

National University of Computer and Emerging Sciences



Obstacle Detection Application for Blind People

Hamna Faisal……………16L-4006

Muneeb Akhtar…………16L-4084

Roshan Ul Haq…………16L-4143

Zain Ghazanfar…………16L-4030

Supervisor:Dr. Kamran Lodhi

B.S. Computer Science

Final Year Project

June 2020

Anti-Plagiarism Declaration

This is to declare that the above publication produced under the:

**Title: Obstacle Detection Application for Blind People**

Is the sole contribution of the author(s) and no part hereof has been reproduced on **as it is** basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: 3rd June, 2020

Student 1

Name: Hamna Faisal

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student 2

Name: Muneeb Akhtar

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student 3

Name: Roshan Ul Haq

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student 4

Name: Zain Ghazanfar

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

[Table of Contents i](#_Toc42076244)

[List of Tables iii](#_Toc42076245)

[List of Figures iv](#_Toc42076246)

[Abstract 1](#_Toc42076247)

[Chapter 1: Introduction 2](#_Toc42076248)

[1.1 Goals and Objectives 3](#_Toc42076249)

[1.2 Scope of the Project 3](#_Toc42076250)

[Chapter 2: Literature Survey 4](#_Toc42076251)

[2.1 Detailed Literature Review 4](#_Toc42076253)

[2.1.1 Definitions 4](#_Toc42076254)

[2.2 Obstacle Detection Using Linear Regression 5](#_Toc42076255)

[2.3 Obstacle Detection Using Grid Algorithm 5](#_Toc42076256)

[2.4 Obstacle Detection Using 3D Geometric Vision 7](#_Toc42076257)

[2.5 Obstacle Detection Using Neural Networks 9](#_Toc42076258)

[Chapter 3: Requirements and Design 14](#_Toc42076259)

[3.1 Functional Requirements 14](#_Toc42076261)

[3.2 Non-Functional Requirements 14](#_Toc42076262)

[3.3 Hardware and Software Requirements 15](#_Toc42076263)

[3.4 System Architecture 16](#_Toc42076264)

[3.5 Architectural Strategies 18](#_Toc42076265)

[3.5.1 User Interface Paradigm 18](#_Toc42076266)

[3.5.2 Use of a specific Product 18](#_Toc42076267)

[3.5.3 Reuse of existing software components to implement various features 18](#_Toc42076268)

[3.5.4 Memory management policies 18](#_Toc42076269)

[3.5.5 Error detection policies 18](#_Toc42076270)

[3.5.6 Future plans for extending the system 18](#_Toc42076271)

[3.6 Use Cases 19](#_Toc42076272)

[3.6.1 User Sign Up 19](#_Toc42076273)

[3.6.2 User Login 20](#_Toc42076274)

[3.6.3 Select Day Mode 21](#_Toc42076275)

[3.6.4 Select Night Mode 22](#_Toc42076276)

[3.6.5 Detect Obstacle Position and Type 23](#_Toc42076277)

[3.6.6 Sign Out 24](#_Toc42076278)

[3.7 GUI 25](#_Toc42076279)

[3.8 Database Design 28](#_Toc42076280)

[3.8.1 ER Diagram 28](#_Toc42076281)

[3.8.2 Data Dictionary 29](#_Toc42076282)

[3.9 System Requirements 29](#_Toc42076283)

[3.10 Design Considerations 30](#_Toc42076284)

[3.10.1 Assumptions 30](#_Toc42076285)

[3.10.2 General Constraints 30](#_Toc42076286)

[3.11 Development Methods 31](#_Toc42076287)

[3.12 Class diagram 32](#_Toc42076288)

[3.13 Sequence diagram 33](#_Toc42076289)

[3.13.1 Get the position of the obstacle 33](#_Toc42076290)

[3.13.2 Get the type of obstacle 34](#_Toc42076291)

[3.14 Policies and Tactics 35](#_Toc42076292)

[3.14.1 Database 35](#_Toc42076293)

[3.14.2 Standard Guidelines 35](#_Toc42076294)

[3.14.3 Algorithms 35](#_Toc42076295)

[3.14.4 Testing 35](#_Toc42076296)

[3.14.5 Installation 35](#_Toc42076297)

[Chapter 4: Implementation and Test Cases 36](#_Toc42076298)

[4.1 Implementation 36](#_Toc42076299)

[4.1.1 Common Object Detection 36](#_Toc42076300)

[4.2 Test case Design and description 39](#_Toc42076301)

[4.2.1 User Login 39](#_Toc42076302)

[4.2.2 User Sign Up 40](#_Toc42076303)

[4.2.3 Select Day Mode 41](#_Toc42076304)

[4.2.4 Select Night Mode 42](#_Toc42076305)

[4.2.5 Login with incorrect credentials 43](#_Toc42076306)

[4.2.6 Obstacle Detection Label Output 44](#_Toc42076307)

[4.2.7 Obstacle Detection Position Output 45](#_Toc42076308)

[4.2.8 Sign Out 46](#_Toc42076309)

[4.2.9 Usability Check 47](#_Toc42076310)

[4.2.10 Performance Check 48](#_Toc42076311)

[4.2.11 Security Check 49](#_Toc42076312)

[4.2.12 Compatibility Check 50](#_Toc42076313)

[4.2.13 Reliability Check 51](#_Toc42076314)

[4.2.14 Modularity Check 52](#_Toc42076315)

[4.3 Test Metrics 52](#_Toc42076316)

[Chapter 5: Experimental Results and Analysis 53](#_Toc42076317)

[5.1 Obstacle Detection Experimental Setup 53](#_Toc42076318)

[5.1.1 Comparison of different models 53](#_Toc42076319)

[Chapter 6: Conclusion 56](#_Toc42076320)

[6.1 Initial Research 56](#_Toc42076321)

[6.2 Work Completed 56](#_Toc42076322)

[References 57](#_Toc42076323)

[Appendix 58](#_Toc42076324)

[Appendix B: Tflite File Creation Code 58](#_Toc42076325)

List of Tables

[Table 1: Comparison of the processing time [10] 7](#_Toc26261421)

[Table 2: Blind User data dictionary 2](#_Toc26261422)9

[Table 3: Image data dictionary 2](#_Toc26261424)9

[Table 4: Pre-Requisites Installation 3](#_Toc26261424)6

[Table 5: Comparison between different models 53](#_Toc26261424)

List of Figures

Figure 1: Unstable Vertices condition [10] 6

Figure 2: Result 1 [10] 6

Figure 3: Result 2 [10] 6

Figure 4: Algorithm flowchart [3] 7

Figure 5: Flowchart of the proposed methodology of obstacle detection and audio warning [6] 9

