Blind Navigation using Deep Learning-Based Obstacle Detection

W.A.D.P.M Gunethilake

2020

Blind Navigation using Deep Learning-Based Obstacle Detection

A dissertation submitted for the Degree of Master of Science in Computer Science

W.A.D.P.M Gunethilake
University of Colombo School of Computing
2020

DECLARATION

I hereby declare that the thesis is my original work, and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Student Name: W.A.D.P.M Gunethilake

Registration Number: 2017MCS034

Index Number:17440348

_____

Signature of the Student & Date

This is to certify that this thesis is based on the work of Mr. /Ms. _____ under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by,

Supervisor Name:

_____

Signature of the Supervisor & Date

ABSTRACT

Blind navigation has become a challenging task in the present. Blind people cannot detect and avoid obstacles similar to the people with good vision and they need guidance to avoid such obstacles. The white cane is the most widely used device by many blind navigators to detect and avoid obstacles. But with the limited reachability of the white cane, it is not possible to detect all the potential threats to the navigator. Therefore, the white cane is not an adequate aid to navigate safely. To secure the safe and independent navigation of the blind people, more insights of their current surroundings must be provided. This study proposes a novel approach for obstacle detection based on deep learning to assist in blind navigation.

In this study, a prototype was developed using deep neural networks (DNN) for obstacle detection and distance estimation due to real-time performance and high accuracy of DNNs. To train the DNN for obstacle detection data was gathered using a simulation environment. The output of the obstacle detection model was used to estimate the distance of the obstacles. The final result by combining the feedback of obstacle detection and distance estimation is communicated to the user via audio queues. The prototype system is deployed in a smartphone and the real-time video stream captured through the smartphone camera is processed to detect obstacles. To train the DNN for obstacle detection SSD MobileNet Architecture was used and the data to train the DNN was generated using a simulation environment. To estimate the distance of the detected obstacles, DNN based MonoDepth algorithm was used.

The mean average precision (mAP) value of all the classes of the DNN for obstacle detection reached more than 70%. Higher accuracy and a higher speed for obstacle detection can be achieved by the system prototype but there is a latency when estimating distance. The usability and the effectiveness of the prototype system exceeded 65% according to the feedback from the usability evaluation.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor Dr. Prasad Wimalaratne, senior lecturer of The University of Colombo School of Computing - UCSC, for his guidance, constant supervision and providing valuable resources throughout the research.

Thank You
W.A.D.P.M Gunethilake
17440348 (2017 | MCS | 034)

Contents

Table of Figures (TOF)

List of Tables (LOT)

ABBREVIATIONS

AP	Average Precision
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DNN	Deep Neural Network
HTTP	Hyper Text Transfer Protocol
mAP	Mean Average Precision
SLAM	Simultaneous Localization and Mapping
SSD	Single Shot MultiBox Detector
YOLO	You Only Look Once

CHAPTER 1

1. Introduction

1.1 Overview

According to the statistics of the World Health Organization (WHO), the number of visually impaired persons are estimated to be 1.3 billion, of whom 36 million are blind in 2018. By 2019 a total of 2.2 billion people suffers from some form of visual impairment [1]. According to these statistics, the blind and visually impaired community is increasing yearly.

Engaging in day-to-day activities without hazel is an extremely difficult task for a visually impaired / blind person. It becomes more difficult when it requires traveling through unfamiliar locations without a close companion to assist them along the way. Guide dogs are used in assisting visually impaired persons, but it is not easy to get a trained animal due to the high cost. Furthermore, traveling in familiar environments without help could also be challenging since the dynamic situations along the way cannot be predicted earlier, and responding to those situations in real-time is not possible for a blind person. Blind people navigate without a clear visual map about the obstacles in their path. Therefore, it is not possible to take precautions to avoid such obstacles similar to a normal person with good vision. This study focuses on improving the independent navigation of the blind and ensuring their safety while navigating.

1.2 Problem Statement

The white cane is the most commonly used assistive device by the blind community for their navigation. It is being used to detect and avoid obstacles that could collide with them along the way. But when considering the rapidly changing surroundings in the present, putting all the trust in the white cane does not ensure the safe independent navigation of the blind people. Furthermore, the limited reachability of the white cane reduces the possibility of detecting potential threats to the blind navigator. Considering the above limitations of the white cane, a solution that improves the safety and ability of independent navigation by providing more insights about the current environment is very convenient and useful for the blind community.

Replacing the white cane with a different assistive device is not a practical solution for many blind users since it is also an indicator for informing others about the blind navigator. Although there are white canes available with advanced technologies, they can be very expensive. Therefore, introducing a solution for blind navigation that can co-use with a white cane and inexpensive can be extremely helpful for blind people. Furthermore, to be used in blind navigation, the assistive methods and solutions must be accurate to guide and be convenient to use while navigating.

1.3 Background

In recent years many researchers have focused on the topic of blind navigation and proposed many approaches to accomplish safe navigation with obstacle detection/avoidance, distance estimation, and providing feedback [2]-[5]. Detecting obstacles is a major area of blind navigation and calculating the distance to obstacles and giving necessary feedback to avoid them also are important features of a blind navigation system.

Most of the existing outdoor navigation systems use GPS (Global Positioning System) technology for localization [6]-[8]. But their low accuracy and signal loss issues have made such systems unreliable to be used by the blind people. Major drawbacks in GPS systems are the inability of giving any feedback on moving obstacles and the inability of giving information on the obstacles near the user [9]. Hence such systems are not useful in assisting the blind.

Computer vision-based approaches, named ETAs (Electronic Travel Aids) are proposed in recent years [10]-[13]. ETAs should be reliable, affordable, and light. Electronic Travel Aid (ETA) named "EyeCane", is designed in [14] which translates point-distance information into auditory and tactile cues. According to [15], a wearable obstacle avoidance ETA survey, ETAs giving audio feedback and tactile feedback are identified as Echolocation, Navbelt, vOICe and Tactile Handle.

Dynamic and static obstacle detection plays an important role when guiding the blind and research has been carried out to discover efficient obstacle detection methodologies [2]-[4]. These existing systems have adapted vision-based techniques by using Monocular cameras, Stereo-cameras, RGB-D cameras, Time-of-flight (TOF), etc. and other sensors such as Ultrasonic sensors, SONAR, LIDAR, etc. to capture the environment and the relevant obstacles. Structure from Motion

algorithm, RANSAC algorithm, SIFT, Multicase Lucas-Kanade algorithm and Event-based algorithm are some of the algorithms that were adapted for obstacle detection.

Deep learning-based approaches are widely adapting to blind navigation in recent years [5][16][17]. Convolutional Neural Network (CNN) techniques are commonly applied in object recognition tasks due to their high accuracy although there are concerns regarding collecting a large set of training data and overfitting due to noisy data when using CNNs.

Smartphone usages of the blind persons are increasing due to their features such as screen reader, haptic feedback, audio feedback, adjustable contrast, audible battery indicator, and vibration. Light weight of a smartphone is also an advantage and it does not affect the navigation. Therefore, smartphones are widely being used in assisting the blind navigation [2]-[5]. Further laptops and other customized devices are used in designing assistive devices for blind navigation [18]-[21].

1.4 Aims and Objectives

Navigation is a difficult task for blind people. Since many blind people use only the white cane for navigation, there is a higher possibility of facing to harmful situations. Therefore, there is a critical need of providing more information of the environment around them to ensure their safety. The main objective of this research is to propose a novel obstacle detection mechanism to improve the safety and independent mobility of the blind people.

The aims of the study are described below.

- The main aim of the proposed approach is real-time obstacle detection with higher accuracy. To achieve this, a deep learning-based approach is used to obstacle detection due to the higher accuracy and real-time prediction of results of deep neural networks (DNN).

- It is expected to fill the gap of detection while using a white cane due to its limited reachability. Mainly when considering obstacles that are above ground-level are not reachable using the white cane and thus not detected by the white cane. With this proposed mechanism it is aimed to cover such obstacles.

- Furthermore, communicating to the user about potential threats in their surroundings in an effective manner to avoid such threats successfully is also an aim of the study.

1.5 Proposed Approach

With the improvements of the technology, it is possible to create deep learning models that provide real-time results with high accuracy [22]. Therefore, deep learning is used for object detection tasks in various applications. Similarly, this proposed system uses a deep learning-based obstacle detection mechanism. Training a deep learning model requires a large amount of data. To train this object detection model, the data was collected using a simulation environment instead of using real-world data.

There are different architectures and frameworks which are available for deep learning. Faster R-CNN [23], Masked R-CNN [24], YOLO [25], SSD [26] are such popular architectures for object detection. Among these, some architectures are fast enough to be running on mobile devices such as YOLO (YOLO family) and SSD (Single Shot MultiBox Detector). It is a challenging task to select a suitable architecture and considering requirements to perform a task and its deployment requirements an architecture must be selected. Here, considering the requirement of mobile deployment, the SSD MobileNet architecture is used to train a deep learning model.

The system prototype consists of three main modules namely object detection model, depth estimation module and feedback module. After detecting an obstacle through the object detection model, its depth estimation will be carried out using the depth estimation module. The depth estimation module is deployed in an external server. To perform depth estimation MonoDepth [27], a deep learning-based approach was used. This proposed approach is a deep learning-based obstacle detection mechanism that combines simulation, deep learning, and computer vision techniques.

1.6 Contribution

The main contribution of this research is to propose a novel obstacle detection mechanism to assist blind navigation that targets,

1. Obstacle detection with high accuracy and in real-time.
   For obstacle detection, a deep neural network is used. In the proposed system, a novel approach was adapted that uses data generated from a simulation environment instead of real-world data to train the deep neural network. Simulation platforms are used in various domains as a mechanism of data generation. Similarly, a simulation platform was used here and to generate data a 3D realistic environment was created.

2. A common mechanism that can be used to estimate the distance of both ground-level, above ground-level obstacles.
   A deep learning-based monocular depth estimation methodology is combined with binary thresholding to identify the obstacle with possible threats to the navigator.

1.7 Scope

This research will be conducted to propose an obstacle detection mechanism to assist blind people to navigate safely and independently. As it is clear that using only a white cane is not adequate for a blind person to navigate safely in outdoor environments, it is important to provide more information about the obstacles in their path. Due to the limited reachability of the white cane, especially the obstacles that exist above the ground-level are not detectable. Such obstacles are mainly focused to detect using the obstacle detection model.

To train the deep learning model, the required data is generated using a simulation environment. The Deep Learning Model (DNN) will be trained using a machine learning library that supports mobile conversion since this model is deployed in a smartphone. To estimate the distance of the detected obstacles a pre-trained MonoDepth algorithm [27] based implementation will be used. The feedback of the system will be communicated to the blind user via pre-recorded audio queues.

1.8 Thesis outline

The thesis is organized in the following format. The first chapter of the thesis identifies the problem of blind navigation and describes the proposed solution to overcome the issue. Chapter two presents the literature review on proposed approaches to assist blind navigation, background on simulation platforms, and usages of simulation in real-world and deep learning algorithms for obstacle detection. Chapter three presents the architecture of the prototype and the research methodology. Chapter four includes the details of the development of the proposed solution. Evaluation results of the obstacle detection model and the system prototype are presented in chapter five and the chapter concludes with the discussion section. The final chapter of the thesis contains the conclusion and future work.

CHAPTER 2

2. Literature Review

2.1 Overview

For years many researchers have focused on improving the safe navigation of the visually impaired in outdoor environments. A navigation system consists of three parts. Such a navigation system includes localization, providing information about the current location and orientation, and pathfinding. For the localization purpose, a particular system must extract the details and features of the current surrounding of the blind person. Global Coordinates (GPS), Local coordinates, environmental features, and dead reckoning are some of the feature identification methods for localization. Providing information about the current location includes static and dynamic obstacle detection and correct orientation. Pathfinding or wayfinding is finding the optimal or best route to the destination.

Many researchers have focused on the area of blind navigation from earlier years and many technologies have been adapted to develop assistive devices for blind navigation over the years. Figure 2.1 provides a classification of blind navigation systems and research carried out under different aspects.

