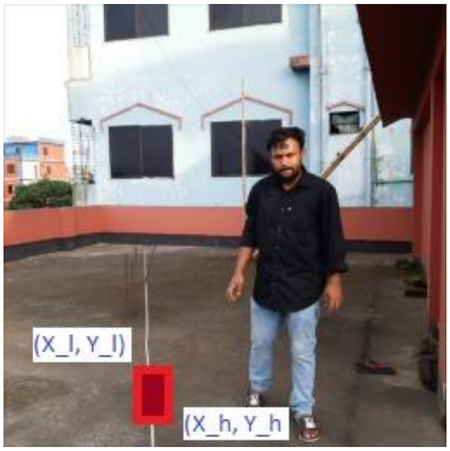
Original Image	**Annotated Image**
	

Figure 3.2.2.2.1: Annotation of reference objects

The annotation data was saved into JSON files. Each metadata files contain two vital coordinates information. The lower pixel coordinates (X_l, Y_l) and higher coordinates (X_h, Y_h) of the reference object as shown in Figure 3.2.2.2.1.

3.2.2.3 Extraction of Ground Coordinates

As per the proposal, we're using only one coordinate of an object to estimate the distance between it and the camera. However, objects can be of different sizes and shapes. It is difficult to tell the exact distance of an object in an image from the camera without any specialized hardware such as - depth sensor. As the problem domain is only concerned about objects in the footpath, we figured most of the obstacles (such as – humans, parked vehicles, animals, etc.) will always be on the ground. Hence, to estimate the distance of any object lying in the footpath we just need to calculate the point distance of the center point of the lower base of the object bounding box. Based on this theoretical knowledge, to apply this in practice a regression model is needed to be trained with relevant data. As the annotation files contain (X_l, Y_l) and (X_h, Y_h) coordinates of the bounding boxes, it is needed to calculate the lower center points before indexing each data points in the CSV file. The center coordinates can be easily calculated by the Equation 3.2.2.2.1.

$$X_c = \frac{X_l + X_h}{2} \qquad (3.2.2.2.1)$$
$$Y_c = Y_h$$

This equation can easily be realized by referring to the annotated reference object in Figure 3.2.2.2.1.

3.3 Dataset Analysis

This research consists of two datasets. Each of them is prepared after following multiple procedures. In this section, we will briefly look at the dataset definitions and some relative analysis.

3.3.1 Segmentation Dataset

The processed dataset contains 6,000 RGB images along with the same number of ground-truth mask images. Table 3.3.1.1 provides a summarized overview of the dataset.

Table 3.3.1.1: Segmentation dataset definition

Data Type	Quantity	Color Channels	Resolution (For Images)	Total Size	Format
Raw data	3,200	3 (RGB)	720 × 720, 1200 × 1200, 960 × 1280	1.88 GB	JPG
Selected sample	3,000	3 (RGB)	720 × 720, 1200 × 1200, 960 × 1280	1.75 GB	JPG
Resized sample	3,000	3 (RGB)	256 × 256	117 MB	JPG
Annotation files	2,962	NULL	NULL	77.5 MB	JSON
Processed image	6,000	3 (RGB)	256 × 256	190 MB	JPG
Ground-truth masks	6,000	1 (B&W)	256 × 256	46.2 MB	JPG

3.3.2 Distance Estimation Dataset

In the initial procedures, the dataset has a definition as given in Table 3.3.2.1.

Table 3.3.2.1: Primary form of distance estimation dataset.

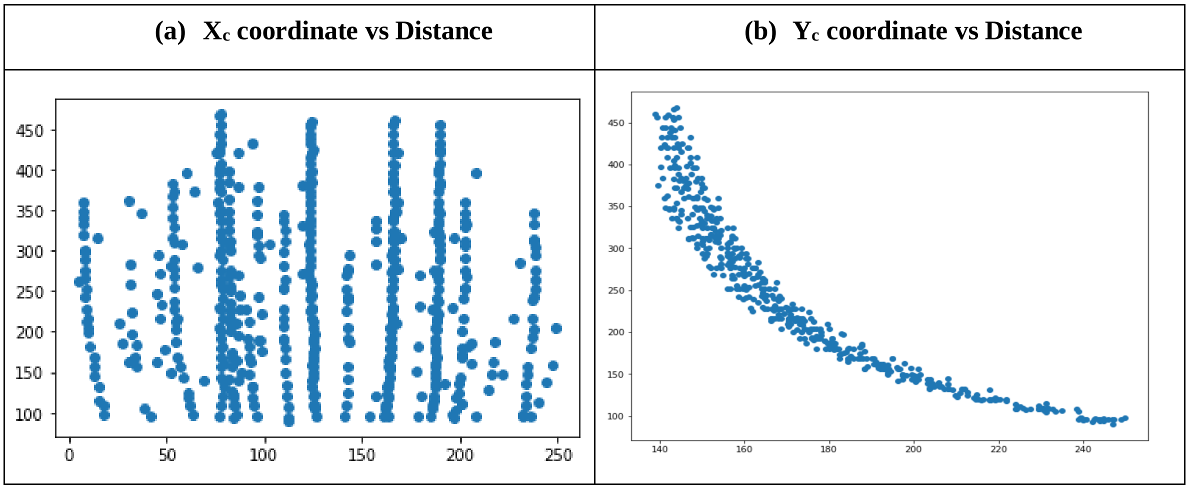
Data Type	Quantity	Color Channels	Resolution (For Images)	Total Size	Format
Raw data	500	3 (RGB)	720 × 720	400 MB	JPG
Resized sample	500	3 (RGB)	256 × 256	10 MB	JPG
Annotation files	500	NULL	NULL	14 MB	JSON

However, for convenience of use to train the regression model important data from annotation files were extracted to make a CSV dataset. The final dataset definition can be found in Table 3.3.2.2.

Table 3.3.2.2: Final Form of Distance Estimation Dataset.

Field	# of Entry	Data Type	Description
filename	500	String	This field contains the filename of the reference image.
x_center		Float	This field contains the horizontal pixel coordinate of the reference object's ground center point.
y_center		Float	This field contains the vertical pixel coordinate of the reference object's ground center point.
dist		Integer	The real-world distance of the reference object's center point from the camera in inches.

With the general idea obtained from the dataset definition now let's consider the scatter plots plotted by data points in Figure 3.3.2.1.



(a) X_c coordinate vs Distance

(b) Y_c coordinate vs Distance

Figure 3.3.2.1: Scatter Plots of X, Y Coordinates & Distance.

The scatter plots illustrate different observations of the reference object's distance from the camera w.r.t. X_c and Y_c pixel coordinates in the image. Here, the vertical axis in both figures is representing the distance in inches and the horizontal axis representing X or Y coordinate depending on the figure. If we look closely at the figures it can be observed that we can not deduce any important information from Plot (a) as it is simply plotting different data points in a scattered manner. It is expected as different points in the horizontal axis of an image can have the same distance. However, in Plot (b) we can easily observe the relation of the vertical axis and the distance. It is figured that as Y_c increases, the distance decreases maintaining a flow. It is worth mentioning that the reason behind this negative relation is that unlike the conventional coordinate system, for an image of shape (M, N) we indicate the top-left pixel as (0, 0) and the bottom-right pixel as (M, N). This plot also reveals the non-linearity of the data points as well. It can be easily concluded that a linear model won't work well for this dataset as a straight line will not properly fit into the data points. Therefore, it is clear that linear regression will not be a proper solution rather polynomial regression will work better in the scenario.

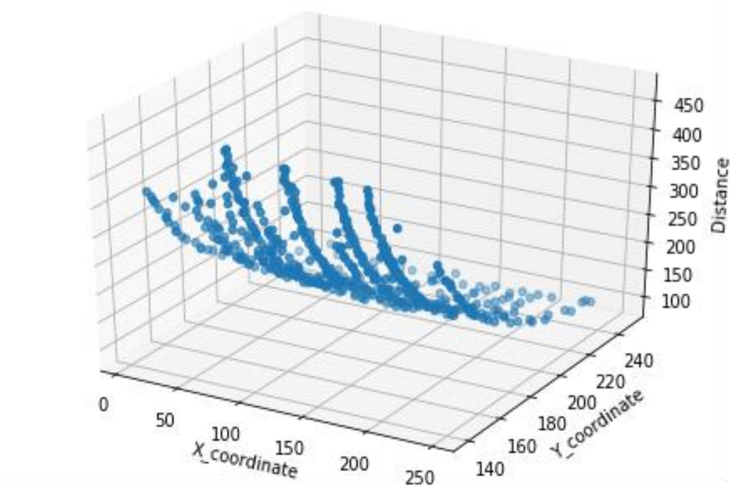


Figure 3.3.2.2: 3D Scatter Plot of the Data Points.

We can observe the relation of distance and the coordinates more vividly in Figure 3.3.2.2 where the data points are plotted in 3D. This plot represents the relation between the object distance and its coordinates. This discussion enlightens the dominance of Y coordinate in distance estimation. This analogy will be proved by Pearson r [32] correlation analysis shortly. It is a common method to appraise the relationship between two variables. It is given by the Equation 3.3.2.1.

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}} \qquad (3.3.2.1)$$

Where n = number of pairs

x, y = variables

The resultant value ranges $-1 \leq r \leq 1$, where -1 represents a perfect negative relationship and +1 represents a perfect positive relationship. For any $r > 0$, it represents that if one variable increases or decreases the other one tends to increase or decrease respectively as well. In contrast, for any $r < 0$, the relationship works oppositely.

After applying this equation for X and Y coordinates and respective distances separately in the dataset we get, $P_Y = -0.91$, $P_X = -0.04$.

We can observe the degree of correlation of Y coordinate to the distance from these coefficients as the value of P_Y is very close to -1.

3.4 Proposed Methodology

In Chapter 2, we studied several approaches for a problem domain to solve each of them and proposed the most feasible solution. This section encloses the encyclopedic explication of each method separately. The workflow diagram given in Figure 3.1.1 signifies that stage 3 and stage 4 are training and evaluating phases respectively. Both of them are explained separately for a better portrayal.

3.4.1 Training Phase

The training phase is further divided into several segments as the research requires multiple models for different purposes. The models were trained either in local machines or in cloud-platform like Google Colaboratory [33] (a cloud platform that provides free RAM, GPU, and storage for research) if there was not enough computational resource for training.

3.4.1.1 Segmentation Model

The U-Net architecture is adopted for the semantic segmentation of footpath in an image. Several modifications were done to reduce computation complexity as well as to obtain high accuracy in different metrics. The model is composed of end-to-end ConvNets and is divided into two parts namely - encoder and decoder. Figure 3.4.1.1.1 visualizes the whole architecture of the model. The encoder part in the diagram is responsible for extracting features from an image. The encoder part of the model is built by using several downsampling blocks in series. The shape of the feature map gradually decreases as consecutive convolution and pooling operation takes place in downsampling blocks. However, it is worth mentioning that, each convolution block in this model is made up of 2 consecutive convolution operations. In contrast in the encoder part, the image size gets restored to the input size along with the localization of classified pixels by consecutive transposed convolution operations in the upsampling blocks. Skip connections [34] are used by concatenating with the result of transposed convolution operation to get a better result in localization. Green arrows in Figure 3.4.1.1.1 are symbolizing skip connections.