Figure 6: System Overview [1] 10

Figure 7: Feature Recognition Module [1] 10

Figure 8: Architecture for YOLO model [9] 12

Figure 9: System Architecture Diagram 17

Figure 10: Splash Screen 25

Figure 11: Sign Up Screen 25

Figure 12: Sign-In Screen 26

Figure 13: Home screen 26

Figure 14: Light Mode Screen 27

Figure 15: Night Mode Screen 27

Figure 16: ER Diagram 28

Figure 17: Class Diagram 32

Figure 18: Get position of obstacle Sequence diagram 33

Figure 19: Get type of obstacle Sequence diagram 34

Figure 20: Configuration File 37

Figure 21: Inference Files 38

Figure 22: Object Detection through webcam 54

Figure 23: Night Mode Detection with Flash 54

Figure 24: Day Mode Detection 54

# Abstract

Obstacle detection application aims to improve the mobility and the safety of the visually impaired people by detecting the obstacles in their path. The main objective of our project is to describe and evaluate the specification, design, and implementation of a technologically-advanced yet accessible exploration aid, an Android app that uses the camera on a smartphone to detect and inform visually impaired users about obstacles in their path. If a person or any object is found in the frame of the user the application would detect it and notify the user through voice command. In this report we will give the details of the research we did and the work done till now. Additionally, the blind can interact with the application through the use of voice commands. Furthermore, the methodology we adopted is based on use of Machine Learning (ML) and Object Detection using Computer Vision (CV) techniques to recognize the hurdle and label them accordingly. This document is intended to focus on the design of an obstacle detection system for assisting the visually impaired people. All of the previous technologies used to detect the obstacles are thoroughly analyzed in the document. This document covers the literature review needed to evaluate the methods employed in the past and how will our project overcome the limitations of these methods by incorporating an effective and fast algorithm to benefit the blind people and extending the previous studies by introducing an innovative feature for the blind people to hunt for their belongings thus diminishing reliance on others. Furthermore, the document covers the hardware and software requirements essential for the development of our application. Adopting different architectural strategies into the system and evaluating each strategy helps to ensure an optimized architecture develops. The class diagram and sequential diagrams, both helps to understand the high-level and the low-level design of the system. Graphical User design is provided to give an idea about how our application will look like in real life. Implementation details about the application have been provided. Our system takes real time images of objects through a smartphone camera as input and predicts the type and position of the obstacle detected. A deep learning model is trained on a dataset, after which each obstacle is taken and compared with a stored dataset of objects to predict its type and then through voice command informs the use. Test cases and their results have also been given in the test case section. Highlighting the results obtained, we selected the YOLO model, a single shot detector for faster obstacle detection and recognition as it had 45 frames per second, more than any other detection algorithms. All the algorithms used in the existing work are more concerned about the accuracy of detection but in our research, we were more scrutinized to find the pertinent model that has a greater speed of detection.

# Chapter 1: Introduction

For visually impaired people, a lot of methods have been employed to reduce their dependency on others and assist them to commute properly and safely to their destination. In the old times, people used white canes to detect the approaching hurdles. So, the visually impaired people cannot determine the type of obstacle ahead of them and solely depend upon others to move safely in the desired path. Furthermore, in unseen circumstances the blind people often require assistance to guide them through their surroundings. People with visually impaired eyes cannot rely completely on guide cane. There are some hurdles such as parking car which cannot be predicted. Therefore, such obstacles could cause harm to an unaccompanied blind person.

There is another common method employed that captures static images and explains the user about the objects and their position. The issue with this method is that user first has to take snapshot and then wait for the output feedback, which overall is a very time-consuming process. Regarding depth camera recognition systems, it takes images of the field of view of the user and analyzes the object depth data to determine hurdles and alert the user of any hurdle ahead. Depth cameras provide more information about obstacles as they provide 3D information of an object. Moreover, depth cameras are more costly as compared to standard cameras. That’s why depth cameras are seldom used in actual applications.

In summary, according to the implementations in preceding studies, assistive devices focus on obstacle detection and recognition of the obstacles for the blind person. To overcome the drawbacks of the previous assistive technologies, our research considers smartphone application. Therefore, our project proposes a detection system for visually impaired people; this system utilize a smartphone and machine learning algorithms to recognize various obstacles and help the blind person to identify his belongings without need for assistance from other people. It will have an intuitive interface, ability to detect and recognize the objects, input voice command, perceive for potential obstacles from the dynamic environment, inform the blind person about the hurdle through audio feedback. This will be a low-priced android application than designed equipment which will be easily accessible to the blind community.

In conclusion, this document presents introduction of our application, literature review in detail which include the work done in the field of obstacle detection for the blind people. Requirement and design chapter includes the function, non-function, system, hardware and software requirements for the successful development of our application. System architecture and its strategies are discussed in detail which describes the software technology used in our project, future plan for extending application, hardware and software interface paradigm. The design considerations and the incremental development model of our project is discussed which gives the details of different modules of the application. Graphical User Design is provided for the reader’s convenience to get an idea about how the application looks like. Implementation chapter gives the details about the prototype design and work done so far. Conclusion chapter lists down a plan of the work to be done for FYP-2.

## Goals and Objectives

This app will intend to create a new virtual world for the visually impaired people. The blind can now perceive objects in the environment ahead of time as they cannot use their sense of sight. The blind persons experience will be completely revived, by using the app they don’t need to bounce into objects; they can commute safely to their destination.

To accomplish the main objective of the project, these are some main functionalities which need to be implemented.

* The goal is to identify and recognize the type of obstacles which are the crucial features of this project that will fulfill the primary requirement for aided mobility of the visually impaired by using obstacle detection application.
* The goal is to deliver the obstacle information to the blind by using voice feedback, informing them about the potential obstacles in their path.
* The goal is to allow the visually impaired to detect and recognize the obstacles easily without taking help from other people by using phone camera with the aid of position, for example if a blind person wants to look for the position of the shoes in the house.
* The goal is to store all the real time images captured by the camera into a database so that obstacles could be identified and informing the user of the position or type of obstacle in real time.
* The goal is to develop a system that responds within 3 seconds to the user.
* The goal is to develop a simple and responsive user interface.

## Scope of the Project

This project will benefit the visually impaired community of Pakistan. The lives of the blind people will be at ease while operating this application and the rate of dependency on others for their daily tasks will be reduced. The motive of our project is to make the blind aware of the approaching obstacle and find their belongings that will be asked by the system and they will input their response to the application through voice if that object belongs to them, the system will store that object in its database. There are main things that our users will be able to do using our application:

* Create an account.
* Safely commute to their destination.
* Identify the type and position of obstacle detected.
* Output feedback to the blind about the obstacle.
* Input their voice to the system to respond if an object belongs to them.

With the goals defined and scope decided, the next chapter will be literature review; shadowing over the current technologies used till now to cater this need and how our application will be different from others. Further, all types of requirements will be discussed in order to make this application operational. Implementation and test cases will be examined and following that the experimental results will be analyzed and appropriately concluded.

# Chapter 2: Literature Survey

The following literature review will tell us about the different approaches towards obstacle detection. Each approach is briefly discussed along with its method to apply and their findings. It throws light on different ways to solve the obstacle detection problem for the blind people. By comparing each method’s findings, we came to know which method was used previously to recognize the obstacle and to deduce which approach is on its peak today.



## Detailed Literature Review

This section provides the detailed literature review of what we’ve studied so far which would help us in developing our application.

### 2.1.1 Definitions

Before moving ahead, we will give some definitions for terms which would be used.

* **Structure from Motion (SFM):**

It detects the features in an image, then matches features between each pair of images, refine matching using RANSAC algorithm and 3D model is reconstructed from 2D information [5].

* **You Only Look Once (YOLO):**

A convolutional neural network that train full images and directly upgrade the detection execution. The network divides the image into a grid and boxes are drawn around images and probabilities are predicted for each of these regions and are weighted accordingly [1, 4, 9].

* **Faster R-CNN**

It takes image as an input and provides a convolutional feature map. A separate network is used to predict the region proposals [1].

* **Oriented FAST and rotated BRIEF (ORB)**

Oriented FAST and rotated BRIEF (ORB) is a robust local feature detector that can be used for [object recognition](https://en.wikipedia.org/wiki/Object_recognition) or [3D reconstruction](https://en.wikipedia.org/wiki/3D_reconstruction) [5].

* **Random sample consensus** (**RANSAC**)

The RANSAC algorithm is an iterative learning technique. It does random sampling of the input data. RANSAC uses the voting scheme to find the optimal result by cutting off the outliers [6, 7].

* **Histogram of Oriented Gradients (HOG)**

The histogram of oriented gradients (HOG) is a [feature descriptor](https://en.wikipedia.org/wiki/Feature_descriptor) used for the purpose of [object detection](https://en.wikipedia.org/wiki/Object_detection). The technique counts occurrences of gradient orientation in localized portions of an image [1].