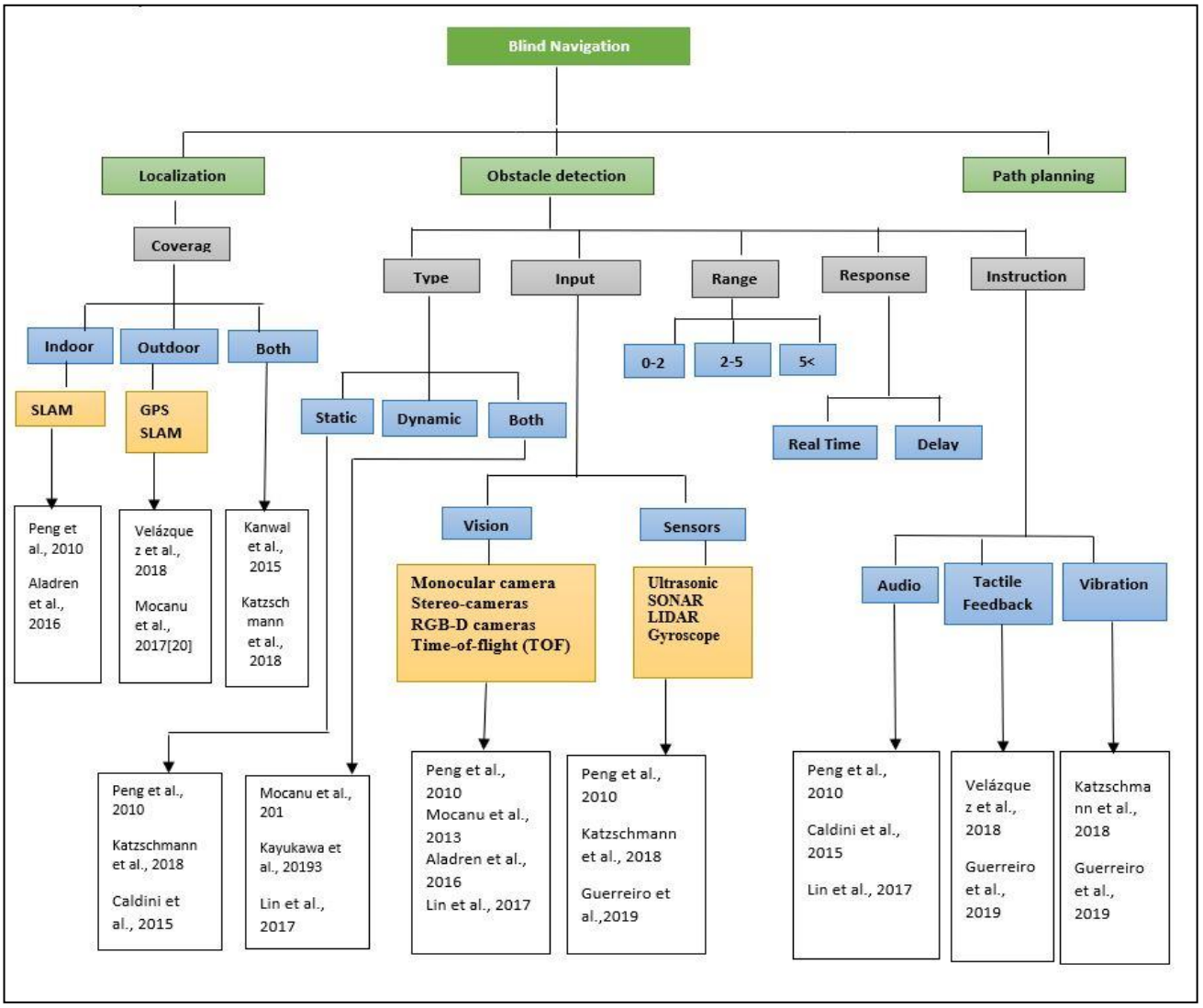


Figure 2.1: Classification of Blind Navigation systems

A detection system for obstacles on the ground at any height using a handheld smartphone is proposed in Peng et al. [2]. This proposed method uses computer vision techniques such as color histograms and edge detection for obstacle identification and obstacles are captured through the smartphone camera. A color histogram is created based on the images acquired from smartphones and histograms are created for each frame and chooses the simplest RGB color space. Then, a binary RGB histogram is built for the safe region which is derived from the image region.

In Caldini et al. [3], a calibrated smartphone on users' chest which installed a gyroscope is used to detect obstacles. It implements a modified Structure from Motion (SfM) algorithm for scene reconstruction. Visual data obtained from the camera and measurements obtained from the

gyroscope are used in the developed algorithm. The proposed vision-based system focuses on obstacle detection to help visually impaired people to move autonomously in unknown indoor and outdoor environments.

Bai et al. [28] presents a navigation device for the visually impaired people to reach a destination in an indoor environment. Here, to solve the problems of indoor localization and virtual-blind-road building, the visual SLAM algorithm was used. Furthermore, a PoI (Point of Interest) -graph was generated by the A* based way finding algorithm to find a globally shortest virtual-blind-road. A dynamic sub-goal selecting based route following algorithm was used to help the blind follow the globally shortest virtual-blind-road and to avoid obstacles. The system is deployed on a pair of wearable optical glasses. Fisheye camera and depth camera are used for obstacle detection and ultrasonic sensor is used for distance calculation. All the algorithms such as visual SLAM, obstacle detecting, way-finding, route following and speech synthesis are performed on a CPU. The feedback from the CPU is provided to the user via earphones and AR glasses.

Providing necessary information feedback in real-time is a critical requirement in blind navigation assisting. A real-time object detection and classification system design is proposed in Mocanu et al. [4]. In this paper moving obstacles are detected using video streams and interesting points are selected and tracked using the multiscale Lucas-Kanade algorithm [29]. Then, by applying the RANSAC algorithm [30] recursively on the set of matched key points, the background motion is identified. The outliers are merged into clusters using the associated motion vectors.

A Sensor-based navigation system for visually impaired persons using a fusion of vision and depth sensors that uses Microsoft Kinect Sensor is introduced in Kanwal et al. [31]. In the research, the expected outcome is to utilize the usage of both sensors by detecting obstacles using corner detection and then estimating their distance using a depth map of the corresponding scene. The data acquired from the Microsoft Kinect camera is processed to detect Edges using the Sobel edge detector, Corners using Harris&Stephens detector and Blobs using SIFT detector. But according to the paper, the Kinect sensor is not reliable as expected in outdoors and some sensors are unable to work in different environmental conditions and the infrared sensor has issues with blind spots in sunlight.

A wearable assistive system for visually impaired which provides the user detailed feedback about the obstacles surrounding the user is proposed in Katzschmann et al. [18]. The proposed system is designed as wearable by visually impaired and composed using a sensor belt and a haptic feedback device. Here infrared time-of-flight (TOF) distance sensors and sonar sensors were used in identifying the surroundings near the visually impaired person and different vibration patterns are introduced to notify the user.

A wearable system for indoor navigation called NAVI (Navigation Assistance for Visually Impaired) was introduced in Aladren et al. [19] which combines both range and color information to address the task of NAVI. In this research, due to the limitations of the range sensor (RGB-D sensors), the color information has been joined for the segmentation of the floor. A user interface containing a sound map information, created using stereo beeps where its frequency depends on the distance from a user to an obstacle and other voice commands are used as mechanisms for alerting the users.

Velázquez et al. [6] proposes a system that combines GPS for outdoor localization and tactile-foot simulation for providing instructions to assist in blind navigation in pedestrian environments. A smartphone was used to get the user GPS coordinate and orientation acquisition and when the user connected to the internet, his geospatial information is transmitted to a cloud server. A text file containing longitude, latitude, and orientation is constructed in the server and discards other information. A remote computer can access the server's text file via the internet and locate the user. Route waypoints returned from YOURS (open source route planner) are stored in the GIS(OpenStreetMap). Directions are translated to actuator commands and transmitted to the user via radiofrequency (RF) and signals are interpreted by a microcontroller and it controls the actuators in the tactile display using vibrations.

An assistive device that combines computer vision techniques and deep convolutional neural networks to detect, track and recognize objects encountered during the outdoor navigation is proposed by Mocanu et al. [16] and it aims on real-time obstacle detection. Object detection and recognition, object tracking and acoustic feedback are the main modules in the proposed framework. The initial object detection is performed by the YOLO Algorithm [25] and the module also consists of a generic object tracker based on two convolutional neural networks (CNNs)

trained offline that can track generic objects using motion patterns and visual attention models. A modified version of the YOLO algorithm is used for classification. The feedback to users is transmitted as acoustic warning messages through headphones. Some of the improvements of the proposed system are ability to handle occlusions, sudden movements of camera/object and rotation or various complex changes.

Everding et al. [32] proposes a wearable lightweight device for facilitating autonomous navigation and obstacle avoidance to assist the blind. The system has two retina-inspired dynamic vision sensors for visual information gathering and data acquired through dynamic vision sensors are processed through hardware by depth extraction, event selection, and 3D sound generation. The visual information is encoded as a stream of single-pixel events that are generated asynchronously and it is a different approach compared to classical camera systems. The Event-based algorithms operating on the visual data stream extract depth information in real-time and translates into the acoustic domain. The total latency of the system is approximately 50ms due to aggregation of the sounds and the system is not validated in dynamic environments.

Another assistive system called BBeep is proposed in Kayukawa et al. [20] that notifies the user and the nearby pedestrians of the potential risk of collision. An RGBD camera is used in the system to detect, track and predict the motion of nearby pedestrians. A stereo camera mounted on a suitcase is used to record RGB images and depth data. The system detects pedestrians using the RGB images and finds their position using the depth data. The RGB image is used to detect people using a convolutional neural network (CNN) and the depth data is used to estimate the distance to pedestrians (YOLOv2 [33]). After predicting the future positions of each pedestrian, BBeep emits an audible signal indicating the risk of collision with the blind navigator. After predicting the future position of the pedestrians in real-time, the system provides sound notifications only when there is a possibility of collision. The higher amount of unnecessary sound emissions is a drawback here.

Guerreiro et al. [21] presented CaBot (Carry-on roBot), an autonomous suitcase-shaped navigation robot guides blind navigators by avoiding obstacles. CaBot relies on a floorplan with relevant Points-of-Interest (POIs) and on LiDAR (Light Detection and Ranging) for localization and path planning. CaBot uses a LiDAR to map the environment, localize itself and for path planning. A stereo camera is used to convey semantics about objects in the environment by using a

convolutional neural network (CNN). The annotated floorplan with the location of relevant POIs are conveyed via speech and a modified suitcase handle is available to provide vibrotactile feedback about CaBot's directional actions. According to the authors the robot has only been tested in a controlled environment and dynamic elements have not been considered.

The proposed system in Lin et al. [5], consist of a computer image recognition system and a smartphone application that form a guiding system for the visually impaired. This system uses deep learning algorithms to recognize various obstacles and implements offline image recognition through the Haar feature and histogram of oriented gradients (HOG) feature to sort through detected objects. Faster R-CNN [34] and YOLO [25] algorithms were used to perform object recognition for real-time performances. This system informs the user of the type of obstacle in front of him or her and reveal the approximate distance between the user and the object. According to the authors, the system can detect objects in Range > 10m but limitations are arising with the distance. Large information computing is considered a burden in this system.

A summary of the above approaches with respect to coverage and type is shown in the following table.

Paper	**Coverage**	**Type**	**Range(m)**
Peng et al., 2010 [2]	Indoor	Static	< 5
Mocanu et al., 2013 [4]	Indoor, outdoor	Static, dynamic	5 <
Caldini et al., 2015 [3]	Indoor, outdoor	Static	< 5
Kanwal et al., 2015 [31]	Indoor, outdoor	Static, dynamic	2-5
Aladren et al., 2016 [19]	Indoor	Static	< 3.5
Everding et al., 2016 [32]	Indoor, outdoor	Static	> 8
Lin et al., 2017[5]	Indoor, outdoor	Static, dynamic	> 10
Mocanu et al., 2017[16]	Outdoor	Static, dynamic	<5

Velázquez et al., 2018 [6]	Outdoor	Static, dynamic	<5
Katzschmann et al., 2018 [18]	Indoor, outdoor	Static	< 5
Bai et al., 2018 [28]	Indoor	Static, dynamic	<2
Kayukawa et al., 2019[20]	Indoor, outdoor	Both	< 10
Guerreiro et al., 2019 [21]	Indoor	Static	<10

Table 2.1: Summary of approaches based on coverage and obstacle type.

The table 2.2 below shows a summary of data input and feedback delivering methods adapted in research described above.

Paper	**Input**	**Instruction**	**Drawbacks**
Peng et al., 2010 [2]	Embedded camera on the smartphone	Auditory feedback	Maintaining the tilt of the smartphone
Mocanu et al., 2013 [4]	Video camera	Auditory feedback	Displacement of the camera
Caldini et al., 2015 [3]	Smartphone camera, gyroscope	Auditory feedback	Maintaining the tilt of the smartphone
Kanwal et al., 2015 [31]	Microsoft Kinect Sensor	Auditory (verbal) feedback	The Kinect sensor is not reliable in outdoors. The user requires to carry a standard Kinect sensor, a battery, and a laptop /processor.
Aladren et al., 2016 [19]	RGB-D sensors	Auditory feedback	Blockage of natural sounds.

Everding et al., 2016 [32]	Retina-inspired Dynamic Vision Sensors	Auditory signals	Latency of the system
Lin et al., 2017 [5]	Smartphone camera	Auditory feedback	Large information computing burden.
Mocanu et al., 2017[16]	Video camera	Acoustic warning messages	Issues in Object tracking module.
Velázquez et al., 2018 [6]	Smartphone was used for using GPS coordinate	Tactile display using vibrations.	GPS signal loss and low accuracy.
Katzschmann et al., 2018 [18]	Sensor belt (infrared time-of-flight (TOF) distance sensors and sonar sensors)	Haptic feedback device Vibration patterns	Displacement of the sensor belt.
Bai et al., 2018 [28]	Fisheye and Depth cameras	Auditory feedback (earphones)	Displacement of the device and its components.
Kayukawa et al., 2019[20]	RGB camera	Audible signal	Unnecessary sound emissions
Guerreiro et al., 2019 [21]	LiDAR and stereo camera	Vibrotactile, Audio feedback	Suitcase-shaped navigation is not able to navigate in staircases etc.

Table 2.2: Summary of approaches based on input and output methods

A summary of used object detection algorithms and frameworks is shown in the table 2.3 below.