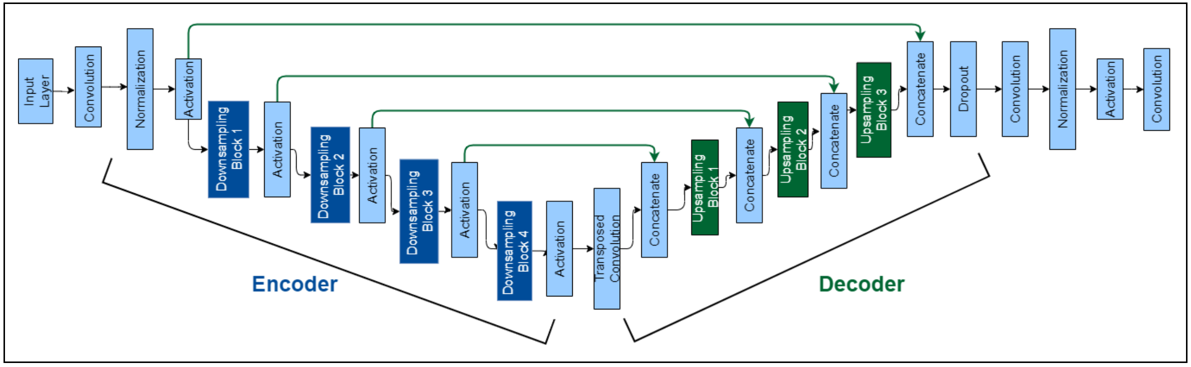


Figure 3.4.1.1.1: Proposed segmentation model architecture

Each downsampling and upsampling blocks are analogous to the other blocks of the same class. These blocks are composed of other sub-blocks. Figure 3.4.1.1.2 shows the structure of these blocks.

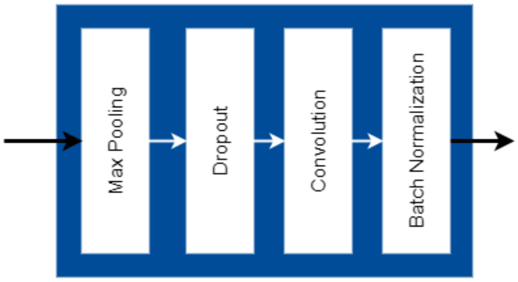
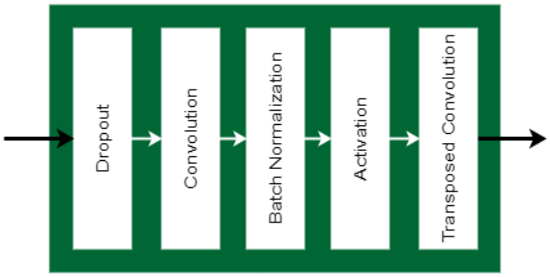
Downsampling Block	Upsampling Block
 Max Pooling Dropout Convolution Batch Normalization	 Dropout Convolution Batch Normalization Activation Transposed Convolution

Figure 3.4.1.1.2: Illustration of downsampling & upsampling block

3.4.1.2 Downsampling Block

Inside each downsampling block, several pivotal operations are carried out.

Max Pooling: Background theory of max pooling has already been discussed in the background chapter. After each convolution block. max pooling is used to reduce the feature map. Each pooling layer contains a pooling window of size (2, 2).

Dropout: Neural networks tend to overfit the model frequently. An approach to solve this problem is turning random neurons off during training. It is known as dropout [35]. By dropping of nodes also makes the network computationally faster as any incoming and outgoing connections on dropped nodes are also not considered in training. It is a very potent technique to reduce overfitting problem along with generalization error. Thus in this model dropout of 0.1 is used.

Activation Functions: Rectified Linear Unit (ReLU) is used as the activation of the convolution layers as it tends to work better with ConvNets. It is interpreted as –

$$y = \max(0, x) \tag{3.4.1.2.1}$$

That is, ReLU takes the bigger number between 0 and the input so any negative number that goes through it instantaneously becomes 0. This function is very faster as it has no complicated math. ReLU also gets rid of the vanishing gradient problem. On the other hand, the sigmoid function is used as the activation function of the output layer as it is a binary classification task. The function is given by –

$$\emptyset(x) = \frac{1}{1 + e^{-x}} \tag{3.4.1.2.2}$$

The sigmoid function always returns the output in the range of [0, 1]. It signifies a pixel's probability to fell in a certain class.

Convolution: Convolution operation has been elucidated in the background chapter. The first convolution block starts with 32 filters. As for each downsampling block onward, the number of filters increases by 2 times. As the convolution layers use the ReLU activation function, we used "He" [36] initializer because it tends to work better with ReLU. A 3 × 3 kernel with "SAME" padding is used.

Batch Normalization: As it is discussed earlier, a deep neural network always works better and converges faster on normalized data hence the dataset was normalized. However, inside a neural network during backpropagation, each layer fine-tunes their weights & biases independently to minimize the loss. But the distribution for each layer might end up shifting. Batch normalization [37] reduces this shift and speeds up the process of training. Batch normalization has been applied between every hidden layer of the model. As this paper explained, batch normalization can be calculated by the Equation 3.4.1.2.3.

$$\widehat{x_\iota} = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \tag{3.4.1.2.3}$$

Where, $x =$ the activation values over a mini batch β

$\sigma_{\beta}^{2} =$ variance of the mini batch

$\mu_{\beta} =$ mini batch mean

3.4.1.3 Upsampling Block

Upsampling block takedowns almost similar operations as downsampling blocks in a reverse manner except for transposed convolution. The output of each upsampling block is concatenated with the output of layers from downsampling blocks by skip connections to produce an input shape output with better precision in localization.

Transposed Convolution: The background theory of transposed convolution is explained in the Background chapter. It is the core operation of the upsampling block. The first transposed convolution starts with 256 ($32 \times mul\_factor$, where $mul\_factor = 8$) filters. For each consecutive transposed convolution the factor reduces by 2 times. Identical to the convolution kernel, it has also a 3 × 3 kernel with "SAME" padding but with a (2, 2) stride.

3.4.1.4 Proposed Segmentation Model Implementation

The model has several skip connections where some layers feed their output to some other layer skipping some intermediate layers. We used the Keras [38] functional API for implementation for convenience as it provides robust features for implementing directed acyclic graphs. One of the key findings while training is that converting the dataset images into grayscale speeds up the training a few times faster without losing accuracy mark. Therefore, each image was converted to grayscale before feeding into the network. Several other techniques have been applied to implement in the training phase. Some of them are explained here.

Dataset Normalization: An intuition of training any deep learning model is that deep neural networks tend to work better for a normalized dataset. That being the case, each pixel value P ranging from 0 to 255 was normalized into 0 to 1 by using Equation 3.4.1.4.1.

$$P_{normalized} = \frac{P - P_{min}}{P_{max} - P_{min}} \qquad (3.4.1.4.1)$$

Optimizer: Optimizers are responsible for fine-tuning weight values in a neural network so that it can minimize the loss function. We used Adam [39], an adaptive gradient algorithm as the optimizer of the model. It is very much well suited for image-based models for its proficiency in problem domains with sparse gradients and requires a little memory.

Metrics: Several metrics are used for the model evaluation. Such as – accuracy, IoU, IoU threshold, dice coefficients, loss, precision, recall, etc. For loss calculation, binary cross-entropy/log loss is used as loss function because the objective of this model is to classify a pixel if its footpath or not. It can be calculated by Equation 3.4.1.4.2.

$$J(w) = -\frac{1}{N}\sum_{n=1}^{N}[ylog\widehat{y_n} + (1- \ y)\log{(1-\widehat{y_n})}] \qquad (3.4.1.4.2)$$

Where, $N =$ number of classes

$y =$ binary indicator for correct classification

$\widehat{y_n} =$ predicted probability

Training: With all these procedures done, the model is prepared to train. The model has an input and output shape of (256 × 256 × 1) as the processed dataset is grayscaled images of 256 × 256 resolution and the output is a pixel-wise mask of prediction. It was trained in Google Colaboratory as it requires a good amount of computing resources. We used early stopping so that the model may not overfit. The learning rate of the model is reduced by a factor of 0.1 up to minimum learning rate of 1×10^{-5} on plateau. Table 3.4.1.4.1 illustrates a summarized report of the training of the segmentation model. However, proper visualization and analysis of the training result are given in chapter 4.

Table 3.4.1.4.1: Segmentation Model Training Summary.

Train Size	Test Size	Batch Size	Epochs	Time Taken	RAM Used	GPU Used	Disk Used	Accuracy
4,800 (80%)	1,200 (20%)	15	49	2 Hrs	12 GB	16 GB	32 GB	**96%**

Figure 3.4.1.4.1 visualizes the train vs validation learning curve and loss curve of the model.

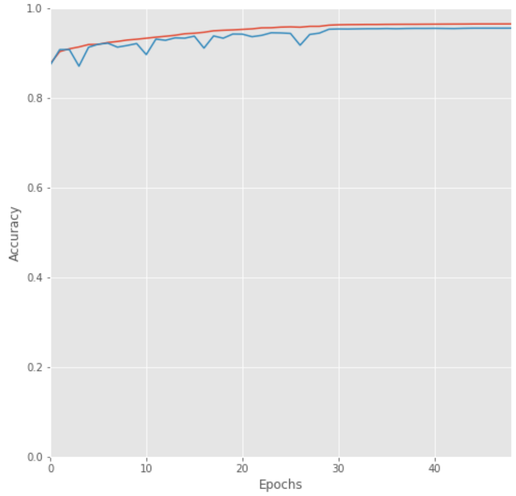
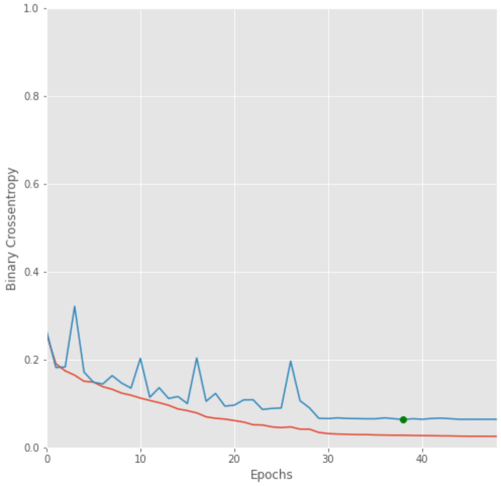
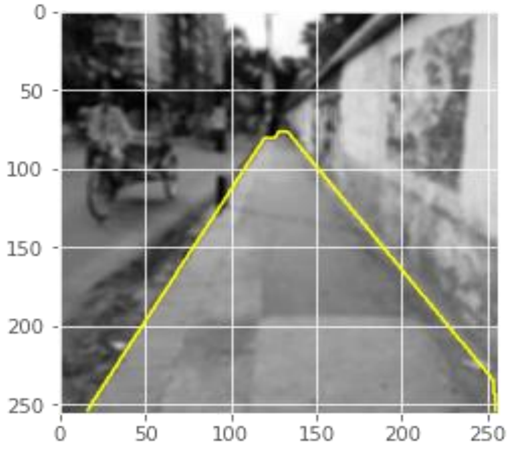
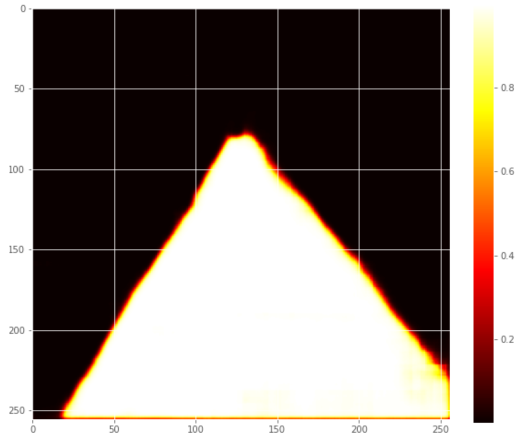
Learning Curve	Loss Curve
	

Figure 3.4.1.4.1: Train vs Validation Learning Curve and Loss Curve.

The curves demonstrate the convergence of the training. A very little fluctuation between the training and validation ensures the model's fitness.

A few predictions on the trained model can be observed in Figure 3.4.1.4.2.