* **Haar feature classifier**

The values of the kernel in a Haar-Feature are manually predicted. A small classifier is created with a small dataset and the weights are trained for each feature. It involves fewer computations, so higher execution speed [1].

* **Lucas-Kanade algorithm (LKA)**

The Lucas-Kanade optical flow algorithm is a technique for tracking the movement of features in images of a scene by associating a vector to every pixel in the scene, obtained by comparing the two consecutive images [7].

## Obstacle Detection Using Linear Regression

This work shadows a technique that used Mean Square Error which is a metric for regression evaluation.

According to the study of Sami et al. [2], aimed at detecting obstacles on the floor for indoor navigation. The input image of the floor was taken, segmented to obtain the region of interest (ROI), compared with all the observed data for different floor types, computed mean square error for every floor type and compared with the heuristic threshold value to obtain the result. Mean Square Error is computed as follows in Equation 1 below:

(1)

Where CF represents the current floor, (F) k represents floor k; m and n represent the number of rows and columns in both images. It provided an accuracy of 96%. Though, Mean Square Error is the most common and simple metric but also the least useful because such a technique where the error is squared may skew the metric towards overestimating or underestimating the model’s badness. Moreover, the blind is not being addressed about the proximity of the obstacle through voice feedback which remains a matter of concern for the further studies.

To conclude, this technique is primitive as now neural networks has taken over that has a more accuracy and low error rate.

## Obstacle Detection Using Grid Algorithm

Maneuverability has been a challenging task for the visually impaired people in this technologically advancing era. A lot of algorithms have been employed to detect the obstacle either they had high computational cost or produced large depth variations.

So, Karthikeyan [10], proposed an algorithm to detect the obstacle using Deformable Grid algorithm. It is a regular grid shaped structure and its deformity depends upon the movement of the obstacles in the view of the camera. The deformable grid holds a property named floating point. This property stated that the vertices are free to move around their current position in its current frame. The magnitude of the vertex’s movement is increased when camera is brought nearer to the obstacle and vice versa. If a region is found to be deformed, the vertices are located at a certain distance from their original position and the edges are elongated or compressed. The unstable vertices are combined to obtain the risk of collision and an obstacle is detected. After deformation the vertices are restored to their original position. Figure 1 below shows the unstable vertices condition.

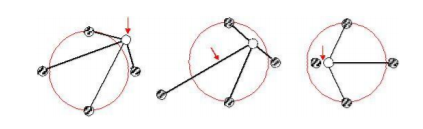


Figure : Unstable Vertices condition [10]

*The above image shows the result when vertices become unstable*

The two tests that were carried out, in the first test the obstacle is approaching the fixed camera as seen in Figure 2.

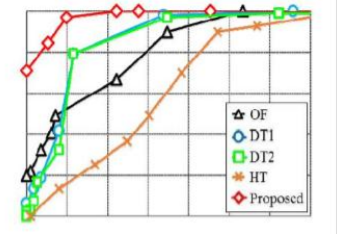


Figure : Result 1 [10]

*The above image shows the result when object is moving towards the camera*

In the second test the camera moving towards the obstacle as seen in Figure 3.

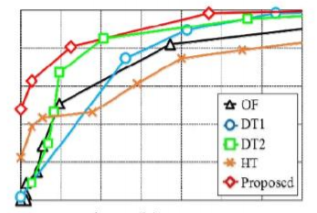


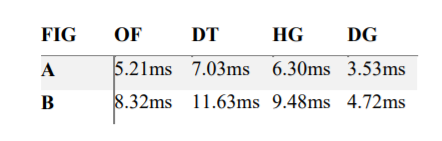
Figure : Result 2 [10]

*The above image shows the result when camera is moving towards the object*

The comparison of the processing time is given in the below table:

Table : Comparison of the processing time [10]

*The below table shows the comparison of different algorithms*



In a nutshell, this method is found to reduce the tracking error and improve the ego-motion of the camera but the chance of false positive rate increases (obstacles detected even if they are not present) as the vertex is out of its region and edges are elongated or compressed as seen in Figure 3. So, in future the chance of false positive rate was tackled by the 3D geometric vision.

## Obstacle Detection Using 3D Geometric Vision

Computer stereo vision is the extraction of 3D information from the images by examining the relative positions of objects in the plane.

#### Obstacle Detection using stereovision for Android-based mobile devices

Andra et al. [3] represented a detection of obstacles using stereovision on mobile devices to avoid traffic accidents. The image is acquired from the camera, rectified, edges detected using Canny-edge algorithm, sparse reconstruction and extended to super pixels obtained from color segmentation. Based on 3D information extracted from stereo reconstruction, the pixels are then grouped together and a bounding box is obtained for each group of super pixels representing an obstacle. The main steps of the procedure are represented in Figure 4.

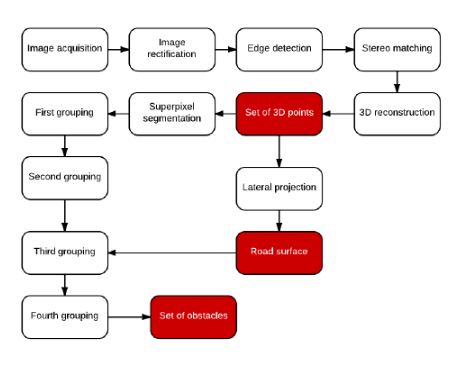


Figure : Algorithm flowchart [3]

*The above image shows the flowchart of the algorithm*

The experiment was carried on static cars in traffic because in real-traffic scenario another stereo system would be required which will make the application really costly and not feasible for the people. Due to the limitations introduced by the hardware of the mobile devices, we obtain small resolution of images, so the system could only perceive obstacles at a distance less than 8 meters.

For future enhancements, an audio feedback and an input to the system would be required to notify the blind about the impending obstacle and to guide him to his destination which is an essential necessity for them.

#### Computer Vision-Based Clear Path Guidance for Blind Wheelchair Users

Ivanchenko et al. [8], aimed at automating the wheel chair to avoid the obstacles in its way that is operated by the visually impaired. The methodology adopted is stereovision by incorporating two stereo cameras in a single package to capture the 3D images.

The images are divided into frames and each frame is analyzed by a ground plane algorithm that tells how high each point lies above or below the ground plane. Moreover, white cane is employed with this system that scans the area in front of the wheelchair for any obstacle. The location and the direction are tracked by the system that should alert the user if any obstacle is detected in a pre-specified range (1.8-3 meters). There were certain limitations to the above approach that the computer vision algorithms, the processing speed was reduced to 5 frames per second. Performance of the system is limited by errors which lead to false positives and false negatives. Many of these errors originate from conditions that degrade image quality. Furthermore, white cane is not feasible and fluid and can get stuck in cracks. Weather conditions can negatively impact the cane traveler’s in-case of a snowfall to check the location of the cane.

Though for future work optimizations of the computer vision algorithms and the use of a faster computer would increase this speed.

#### Smartphone-Based Obstacle Detection for the Visually Impaired

Caldini et al. [5] used Structure from Motion (SFM) algorithm to identify the obstacles aiding the visually impaired to manipulate objects and acoustically perceive their surroundings. This algorithm makes use of gyroscope installed in the smartphone to retrieve the velocity and the angles of the image sequences captured by the mobile camera to compute a rotation matrix which is input as a parameter to the SFM algorithm along with a pair of images.

The SFM first compute the features using ORB feature descriptor, then match images and the features to find the inlier feature correspondences and then reconstruct a 3D image. The 3D objects were labeled obstacles. Two outdoor and one indoor test were carried out where the algorithm computed the correct depth values and obstacles were clearly identifiable. Furthermore, this technique successfully detected all the obstacles on the way but Structure from Motion algorithm has a high computational complexity so other techniques could be used that are easy to implement.

To conclude, it lacked the inability to classify the type of obstacle and most importantly an acoustic interface to provide feedback to the visually impaired people about the obstacle.