Paper	**Obstacle Detection Algorithm/s and Framework**	**Deployment Platform**
Peng et al., 2010 [2]	Combination of color histograms, edge cues, and pixel-depth relationship.	Smartphone
Mocanu et al., 2013 [4]	Multiscale Lucas-Kanade algorithm, RANSAC algorithm	Smartphone
Caldini et al., 2015 [3]	Structure from Motion (SfM) algorithm	Smartphone
Kanwal et al., 2015 [31]	The data acquired from the Microsoft Kinect camera is processed to detect Edges using the Sobel edge detector, Corners using Harris&Stephens detector and Blobs using SIFT detector.	Standard Kinect sensor, a battery, and laptop or processor. User will carry all in a backpack
Aladren et al., 2016 [19]	RANSAC algorithm is used for plane detection, Range-based plane extraction algorithm, segmentation algorithm based on range information, and Back Projection Algorithm are used.	RGB-D device (camera), a laptop and headphones
Everding et al., 2016 [32]	Event-based algorithms on visual data is used for depth extraction.	Lightweight device containing headphones connected to a USB sound adapter.
Lin et al., 2017 [5]	Faster R-CNN and YOLO algorithms	Smartphone

Mocanu et al., 2017 [16]	YOLO approach with extensions.	video camera, processing unit, graphical board, headphones.
Velázquez et al., 2018 [6]	GPS, GIS	Smartphone, tactile display
Katzschmann et al., 2018 [18]	After detecting obstacles commands are sent from sensor belt to haptic strap via a Bluetooth transceiver.	Sensor Belt, Haptic Strap
Bai et al., 2018 [28]	Visual SALM and PoI-graph for indoor localization and virtual-blind-road building (By fusing data from cameras and ultrasonic sensor). Way Finding by applying A* algorithm to the PoI-graph	AR glasses connected to CPU and cameras (Fisheye, Depth)
Kayukawa et al., 2019 [20]	Combination of stereo images and CNN-based generic object detector (YOLOv2)	Laptop, Speakers
Guerreiro et al., 2019 [21]	Convolutional Neural Network (CNN), LiDAR for localization and path planning.	Autonomous navigation robot

Table 2.3: Summary of object detection methods adapted in research.

2.2 Object Detection Algorithms

When considering the above summary, in recent years deep learning algorithms have been used for object detection in assisting the blind. RCNN model family, SSD and YOLO (You Only Look Once) algorithm families are popular in object detection [46].

Convolutional Neural Networks (CNN) are commonly used in analyzing images. It divides the image into multiple regions and classifies each region into various classes. The drawback of CNN's is it is not possible to classify more than one different object in one image. Region-based Convolution Neural Network (R-CNN) uses pre-trained CNN and Selective search to create bounding boxes or region proposals. R-CNN has three modules named region proposal, feature extractor and classifier. Slow object detection, higher training time and space are some of the drawbacks of this algorithm. Furthermore, R-CNN training is a multi-stage pipeline.

Fast – RCNN is an improved version of RCNN and still uses the Selective search. Therefore, due to the Selective search used in Fast-RCNN the computation time is still higher. Faster – RCNN replaced the Selective search with a region proposal network. Therefore, Faster- RCNN is faster than Fast-RCNN. Faster- RCNN has two modules. The first module is a CNN to propose regions and the second module is named the Fast-RCNN detector uses proposed regions. The overall performance of Faster-RCNN depends on the performance of each module are they execute sequentially. Faster R-CNN is widely used for object detection tasks. Class label and bounding box coordinates for each object can be obtained from Faster – RCNN. Mask R-CNN is an extension of Faster R-CNN for pixel level segmentation. Mask R-CNN is a combination of a Faster R-CNN that does object detection FCN (Fully Convolutional Network) that does pixel-wise boundary.

YOLO is another popular algorithm family in object detection. It is faster than Faster - RCNN due its simpler architecture. When comparing YOLO and Faster – RCNN, YOLO has difficulty detecting objects that are small and close to each other. But Faster RCNN detects small objects well. However, there is a latency in Faster- RCNN with real-time object detection.

2.3 Simulation Platforms

To perform deep learning, a large number of datasets are required. In autonomous vehicle research, simulators are used to generate a lot more data efficiently, than a real vehicle with a driver could collect at the same time. Further, dangerous situations in simulations do not cause real damage and it is cost-efficient to simulate different scenarios in virtual environments. An autonomous/self-driving vehicle must be accurate all the time to detect, avoid obstacles and take necessary precautions. To develop such an autonomous vehicle, large data sets must be collected to train machine learning models. However, practically it is impossible to collect a large number of data sets from real-world scenarios using a driver. Therefore, to overcome the issue of data generation, simulation platforms are used.

AirSim [35], CARLA [36], DeepDrive are some of the simulation platforms which are being used for autonomous vehicle research. Microsoft AirSim is a simulator for drones and cars built on the Unreal Engine to support research and experiment with deep learning, computer vision, and reinforcement learning algorithms. According to [35] Microsoft AirSim includes a physics engine and the simulator is designed to support new types of vehicles, hardware platforms, and software protocols. CARLA is an open-source simulator for autonomous driving research, developed to support development, training, and validation of autonomous urban driving systems [36]. These simulation platforms provide APIs to generate, train and test data.

- Modeling: These platforms allow us to model vehicles, drones and other object types in the simulation environments. They also expose APIs to interact with the model in the simulation programmatically.
- Sensors: The main sensors available in these simulators are cameras. RGB, Depth, Thermal / IR cameras are supported as well as other sensors such as Lidar and Radar.
- Weather conditions: Using different weather conditions feature is also available in some simulation platforms.

A comparison of Simulation Platforms is shown in the table 2.4 below.

Platform	**Modeling**	**Camera**	**Other sensors**	**Weather**	**Game Engine/ Platform**	**Environments**
AirSim	Vehicle, Drone, Other	RGB, Depth, Thermal, IR	Accelerometer, Gyroscope, Barometer, Magnetometer, Lidar, GPS.	Yes	Unreal Unity	Urban, Other
CARLA	Vehicle	RGB, Depth, Semantic segmentation	Lidar	Yes	Unreal	Urban
DeepDrive	Vehicle	RGB, Depth,		-	Unreal	Urban
NVIDIA Drive	Vehicle	Yes	Lidar Radar, GNSS / IMU	-	NVIDIA DRIVE PX AI computing platform	Urban

Table 2.4: Comparison of Simulation Platforms

Furthermore, these simulators are being used to research purposes in various domains other than in autonomous vehicle research. [37], AirSim-W, a simulation environment which is designed for the domain of wildlife conservation and it includes an African savanna environment. This proposed environment contains improved APIs that follow objects of interest or to fly in zigzag patterns to generate simulated training data which would assist in Identifying poachers and animals.

In [38] a novel application called SPOT (Systematic POacher de-Tector) which is based on [37] is explained. African savanna environment is created using AirSim that allows users to interactively fly through a savanna, while also running SPOT for live, near real-time detection. The research [39] proposed a drone detection model that uses the UAV simulator of the Microsoft AirSim to construct simulation environments. The drones are instantiated inside the simulation environment. In this research, a custom environment is built in Unreal Engine and using the internal camera model of AirSim, images of drones were generated in different angles.

2.4 Conclusion

In this section, a detailed literature review of the background related to blind navigation is provided. Past and current research in the area of blind navigation are analyzed in depth in this section and the evolution of object detection algorithms from CNN, R-CNN to SSD are further discussed here. Due to the high accuracy that can be achieved by deep neural networks, many object detection systems are adapting deep learning-based approaches. This can be seen similarly in blind navigation assistive systems. In the 2.3 section, details of simulation platforms available today and a comparison of such platforms are presented. Simulation platforms are used in various domains as an approach for data generation. The next chapter describes the research methodology and the implementation process of the proposed proof of concept system.

CHAPTER 3

3. Methodology

This chapter presents the design, the process flow, and the methodology of the proposed object detection prototype to assist the blind navigation. The chapter starts with an overall idea of how the system works at a higher level. Section 3.1 describes the design of the system and the process flow and the methodology to build the system is described in section 3.2.

3.1 Research methodology

The research methodology is based on constructive research. Constructive research consists of several phases. A relevant and practical problem must be available to apply the constructive research. After analyzing the problem domain using relevant literature, practical experiences or using other methods a thorough understanding must be gained on the research area. Then considering the identified problems and research gaps, a novel prototype solution must be constructed and the solution could be presented as an algorithm, model or a framework. The constructed solution must be tested and evaluated thoroughly to be considered a theoretical contribution to the field.

Considering the threats that the blind people face while navigating, this research is focused on finding a practical solution to assist them. Although the blind people use the white cane to assist them, due to its limited reachability, it is not an adequate solution. Therefore, more insights of the surroundings must be provided to the navigator. Information of the obstacles with potential threats must be informed to the user based on the distances.

The proposed prototype uses deep neural network-based approaches to detect obstacles and estimate the distance of detected obstacles. This prototype will be deployed in a smartphone and therefore, it is developed as a mobile application. The blind navigator will use the smartphone camera to capture the surroundings in a video stream and after processing the data, the application will provide necessary feedback through audio queues to avoid threatening obstacles in their path.

3.2 Design of the PoC (Proof of Concept Prototype)

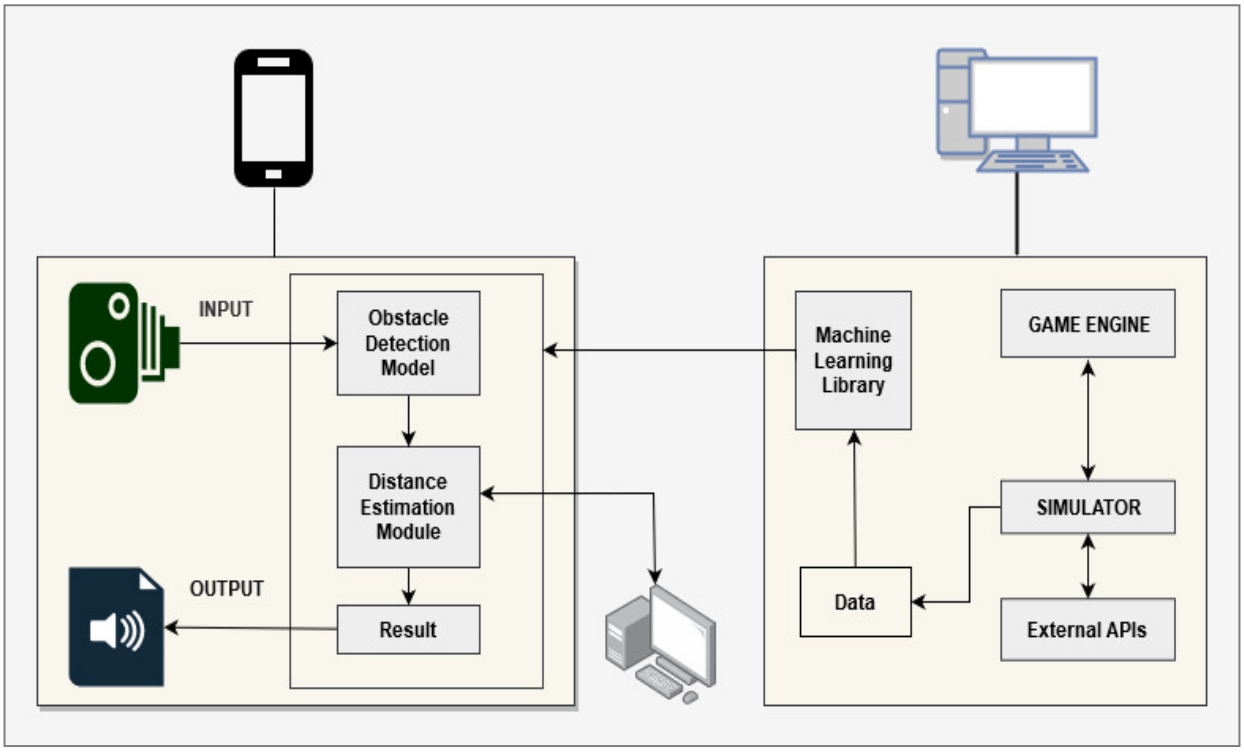


Figure 3.1: Design of the PoC

According to the figure 3.1, to design the prototype following software and hardware are used.

1. Simulation platform
2. Game Engine (used simulation platform is a plugin for game engines)
3. Machine learning Library
4. External sever to perform depth estimation.
5. Mobile device: A suitable android smartphone with a quality camera and an adequate RAM to install the application.

A simulation platform is used to generate data to train a deep learning model. The simulation platform works as a plugin for a game engine and there are APIs available in the simulator that can be extended using external APIs. By extending the available APIs or using other available methods provided by the simulation platform, the desired data generation can be performed. The trained deep learning model is deployed in a mobile platform therefore it's important to choose a suitable architecture that supports mobile deployments.

3.3 Process flow of the (PoC) System

Figure 3.2 illustrates the process flow of the prototype.