Original Image	Prediction
	

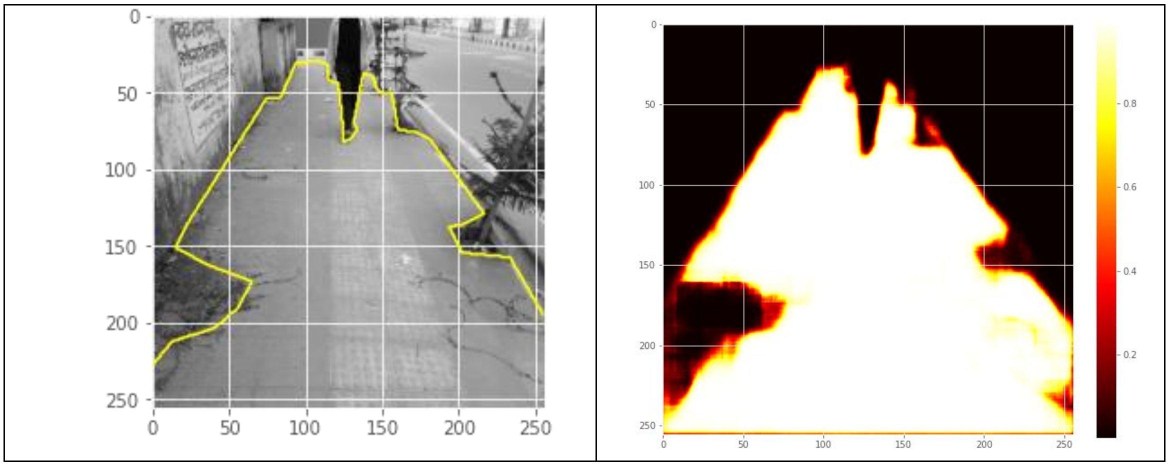


Figure 3.4.1.4.2: Predictions on trained model

3.4.1.5 Object Detection Model

The system adopts the YOLOv3 framework for object detection as mentioned earlier. Nevertheless, a question can be raised that why to use a separate object detection framework while semantic segmentation itself can do the work. There are two main reasons behind that. They are –

1. **Multi-label Annotation Dataset:** To build a segmentation model the first requirement is an annotated dataset. As there was no suitable footpath dataset for this research, we built one. However, annotating the dataset for multiclass segmentation is a very rigorous, labor-intensive, and time-consuming procedure. The segmentation model may become very heavy as for the need for more filters which is not desirable and feasible here. Thus it is easier to use an existing framework that can detect the required objects faster.
2. **Distance Measurement Technique:** The distance prediction technique used in this research uses the bounding-box coordinates of detected objects returned by the object detection framework to predict an object's distance. It helped to build such a system that doesn't require any additional hardware for distance measurement with a little loss. It is also very fast and efficient to use. The technique is broadly described in section 3.4.1.7.

3.4.1.6 Object Detection Model Preparation

There is no necessity of training the model as the pre-trained weights [40] on the COCO dataset can be used for object detection and it has all the necessary classes of attraction. The model is trained with 80 classes and 330K images. However, not all of the 80 classes are needed in our system. Therefore, we only kept classes that could be necessary while walking in a footpath and dropped others for the sake of simplicity and faster detection. Classes of consideration can be found in Table 3.4.1.6.1.

Table 3.4.1.6.1: Classes of Consideration for Object Detection.

Classes of Consideration						
person	bicycle	car	motorbike	bus	train	traffic light
fire hydrant	stop sign	bench	bird	cat	dog	cow
backpack	umbrella	handbag	suitcase	chair	cell phone	

We integrated OpenCV's [41] Deep Neural Network (DNN) module to load the weights of YOLOv3 and get it to work very lucidly. Its CPU implementation is extremely fast compared to other Neural Network (NN) architectures like Darknet [42] running on CPU. As it is already described, the algorithm yields bounding boxes with a confidence score assigned for each. Primarily, boxes with lower confidence scores get discarded. The remainder boxes go through Non-Maximum Suppression (NMS) which takes care of bounding boxes that are overlapped. The procedure only lets a box to pass if it scores more than the suppression threshold. NMS algorithm is as following –

Algorithm:

```
    Bounding boxes, B <= {$b_1$, $b_2$, …, $b_n$}
    Confidence, C <= {$c_1$, $c_2$, …, $c_n$}
    T <= Threshold Value
    Proposal list, P <= {}
    WHILE $B \neq Ø$
        m <= argmax(S)
        M <= $b_m$
        P <= $P \cup M$
        B <= B – M
    FOR EACH $b_i$ IN B
        IF IOU(M, $b_i$) ≥ T THEN
            B <= B - $b_i$
            C <= C - $c_i$
```

The algorithm returns filtered proposals P and their corresponding confidences C.

3.4.1.7 Object Distance Estimation Model Preparation

We discussed in the Background chapter that for a monovision approach it is a common practice to keep several parameters constant. This research is not an exception either. Several parameters were kept constant for a better result as the whole process is highly dependent on image processing. Table 3.4.1.7.1 tracks several parameter constants –

Table 3.4.1.7.1: Parameter Constants.

Parameter	Value
Captured Image Resolution	720 × 720
Converted Image Resolution	256 × 256
Focal Length	3.92mm
Aperture	1.7
Camera Height above Ground	50″
Angle of View	90° to the ground

Keeping the parameters constant all that is left is to take some reference image, compute the distance of reference objects real-world distance. However, we have already done that in the data preparation section. As described earlier, the dataset contains the center point of the reference object's ground coordinates and real-world distance of the objects from the camera in the CSV dataset.

With this, the only thing left is to extract these data from the dataset and train the model. For the training, polynomial linear regression was adopted as the dependent variable (D_o, the real-world distance of the object) in the dataset has a non-linear relationship with the independent variables (X_c, Y_c pixel coordinates of the reference object) which has already been observed in Section 3.2.2.

The equation for multivariate linear regression can be given by Equation 3.4.1.7.1.

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_n X^n + \epsilon \qquad (3.4.1.7.1)$$

Where, $X =$ Independent variable

$Y =$ Dependent variable

$\varepsilon =$ Random error

$n =$ Degree

In our case, the degree of the polynomial is 2.

The cost function of the regression model is given in Equation 3.4.1.7.2.

$$Cost\ function, J = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y^{i})^{2} \qquad (3.4.1.7.2)$$

Where, $\hat{y} =$ Prediction value

$y^{i} =$ Real Value

$m =$ Number of observations

The gradient descent algorithm is used to minimize the cost function. The parameters are updated by the gradient descent algorithm by the Equation 3.4.1.7.3 –

$$w_k = w_k - \alpha \frac{\partial}{\partial w_k} J(w_k) \qquad (3.4.1.7.3)$$

Where, $\alpha =$ Learning rate

$w_k =$ parameter for feature x_k

The model was trained locally. The dataset was split by 4:1 for train and test data respectively. The training summary can be found in Table 3.4.1.7.1.

Table 3.4.1.7.2: Object distance estimation model training summary

Train Size	**Test Size**	**Time Taken**	**RAM Used**	**GPU Used**	**Accuracy**
400 (80%)	100 (20%)	< 2 Mins	< 100 MB	NULL	94%

The model fits with –

Intercept: 2200.49

Coefficients: 0.00, -3.0609^{-1}, -1.8294^{01}, -4.1425^{-4}, 1.774^{-3}, 3.9805^{-2}

3.4.2 Evaluation Phase

Integrating all the pre-trained models into a workable system is the first step of the evaluation phase. The class diagram given in figure 3.4.2.1 should provide an overview of the system implementation.

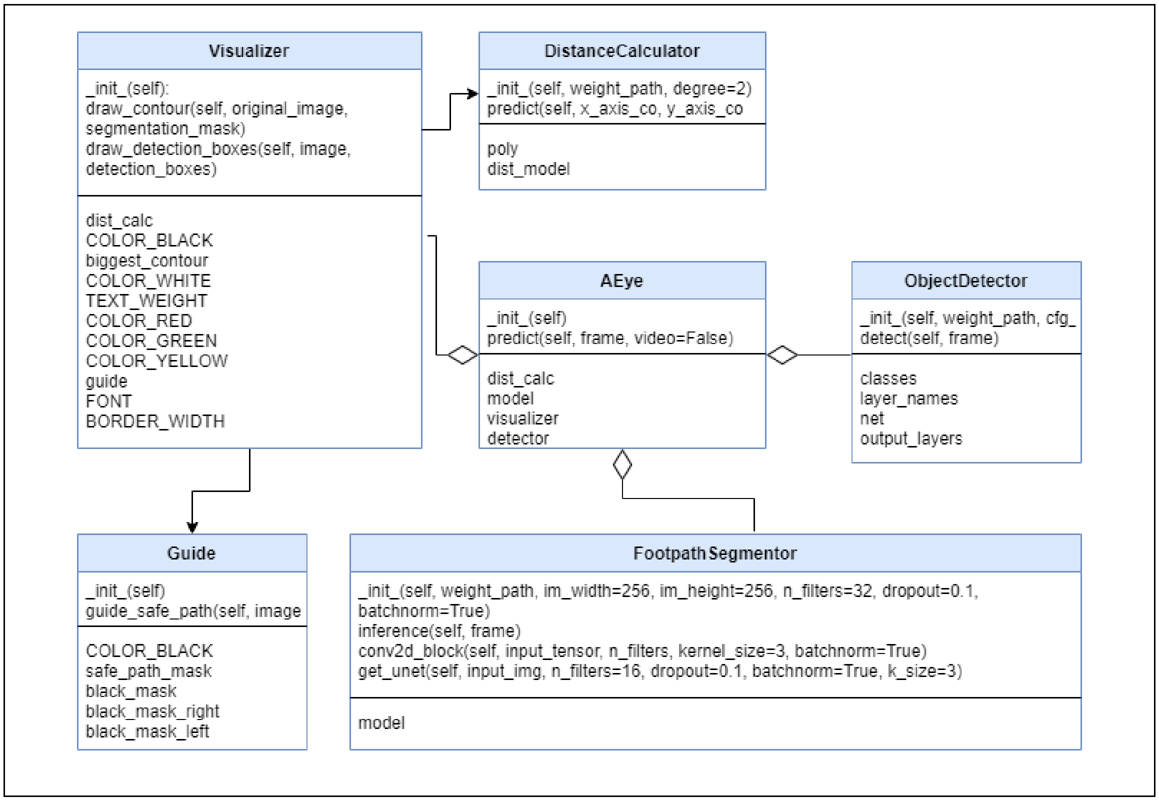

Visualizer
init(self):
draw_contour(self, original_image, segmentation_mask)
draw_detection_boxes(self, image, detection_boxes)
dist_calc
COLOR_BLACK
biggest_contour
COLOR_WHITE
TEXT_WEIGHT
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
guide
FONT
BORDER_WIDTH
DistanceCalculator
init(self, weight_path, degree=2)
predict(self, x_axis_co, y_axis_co
poly
dist_model
AEye
init(self)
predict(self, frame, video=False)
dist_calc
model
visualizer
detector
ObjectDetector
init(self, weight_path, cfg_
detect(self, frame)
classes
layer_names
net
output_layers
Guide
init(self)
guide_safe_path(self, image
COLOR_BLACK
safe_path_mask
black_mask
black_mask_right
black_mask_left
FootpathSegmentor
init(self, weight_path, im_width=256, im_height=256, n_filters=32, dropout=0.1, batchnorm=True)
inference(self, frame)
conv2d_block(self, input_tensor, n_filters, kernel_size=3, batchnorm=True)
get_unet(self, input_img, n_filters=16, dropout=0.1, batchnorm=True, k_size=3)
model

Figure 3.4.2.1: UML class diagram of the implemented system.