#### Assisting the Visually Impaired: Obstacle Detection and Warning System by Acoustic Feedback

Rodriguez et al. [6], proposed an effective technique for detecting the obstacle and thereby informing the user by beep about the approaching obstacle. A disparity map is generated from the images of a stereo camera since every pixel has a disparity value, we are aware of the 3D position (with respect to the camera coordinate frame) we can utilize the 3D information to detect the obstacles in any condition.

RANSAC, a ground planner algorithm produces the grid frame of images. Following that, an image analysis is carried by grid filter and analyzed to account for the potential obstacles in the frame. On detecting the obstacle, a beep is generated to alert the visually impaired about the imminent obstacle and the system is moved to the next frame. The flowchart in Fig. 5 shows the process

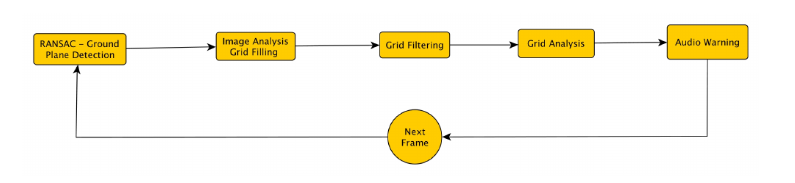


Figure : Flowchart of the proposed methodology of obstacle detection and audio warning [6]

*The above image shows the flow of the proposed method*

Numerous experimental results in indoor and outdoor environments were carried out by the visually impaired people and obtained an accuracy of 95.8% of correct detection and audio warning. Further the stereo camera is attached to the chest of the blind participant so the wear-ability of the system can be improved by incorporating the main components in a smartphone. Stereo camera is not feasible for the blind people as they are an expensive hardware component. Moreover, the accuracy of the system can be enhanced by implementing the system with neural networks so that it can detect the presence of holes and sidewalk curbs which it is not doing in the current system.

## Obstacle Detection Using Neural Networks

Neural Networks, being the emerging advancement in scientific and technological research has been considered the optimal choice for obstacle detection recognition.

#### Simple Smartphone-Based Guiding System for Visually Impaired

Lin et al. [1], in their study regarding the Obstacle Detection aims at assisting the visually impaired people to navigate safely to their destination with comprehensive understanding of the surroundings by diminishing the probability of colliding with the objects.

The system consists of online and offline modes. In an environment with a stable Internet connection, the stable mode is selected to achieve better recognition accuracy and the fast mode to achieve better recognition speed.

The proposed structure consisted of three modules

1. Deep Recognition Module
2. Direction and Distance Module
3. Feature Recognition Module

Below is the structure of this network

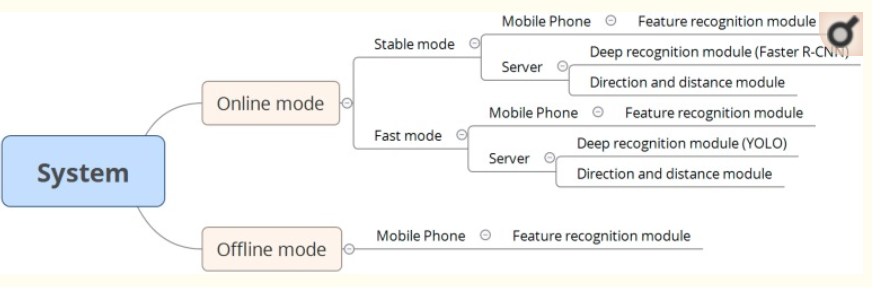


Figure : System Overview [1]

*The above image shows an overview of the proposed system*

* **Feature Recognition Module**

After a smartphone has captured an image, the image is sent to the feature recognition module. The original image will be sent to the remote server when the smartphone is in online mode. The original image will not be sent or delivered anywhere if the device is in offline mode. The module then uses the hair function to process the stairs in a picture and identifies the form of obstacle.

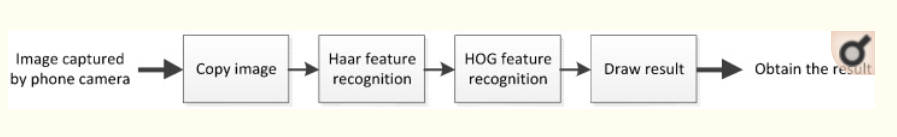


Figure : Feature Recognition Module [1]

*The above figure shows the process of feature recognition module*

* **Deep Recognition Module**

This module executes Faster R-CNN algorithm in the stable mode and YOLO algorithm in the fast mode. The image captured by the smartphone is directly send to the server and is transferred to the deep recognition module based on Faster R-CNN algorithm. A convolutional response map is obtained that is slide through the sliding window operation to get a unique anchor. A coordinate graph is generated and passed to the ROI. Eventually, to evaluate the recognized object type and its position, the non-maximum suppression algorithm is performed. To conclude, Faster R-CNN algorithm is more accurate, but slower. The YOLO algorithm, on the other hand, includes only one network used to determine the position of an obstacle and its class type.

* **Direction and Distance Module**

After the deep recognition module has processed an image, the server sends the recognition results to the direction and distance module. The formula given below is used to calculate the distance and direction of an obstacle from the camera.

(2)

For training the model, Pascal dataset images and images captured by the researchers of this study were mixed together for training and testing and achieved an accuracy of 60%.

There were certain limitations to the above-mentioned approaches that included that these experiments were only based on seven types of obstacles and involved server that enforced large information computing burden thus slowing down the processing speed. Further, there was no method to inform the blind person about the approaching obstacle.

#### Obstacle Detection with voice feedback- YOLOv3+gTTS

Yip [4], in his study regarding obstacle detection focused on a more recent technique to detect the instances of objects in real time and also let the blind know the position of the obstacle through voice feedback which is really helpful for them. The ANN model used You Only Look Once (YOLO) algorithm and was trained with Common Objects in Context (COCO) dataset. It used probability rule to find the probability of all the 80 classes. The class with the highest probability will be the predicted object and is output by voice using the Google’s API.

#### A Smartphone-Based Obstacle Detection and Classification System for Assisting visually impaired people

Tapu et al. [7] in their study discussed the challenges and limitations confronted by the visually impaired people and overcoming the shortcomings and providing convenience for disabled people by adopting the methodology that includes interest point extraction from a stream of images captured by the mobile camera, tracking of the interest point using Lucas- Kanade algorithm , identifying the background motion by RANSAC algorithm, computing the proximity between two interest points by sorting them in descending order on the basis of their occurrences and forming clusters. When all the interest points are included in clusters, the K-NN algorithm is implemented to detect and classify the degree of risk of the obstacles as low or high according to their distance from the camera.

Regarding the classification of the obstacle, the detected objects are divided into 128 cells. From each cell an interest point is extracted that is characterized by the HOG feature descriptor. It captures the object structure from its surrounding regions, using k-means clustering algorithm and map the image patches to the nearest visual word. The total time required for object detection and classification was 132ms/frame which leads to a processing speed around 7 frames/second. Indeed, the adoption of such advanced methodology provided a great deal of facilitation that helped bridge the lacking in the previous adopted models such as the Linear Regression model.

The limitations were that the system lacked audio feedback to alert the visually impaired about the forthcoming obstacle. Secondly, the Lucas- Kanade algorithm has a high computational cost and does not ensure brightness consistency so a more effective technique could be used that obtains better estimation keeping in view the illumination inconsistency.

#### Let Blind People See: Real-Time Visual Recognition with Results Converted to 3D Audio

Jiang et al. [9], aims at exploring the possibility of using the sense of hearing to understand visual objects in the surrounding and inform about the spatial position using 3D audio. The technique adopted is YOLO algorithm. It divides the image into a grid. Each grid calculates confidence score and detects if the class falls in the boxes. The confidence of each box is given below by the Equation

 (3)

The convolutional network for YOLO model contains 24 layers to extract features from the image and 2 fully connected layers to predict the confidence score. The Convolutional architecture for YOLO model is given in Figure 8.