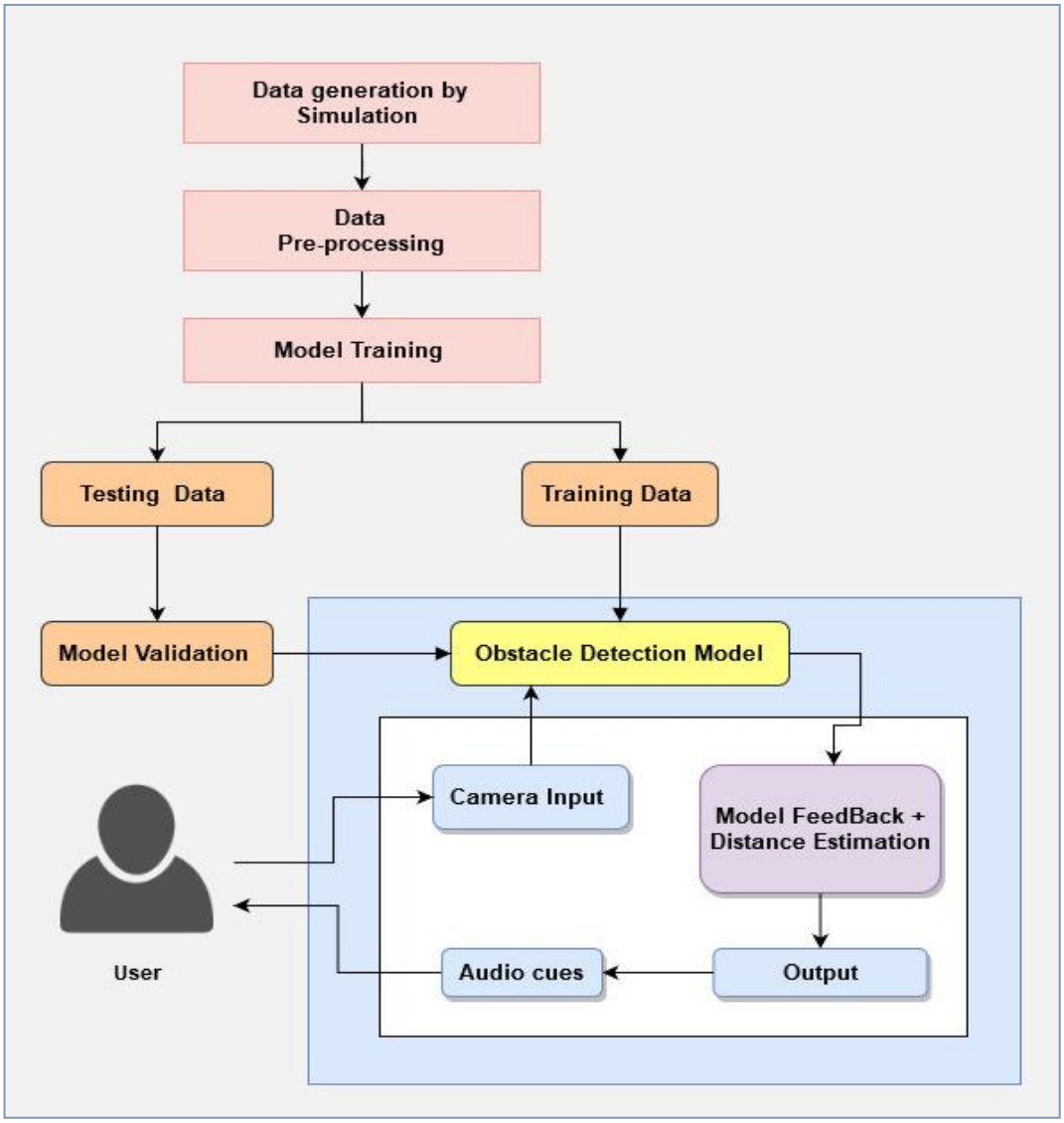


Figure 3.2: Process flow of the prototype.

The process flow to build the prototype system is described in the following section.

3.3.1 Generation of training and testing data

Creating a simulation Environment

For this step, a simulation platform will be used and such available simulation platforms are mostly built on game engines such as Unreal or Unity, etc. It is possible to create highly realistic 3D-scenes for games using such game engines. Similarly, it is possible to create new and customized environments after adding the plugins of the simulator. To create a suitable simulation environment/s it is to possible use environmental modules already available in the respective asset stores of the game engines. Further, it is possible to use single assets and create customized environments for the required purpose.

Data Generation

The simulation platforms allow modeling different types of physics models such as drones, vehicles and other objects. These physics models have cameras dedicated to them. These camera views can be accessed through API calls. For example, the cameras of the car model in AirSim can be accessed via front_center, front_right, front_left, fpv and back_center API calls. AirSim has car, multirotor and other physics models. Furthermore, some platforms allow to simply move a camera using keyboard keys through the simulation environment and retrieve data.

By using the capabilities of the simulation platforms, it is possible to model different objects with sensor capabilities. For an example a person with sensors attached to the body (wearing a headband with two cameras, etc.) can be modeled within the environment, and while simulating the way that person walks, data can be generated. Similarly, methods without physical engines can also be adapted into the data generation process. The simulation platforms allow us to modify or extend the behavior of physics models and cameras through APIs using different programming languages. Data collected through simulation are used to train a deep learning model.

3.3.2 Object detection model training

Deep learning algorithm and machine learning library selection

Using the collected data from the simulation, a deep learning model is trained for obstacle detection. There are machine learning libraries available for this purpose and it is important to select a machine learning platform that supports mobile conversions. Keras [40], pyTorch, Caffe and TensorFlow are some of the popular deep learning libraries. Among such libraries, TensorFlow provides a mobile framework called TensorFlow Lite and pyTorch provides a framework called pyTorch Mobile that supports mobile deployments. To train a deep learning model it is important to have the supported hardware such a good GPU preferably Nvidia, CPU as Core i5 and a RAM of 8GB.

Table 3.1 presents a summary based on whether it is possible to deploy an object detection algorithm on a mobile device or not.

Detection Algorithm	Possible to Implementation on mobile devices
R-CNN	No
Fast R- CNN	No
Faster R-CNN	No
Mask R-CNN	No
SSD	Yes
YOLO	Yes
YOLO 2	Yes

Table 3.1: Summary of DNN algorithms on the possibility of deploying on mobile devices.

Faster R-CNN algorithm uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. But this reduces the real-time performance. Single Shot MultiBox Detector (SSD) is designed for object detection in real-time. SSD speeds up the process by eliminating the region proposal network and to reduce the accuracy loss, SSD includes multi-scale features and default boxes.

SSD's architecture builds on the VGG-16 architecture and discards its fully connected layers. The VGG- 16 network functions as a feature extractor. VGG-16 was used due to its strong performance in high-quality image classification tasks.

Training the Deep Learning model

There are two deep learning methods that perform object recognition.

I. **Training a new model from scratch**

This approach involves using a large dataset and designing a network architecture from scratch. It requires a large amount of training data and needs to set up the layers and weights of the CNN manually. Before train a new model from scratch it is necessary to get a thorough understanding of the outcome of the model. The training time for this type of model is very time-consuming.

II. **Using transfer learning**

A machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular method in computer vision since it reduces the time to train a model from scratch. This method is faster because the model has already been trained on thousands of images.

3.3.3 Depth estimation

Distance estimation using a single camera is considered a challenging task. There are several approaches available for this purpose.

Approach I

To calculate the distance to an object from a single camera, several parameters must be predefined. The ratio of the size of the object in the picture and the height of the object in real-world is the same as the ratio between the focal length and distance of the object and camera. This relationship can be defined as follows.

$$\text{Distance} = \frac{\text{focal length} * \text{ real height of the object } * \text{camera frame height}}{\text{Image height} * \text{ sensor height}} \quad 3.1$$

But there are some drawbacks in this method. The real height of a particular object must be known in advance to calculate the distance. Practically it is not possible to know the height of a random unknown object. Furthermore, the frame height and the sensor height mainly depend on the camera type and the brand. Therefore, every time this calculation is performed on a different type of camera, camera frame height and sensor height must be recalculated.

Approach II

There is a geometric correlation between the focal length of a camera lens (F), the distance from the lens to the target object(O), and the distance between the lens and the projected image (I). This relationship can be defined as follows.

$$\frac{1}{F} = \frac{1}{I} + \frac{1}{O} \quad 3.2$$

Figure 3.3 describes the image formation information for distance estimation in a camera lens.

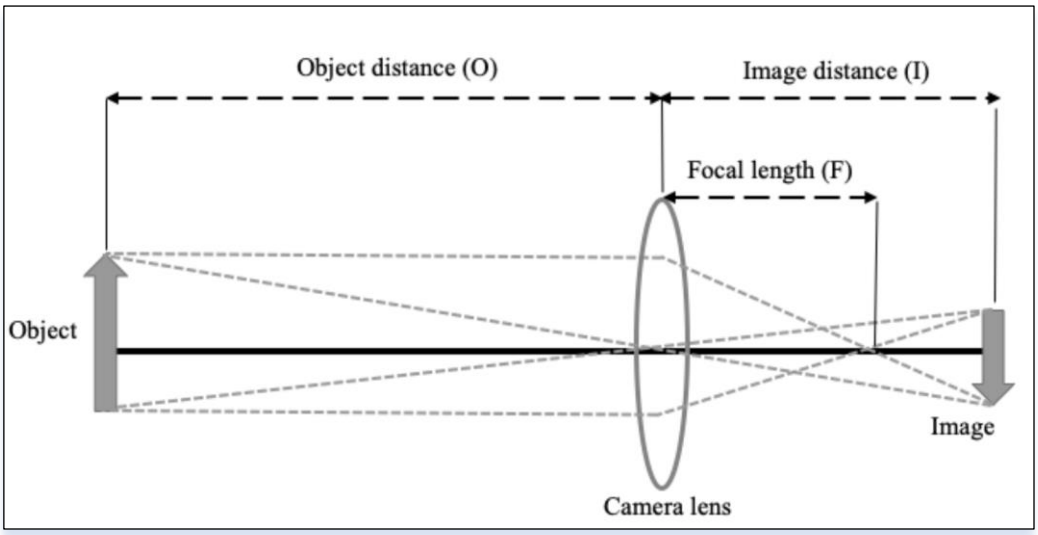


Figure 3.3: Image formation for distance estimation in a camera lens

Approach III

Another approach of estimating distance is based on monocular images. Monodepth [27] is an unsupervised CNN based approach for distance estimation. A deep learning model is trained on stereo images and it makes inference on monocular images. Using this approach, it is possible to get a disparity image of a particular image. In a disparity image, pixels with larger disparities are relatively closer to the camera than the pixels with smaller disparities.

Furthermore, to get only an estimation of the distances, it is possible to apply image binarization by giving a global threshold value to the disparity image. If we use binary thresholding on the disparity image, it will be divided into two black and white segments. From that segmented black and white image, we can get a relative distance estimation for each obstacle in it. As the obstacles in the white area are closer to the camera than the obstacles in the black area.

If we only consider the obstacles on the ground-level it is possible to change the tilt of the smartphone. For example, if the tilt is set to 45 degrees and the smartphone is held 1m above the ground-level, an area of nearly one-meter can be detected. But it is not useful when detecting obstacles above ground level. To reduce the view of the angle of the smartphone camera a small tilt can be used depending on the user's height. Thus, it is possible to reduce the detections of the obstacles that are above the head level of the user.

3.3.4 Audio feedback module

The final output of the obstacle detection system by combining the results of both the obstacle detection model and the distance estimation module will be provided to the user. For this prerecorded audio will be used. When an obstacle is detected the user will be notified about the obstacle type and the directional recommendations.

3.4 Conclusion

In this section, the research methodology, design of the proposed proof of concept system and the process flow of the prototype are presented. As the research methodology, constructive research methodology is adapted in this study. The prototype is deployed in a mobile platform and therefore it is important to select a suitable deep learning architecture that supports mobile deployments. YOLO and SSD are such deep learning algorithms that are fast enough to run on mobile devices. The features and methods provided by the simulation platforms to enable data generation are discussed here. Furthermore, approaches to estimate the distance from a single camera to an obstacle are described here. In the next chapter, the implementation details of the proposed prototype are presented.

CHAPTER 4

4. Implementation

This chapter describes the process carried out to implement the proposed system. Data generation using a simulation platform is described in section 4.1 and deep learning object detection model training is described in section 4.2. Section 4.3 explains the modification of the prototype to integrate the distance estimation module to the prototype.

The overall implementation process is elaborated in figure 4.1 below.