3.4.2.1 System Integration

Figure 3.4.2.1 suggests that the overall system maintains modularity as different classes are being used for different purposes. Previously trained segmentation model, object detection model, distance measurement models are brought into play in FootpathSegmentor, ObjectDetector, and DistanceCalculator classes respectively. The Main class is composed of those 3 different classes that load pre-trained models into memory every time the system starts. After that, each time it is fed with an image it makes 3 copies of the image frame of which 2 are sent to the segmentation model and the object detection model, and 1 is kept to the Main class for combining all the results. Finally, after combining inferred results from different classes the final frame is sent to the Guide class which in turn after processing the image guides the user about the decision it makes. The simple pipeline of the system's processing is illustrated in Figure 3.4.2.2.

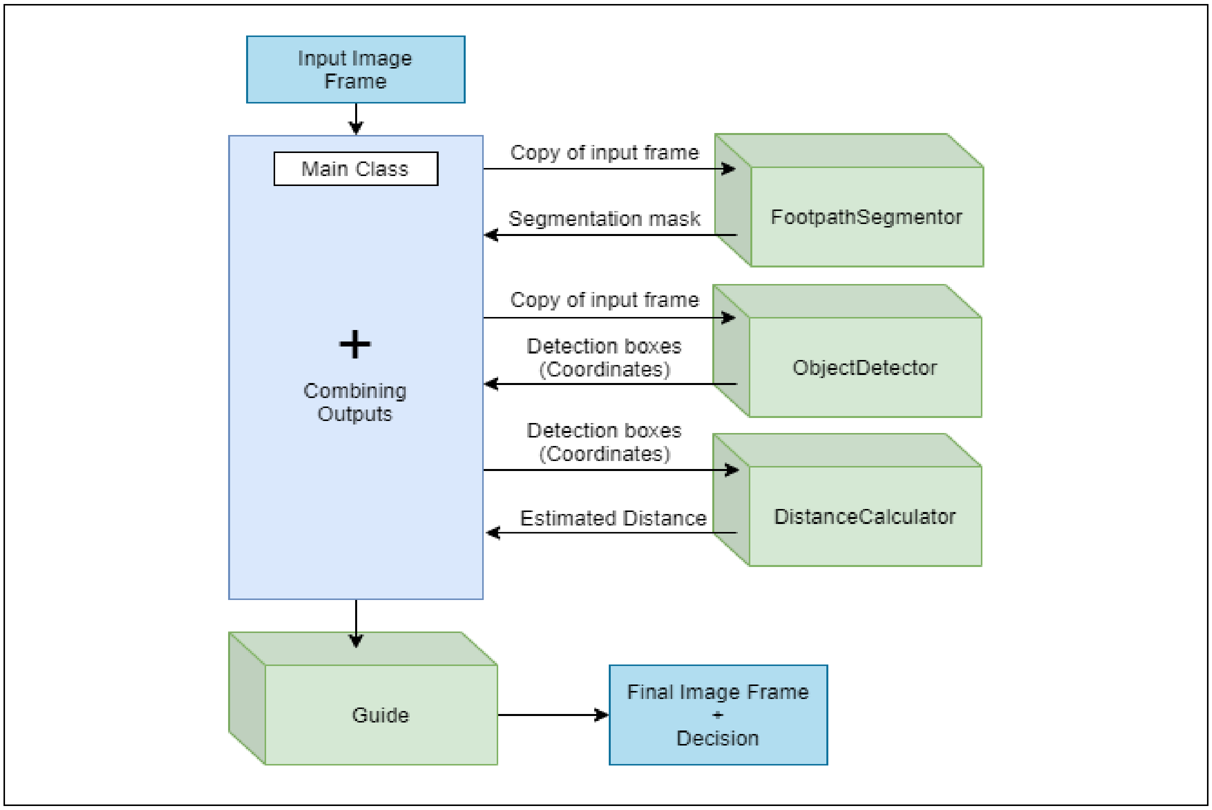


Figure 3.4.2.2: System Pipeline.

So far in this study, we have explained everything from Figure 3.4.2.2 except the Guide class and its working procedures. It is responsible for analyzing images and come up with a decision about a safe pathway. It can deduce a conclusion after processing the image whether the straight-ahead path is walkable, or there is any obstacle in the pathway. If there is any it can tell which way the person should shift, left or right, or should they stop. Even if the object detector fails to detect any unrecognized object, this class is capable of perceiving knowledge of other problem scenarios like an open manhole or a damaged pathway and may help the user to make a decision. Let's dive into it and get intuitions about its working procedures or see how it works.

3.4.2.2 Guide

This particular class takes an input image frame that is the resultant image frame after being inferred by the 3 discussed models. Let's consider the images in Figure 3.4.2.3.

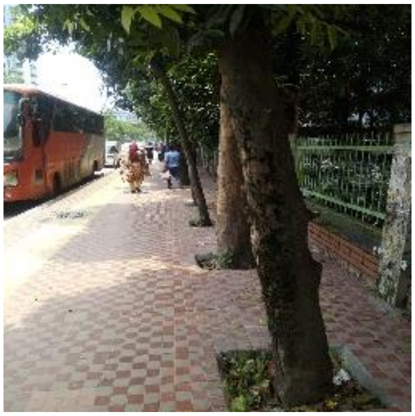
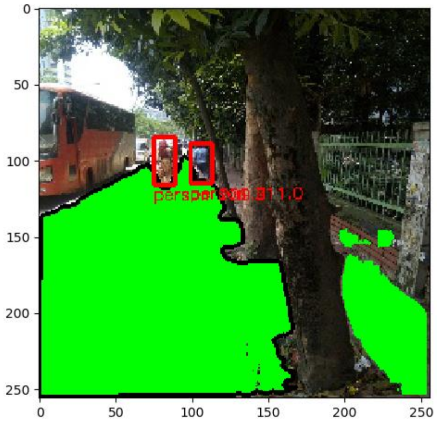
Sample Image	**Result of different model inference**
	

Figure 3.4.2.3: System's inference in a sample image.

In Figure 3.4.2.3 an inference of the system in an input image shown. The green marked area is the footpath and the red marked boxes are detected objects in the footpath. However, we can notice a tree in the pathway. But how can the system tell which path should the user take? This is where the Guide class comes in. This class partitions the lower part of the image in 3 segments and extracts the green channel (as in RGB image) from the image. As our segmentation model marks the safe footpath area keeping the green channel intensity to the maximum (i.e. R=0, G=255, B=0), the segment that has the maximum green pixels will be the safest one. Now, all we need is to count the green pixels in each segment. This analogy leads to a very computationally efficient but robust solution. After extraction of the green channel from the original RGB image it is simply a B&W image. Therefore, we only need to check the pixels with the highest intensity. The algorithm is as following –

Algorithm:

```
FUNCTION safest_path(segments, im_height, im_width):
    safe_area <= []
    FOR EACH segment IN segments:
        sum <= 0
        FOR i <= 0 TO im_height:
            FOR j <= 0 TO im_width:
                IF segment[i][j] == 255:
                    sum += 1
        safe_area.ADD(sum)
    RETURN ARGMAX(safe_area)
```

After applying this algorithm to the segments of the sample image of Figure 3.4.2.3 we get the result illustrated in 3.4.2.4.

*GPC = Green Pixel Count

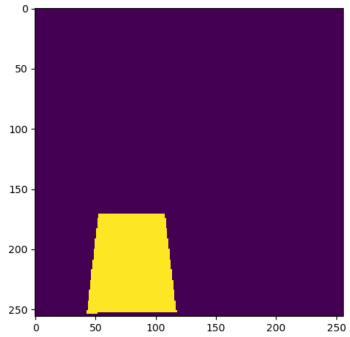
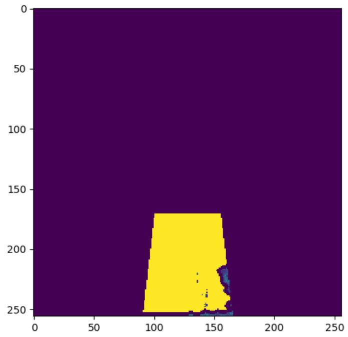
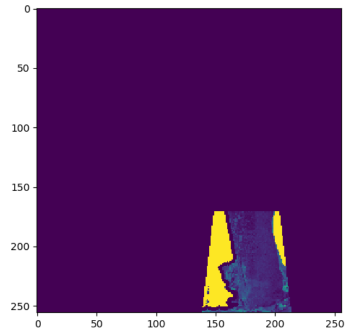
Left Segment	Middle Segment	Right Segment
		
GPC: 5319	GPC: 5030	GPC: 1483

Figure 3.4.2.4: Segment-wise green pixel counting.

It is observed in the figure that the left segment has the highest green pixel count. Therefore, it should be the safest path to take and the system will tell the user to shift left. Now combining each part we get the final result in Figure 3.4.2.5 for the sample image.

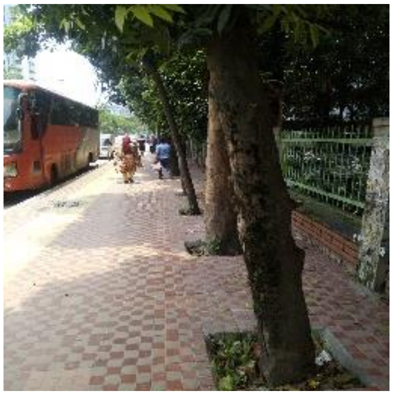
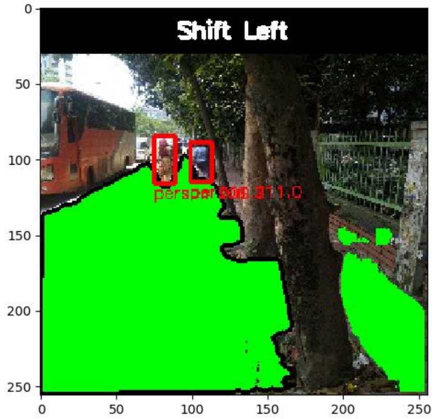
Sample Image	Final Output
	 Shift Left person 0 50 100 150 200 250 0 50 100 150 200 250

Figure 3.4.2.5: Final output with Guide.

3.5 Implementation Requirements

To implementation is divided into two phases as given in Figure 3.1.1. The system is trained on the cloud and assembled in the local machine. It requires different hardware and software to implement. Thus, the implementation requirements are divided into 3 different types.

3.5.1 Cloud System (Training)

- Platform: Google Colaboratory
- CPU: Intel ® Xeon ®
- Clock Speed: 2300 MHz
- GPU: Tesla K80
- Available RAM: 25 GB
- Available GPU: *Variable* [> 8 GB]
- Total Storage: 68 GB
- L3 Cache: 46080K

3.5.2 Local System (System Assembling)

- CPU: Intel® Core™ i5-4200U
- Clock Speed: 2600 MHz
- GPU: Nvidia Geforce 820M
- RAM: 8 GB
- Video Memory: 2 GB
- L3 Cache: 3 MB
- Operating System: Windows 10 Pro

3.5.3 Software Requirements

- Language: Python (3.5+)
- IDE: PyCharm, Jupyter Notebook
- Dependencies: Dependencies of the project along with the preferred version are given in Table 3.5.3.1.

Table 3.5.3.1: Project Dependencies and Version.

Package	Version	Usage
imutils	0.5.3	Image processing
jupyter	1.0.0	IDE
Keras	2.3.0	Deep Learning library
labelme	4.2.9	Image annotation
matplotlib	3.1.0	Plotting/Visualization
numpy	1.16.4	General-purpose computing
opencv-python	4.1.1.26	Computer vision library
pandas	0.24.2	Data analysis/manipulation
scikit-image	0.15.0	Image processing
scikit-learn	0.21.2	Machine learning library
tensorflow	1.15.0	Math library for ML

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

The sole purpose of this chapter is to provide experimental results of the methodologies used in this research and their effectiveness compared to other similar methods. Illustrations will be provided as well as comparative analysis in a tabular manner.