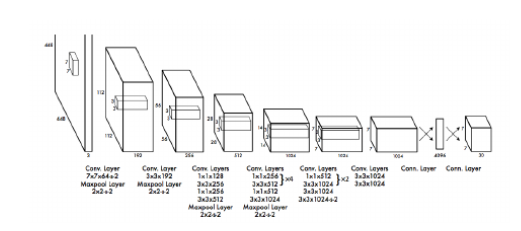


Figure : Architecture for YOLO model [9]

*The above figure shows architecture of the YOLO model*

For depth estimation, GoPro Hero 3 was used because it has a large field of view. Now to calculate distance, the person’s class is given estimated height of 5.5 feet because it’s the average height of human beings. The height values are hard coded for each of the 20 classes in the classifier. Then from the height of the bounding box and the height of the object, the depth of the object was calculated.

Finally, YOLO outputs the probability of each class for each frame. The class with highest probability is taken as confident detection result. To generate 3D sound, Unity’s 3DCeption plugin is used. The two tests carried on the blind people correctly identified the obstacles and the whole process took only 15 seconds. The limitations drawn were that the model can only correctly detect an object if it lies within 2-5m range of the person. Secondly, there is information overload that is, when the system is trying to notify user of multiple objects at the same time. This can be solved by delayed notifications that is, there should be a pause of limited seconds before the next notification is delivered. Moreover, YOLO provides relatively good objective detection with extremely fast processing speed as compared to the Faster R-CNN algorithm studied earlier.

Referring to the above-mentioned literature, further enhancements in the Artificial Neural Network Architecture with multiple hidden layers would result in great accuracy. After reading and analyzing the research articles, we have reached to a conclusion that informing the blind through audio, detecting obstacles at night and the speed of detecting the obstacles in real time has not been dealt in any of the articles which is a major concern for the blind people to reduce their reliance on others. So, we are covering this need in our project, recognizing the obstacles in real time with a greater speed can provide a breakthrough in the matter of obstacle detection and recognition that would benefit a great deal to the blind community.

# Chapter 3: Requirements and Design

This chapter contains the details of functional, non-functional, hardware and software requirements. Architectural design of Obstacle Detection application is explained along with design and implementation details. GUI of application and database design is shown to help readers to understand the use cases. For further details class diagram is shown at the end of this chapter.



## Functional Requirements

This sub-section includes the functional requirements of our project.

* Our application shall allow the blind person to select one of the two modes by tapping on the screen. By one tap, day mode is selected. By two taps, night mode is selected.
* Our application will allow the visually impaired to open the camera through voice input or by tapping on screen.
* Our application shall provide audio output to the vision deficient person about an approaching obstacle.
* Our application shall provide acoustic feedback to the vision deficient person about the type of obstacle detected.
* Our application shall provide voice feedback to the vision deficient person about the distance and position of the impending obstacle from them.
* Our application will allow blind people to save their personal belongings in a database so that they can easily search for their assets later whenever they are recognized.

## Non-Functional Requirements

This sub-section includes the functional requirements of our project.

* **Usability:**

Our interface shall be interactive, responsive and easy to use and shall require no time training. This requirement is non-verifiable.

* **Performance:**

The application shall be optimized to reduce the time cost in retrieving the frames from server and combining them to detect an obstacle. Any number of users will be able to use this application at a time. Response will get back within 5ms and this requirement is testable.

* **Security:**

Data about belongings of a blind person is only accessible to them and not to any other user. Every user will have a separate account and only their belongings data will be available to them.

* **Compatibility:**

The application is Android based so it will be compatible with minimum Android version 6.0 and API level 23 and this requirement is testable.

* **Reliability:**

The accuracy of the obstacle detection by the application shall be high and would have minimum ratio of errors and crashes. The detection of an obstacle will yield an accuracy of around 68% and this requirement is testable.

* **Extensibility:**

Multiple new features could be added to the application over the time and old features could be updated or removed easily.

* **Modularity:**

The application would be divided into different modules which would be easier to maintain.

## Hardware and Software Requirements

Mentioned below are the hardware and software requirements that will be required to develop and deploy the project.

The basic hardware required for our model to work on is

* Android smartphone with minimum 6.0 OS version and API level 23.
* Memory: minimum 2 GB because high computations required.
* Storage: 16GB minimum.
* Rear camera can be 8MP or more but not less because most of our work will be done through it.

If we finish making the app early, we will try to convert the app into a product. For that purpose, the required hardware would be

* Raspberry pi 3
* Memory card
* 8MP minimum camera
* Headphones for audio feedback

The software required to develop each module and then integrate all of them are

* Android Studio 3.5 or 4.0
* Anaconda
* JetBrains’s PyCharm

## System Architecture

The system architecture for the project has been divided into 3 layers, with each layer providing interfaces and communication capabilities to the layers around it.

The 3 layers include:

* **Presentation layer**

This layer provides user interface which is accessible through an android-based application. It will display content useful to a visually impaired person. By tapping on the screen, the corresponding mode will be selected or camera could be opened.

* **Logic Layer**

This module contains the code for our obstacle detection and recognition part. It provides the necessary functions and methods needed to detect and identify an obstacle. Our application is user-oriented so the user module is large enough. To modularize the user module, different modules are formed. The example of such modules includes: detection of obstacle, recognition of the type of obstacle and audio feedback.

* **Data Layer**

The function takes parameters from the logic layer as input and provide the optimal action to be performed as their output. It provides storage of blind’s belongings in a database and retrieval of the obstacles from the COCO Dataset.

Presentation Layer

Graphical User Interface

Interface

Logic Layer

Program Driver Script

Data Layer

Datasets

COCO Dataset

Figure : System Architecture Diagram

*The above figure depicts our project’s architecture diagram*

The basic architecture described can further be elaborated further as shown in Figure 9. The three-layered architecture can be understood as the client who interacts with the interface which acts as top layer, asks the obstacle detection system for the approaching obstacle which is the middle layer and the application accesses data from the database which is the bottom layer. On the basis of the above architecture, the technique adopted is a layered architecture.

## Architectural Strategies

This section includes the architectural strategies that would be incorporated in the system. The strategies are formulated towards the better and the most optimized version of the architecture. The following strategies implemented in the architecture enhances the overall performance of the system.

### User Interface Paradigm

According to the Golden Rules for the Human Computer Interaction, the graphical user interface should be consistent and easy to understand and comprehend. It should keep the user in control and the user interface should be intuitive. It should be made sure that the user interface is easy to use by the blind people.

### Use of a specific Product

The type of programming languages that would be used is Java for android development and python for the background implementation of machine learning algorithm. The Real-time database of Firebase would be used. It provides easy API’s for handling database queries. Android dependencies and the libraries such as Retrofit and other machine learning libraries provided by the Google ML toolkit, such as the Tensor Flow Lite would be incorporated to provide the information about the obstacles.

### Reuse of existing software components to implement various features

Tools such as the Firebase integrations would be used for devising the structure of the application. Built in Machine learning toolkit and its functions would be used for prediction performing functions.

### Memory management policies

Data should be retrieved and inserted into the database using the active internet connection. Insertions in the database cannot be made offline. There should be no memory leaks in the code so that the running application is glitch free.

### Error detection policies

There is basically the usage of the try catch statements that indicate the error detection and policies. Firebase analytics shall be used to detect the errors and can be fixed.

### Future plans for extending the system

For the final year project, the system is only limited to the dataset for retrieving information about the obstacle detected. However, there are future plans for increasing the number of the services and the limitation of the user. In future, we would try to increase to enrich the features also.

## Use Cases

These use cases help the users to understand the functionalities of each actor in the system.