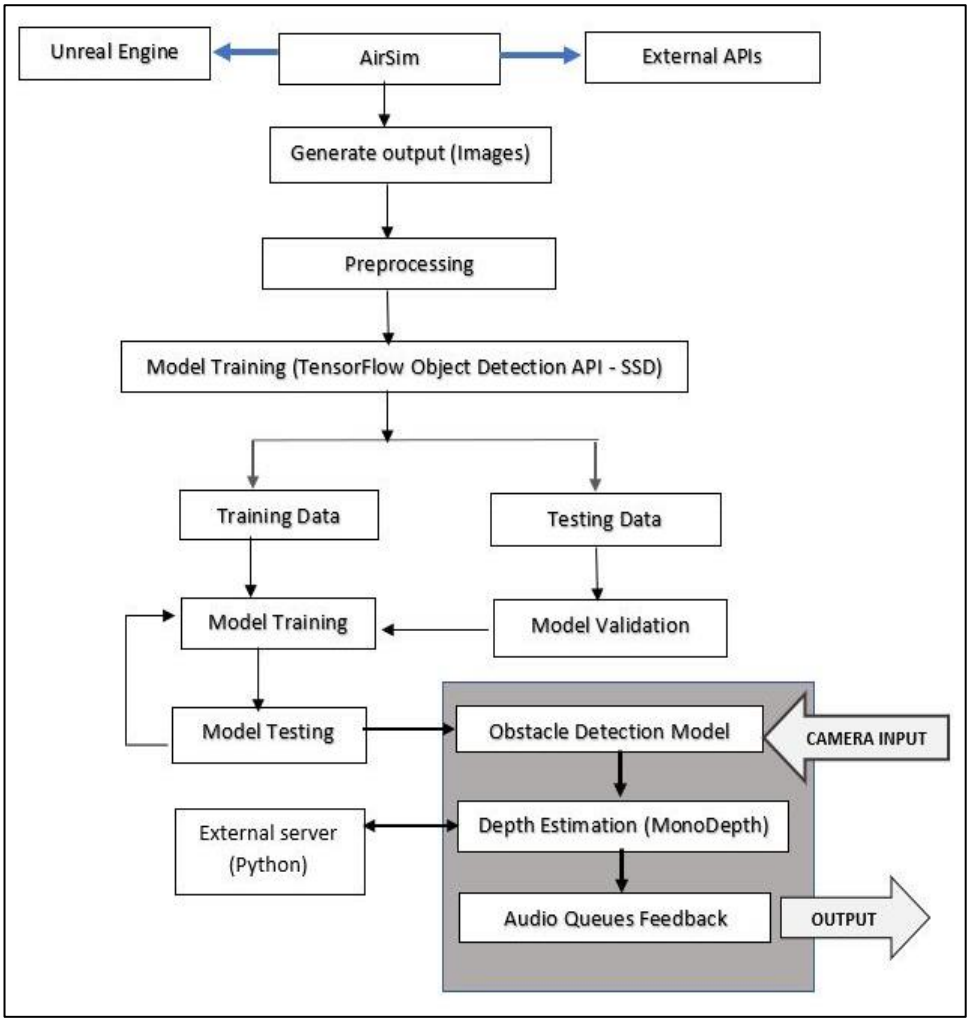


Figure 4.1: Implementation of the system

4.1 Data generation with AirSim by simulation

AirSim is a simulation platform that enables simulation in realistic 3D environments. It is open-source and has an active repository with good documentation and an active community. It also has a number of APIs that provide access to cameras and different physics models and such available APIs can be modified using programming languages as Python or C++. AirSim has different physics models such as vehicle, and multirotor models provided for simulation and it is also possible to simulate without any physics model. In AirSim, this approach is called the 'Computer Vision' mode. In this mode the physics engines are disabled and the cameras are moved in desired paths and angles by the keyboard. Furthermore, AirSim provides RGB, depth and thermal camera output. It also provides output of the sensors such as accelerometer, gyroscope and barometer. CARLA is another simulation platform and it supports vehicle physics model and sensors such as Lidar. AirSim provides more features compared to the other simulation platforms such as CARLA and DeepDrive.

Considering the above facts AirSim was used in developing the proposed system. AirSim is used as a plugin for Unreal Engine which is a suite of integrated tools for game developers to design and build games, simulations, and visualizations.

4.1.1 Setting up AirSim: Creating Unreal projects and integrating AirSim

The AirSim plugin can be downloaded from the Microsoft Git hub repository. Using the Unreal project browser, we can create a new empty project with basic code C++. After creating the project, a suitable simulation environment is downloaded from Unreal Market Place and merge the configuration files and plugins with the project configurations. This process is also possible to do automatically without manually transferring files by directly adding an asset/s into a project. Then the AirSim plugin must be enabled in the Unreal project. This setup was completed using Unreal Engine version 4.23 and Visual Studio 2017 was used as an editor for Unreal projects.

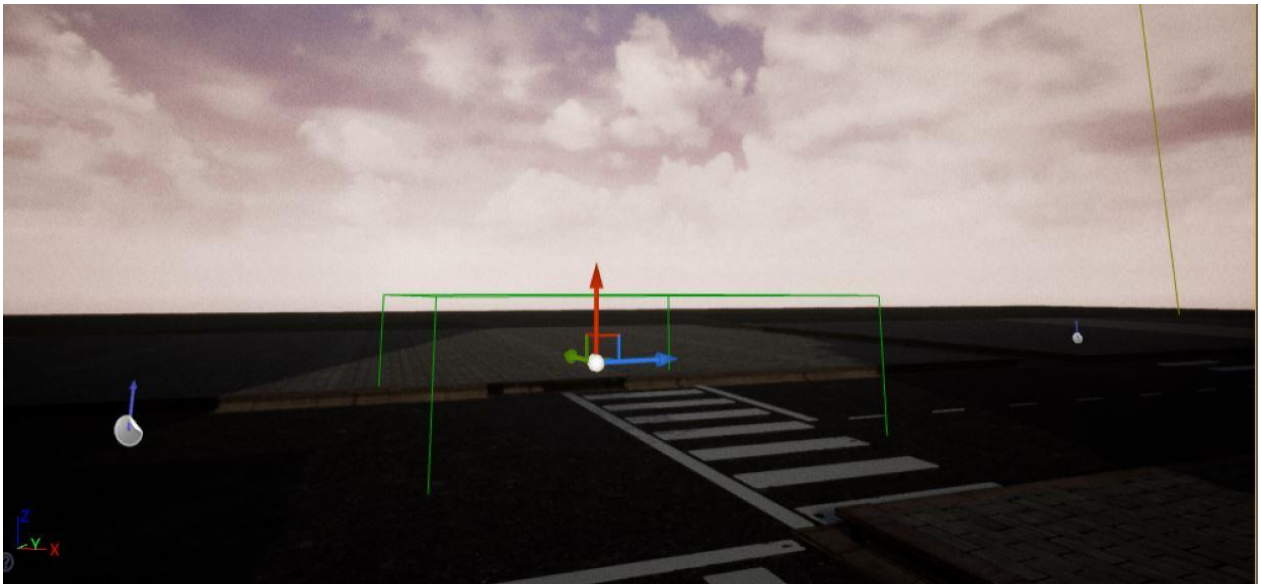


Figure 4.2: Creating Simulation Environments in Unreal Engine – Crossover

4.1.2 Data generation

For this prototype, both custom and already available environments that contain ground-level obstacles such as cars, crosswalks, roadblocks, staircases and above ground-level obstacles such as branches, windows and banners were used.

To generate data the “Computer Vision" mode was used. In this mode, the physics engine is disabled and it is possible to move the camera around using the keyboard. Here the data capturing is controlled by the user. By moving the camera in the simulation environment, it is possible to capture the view of the obstacles in different angles. To capture the images settings json file provided by AirSim can be modified as preferred. After pressing the Record button, AirSim will continuously generate images according to the specifications in settings json file. The collected RGB images with 800px width and 600px height were saved in PNG format. This process continued until an adequate number of images were collected to train the deep learning model.

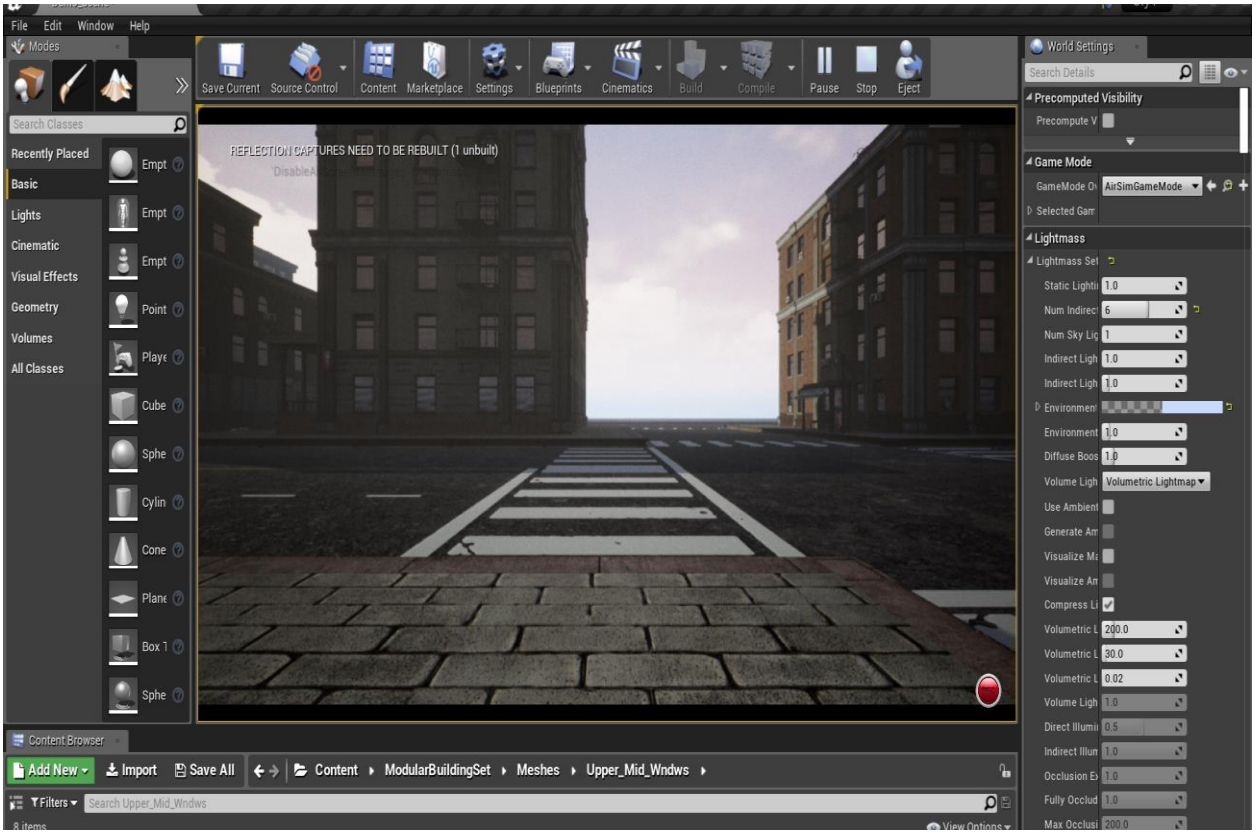


Figure 4.3: Sample of Environment created in Unreal Engine.

4.2 Deep learning model training

TensorFlow [41] is a free and open-source software library that can be used to develop machine learning applications such as neural networks. Furthermore, TensorFlow provides a platform called TensorFlow Lite that enables converting TensorFlow models into mobile compatible versions. TensorFlow Lite [42] is a set of tools to run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and small binary size. Therefore, TensorFlow was used as the machine learning library to train the deep learning model.

TensorFlow has introduced a framework called TensorFlow Object Detection API, which is a framework for creating deep learning networks for object detection. TensorFlow Object Detection API has been introduced to perform object detection in images and videos. The TensorFlow CPU version 1.15 and Python version 3.6 which were installed inside an Anaconda [43] (2019.10) environment, and the TensorFlow Object Detection API were used to train the model.

4.2.1 Data preprocessing

To preprocess the collected data following steps were followed.

1. Creating the relevant dataset
    1.1. Reduce the sizes of images. (800 * 600 pixels)
        Python scripts are available for this reduction.
    1.2. Change the image format from PNG to JPG format.
    1.3. Rename and separate the captured images into two folders. Training folder (80%) and Testing folder (20%).
    1.4. Label the training images using the "labelImg" library.
        LabelImg is a graphical image annotation tool written in Python. The labeling process is done manually. Rectangles should be drawn around the objects which are planned to detect. Every image in training and testing folders must be labeled manually.
2. Convert the images to "TFRecord" file format
    2.1. TensorFlow Object Detection API uses the TFRecord file format. A python script was used to convert the image dataset in training and testing folders to the TensorFlow record format.
3. Create RECORD files for train and test data sets.

4.2.2 Setting up TensorFlow Object Detection API

1. Setting up the TensorFlow Object Detection API Environment.
    1.1. Clone the TensorFlow object detection API
        Using https://github.com/tensorflow/models.git command the repository can be cloned into a specified directory on the computer.

4.2.3 Preparing for Transfer Learning and Training the model

To initialize the deep learning model, the SSD MobileNet architecture was used. MobileNet is an efficient architecture introduced by Google (using depth-wise and pointwise convolutions). It can be used for classification purposes, or as a feature extractor.

1. Downloading the latest checkpoint of the pre-trained ssd_mobilenet_v2_coco version from the TensorFlow model zoo and setting up the configuration.
2. Model Evaluation
   2.1. COCO APIs were installed to use COCO evaluation metrics to evaluate the accuracy of the model during training.
3. Model training
   3.1. The training of the model was carried until the total loss value becomes smaller.

4.2.4 Converting to a mobile version

TensorFlow Lite framework will be used for the conversion.

1. After completing the model training, the inference graph was converted into a tflite file. This file size is around 4.5 MBs. This is the format of TensorFlow Lite which can be loaded into a mobile or embedded device. It is important to reduce the tflite file size below 5MB by compressing.
2. To test this file in an android app, the set-up of the Object detection android example by TensorFlow was used. Android Studio version 3.5 was used to set up this application. Upon the successful installation, its tflite file was exchanged with the trained models tflite file and successfully installed the android application on a Samsung Galaxy J7 and on a Huawei MediaPad t3.