4.1 Experimental Results

We proposed a U-Net architecture for semantic segmentation that achieves as high as 96% accuracy but with very few parameters thus make the model highly usable for real-time segmentation with low processing power. A comparative analysis between other similar models in our dataset in terms of different metrics is given in Table 4.1.1 & Table 4.1.2. The data shown in the tables are instances when validation loss was at the minimum during training.

Table 4.1.1: Comparative results between segmentation models in the dataset for training data.

Model	**Backbone**	**Acc.**	**Loss**	**IOU**	**IOU Thresh.**	**Dice Coeff.**	**Prec.**	**Recall**	**Params (million)**
LinkNet	VGG-16	0.96	0.10	0.94	0.95	0.97	0.94	0.92	15.6
U-Net	VGG-16	0.96	0.10	0.95	0.95	0.97	0.95	0.92	19
PSPNet	VGG-16	0.96	0.07	0.91	0.94	0.95	0.94	0.94	10
Proposed U-Net	**None**	**0.96**	**0.03**	0.94	**0.96**	**0.97**	**0.96**	**0.94**	**4.7**

Table 4.1.2: Comparative results between segmentation models in the dataset for validation data.

Model	**Backbone**	**Acc.**	**Loss**	**IOU**	**IOU Thresh.**	**Dice Coeff.**	**Prec.**	**Recall**	**Params (million)**
LinkNet	VGG-16	0.95	0.17	0.92	0.93	0.96	0.94	0.92	15.6
U-Net	VGG-16	0.95	0.16	0.91	0.92	0.95	0.95	0.92	19

PSPNet	VGG-16	**0.96**	0.09	0.90	0.92	0.95	0.94	0.94	10
Proposed U-Net	None	0.95	**0.06**	0.91	**0.93**	0.95	0.94	**0.93**	**4.7**

By analyzing data given in Table 4.1.1 and 4.1.2, it is observed the proposed U-Net architecture is providing almost similar performance but with a very few parameters. It can also be deducted that the proposed model has the lowest fluctuation and higher convergence. Thus, it can be concluded that for a system with limited power and real-time use the proposed network works better than other similar networks. A few predictions of the model are given in Figure 4.1.1.

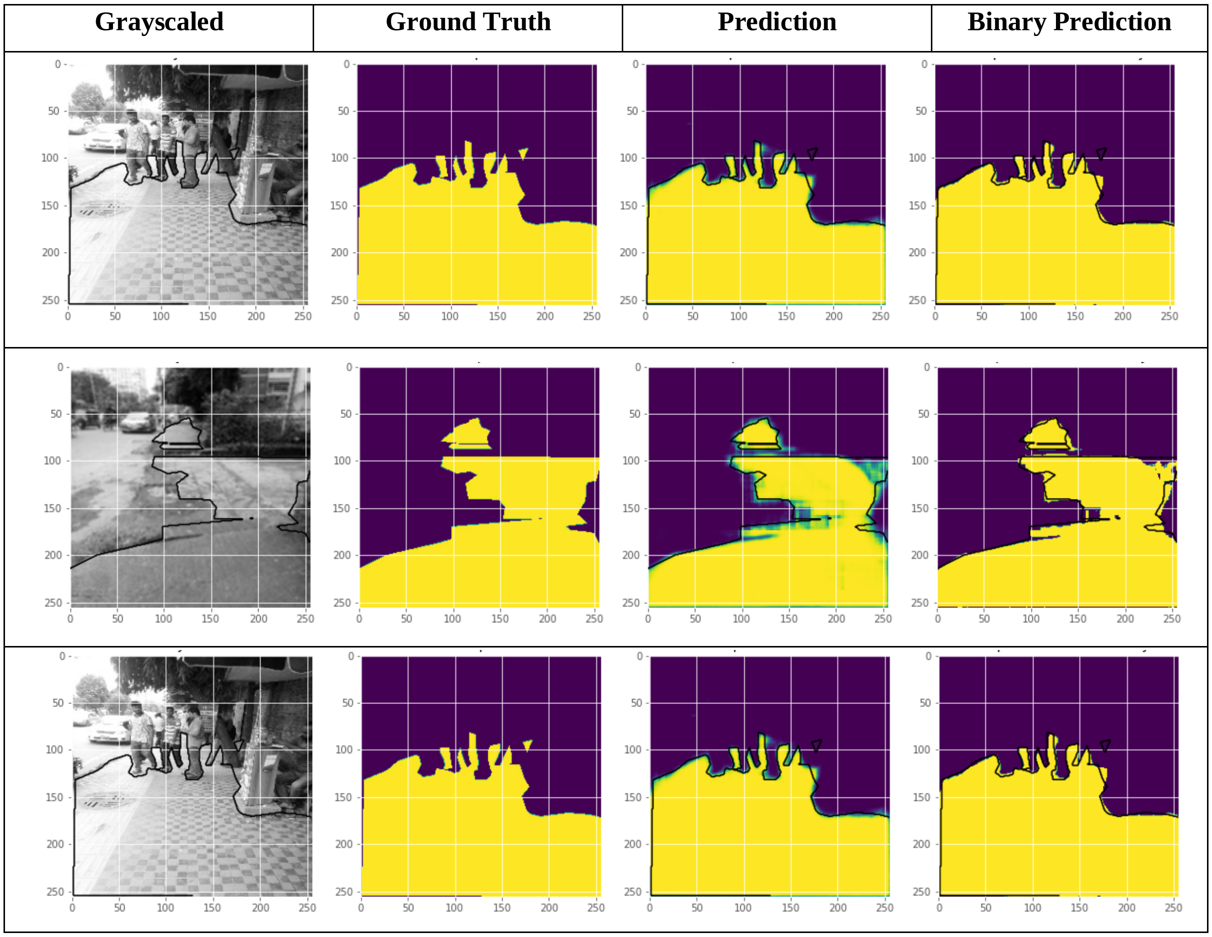


Figure 4.1.1: Prediction results of the segmentation model.

On the other hand, we have used a polynomial regression model for distance estimation. In Table 4.1.3 we shall observe the model performance compared to the linear regression model in the test dataset.

Table 4.1.3: Comparison of performance between LR & PR for distance estimation.

*RMSE = Root Mean Square Error, *MAE = Mean Absolute Error,

*EVS = Explained Variance Score

	Score	**R2-Score**	**RMSE**	**MAE**	**EVS**
Linear Regression	0.84	0.84	37.29	18.5	0.82
Polynomial Regression	**0.94**	**0.95**	**24.3**	**30.71**	**0.94**

In Table 4.1.4 we illustrate some outputs of the distance estimation model compared to the real distance.

Table 4.1.4: Comparison of the predicted and real distance.

Observation No.	**X_c**	**Y_c**	**Prediction**	**Real Value**
1	125.81	234.90	106.8	106
2	78	185.81	175	180
3	187.72	187	162.2	168
4	82	200.36	136	141
5	142.72	209.8182	119.4	124

Thus, we can evaluate the efficiency and performance of the different models working on this proposal. With all these being described, now we shall observe some of the final outputs the system predicts for an input frame.

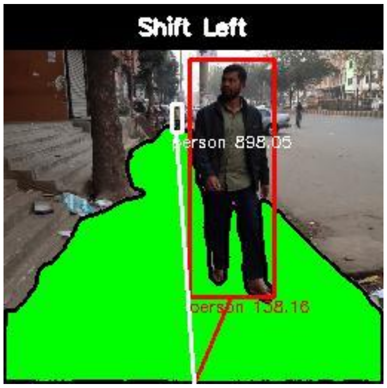
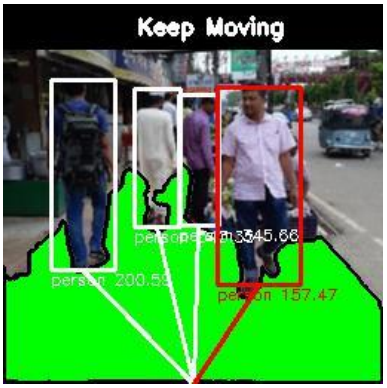
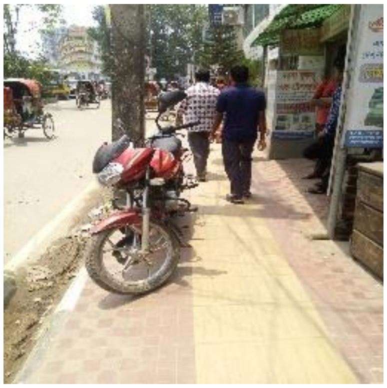
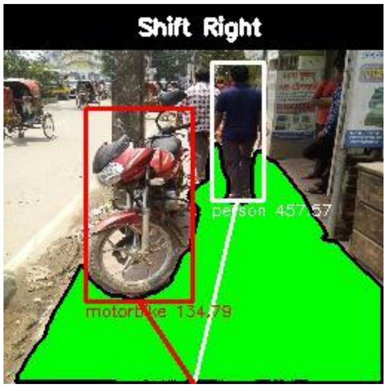
No.	Input Image	System Output
1		
2		
3		

Figure 4.1.2: System predictions for input images.

In Figure 4.1.2 we can observe some predictions where the system is capable of outputting the right decisions by inferencing images in different models. In these observations, some unseen images were fed into the system and we see it perfectly outputted the result. So, it is safe to assume that the system is working as expected. However, in some cases, the system may struggle to perceive and come to the right decision when the scene gets too complicated, or some outlier objects exist in the scene, or when there is maybe no footpath exists in the scene.

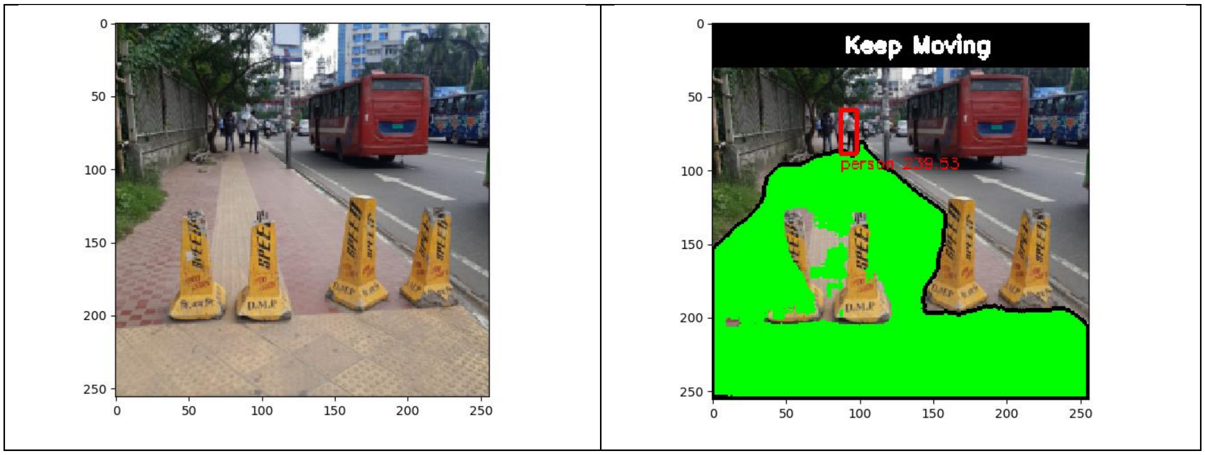


Figure: 4.1.3: System producing the wrong decision.

It is possible that when an input image contains a new environment that has something unfamiliar to the model, it may end up producing a wrong decision. Figure 4.1.3 is somewhat similar to that kind of scenario. We can see the segmentation here is not perfect for which the model ended up producing the wrong decision. The reason behind this is that the model didn't see enough data like this during training.