### User Sign Up

The following use case describes the User Sign-Up functionality and the basic flow in order to operate it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Sign Up | | |
| Actors | | User, Admin | | |
| Summary | | The user shall provide their name, email and password on the Sign-Up form and after successful verification, redirect the user to the home page. | | |
| Pre-Conditions | | The user must not be in the database records either added by any of the authorized users or added manually by a developer.  The user must not have an account. | | |
| Post-Conditions | | The user’s session is successfully established and shall be redirected to the home page after Sign-Up and verification through email. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user opens the Sign-Up page. | | 2 | The Sign-Up page is displayed asking for name, email and password. |
| 3 | The user enters valid email. | | 4 | The system verifies the email, establishes a session for the user and redirects the user to the home page. |
| 5 | The user should enter a unique username | | 6 | The system verifies the username in its database. |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email. | | 4-A | The system responds with an error message: *Incorrect email entered.* |
| 5 | The user name should be unique. | | 6-A | The system responds with an error message: *User with this name Already exist.* |

### User Login

The following use case describes the User Login functionality and the basic flow in order to operate it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Login | | |
| Actors | | User, Admin | | |
| Summary | | The user shall provide their email and password on the login form and after successful verification, redirect the user to the home page. | | |
| Pre-Conditions | | The user must be in the database records either added by any of the authorized users or added manually by a developer.  The user must not already be logged in. | | |
| Post-Conditions | | The user’s session is successfully established and shall be redirected to the home page. | | |
| Special Requirements | | None | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user opens the login page. | | 2 | The login page is displayed asking for email and password. |
| 3 | The user enters valid email and password. | | 4 | The system verifies the email and password, establishes a session for the user and redirects the user to the home page. |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email or password. | | 4-A | The system responds with an error message: *Incorrect email or password entered.* |

### Select Day Mode

The following use case describes the select day mode functionality and the basic flow in order to operate it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Day mode | | |
| Actors | | User and System | | |
| Summary | | The system should take a stimulus from the user in the form of one tap on the screen and produce a response in the form of activating light mode of the application. | | |
| Pre-Conditions | | The user must not touch for too long that the set time of release is out. | | |
| Post-Conditions | | The user’s camera is successfully opened and the system starts detecting the obstacles in the surroundings. | | |
| Special Requirements | | User must be logged into the system | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user activates the day mode by a single tap. | | 2 | System open the camera and start capturing photos in burst mode |
| **Alternative Flow** | | | | |
| 1 | The user activates the day mode by a single tap. | | 2-A | The system responds with an error message: *Day mode is not activated.* |

### Select Night Mode

The following use case describes the select night mode functionality and the basic flow in order to operate it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Night mode | | |
| Actors | | User and System | | |
| Summary | | The system should take a stimulus from the user in the form of two taps on the screen and produce a response in the form of activating night mode of the application. | | |
| Pre-Conditions | | The user must not touch for too long that the set time of release is out. | | |
| Post-Conditions | | The user’s camera is successfully opened and starts detecting the obstacle in the nearby area using UV light diction or by Flash mode. | | |
| Special Requirements | | User must be logged into the System | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user activates the night mode by double tap. | | 2 | System opens the camera and start taking photos in burst mode |
| **Alternative Flow** | | | | |
| 1 | The user activates the night mode y double tap. | | 2-A | The system responds with an error message: *Night mode is not activated.* |

### Detect Obstacle Position and Type

The following use case describes the detect obstacle type and position functionality and the basic flow in order to operate it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Obstacle Detection | | |
| Actors | | User and the System | | |
| Summary | | The system will take the real time images of the surroundings and compares it with those stored in the dataset. If found, the type ad position of the obstacle will be informed to the blind through audio feedback. | | |
| Pre-Conditions | | The user must have the camera opened, so that on the time of finding obstacles system helps the user by comparing those stored images with images captured by the smartphone camera to detect it as an obstacle. | | |
| Post-Conditions | | The user’s camera is successfully opened and starts detecting the obstacles in the nearby area using UV light diction or by Flash mode in Night mode and only by camera in Day mode. | | |
| Special Requirements | | User must be logged into the System. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user wants to detect the obstacles | | 2 | System opens the camera and start taking photos in burst mode until an obstacle is detected. |
| **Alternative Flow** | | | | |
| 1 | The user wants to detect the obstacles | | 2-A | The system responds with an error message: *No Obstacle found.* |

### Sign Out

The following use case describes the sign out functionality and the basic flow in order to operate it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Sign Out | | |
| Actors | | User and the System | | |
| Summary | | The user swipes left to log out of the application. | | |
| Pre-Conditions | | The user must be logged into the application | | |
| Post-Conditions | | The user’s session is destroyed and the user is logged out of the application. | | |
| Special Requirements | | User must be logged into the System. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user swipes left to sign out | | 2 | System destroys the user’s session and the user is logged out of the application. |
| **Alternative Flow** | | | | |
| 1 | The user swipes left to sign out | | 2-A | The system responds with an error message: *Not signed out.* |

## GUI

This section includes the GUI dumps that will show the users how functional requirements will be performed on interface.

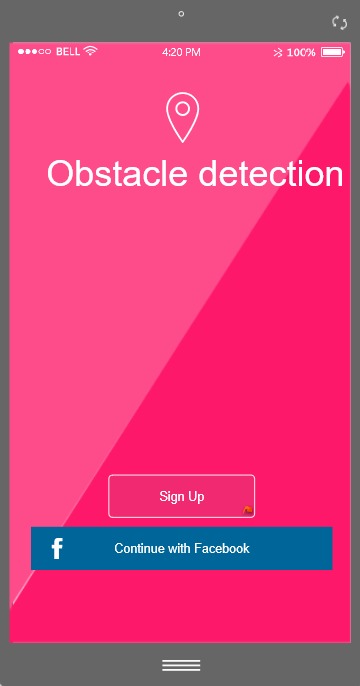


Figure : Splash Screen

*This is the splash screen that will prompt the user to sign up with Facebook or application*

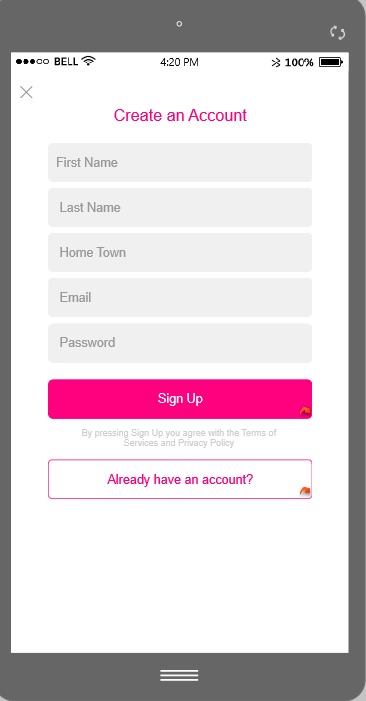


Figure : Sign Up Screen

*If a user is not registered already, he shall sign up to the system*

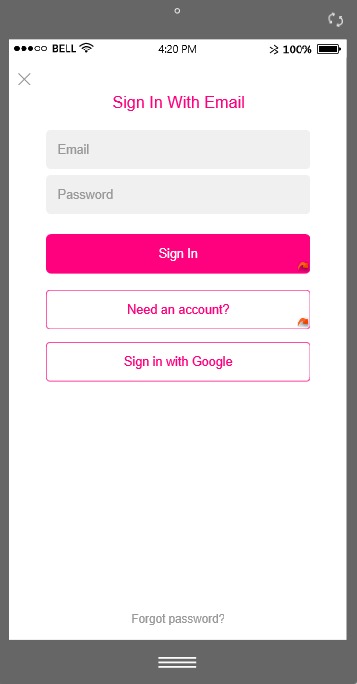


Figure : Sign-In Screen

*If the user has already signed up, they sign in the next time*

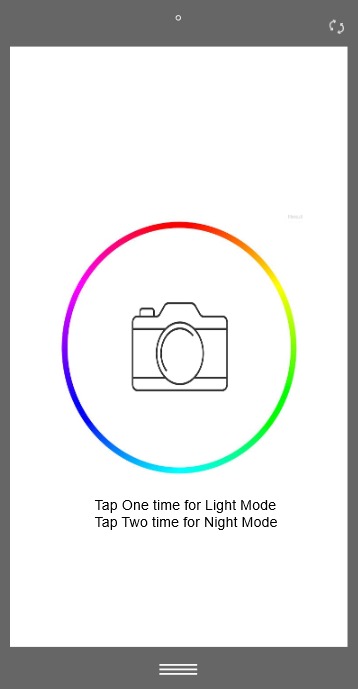


Figure : Home screen

*The home screen shows two modes: Light Mode and Night Mode*

The above screen shows an example of how our application’s home screen will look like. The user single tap on the screen to switch to the light mode and double tap on the screen to switch to the night mode.