4.3 Modifying the android application

4.3.1 Implementing the depth estimation module

For the depth estimation process, MonoDepth approach - which is a method to estimate monocular depth from a single image is used. There are several other approaches available for depth estimation from a single camera as described in section 3.3.3.

Approach 1: the ratio of the size of the object in the picture and the height of the object in real-world is the same as the ratio between the focal length and distance of the object and camera. In this approach the real height of the object, camera frame height and sensor height along with other parameters must be known in advance to calculate the distance. Since some parameters are specific to a particular brand or a model of the same brand, such parameters must be re-defined when using a camera of a different model /brand.

Approach 2 is using the geometric correlation between the focal length of a camera lens (F), the distance from the lens to the target object(O), and the distance between the lens and the projected image (I). The focal length of the smartphone camera may not be similar in different brands/ models and therefore the parameter F must be re-defined when using a different smartphone. Furthermore, calculating the parameter I in real time can be an issue when considering the limited calculation power of a smartphone.

Approach 3 is estimating distance is based on monocular images. MonoDepth is such a mechanism and a deep learning model is trained on stereo images and it is used to retrieve a disparity image of a given image. The advantage of this approach compared to the above approaches is that this model can work as an independent module so that it is not required to use pre-defined parameters of a particular smartphone. Therefore, MonoDepth is used in this prototype to estimate the distance.

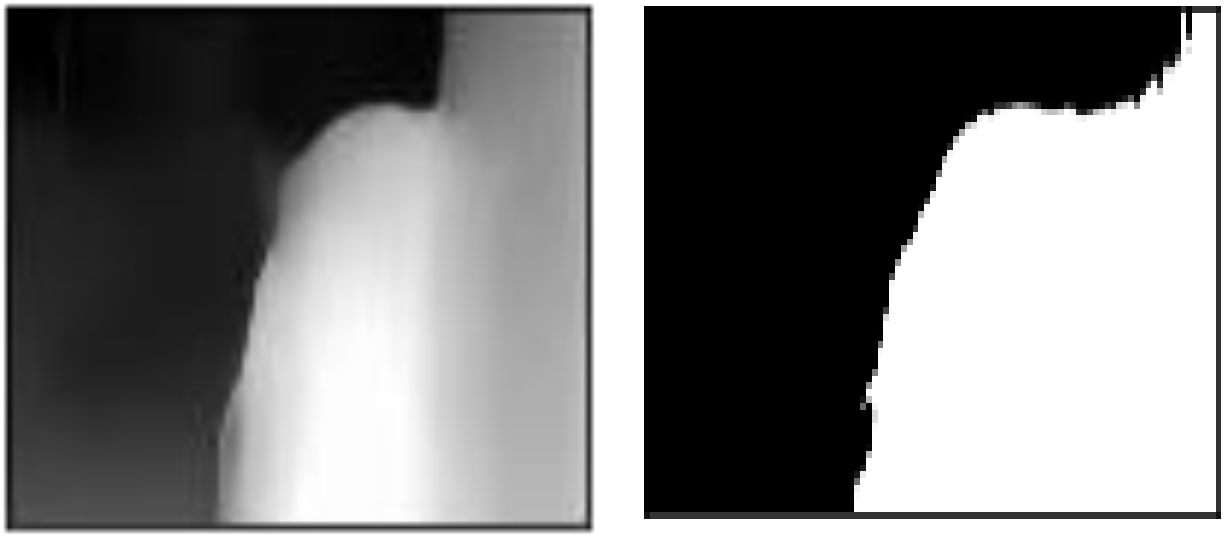
When an image is given as the input to the model a corresponding disparity image can be obtained as the output. After an obstacle is successfully detected by the obstacle detection model, an image of the current view will be immediately captured via the smartphone camera. That image will be used to get a corresponding disparity image. To get the disparity image, a MonoDepth based PyTorch implementation [44] with pre-trained model on Kitty [45] data set was used.

In the following figure 4.4, the original image is in the left side and the corresponding disparity image is in the right side.

Original Image Disparity image

Figure 4.4: Original image and corresponding disparity image

After capturing the disparity image, to capture the areas with low disparity values which are closer to the camera compared to the high disparity areas, we can use a thresholding value to the image. This can be achieved by adding a binary threshold to the disparity image. It will divide the image into two black and white segmented areas. The areas of the white color can be considered as closer regions. In the figure 4.5 below, the greyscale image can be seen in the left side and the image after adding binary thresholding can be seen in the right side.



Greyscale image (Left) Image after adding binary Thresholding using (Right)

Figure 4.5 : Grey Scale image and Image after adding binary thresholding

When the model gives the output of a detected obstacle it also gives the region of the obstacle in a rectangle. Therefore, it is possible to acquire the coordinates of the bounding box of a particular obstacle. Then by comparing the overlapping white areas of the segmented image and the obstacle region of the original image, we can decide whether the detected image is near the user or not. The priority will be given to the obstacles detected in white areas. As the output of the distance estimation module, the obstacles detected in white areas of the segmented image will be selected.

4.3.2 Server-side implementation of the Distance estimation module

The distance estimation module is deployed in an external server. A Python server is needed for the calculation of the disparity image. The captured image from the smartphone camera is sent to server via HTTP from the Android application and the result is received back to the smartphone. After the server receives the image and the bounding box information sent from the smartphone, the corresponding disparity images is produced. Then the binary thresholding is applied on the disparity image. After applying the thresholding, the bounding box coordinates of the obstacle is checked against the white segments of the image with thresholding. After identifying the closer obstacles, the feedback is sent to the smartphone.

To implement the binary thresholding, OpenCV (version 3.3.1) library was used and a global thresholding value of 127 / 255 was used for the thresholding purpose. This divides the image into two black and white segments. For the comparison of bounding box coordinates and white segments another python implementation was used.

4.3.3 Integrating the audio queue feedback module

After receiving the detected obstacle details and the distance estimation details, the obstacles in the close range are communicated to the blind user through audio queues. Prerecorded audio queues are used for this process.

4.4 Conclusion

This chapter describes the complete implementation details of the proposed prototype. AirSim is a simulation platform introduced by Microsoft to support experimentation on machine learning and deep learning. To build this prototype, data sets were generated using AirSim to train the deep learning model for obstacle detection. SSD MobileNet Architecture was used to train the deep learning model using the TensorFlow Object Detection API and to estimate distance, MonoDepth PyTorch implementation was used. To give audio feedback to the user, pre-trained audio queues are used. Using the TensorFlow Lite library the prototype was converted into a mobile compatible version to be deployed in a smartphone.

The next chapter presents the evaluation details of the prototype. Under that section, evaluation of the deep learning model and the usability evaluation of the proposed system are presented.

CHAPTER 5

5. Evaluation and Results

This chapter presents the evaluation and results of the study. The evaluation of the research was carried out in two phases. The first section discusses the evaluation results of the trained deep learning model and the second section presents the evaluation of the usability of the proposed obstacle detection system. This chapter concludes with the discussion on the obtained results of the prototype evaluations.

5. 1 Evaluation of the Object Detection Model

The deep learning model for obstacle detection was trained using the SSD MobileNet architecture using the TensorFlow Machine Learning library. The data to train the deep learning model was generated using simulation in a 3D environment. For this task 1500 images were generated and 1200 (80%) images were used for as training data and 300 (20%) images were used as testing data. Using cameras available in the simulation environment, generating images of various objects from different angles can be performed in a lesser time compared to taking pictures using a camera.

The real-time video stream captured by the smartphone camera is used for the object detection process. Mainly to evaluate an object detector it is important to consider 2 factors. The first factor is the correct detection of objects which is determining whether an object exists in a particular view. The second factor is determining the locations of the object/s with bounding boxes correctly.

For the evaluation of the model, COCO detection metrics were used while training the TensorFlow Object Detection API. The COCO metrics are the official detection metrics used to score the COCO dataset. In COCO evaluation according to the documentation, the Intersection over Union (IoU) threshold ranges from 0.5 to 0.95 with a step size of 0.05 represented as AP@[.5:.05:.95].

The IoU is given by the ratio of the area of intersection and area of union of the predicted bounding box and ground truth bounding box. To decide the correction of a predicted object, Intersection over Union is used. A prediction is considered to be a true positive if IoU is greater than the threshold and a false positive if IoU is lesser than the threshold.

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \qquad 5.1$$

The following table describes how to classify the output from the trained DNN.

		Actual values	
		Positive	Negative
Predicted values	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

Table 5.1: Predicted and Actual values matrix

Precision is defined as the number of true positives divided by the sum of true positives and false positives

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

The recall is defined as the number of true positives divided by the sum of true positives and false negatives. It is the True Positive Rate i.e. Of all the actual positives, how many are True positives predictions.

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

In this study, to evaluate the performance of the trained DNN, Average Precision (AP) was adapted. After the training process was performed up to 5000, 15000, 25000, and 35000 steps the respective AP values were calculated. The mAP for object detection is the average of the AP calculated for all the classes. TensorFlow Object Detection API provides python scripts to evaluate the models that are being trained.

5. 2 Results of the Obstacle Detection Model Evaluation

Class name	AP (AP@0.5IOU)
Car	0.27
Crosswalk	0.31
Window	0.34
Branch	0.41
Tree	0.38
Banner	0.29
mAP (mAP@0.5IOU)	0.33

Table 5.2: AP calculation at checkpoint 5000

Class name	AP (AP@0.5IOU)
Car	0.52
Crosswalk	0.59
Window	0.61
Branch	0.57
Tree	0.54
Banner	0.53
mAP(mAP@0.5IOU)	0.56

Table 5.3: AP calculation at checkpoint 15000

Class name	AP (AP@0.5IOU)
Car	0.60
Crosswalk	0.66
Window	0.74
Branch	0.72
Tree	0.65
Banner	0.61
mAP(mAP@0.5IOU)	0.66

Table 5.4: AP calculation at checkpoint 25000

Class name	AP (AP@0.5IOU)
Car	0.69
Crosswalk	0.69
Window	0.77
Branch	0.75
Tree	0.70
Banner	0.63
mAP (mAP@0.5IOU)	0.70

Table 5.5: AP calculation at checkpoint 35000

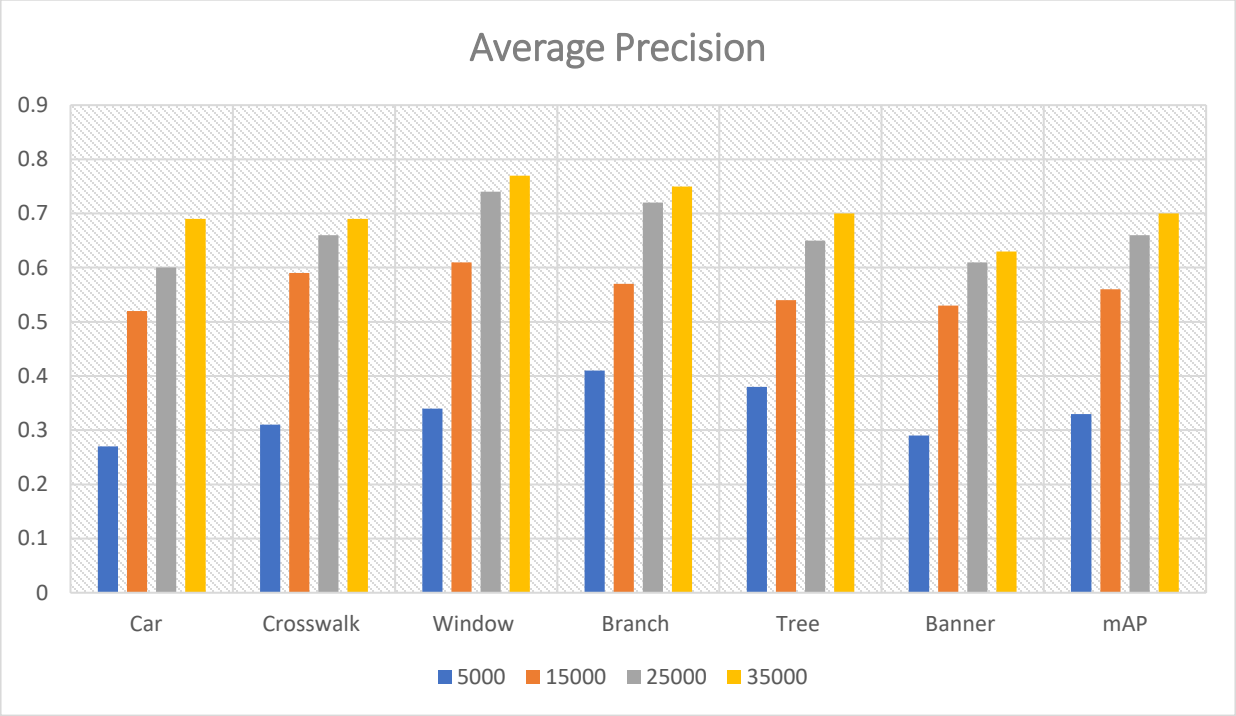


Figure 5.1: Average Precision after 5000, 15000, 25000, and 35000 training steps.