4.2 Discussion

After all the explanatory analysis of results given in Section 4.1, we can conclude that the an-Eye system is capable of providing very robust and efficient assistance in vision. Even though there exist some constraints and limitations in the system as one is given in Figure 4.1.3, it is possible to improve the performance by diversifying the dataset more and feeding it with more data. In future works of this research, these limitations will be highlighted and be worked upon. However, keeping the limitations aside undoubtedly the performance of the system is pretty good according to the resource it is utilizing and accuracy it is providing.

CHAPTER 5

IMPACT ON SOCIETY, ENVIRONMENT, AND SUSTAINABILITY

5.1 Impact on Society

Blindness is always been a social problem since the start of mankind. Eyesight is probably the most important sensory any living being gets by birth. People with blindness have to face lots of problems while moving from one place to another or communicate with others. Sometimes, while walking down a busy footpath they even get hurt. This study is a little step to improve the current condition. With the evolution of computer science, vision-based systems are now being rapidly used in every sector imaginable. So, why not using this blessing for the good of mankind? Hence, this is our effort to make a vision-based solution. The proposed system can be a good assistant for people with visual impairments. It will help the blind ones to navigate to their desired destination with a little less hassle. The system will help to minimize the trouble they face outdoor. It will also help in moving faster as it is a real-time system and will constantly keep giving feedback about the environment. The system will help to avoid a lot of unexpected accidents people with bad eyesight face every day. Hopefully, this study has the potential to impact positively in the society as it is intended to minimize a social problem.

5.2 Impact on Environment

Eyesight is the primary means of communication within our environment and the first approach to collect information about the environment. For blind people that information must be complemented with another very powerful data collection such as navigation from live video footage. To do so, our system gathers and segments video images and produce a safe path through the footpath. For most people who are blind, roaming an unknown environment could be troublesome, uneasy, and dangerous. Over the past years, there has been some research to help people with blindness to navigate safely. But there are a few works that have been done from the perspective of our country. The environment in our country differs from one city to another. Though there is footpath but most of the time it filled with shops and hawkers which left a little area to move. So blind people face a really tough time to walk through these footpaths. No matter which area in our country, the system can easily detect obstacles, holes, and many other interruptions.

5.3 Sustainability

A lot of research has done before choosing any method to implement the system. While it comes to detection of any obstacles, holes in the footpath, or other interruptions we have to be exact about that with the exact distance between the user and the obstacle as the safety of the user is the primary issue. The system was implemented in such a way that whenever there is a hole in the footpath, shops, or any other obstacle it can detect and also warn the user about their safety through a voice command. The system can identify the exact interruption and let know the user about it so that the user can take the necessary steps to avoid those interruptions. Also, the system will suggest the safest path through the footpath.

CHAPTER 6
SUMMARY, CONCLUSION, RECOMMENDATION AND IMPLICATION FOR FUTURE RESEARCH

6.1 Summary

The primary objective of this research was to develop a system that may help visually impaired to navigate while they are using footpaths in Bangladesh. While doing our study on the topic our finding is that most of the footpaths are very much unfriendly for visually challenged people. It is also noticed that many of them face minor accidents now and then and sometimes they also get critically hurt. Considering the scenario and socio-economic condition of the country putting a considerably low-cost but effective system could be of great help. Hence, we came up with a solution named 'an-Eye'. The system used stacks of procedures such as – semantic segmentation, object detection, distance measurement of detected objects, and identifying safe pathways. Each of the procedures is highly dependent on different computer vision techniques, deep learning, and machine learning algorithms. For this study, a novel annotated footpath dataset of Dhaka city was built. The segmentation model has a 96% accuracy with only 4 million parameters which is the current state-of-the-art result compare to other similar segmentation model variants. The system also uses the YOLO object detection algorithm which makes it robust, incredibly fast, and capable of real-time use. Our proposed system introduces a very low computation costly but pretty effective way for object distance measurement with a score of 94% that helps the system to determine the perceive the environment and let the user know whether they should move forward or shift left or shift right or should stop moving.

6.2 Implication for Future Study

We are currently working on the system to make it mobile-friendly so that it can be accessible by everyone. As the processing power of mobile devices has been increased a lot in recent trends, we believe we can come up with a mobile solution in near future. We have a plan to enrich the dataset more with footpath images outside Dhaka city so that the system can work in other cities as well. It is mentioned earlier in the paper that the primary object of this research is assisting visually challenged but it is not limited to it. The system may also come useful to robotic applications as well. For distance estimation, we have some constraints like the fixed focal length, positional height, and angle of the camera sensor. We will be working to remove these constraints and come up with a more flexible solution while keeping the cost minimal. Our plan also includes improving the system at the level from which it can be used by people across the globe.

References

[1] WHO, Global Data on Visual Impairment, available at https://www.who.int/blindness/publications/globaldata/en/, last accessed on 01-07-2020 8:00 PM

[2] Directorate General of Health Services (DGHS), available at https://www.dghs.gov.bd/licts_file/images/Health_Bulletin/HB2012_CH/HB2012_CH18_National-Eye-Care.pdf, last accessed on 27-04-2020 at 5:00 PM

[3] Seva Organization, available at http://www.seva.org/pdf/Seva_Country_Fact_Sheets_Bangladesh.pdf, last accessed on 27-04-2020 at 5:05 PM

[4] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham.

[5] J. Redmon & Ali Farhadi, YOLOv3: An Incremental Improvement, arXiv:1804.02767, 2018

[6] Lin TY. et al. (2014) Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham.

[7] Bimbraw, Keshav. (2015). Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology. ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings. 1. 191-198. 10.5220/0005540501910198.

[8] Fridman, Lex & Brown, Daniel & Glazer, Michael & Angell, William & Dodd, Spencer & Jenik, Benedikt & Terwilliger, Jack & Patsekin, Aleksandr & Kindelsberger, Julia & Ding, Li & Seaman, Sean & Mehler, Alea & Sipperley, Andrew & Pettinato, Anthony & Seppelt, Bobbie & Angell, Linda & Mehler, Bruce & Reimer, Bryan. (2019). MIT Advanced Vehicle Technology Study: Large-Scale Naturalistic Driving Study of Driver Behavior and Interaction with Automation. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2926040.

[9] John Canny. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-8(6):679–698, Nov. 1986.

[10] Gonzalez, Rafael C. & Woods, Richard E., Digital Image Processing, 3rd Edition, Pearson Education, 2008, pp. 738–761.

[11] Jin X., Han J. (2011) K-Means Clustering. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA

[12] Ian Goodfellow, Yoshua Bengio, & Aaron Courville, Deep Learning, MIT Press, 2016, pp. 166-224.

[13] Ian Goodfellow, Yoshua Bengio, & Aaron Courville, Deep Learning, MIT Press, 2016, pp. 330-338.

[14] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2015.7298965

[15] Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid Scene Parsing Network. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2017.660

[16] Chaurasia, A., & Culurciello, E. (2017). LinkNet: Exploiting encoder representations for efficient semantic segmentation. 2017 IEEE Visual Communications and Image Processing (VCIP). doi:10.1109/vcip.2017.8305148

[17] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4), 834–848. doi:10.1109/tpami.2017.2699184

[18] Girshick, R. (2015). Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV). doi:10.1109/iccv.2015.169

[19] Ren, Shaoqing & He, Kaiming & Girshick, Ross & Sun, Jian. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39. 10.1109/TPAMI.2016.2577031.

[20] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. Lecture Notes in Computer Science, 21–37. doi:10.1007/978-3-319-46448-0_2

[21] Alizadeh, P. (2015). Object Distance Measurement Using a Single Camera for Robotic Applications. Available at: https://www.semanticscholar.org/paper/Object-Distance-Measurement-Using-a-Single-Camera-Alizadeh/ddf137c5e767186cc37a3ef55803a67192846808

[22] Murawski, Krzysztof. (2015). Method of Measuring the Distance to an Object Based on One Shot Obtained from a Motionless Camera with a Fixed-Focus Lens. Acta Physica Polonica A. 127. 1591-1596. 10.12693/APhysPolA.127.1591.

[23] Brandt S. (1999) Linear and Polynomial Regression. In: Data Analysis. Springer, New York, NY. Avaiable at: https://doi.org/10.1007/978-1-4612-1446-5_12

[24] Sammut C., Logistic Regression, Webb G.I. (eds) Encyclopedia of Machine Learning, 2011, Springer, Boston, MA. Available at: https://doi.org/10.1007/978-0-387-30164-8

[25] Dionisi, A. & Sardini, Emilio & Serpelloni, Mauro. (2012). Wearable object detection system for the blind. 2012 IEEE I2MTC - International Instrumentation and Measurement Technology Conference, Proceedings. 1255-1258. 10.1109/I2MTC.2012.6229180.

[26] Pradeep, Vivek et al. “Robot vision for the visually impaired.” 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops (2010): 15-22.

[27] Lee Y.H., Medioni G. (2015) Wearable RGBD Indoor Navigation System for the Blind. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8927. Springer, Cham.

[28] Chen, Liang-Bi & Su, Jian-Ping & Chen, Ming-Che & Chang, Wan-Jung & Yang, Ching-Hsiang & Sie, Cheng-You. (2019). An Implementation of an Intelligent Assistance System for Visually Impaired/Blind People. 10.1109/ICCE.2019.8661943.

[29] N. Dey, A. Paul, P. Ghosh, C. Mukherjee, R. De and S. Dey, "Ultrasonic Sensor Based Smart Blind Stick," 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), Coimbatore, pp. 1-4, 2018.

[30] Nadia Kanwal & et al. (2015), "A Navigation System for the Visually Impaired:
A Fusion of Vision and Depth Sensor", Hindawi Publishing Corporation, Volume 2015, Article ID 479857

[31] Kentaro Wada. (2016). labelme: Image Polygonal Annotation with Python. Available at: https://github.com/wkentaro/labelme.

[32] Freedman, D., Pisani, R., & Purves, R. (2007). Statistics (international student edition). Pisani, R. Purves, 4th Edn. WW Norton &amp; Company, New York.

[33] Google Colaboratory, available at https://colab.research.google.com/, last accessed on 04-05-2020 at 3:00 PM

[34] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.90

[35] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, & Ruslan Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting, Journal of Machine Learning Research, Vol. - 15, pp. 1929-1958.

[36] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015 IEEE International Conference on Computer Vision (ICCV). doi:10.1109/iccv.2015.123

[37] Ioffe, Sergey & Szegedy, Christian (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning, Vol - 37, pp. 448–456

[38] Keras, available at https://github.com/keras-team/keras/, last accessed on 04-05-2020 at 10:30 PM

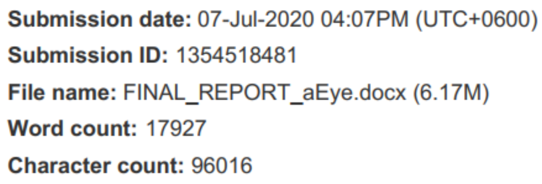
[39] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

[40] YOLO: Real-Time Object Detection, available at https://pjreddie.com/darknet/yolo/, last accessed on 08-05-2020 at 9:00 PM

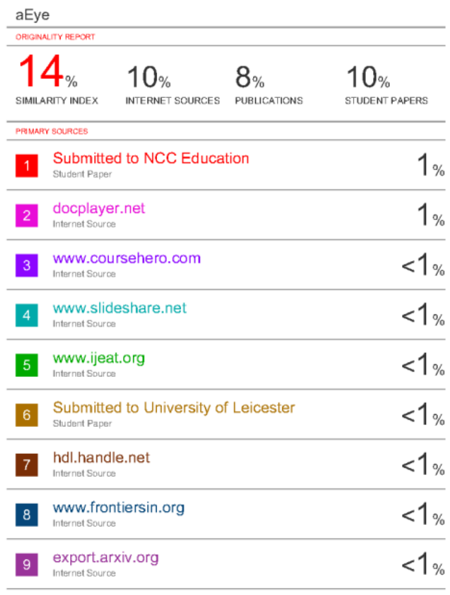
[41] OpenCV (2015), Open Source Computer Vision Library, available at https://github.com/opencv/opencv

[42] Darknet: Open Source Neural Networks in C, available at http://pjreddie.com/darknet/, last accessed on 08-05-2020 at 11:03 PM.