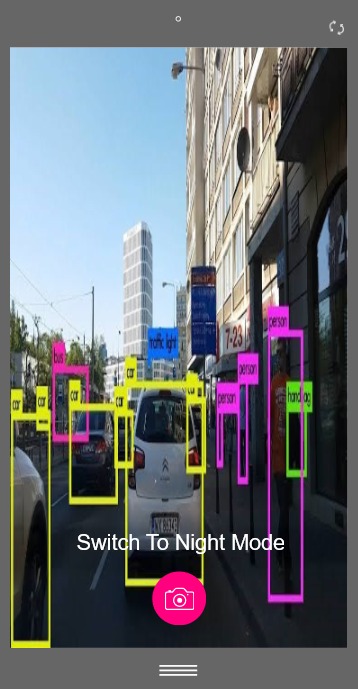


Figure : Light Mode Screen

*By single tap, the light mode of the application is selected and camera is opened*

The above screen shows an example of how our application will look like when light mode is selected. The blind person single tap on the home screen, the light mode screen is switched and the camera is opened to start detecting the obstacles.

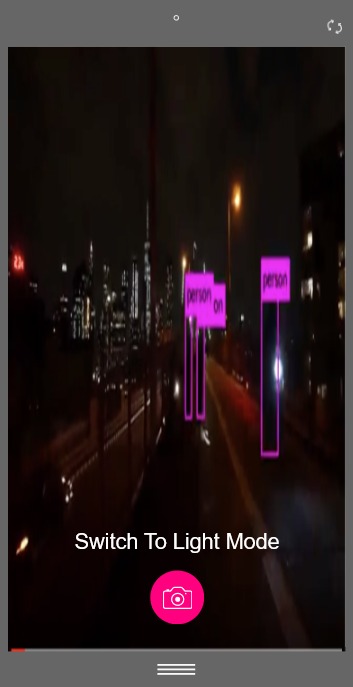


Figure : Night Mode Screen

*By double tap, the night mode of the application is selected and camera is opened*

The above screen shows an example of how our application will look like when night mode is selected. The blind person double tap on the home screen, the night mode screen is switched and the camera is opened to start detecting the obstacles.

## Database Design

This section includes the ER Diagram and the Data Dictionary of our project.

### ER Diagram

The following diagram shows the ER diagram of our project

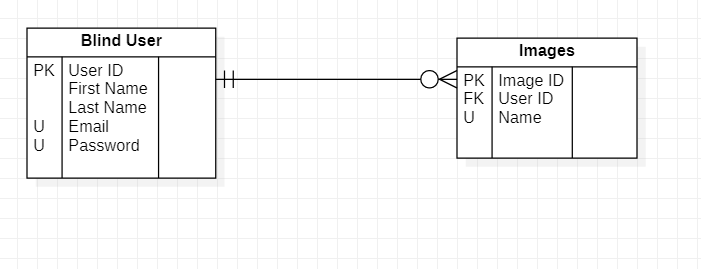


Figure : ER Diagram

ER diagram shows the complete interaction of system entries with other system entries.

The above diagram shows relationship of one entry with other entries and clearly specifies the primary key and foreign keys of each entry.

### Data Dictionary

A Data Dictionary depicts what the data means and how it interrelates and is time-consuming.

Table 2: Blind User data dictionary

Blind User data dictionary shows the attributes data types, field size, description and an example of attribute values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Description** | **Example** |
| User ID | Integer | 5 | Unique User ID | 10245 |
| Email | Text | 25 | Unique User Email address | [user@gmail.com](mailto:user@gmail.com) |
| First Name | Text | 20 | First name of the user | Hamna |
| Last Name | Text | 20 | Last name of the user | Faisal |
| Password | Text | 20 | Password of user account | 123456 |

Table 3: Image data dictionary

Image data dictionary shows the attributes data types, field size, description and an example of attribute values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Description** | **Example** |
| Image ID | Integer | 1 | Unique image ID | 1 |
| User ID | Integer | 5 | Foreign key from user table | [10876](mailto:user@gmail.com) |
| Name | Text | 20 | Name of the images in the COCO dataset | wallet |

## System Requirements

This section includes the hardware and the software requirements that are required to develop and deploy the android application of the system. The system would firstly allow the user to sign in the application or create account. Once the user gets signed in, he would be able to detect the obstacles and recognize the type of obstacle by the phone camera through audio feedback. The blind person would be able to detect obstacles both in day mode and night mode by the help of the flash. By a single tap, day mode would be selected and by double tap, night mode is selected. Moreover, the visually impaired will be able to search for their belongings in their surroundings. This is just the basic workflow or the domain overview of the application.

## Design Considerations

Following are the major requirements and design considerations that must be addressed before the beginning of development procedure. This section includes all the assumptions and all the general constraints that limit the design and its considerations.

### Assumptions

These are the requirements that we are expecting the users of our application to have in order to use and download our application

* The users must have android devices with android version 6.0 (Marshmallow) or above.
* The users have a basic understanding of what the system is trying to achieve.
* The users are familiar with the operationality of the application.
* The devices must meet the minimum storage requirement to download this application.

### General Constraints

The following list consist the global constraints that have a significant impact on the design of system.

* The user should have an android based smartphone with android version 6.0 or above.
* High processing and computation power are required to quickly train the model on dataset and process the query to inform the blind person about the approaching obstacle.
* The system should have enough memory to make sure that system keeps on running and doesn’t stops during process.
* A good internet connection would be required to reduce the transmission time of data between server and application.
* It should be authenticated that the credentials entered are correct and the user should be displayed with their particular account information and no user can access an account other than their own.

## Development Methods

To develop and manage an application that would be doing real-time object detection and at the same time informing the user through audio feedback, many object detection models were studied. Some models had a high accuracy rate, but their speed of detection was low. As we will be detecting objects in real time, it is necessary that our response time is as low as possible. Keeping all of this in mind, we decided to use You Only Look Once model which is known as YOLO. It is a regression-based technique in which it takes an image and simultaneously learns the bounding box coordinates and assigns probabilities to all class labels. We can obtain 45FPS on a GPU using YOLO.

In our first module, we will get images in real-time and convert them in gray scale using OpenCV and python. This is done to reduce the complexity of the image and thus fewer loads on our model. These images will then be sent to our server where we apply our model to detect and recognize obstacles. The second approach would be to do all the detection and recognition inside our application. For night, we will be capturing images with flash on, thus the user can use our app at any time of the day and night.

In our second module, we apply YOLO model on gray scale images. For our dataset, we will be using COCO dataset. It contains 80 classes for objects that can be found in daily life. YOLO will use this dataset to find the probability of each class. Thus, the object will lie in the class with the highest probability. Moreover, we will be calculating the distance of the obstacle from the user so that the user will know when to move aside. Furthermore, our model will also detect the position of the obstacle in the image. Here the user will know if he needs to move left or right.