After the 5000th, 15000th, 25000th and 35000th training steps, the calculated mAP values of the classes were 0.33, 0.56, 0.66 and 0.70 respectively. While collecting the data set of 1500 images, a roughly similar number of images per class were generated to reduce the low performance that occurs due to the limited training data for classes. According to figure 5.1 after the first 5000 training steps, all the classes reach more than 25% of average precision (AP) value. After the next 10000 training steps, the average precision of the classes increases significantly. The AP of the window class reached more than 60% AP after 15000 training steps. After 25000 training steps, the AP values of all the classes were increasing but the increasing rates were lower compared to the earlier steps. Similarly, the increasing rates of the AP of classes were very low after 35000 training steps. The mAP values of all the classes show a similar pattern of increasing where the earlier values were increasing at a higher rate and the latter values were increasing at a lower rate. The window and branch class reached more than 70% AP after 35000 training steps.

5. 3 Evaluation of the Obstacle Detection System

5.3.1 User feedback of the Prototype

To get the user feedback of the prototype, it was tested in a controlled environment that contains similar obstacles that were used to generate data. In this environment, blindfolded human test subjects were used to collect the feedback. The experiment was conducted while the completely blindfolded participants walk through the controlled environment using a white cane and the smartphone which contains the prototype.

5.3.2 Experimental Setup

To experiment on the obstacle detection system prototype, four blindfolded persons had participated. Before the experiment, each participant was thoroughly briefed about using the system and the feedback mechanism. Each participant was given 5 minutes to navigate within the experimental environment and the feedback was collected using a questionnaire. Each participant was asked to rate answers on a 1 to 10 scale. Where 1 being very poor and 10 being excellent. Five participants participated in the experiment and they all were below 30 years of age. The smartphone had an uninterrupted internet connection while the experiment was conducting.

The following table describes the participant details and corresponding experimental setup information.

Participant	Gender	Age	Duration (mins)	Time of the Day	Obstacles
1 (P1)	Male	25	5	Morning	Window, tree, branches, staircase
2 (P2)	Male	23	5	Afternoon	Window, tree, branches, staircase
3 (P3)	Female	29	5	Morning	Window, tree, branches, staircase
4 (P4)	Female	26	5	Afternoon	Window, tree, branches, staircase
5 (P5)	Male	28	5	Afternoon	Window, tree, branches, staircase

Table 5.6 : Personal and experimental details of the participants

The following questions were used to collect feedback from the participants.

1. *Whether the place where the smartphone is positioned on the body is comfortable while navigating with the white cane?*
2. *After receiving the feedback from the audio queue did you get enough time to avoid the obstacle without colliding with it?*
3. *If the notifying frequency of audio queues were satisfied?*
4. *Are you satisfied with the overall usability of the obstacle detection system?*

5. 4 Results of the Prototype Evaluation

5.4.1 Summary of the user feedback

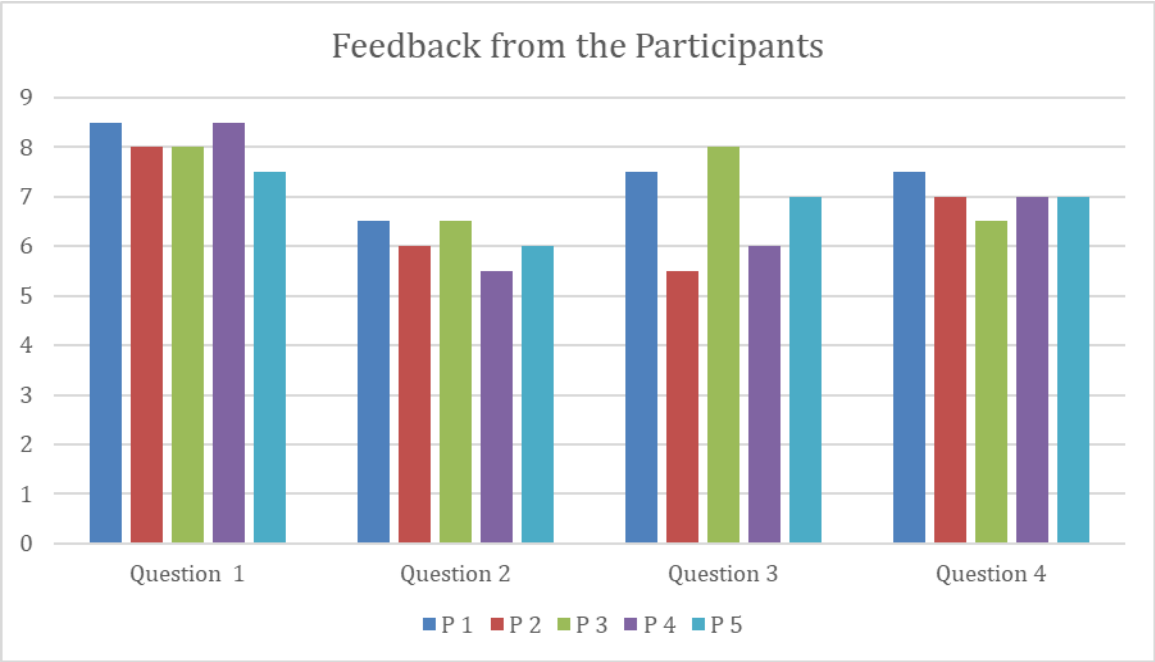


Figure 5.2 : Summarized feedback from the participants

The smartphone was calibrated using a belt around the waist of each participant. Thus, it reduced using the other hand which is not being used to hold the white cane. The participants were satisfied with both hands not being occupied while navigating. According to the instructions of the audio queues, the participants were successfully notified about the obstacles in front of them. But according to them the directional feedback to avoid the obstacles could have been improved.

Notifying frequency of the audio queues received mixed feedback from the participants. To reduce this frequency, only the obstacles which have a confidence score of more than 75% from the obstacle detection model are used for distance estimation. This further reduces the possibilities of false-positive detections. In overall, the participants rated the overall usability of the prototype above 60%.

5.4.2 Analysis of Variance (ANOVA) Test

An Analysis of Variance (ANOVA) test was performed on the feedback data collected from the participants. To perform the ANOVA test, two null hypotheses were developed based on rows and columns of the table 5.7.

	Q1	Q2	Q3	Q4
P1	8.5	6.5	7.5	7.5
P2	8	6	5.5	7
P3	8	6.5	8	6.5
P4	8.5	5.5	6	7
P5	6	6	7	7

Table 5.7: Data to perform ANOVA test

- Null hypothesis for rows
  H0: There is no significant difference between the feedback collected from each participant (regardless of the questions).

- Null hypothesis for columns
  H0: There is no significant difference between the feedback collected for each question (regardless of the participants).

The ANOVA test was performed using Excel Software. ANOVA two-factor without replication test was performed using an alpha value of 0.05. The results of the ANOVA are shown in the figure 5.3.

SUMMARY	Count	Sum	Average	Variance
P1	4	30	7.5	0.66666667
P2	4	26.5	6.625	1.22916667
P3	4	29	7.25	0.75
P4	4	27	6.75	1.75
P5	4	26	6.5	0.33333333
Q1	5	39	7.8	1.075
Q2	5	30.5	6.1	0.175
Q3	5	34	6.8	1.075
Q4	5	35	7	0.125

ANOVA

Source of Variati	SS	df	MS	F	P-value	F crit
Rows	2.95	4	0.7375	1.2919708	0.3272605	3.25916673
Columns	7.3375	3	2.44583333	4.28467153	0.02840766	3.49029482
Error	6.85	12	0.57083333			
Total	17.1375	19				

Figure 5.3: Results from the ANOVA Two- Factor without replication Test

In the top section of figure 5.3, the average and the variance of the feedback recorded from each participant can be obtained and from the bottom section, ANOVA test result can be obtained.

In the first row of the ANOVA result (Rows), the F-critical (value = 3.26) is larger than the F (value= 1.29). The P- value for the Rows is 0.327 which is larger than the alpha value (0.05). Since rows = 0.327 > 0.05 = alpha, the null hypothesis for rows cannot be rejected. Thus, we can observe that there is no significant difference between the feedback of each participant. According to figure 5.2, the feedback received for the overall usability of the system from each participant exceeded 65%. Thus, approvable results on the prototype were obtained from the participants.

In the second row of the ANOVA result (Columns), the F-critical (value = 3.49) is smaller than the F (value= 4.28). The P- value for the Columns is 0.028 which is smaller than the alpha value (0.05). Since rows = 0.028 < 0.05 = alpha, the null hypothesis for columns must be rejected. There is a significant difference between the results obtained from different questions.

According to the figure 5.2, it can be seen that the difference between the results obtained for question one and question two are significantly different. More favorable feedback was collected for question one and the favorability of the feedback collected for question two is lesser.

5. 5 Discussion

The deep neural network trained for obstacle detection reached a mean Average Precision (mAP) of 70% for all the classes. Therefore, the data collected from simulation has shown the validity to be used in object detection tasks. The trained model was used to develop the prototype to assist the blind navigation and to evaluate the usability, an ANOVA test was performed. From the usability test, the hypothesis related to the rows, or the feedback of each participant was recognized as significantly similar. The hypothesis related to the columns, or the feedback on each question was recognized as significantly different. Thus, the participants showed less approval on some features of the prototype, but they showed satisfaction on the overall usability of the system.

The obstacle detection deep learning model which is deployed in the smartphone, functions accurately without an internet connection since it only uses the resources of the smartphone. The inference speed of the trained model is less than one second after deploying it on the smartphone. Therefore, the time to recognize an obstacle takes less than one second. However, for the distance estimation of the detected obstacle, an external server is used. For that, an internet connection must be available in the smartphone. Furthermore, the time taken to retrieve distance estimation results from the server takes around two seconds. This is considerably larger than the obstacle detection time. This makes the overall latency of the system around two seconds. Therefore, an internet connection is mandatory to function the system properly.

However, it is possible to change the tilt of the smartphone only to focus on the ground-level obstacles. With a tilt of 45 degrees towards the ground, the user can detect obstacles on the ground up to one or two meters. This can be used as an alternative approach to estimate distance when an internet connection is not available. The latency of the system is reduced significantly here since the result from the server is not used but the accuracy is also reduced.

Without an internet connection, it is not possible to perform distance estimation of the above ground-level obstacles. Furthermore, this tilt reduces the angle of the smartphone camera significantly. Therefore, a small tilt is recommended to maintain while using the system with an internet connection. The accuracy and latency of the system are reduced when the system is used without an internet connection. In contrast, higher accuracy and a higher latency can be achieved from the system with an internet connection. The obstacle detection of the model is not affected by the internet connection. The table 5.8 shows a comparison when the prototype operates in different conditions.

	With Internet connection	Without internet connection
Accuracy	High	Low
Obstacle detection output from the DNN	Yes	Yes
Reliability	High	Low
Latency	High	Low
Detecting obstacles above ground-level	Possible	Not possible
Detecting obstacles in ground-level	Possible	Possible (< 1.5m)

Table 5.8: Comparison of the prototype in different conditions.

5. 6 Conclusion

This chapter presented the evaluation details of the trained deep neural network (DNN) and the system prototype in depth. To evaluate the usability of the system prototype, experiments were conducted using five blindfolded participants and they showed satisfaction on the usability of the system. Furthermore, the features that need improvements were also identified subsequently. In 5.5 section, the different conditions where the prototype can be used are discussed. In the next chapter includes conclusion and the future work of the thesis.

CHAPTER 6

6. Conclusion and Future Work

This study proposed a deep learning-based obstacle detection mechanism to assist blind navigation. The developed prototype consists of three main modules namely obstacle detection, distance estimation, and audio queue feedback. For obstacle detection, a deep neural network is trained using the data generated by simulation instead of real-world data. To estimate the distance, monocular depth estimation was used and the feedback is communicated to the user via audio queues. The prototype is deployed in a smartphone to improve the user-friendliness. The distance estimation calculation is performed in an external server and the result is sent back to the smartphone.

The main research question focused on this study is,

1. How to find a solution to fill the detection gap of the white cane that arise due to its limited reachability to improve the independent navigation of the blind?

Under that, the following sub-questions were focused.

- 1.1 How to generate data using simulation to detect target obstacles?
- 1.2 How to develop a deep learning model from collected data that can be deployed in a mobile platform?
- 1.3 What object detection algorithms or architecture should be adapted?
- 1.4 What is the most suitable mechanism to estimate the distance of the detected obstacles?

To improve the safe and independent mobility of the blind people, more insights about their current surroundings must be provided to them. Thus, they can avoid threatening obstacles and situations successfully. Considering the real-time performance and the high accuracy of the deep neural networks, a deep learning-based obstacle detection mechanism was proposed in this study to improve the navigation of the blind people.

To generate the data required to train the deep learning model, a simulation platform named AirSim was used. A 3D realistic environment created using the Unreal game engine and to generate data, 'Computer Vision' mode available in AirSim was used. Considering the requirement of mobile deployment of the prototype, to select a suitable deep learning architecture that is fast enough to run on mobile devices, various object detection architectures were studied. The SSD MobileNet architecture was selected to train the deep learning model using the TensorFlow machine library. A deep learning-based distance estimation method using MonoDepth algorithm was used in the prototype to estimate the distance.

After evaluating the deep neural network trained for object detection, the mean Average Precision (mAP) of all the classes reached 70%. To evaluate the usability of the prototype, an experiment was conducted and the feedback was collected using a questioner. According to the feedback the usability and the effectiveness of the prototype system reached more than 60% and some drawbacks of the prototype were identified.