PLAGIARISM REPORT



Submission date: 07-Jul-2020 04:07PM (UTC+0600)
Submission ID: 1354518481
File name: FINAL_REPORT_aEye.docx (6.17M)
Word count: 17927
Character count: 96016



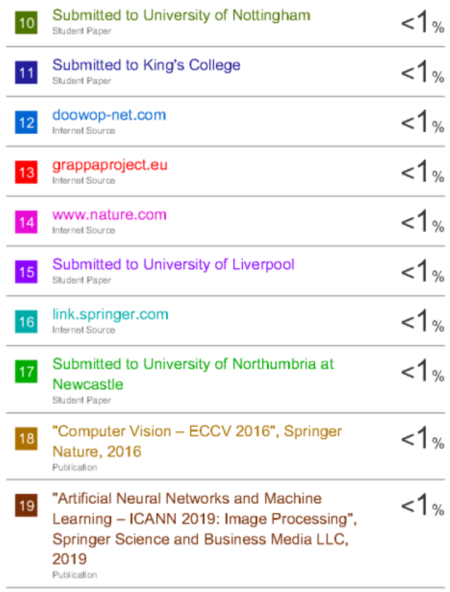
aEye

ORIGINALITY REPORT

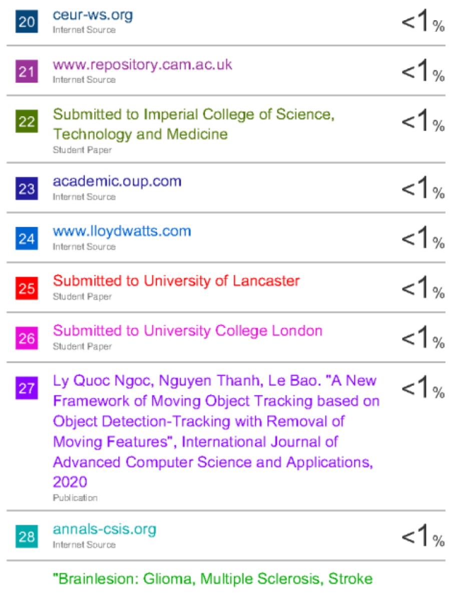
14%	10%	8%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to NCC Education Student Paper	1%
2	docplayer.net Internet Source	1%
3	www.coursehero.com Internet Source	<1%
4	www.slideshare.net Internet Source	<1%
5	www.ijeat.org Internet Source	<1%
6	Submitted to University of Leicester Student Paper	<1%
7	hdl.handle.net Internet Source	<1%
8	www.frontiersin.org Internet Source	<1%
9	export.arxiv.org Internet Source	<1%

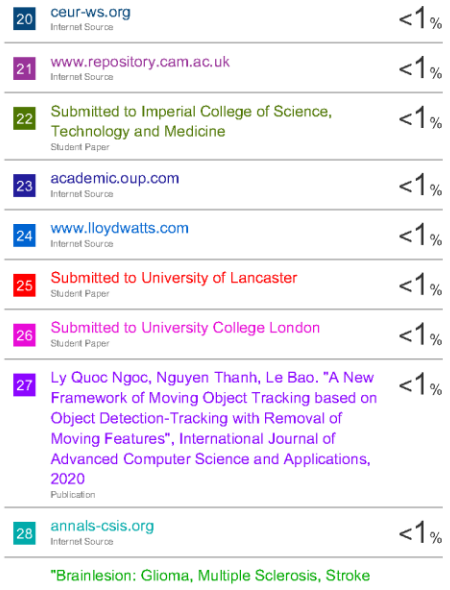


10	Submitted to University of Nottingham Student Paper	<1%
11	Submitted to King's College Student Paper	<1%
12	doowop-net.com Internet Source	<1%
13	grappaproject.eu Internet Source	<1%
14	www.nature.com Internet Source	<1%
15	Submitted to University of Liverpool Student Paper	<1%
16	link.springer.com Internet Source	<1%
17	Submitted to University of Northumbria at Newcastle Student Paper	<1%
18	"Computer Vision – ECCV 2016", Springer Nature, 2016 Publication	<1%
19	"Artificial Neural Networks and Machine Learning – ICANN 2019: Image Processing", Springer Science and Business Media LLC, 2019 Publication	<1%



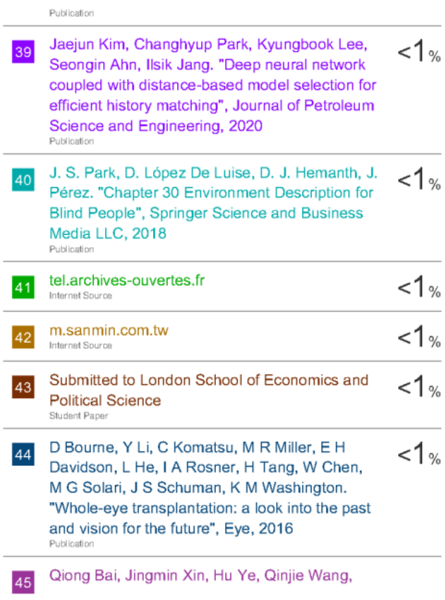
20	ceur-ws.org Internet Source	<1%
21	www.repository.cam.ac.uk Internet Source	<1%
22	Submitted to Imperial College of Science, Technology and Medicine Student Paper	<1%
23	academic.oup.com Internet Source	<1%
24	www.lloydwatts.com Internet Source	<1%
25	Submitted to University of Lancaster Student Paper	<1%
26	Submitted to University College London Student Paper	<1%
27	Ly Quoc Ngoc, Nguyen Thanh, Le Bao. "A New Framework of Moving Object Tracking based on Object Detection-Tracking with Removal of Moving Features", International Journal of Advanced Computer Science and Applications, 2020 Publication	<1%
28	annals-csis.org Internet Source	<1%

"Brainlesion: Glioma, Multiple Sclerosis, Stroke

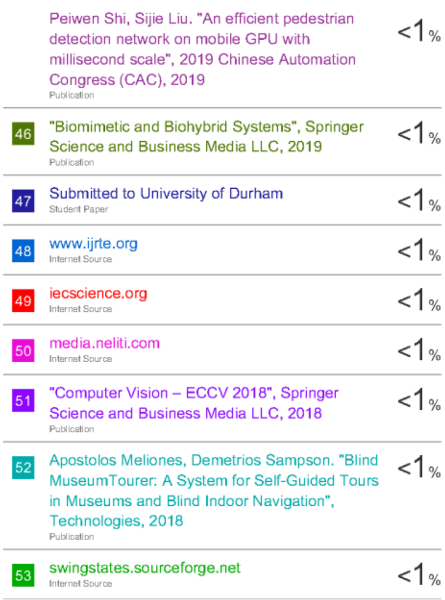


20	ceur-ws.org Internet Source	<1%
21	www.repository.cam.ac.uk Internet Source	<1%
22	Submitted to Imperial College of Science, Technology and Medicine Student Paper	<1%
23	academic.oup.com Internet Source	<1%
24	www.lloydwatts.com Internet Source	<1%
25	Submitted to University of Lancaster Student Paper	<1%
26	Submitted to University College London Student Paper	<1%
27	Ly Quoc Ngoc, Nguyen Thanh, Le Bao. "A New Framework of Moving Object Tracking based on Object Detection-Tracking with Removal of Moving Features", International Journal of Advanced Computer Science and Applications, 2020 Publication	<1%
28	annals-csis.org Internet Source	<1%

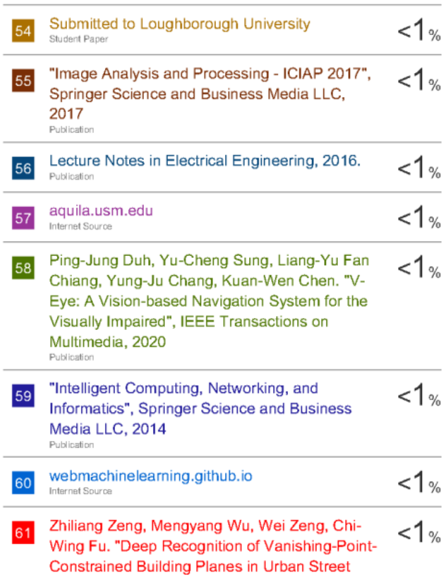
"Brainlesion: Glioma, Multiple Sclerosis, Stroke



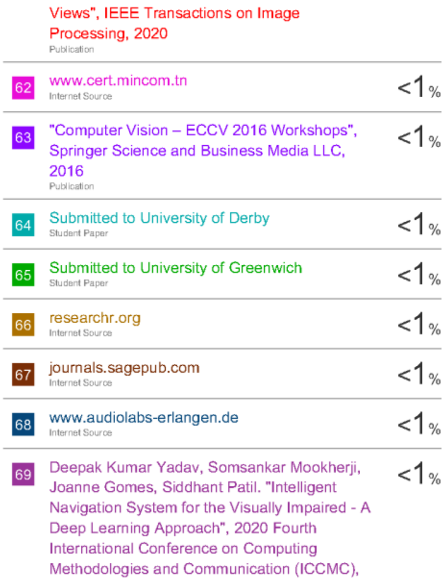
	Publication	
39	Jaejun Kim, Changhyup Park, Kyungbook Lee, Seongin Ahn, Ilsik Jang. "Deep neural network coupled with distance-based model selection for efficient history matching", Journal of Petroleum Science and Engineering, 2020 Publication	<1%
40	J. S. Park, D. López De Luise, D. J. Hemanth, J. Pérez. "Chapter 30 Environment Description for Blind People", Springer Science and Business Media LLC, 2018 Publication	<1%
41	tel.archives-ouvertes.fr Internet Source	<1%
42	m.sanmin.com.tw Internet Source	<1%
43	Submitted to London School of Economics and Political Science Student Paper	<1%
44	D Bourne, Y Li, C Komatsu, M R Miller, E H Davidson, L He, I A Rosner, H Tang, W Chen, M G Solari, J S Schuman, K M Washington. "Whole-eye transplantation: a look into the past and vision for the future", Eye, 2016 Publication	<1%
45	Qiong Bai, Jingmin Xin, Hu Ye, Qinjie Wang,	



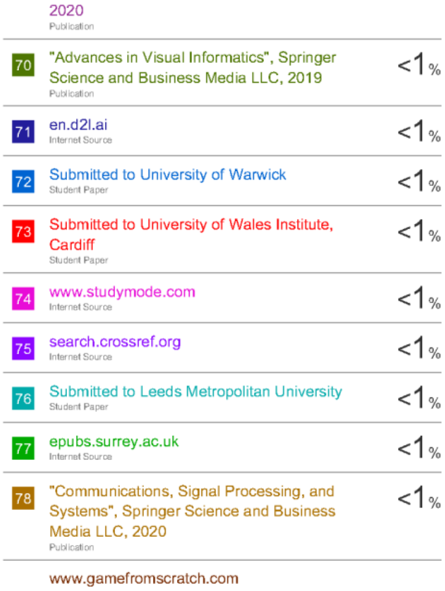
	Peiwen Shi, Sijie Liu. "An efficient pedestrian detection network on mobile GPU with millisecond scale", 2019 Chinese Automation Congress (CAC), 2019 Publication	<1%
46	"Biomimetic and Biohybrid Systems", Springer Science and Business Media LLC, 2019 Publication	<1%
47	Submitted to University of Durham Student Paper	<1%
48	www.ijrte.org Internet Source	<1%
49	iecscience.org Internet Source	<1%
50	media.neliti.com Internet Source	<1%
51	"Computer Vision – ECCV 2018", Springer Science and Business Media LLC, 2018 Publication	<1%
52	Apostolos Meliones, Demetrios Sampson. "Blind MuseumTourer: A System for Self-Guided Tours in Museums and Blind Indoor Navigation", Technologies, 2018 Publication	<1%
53	swingstates.sourceforge.net Internet Source	<1%



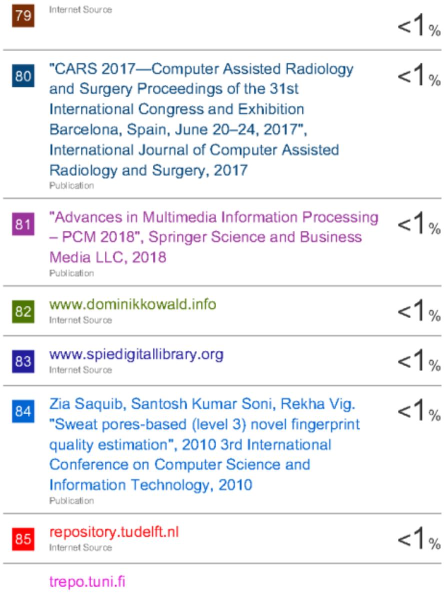
54	Submitted to Loughborough University Student Paper	<1%
55	"Image Analysis and Processing - ICIAP 2017", Springer Science and Business Media LLC, 2017 Publication	<1%
56	Lecture Notes in Electrical Engineering, 2016. Publication	<1%
57	aquila.usm.edu Internet Source	<1%
58	Ping-Jung Duh, Yu-Cheng Sung, Liang-Yu Fan Chiang, Yung-Ju Chang, Kuan-Wen Chen. "V-Eye: A Vision-based Navigation System for the Visually Impaired", IEEE Transactions on Multimedia, 2020 Publication	<1%
59	"Intelligent Computing, Networking, and Informatics", Springer Science and Business Media LLC, 2014 Publication	<1%
60	webmachinelearning.github.io Internet Source	<1%
61	Zhiliang Zeng, Mengyang Wu, Wei Zeng, Chi-Wing Fu. "Deep Recognition of Vanishing-Point-Constrained Building Planes in Urban Street	<1%



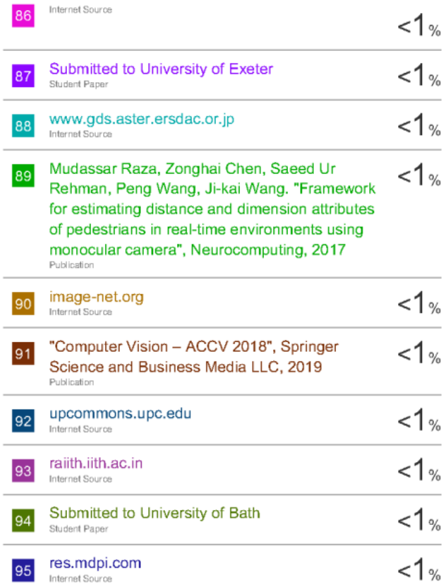
	Views", IEEE Transactions on Image Processing, 2020 Publication	
62	www.cert.mincom.tn Internet Source	<1%
63	"Computer Vision – ECCV 2016 Workshops", Springer Science and Business Media LLC, 2016 Publication	<1%
64	Submitted to University of Derby Student Paper	<1%
65	Submitted to University of Greenwich Student Paper	<1%
66	researchr.org Internet Source	<1%
67	journals.sagepub.com Internet Source	<1%
68	www.audiolabs-erlangen.de Internet Source	<1%
69	Deepak Kumar Yadav, Somsankar Mookherji, Joanne Gomes, Siddhant Patil. "Intelligent Navigation System for the Visually Impaired - A Deep Learning Approach", 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC),	<1%



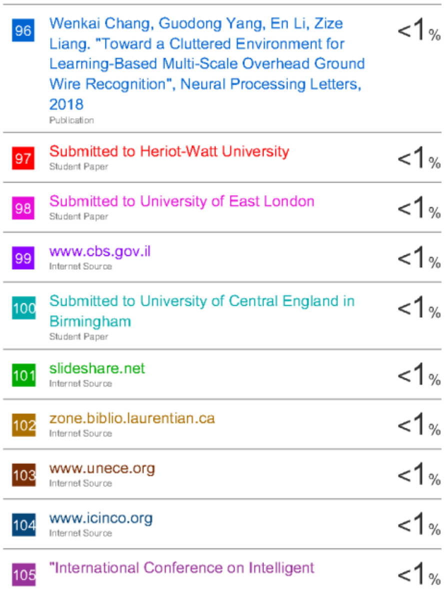
	2020 Publication	
70	"Advances in Visual Informatics", Springer Science and Business Media LLC, 2019 Publication	<1%
71	en.d2l.ai Internet Source	<1%
72	Submitted to University of Warwick Student Paper	<1%
73	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
74	www.studymode.com Internet Source	<1%
75	search.crossref.org Internet Source	<1%
76	Submitted to Leeds Metropolitan University Student Paper	<1%
77	epubs.surrey.ac.uk Internet Source	<1%
78	"Communications, Signal Processing, and Systems", Springer Science and Business Media LLC, 2020 Publication	<1%
	www.gamefromscratch.com	



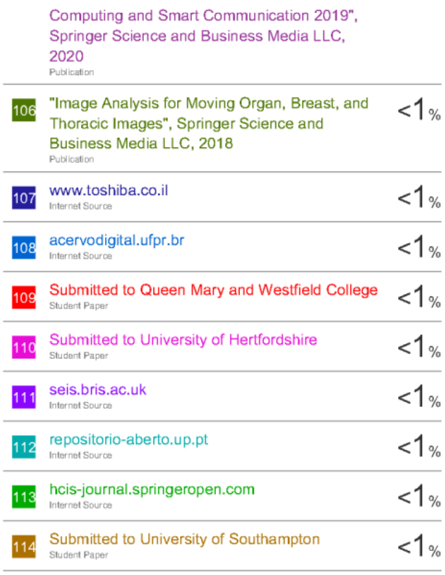
79	Internet Source	<1%
80	"CARS 2017—Computer Assisted Radiology and Surgery Proceedings of the 31st International Congress and Exhibition Barcelona, Spain, June 20–24, 2017", International Journal of Computer Assisted Radiology and Surgery, 2017 Publication	<1%
81	"Advances in Multimedia Information Processing – PCM 2018", Springer Science and Business Media LLC, 2018 Publication	<1%
82	www.dominikkowald.info Internet Source	<1%
83	www.spiedigitallibrary.org Internet Source	<1%
84	Zia Saquib, Santosh Kumar Soni, Rekha Vig. "Sweat pores-based (level 3) novel fingerprint quality estimation", 2010 3rd International Conference on Computer Science and Information Technology, 2010 Publication	<1%
85	repository.tudelft.nl Internet Source	<1%
	trepo.tuni.fi	



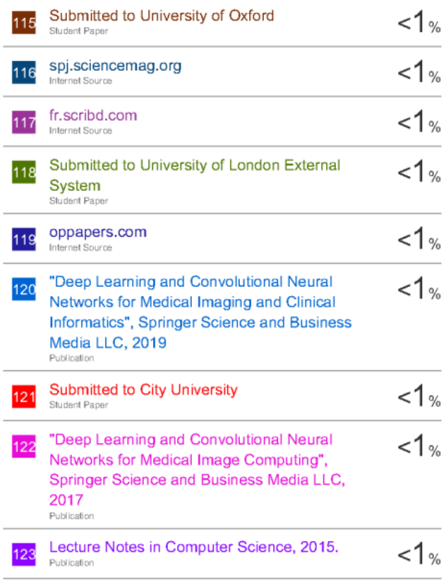
86	Internet Source	<1%
87	Submitted to University of Exeter Student Paper	<1%
88	www.gds.aster.ersdac.or.jp Internet Source	<1%
89	Mudassar Raza, Zonghai Chen, Saeed Ur Rehman, Peng Wang, Ji-kai Wang. "Framework for estimating distance and dimension attributes of pedestrians in real-time environments using monocular camera", Neurocomputing, 2017 Publication	<1%
90	image-net.org Internet Source	<1%
91	"Computer Vision – ACCV 2018", Springer Science and Business Media LLC, 2019 Publication	<1%
92	upcommons.upc.edu Internet Source	<1%
93	raiith.iith.ac.in Internet Source	<1%
94	Submitted to University of Bath Student Paper	<1%
95	res.mdpi.com Internet Source	<1%



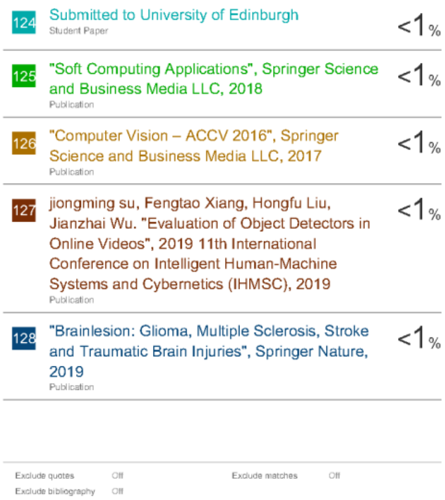
96	Wenkai Chang, Guodong Yang, En Li, Zize Liang. "Toward a Cluttered Environment for Learning-Based Multi-Scale Overhead Ground Wire Recognition", Neural Processing Letters, 2018 Publication	<1%
97	Submitted to Heriot-Watt University Student Paper	<1%
98	Submitted to University of East London Student Paper	<1%
99	www.cbs.gov.il Internet Source	<1%
100	Submitted to University of Central England in Birmingham Student Paper	<1%
101	slideshare.net Internet Source	<1%
102	zone.biblio.laurentian.ca Internet Source	<1%
103	www.unece.org Internet Source	<1%
104	www.icinco.org Internet Source	<1%
105	"International Conference on Intelligent	<1%



	Computing and Smart Communication 2019", Springer Science and Business Media LLC, 2020 Publication	
106	"Image Analysis for Moving Organ, Breast, and Thoracic Images", Springer Science and Business Media LLC, 2018 Publication	<1%
107	www.toshiba.co.il Internet Source	<1%
108	acervodigital.ufpr.br Internet Source	<1%
109	Submitted to Queen Mary and Westfield College Student Paper	<1%
110	Submitted to University of Hertfordshire Student Paper	<1%
111	seis.bris.ac.uk Internet Source	<1%
112	repositorio-aberto.up.pt Internet Source	<1%
113	hcis-journal.springeropen.com Internet Source	<1%
114	Submitted to University of Southampton Student Paper	<1%



115	Submitted to University of Oxford Student Paper	<1%
116	spj.sciencemag.org Internet Source	<1%
117	fr.scribd.com Internet Source	<1%
118	Submitted to University of London External System Student Paper	<1%
119	oppapers.com Internet Source	<1%
120	"Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics", Springer Science and Business Media LLC, 2019 Publication	<1%
121	Submitted to City University Student Paper	<1%
122	"Deep Learning and Convolutional Neural Networks for Medical Image Computing", Springer Science and Business Media LLC, 2017 Publication	<1%
123	Lecture Notes in Computer Science, 2015. Publication	<1%



124	Submitted to University of Edinburgh Student Paper	<1%
125	"Soft Computing Applications", Springer Science and Business Media LLC, 2018 Publication	<1%
126	"Computer Vision – ACCV 2016", Springer Science and Business Media LLC, 2017 Publication	<1%
127	jiongming su, Fengtao Xiang, Hongfu Liu, Jianzhai Wu. "Evaluation of Object Detectors in Online Videos", 2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2019 Publication	<1%
128	"Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries", Springer Nature, 2019 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		