The inference time of the obstacle detection model is always less than one second. But the latency of the depth estimation is higher. Thus, the obstacle detection is performed near real-time, but due to the latency of the distance estimation module, the overall latency of the system is higher. Therefore, as future work, it is expected to find a suitable approach to reduce the processing time of the distance estimation model. Another, important aspect of the system is the ability of functioning without an internet connection since there can be occasions which it is not possible to access the internet. As future work, it is also expected to focus on this aspect.

7. References

[1] "Vision Impairment and Blindness." Accessed November 1, 2019. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment.

[2] Peng, En, Patrick Peursum, Ling Li, and Svetha Venkatesh. "A Smartphone-Based Obstacle Sensor for the Visually Impaired." In Ubiquitous Intelligence and Computing, edited by Zhiwen Yu, Ramiro Liscano, Guanling Chen, Daqing Zhang, and Xingshe Zhou, 6406:590–604. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. https://doi.org/10.1007/978-3-642-16355-5_45.

[3] Caldini, Alessandro, Marco Fanfani, and Carlo Colombo. "Smartphone-Based Obstacle Detection for the Visually Impaired." In Image Analysis and Processing — ICIAP 2015, edited by Vittorio Murino and Enrico Puppo, 9279:480–88. Cham: Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-23231-7_43.

[4] Mocanu, Bogdan, Andrei Bursuc, Titus Zaharia, and tapu. "A Smartphone-Based Obstacle Detection and Classification System for Assisting Visually Impaired People." In 2013 IEEE International Conference on Computer Vision Workshops, 444–51. Sydney, Australia: IEEE, 2013. https://doi.org/10.1109/ICCVW.2013.65.

[5] Lin, Bor-Shing & Lee, Cheng-Che & Chiang, Pei-Ying. (2017). Simple Smartphone-Based Guiding System for Visually Impaired People. Sensors. 17. 1371. 10.3390/s17061371.

[6] Velázquez, R., Pissaloux, E., Rodrigo, P., Carrasco, M., Giannoccaro, N., Lay-Ekuakille, A., 2018. An Outdoor Navigation System for Blind Pedestrians Using GPS and Tactile-Foot Feedback. Applied Sciences 8, 578. https://doi.org/10.3390/app8040578

[7] Liu, Yongqing & Chen, Qi. (2018). Research on Integration of Indoor and Outdoor Positioning in Professional Athletic Training. Proceedings. 2. 295. 10.3390/proceedings2060295.

[8] Kumar Yelamarthi, Daniel Haas, Daniel Nielsen, and Shawn Mothersell. 2010. RFID and GPS integrated navigation system for the visually impaired. In 2010 53rd IEEE International Midwest Symposium on Circuits and Systems. IEEE Press, Piscataway, NJ, USA, 1149–1152. DOI: http://dx.doi.org/10.1109/MWSCAS.2010.5548863

[9] "Accuracy of GPS Data - OpenStreetMap Wiki." Accessed November 1, 2019. https://wiki.openstreetmap.org/wiki/Accuracy_of_GPS_data.

[10] “Electronic Travel Aids for the Blind.” Accessed November 1, 2019. https://www.tsbvi.edu/orientation-and-mobility-items/1974-electronic-travel-aids-for-the-blind.

[11] A. Amedi, et. al “Shape conveyed by visual-to-auditory sensory substitution activates the lateral occipital complex,” Nature Neuroscience, vol.10, no. 6, pp. 687-689, June 2007.

[12] T. Schwarze, M. Lauer, M. Schwaab, M. Romanovas, S. B¨ohm, and T.J¨urgensohn, “A camera-based mobility aid for visually impaired people”,KI-K¨unstliche Intelligenz, pp. 18, 2015.

[13] P. Bach-Y-Rita and S. W. Kercel “Sensory substitution and the human machine interface, “ Trends Cogn Sci., vol. 7, no. 12, pp.541-546, Dec.2003.

[14] Shachar, Maidenbaum, Hanassy Shlomi, Abboud Sami, Buchs Galit, Chebat Daniel-Robert, Levy-Tzedek Shelly, and Amedi Amir. “The “EyeCane”, a New Electronic Travel Aid for the Blind: Technology, Behavior & Swift Learning.” Restorative Neurology and Neuroscience, no. 6 (2014): 813–824. https://doi.org/10.3233/RNN-130351.

[15] Dakopoulos, D., and N.G. Bourbakis. “Wearable Obstacle Avoidance Electronic Travel Aids for Blind: A Survey.” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40, no. 1 (January 2010): 25–35. https://doi.org/10.1109/TSMCC.2009.2021255.

[16] Tapu, Ruxandra & Zaharia, Titus. (2017). Seeing Without Sight — An Automatic Cognition System Dedicated to Blind and Visually Impaired People. 1452-1459. 10.1109/ICCVW.2017.172.

[17] Tapu, Ruxandra & Zaharia, Titus. (2017). DEEP-SEE: Joint Object Detection, Tracking and Recognition with Application to Visually Impaired Navigational Assistance. Sensors. 17. 2473. 10.3390/s17112473.

[18] Katzschmann, Robert K., Brandon Araki, and Daniela Rus. “Safe Local Navigation for Visually Impaired Users With a Time-of-Flight and Haptic Feedback Device.” IEEE Transactions on Neural Systems and Rehabilitation Engineering 26, no. 3 (March 2018): 583–93. https://doi.org/10.1109/TNSRE.2018.2800665.

[19] Aladren, A., G. Lopez-Nicolas, Luis Puig, and Josechu J. Guerrero. “Navigation Assistance for the Visually Impaired Using RGB-D Sensor With Range Expansion.” IEEE Systems Journal 10, no. 3 (September 2016): 922–32. https://doi.org/10.1109/JSYST.2014.2320639

[20] Kayukawa, Seita & Higuchi, Keita & Guerreiro, João & Morishima, Shigeo & Sato, Yoichi & Kitani, Kris & Asakawa, Chieko. (2019). BBeep: A Sonic Collision Avoidance System for Blind Travellers and Nearby Pedestrians. 10.1145/3290605.3300282.

[21] Guerreiro, J., Sato, D., Asakawa, S., Dong, H., Kitani, K.M., Asakawa, C., 2019. CaBot: Designing and Evaluating an Autonomous Navigation Robot for Blind People, in: The 21st International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS '19. Presented at the The 21st International ACM SIGACCESS Conference, ACM Press, Pittsburgh, PA, USA, pp. 68–82. https://doi.org/10.1145/3308561.3353771

[22] Szegedy, Christian et al. "Deep Neural Networks for Object Detection." NIPS (2013).

[23] Ren, S., He, K., Girshick, R., Sun, J., 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497 [cs].

[24] He, K., Gkioxari, G., Dollár, P., Girshick, R., 2018. Mask R-CNN. arXiv:1703.06870 [cs].

[25] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas,NV, USA, 27–30 June 2016; pp. 779–788.

[26] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. arXiv:1512.02325 [cs] 9905, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

[27] C. Godard et al., Unsupervised Monocular Depth Estimation with Left-Right Consistency, CVPR, 2017.

[28] J. Bai, S. Lian, Z. Liu, K. Wang, and D. Liu, "Virtual-Blind-Road Following-Based Wearable Navigation Device for Blind People," *IEEE Trans. Consumer Electron.*, vol. 64, no. 1, pp. 136–143, Feb. 2018, doi: 10.1109/TCE.2018.2812498.

[29] Baker, Simon & Matthews, Iain. (2004). Lucas-Kanade 20 Years On: A Unifying Framework Part 1: The Quantity Approximated, the Warp Update Rule, and the Gradient Descent Approximation. International Journal of Computer Vision - IJCV.

[30] Raguram R., Frahm JM., Pollefeys M. (2008) A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In: Forsyth D., Torr P., Zisserman A. (eds) Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5303. Springer, Berlin, Heidelberg

[31] Kanwal, Nadia, Erkan Bostanci, Keith Currie, and Adrian F. Clark. “A Navigation System for the Visually Impaired: A Fusion of Vision and Depth Sensor.” Applied Bionics and Biomechanics 2015 (2015): 1–16. https://doi.org/10.1155/2015/479857.

[32] L. Everding, L. Walger, V. S. Ghaderi and J. Conradt, "A mobility device for the blind with improved vertical resolution using dynamic vision sensors," *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Munich, 2016, pp. 1-5.doi: 10.1109/HealthCom.2016.7749459

[33] Jk Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR ’17). IEEE, 6517–6525. https://doi.org/ 10.1109/CVPR.2017.690

[34] Ren, S.; He, K.; Girshick, R.; Sum, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Proceedings of the Neural Information Processing System (NIPS), Montreal, QC,Canada, 7–12 December 2015; pp. 1–9.

[35] Shah, Shital, Ashish Kapoor, Debadeepta Dey, and Chris Lovett. “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles.” Field and Service Robotics, July 5, 2017. https://www.microsoft.com/en-us/research/publication/airsim-high-fidelity-visual-physical-simulation-autonomous-vehicles/.

[36] Dosovitskiy, Alexey & Ros, German & Codevilla, Felipe & Lopez, Antonio & Koltun, Vladlen. (2017). CARLA: An Open Urban Driving Simulator.

[37] Bondi, Elizabeth, Lucas Joppa, Milind Tambe, Debadeepta Dey, Ashish Kapoor, Jim Piavis, Shital Shah, et al. “AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs.” In Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS) - COMPASS ’18, 1–12. Menlo Park and San Jose, CA, USA: ACM Press, 2018. https://doi.org/10.1145/3209811.3209880.

[38] Bondi, Elizabeth, Ashish Kapoor, Debadeepta Dey, James Piavis, Shital Shah, Robert Hannaford, Arvind Iyer, Lucas Joppa, and Milind Tambe. “Near Real-Time Detection of Poachers from Drones in AirSim.” In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 5814–16. Stockholm, Sweden: International Joint Conferences on Artificial Intelligence Organization, 2018. https://doi.org/10.24963/ijcai.2018/847.

[39] Carrio, Adrian, Sai Vemprala, Andres Ripoll, Srikanth Saripalli, and Pascual Campoy. “Drone Detection Using Depth Maps.” ArXiv:1808.00259 [Cs], August 1, 2018. http://arxiv.org/abs/1808.00259.

[40] Home - Keras Documentation [WWW Document], n.d. URL https://keras.io/ (accessed 1.20.20).

[41] TensorFlow [WWW Document], n.d. URL https://www.tensorflow.org/ (accessed 1.20.20).

[42] TensorFlow Lite [WWW Document], n.d. URL https://www.tensorflow.org/lite (accessed 1.20.20).

[43] “Anaconda | The World’s Most Popular Data Science Platform,” *Anaconda*. https://www.anaconda.com/ (accessed Nov. 15, 2020).

[44] @article{monodepth2,title = {Digging into Self-Supervised Monocular Depth Prediction},author = {Cl{\'{e}}ment Godard andOisin {Mac Aodha} and Michael Firman and Gabriel J. Brostow}, booktitle = {The International Conference on Computer Vision (ICCV)},month = {October},year = {2019}}

[45] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, Vision meets robotics:The KITTI dataset,Int. J. Robot. Res., vol. 32, no. 11, pp. 12311237,2013.

[46] Chen, Z., Khemmar, R., Decoux, B., Atahouet, A., Ertaud, J.-Y., 2019. Real Time Object Detection, Tracking, and Distance and Motion Estimation based on Deep Learning: Application to Smart Mobility, in: 2019 Eighth International Conference on Emerging Security Technologies (EST). Presented at the 2019 Eighth International Conference on Emerging Security Technologies (EST), IEEE, Colchester, United Kingdom, pp. 1–6. https://doi.org/10.1109/EST.2019.8806222

Appendix

The following questioner was used to collect feedback from the participants in the usability evaluation experiments.

1. ***Whether the place where the smartphone is positioned on the body is comfortable while navigating with the white cane?***

   Please rate your experience according to the following criteria.
   Choose your feedback on a scale from 1 to 10. Select a number between 1 and 10 (inclusive)
   1 -Very Poor 10 – Very Good ☐

2. ***Does the belt holding the smartphone give you any extra hazel while navigating?***

   Yes ☐ No ☐

3. ***After receiving the feedback from the audio queue did you get enough time to avoid the warned the obstacle without colliding with it?***

   Please rate your experience according to the following criteria.
   Choose your feedback on a scale from 1 to 10. Select a number between 1 and 10 (inclusive)
   1 -Very Poor 10 – Very Good ☐

4. ***If the frequency of audio queues were uncomfortable for the ear?***

   Please rate your experience according to the following criteria.
   Choose your feedback on a scale from 1 to 10. Select a number between 1 and 10 (inclusive)
   1 -Very Poor 10 – Very Good ☐

5. ***What is your feedback on the audio queue feedback?***

5.1. Do you like the method of delivering the audio queue to you?

Yes ☐ No ☐

5.2. If you do not like the headphone what alternatives do you suggest?

☐

6. ***Do you have any other improvements that you think the system could have to improve the independent navigation?***

☐

7. ***Please rate the overall usability of the system according to your experience.***

Please rate your experience according to the following criteria.

Choose your feedback on a scale from 1 to 10. Select a number between 1 and 10 (inclusive)

1 -Very Poor 10 – Very Good

☐