In our third module, we will be implementing a new feature, in which the user can store pictures of his belongings beforehand. Now next time if the user comes across his belongings, our system will answer accordingly. This will help the user in knowing that a particular object belongs to him.

In our fourth module, we will convert the result, i.e.name of the obstacle, its distance from the user and the position of it in the picture from our previous module into audio feedback using Google’s text-to-speech API.

For our last module, we will develop an interface for our app which will be suitable for blinds to use. For instance, if the user taps on the screen once, it will open the camera without flash. For night mode, user needs to double tap on screen so that camera and flash both opens simultaneously. The user will need help of someone for first time sign in and when he wants to take picture of his belongings.

## Class diagram

This section includes the detailed class diagram of the whole system to have a better understanding.

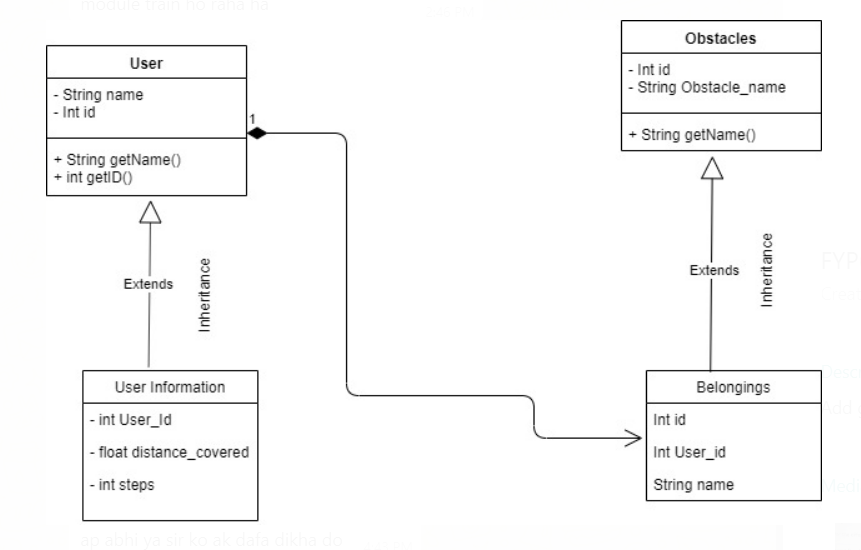


Figure 17: Class Diagram

*The above figure depicts the class diagram for our project*

The above diagram shows the class diagram of our application, clearly exhibiting the relation of the entities with one another.

## Sequence diagram

This section provides an understanding of the low-level design of our system. The sequential diagrams tell us about how the important use cases of our system would be executed.

### Get the position of the obstacle

When the query for recognizing the position of obstacle is requested by the user, the following sequence of actions is performed.

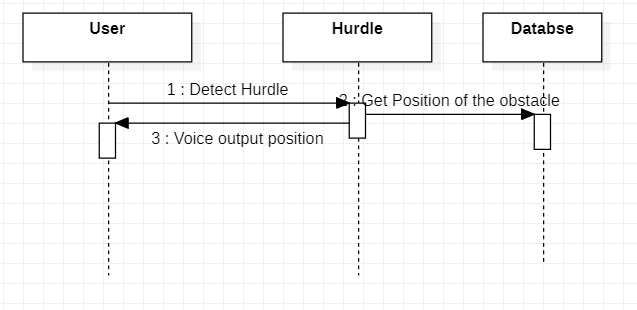


Figure 18: Get position of obstacle Sequence diagram

*The above figure shows the sequence diagram of getting the position of the obstacle*

### Get the type of obstacle

When the query for recognizing the type of obstacle is requested by the user, the following sequence of actions is performed.

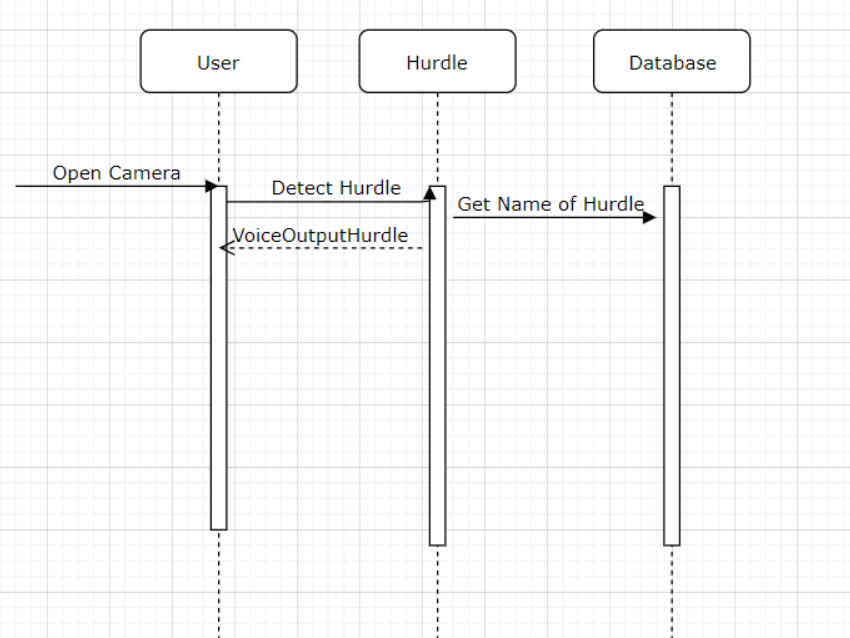


Figure 19: Get type of obstacle Sequence diagram

*The above figure shows the sequence diagram of getting the name of the obstacle*

The above diagram shows the sequence diagrams of our application, clearly exhibiting an understanding of the low-level design of our system.

## Policies and Tactics

Following policies and tactics would be required while formulating the system.

### Database

Third party libraries shall play a very important part in the development of the software. The detection, recognition and searching using the machine learning algorithms shall be using python libraries. Different API’s would be formulated and called on the need basis.

### Standard Guidelines

All the guidelines and the standards would be followed while developing the software. All the good programing practices and use of guidelines and platform documentation shall be considered and referred in order to follow the standards of the platform. Coding guidelines and conventions would be followed while coding throughout the application.

### Algorithms

Tiny Yolo v3 algorithm would be implemented for accurate detection and recognition of the obstacles as accuracy is a crucial factor in our case since the safety of the visually impaired people is highly vital.

### Testing

Testing the system is an important part for our project. White and Black box testing would both be used alongside, to produce a product that is free from all glitches and is smooth when running. In white box testing, detailed implementation of the code is done through by the tester. In black box, only the application is tested before the deployment on the device. Using both these approaches, the chances of error decreases.

### Installation

The application would readily be made available on Google Play Store. The users can download the application and install it into their devices.

The chapter clearly exhibits all the functional, non-functional, hardware and software requirements in order to run the application. Moreover, class diagram, sequence diagrams and GUI screens are provided so that anyone using our application can easily reproduce the system.

# Chapter 4: Implementation and Test Cases

In this chapter, we have discussed some of our strategies, platforms, API’s and algorithms that were implemented in our prototype. In our prototype, we have used COCO dataset for obstacle detection. We have used platforms like Anaconda for the obstacle detection and recognition. Moreover, Python was used for the development of different algorithms such as Tiny YOLO v3. Tensor Flow Lite Library was used to train the model.

The entire process consists of common object detection

## 4.1 Implementation

This chapter will discuss the implementation strategies, algorithms and the dataset used in detail so that the user has a better idea of our application.

### 4.1.1 Common Object Detection

The method implemented in this case is Tensor flow object detection API integrated with COCO dataset. The entire process runs on Tensor flow. The process description is as follow:

* **Tensor flow Installation**

The installation on anaconda is as follows:

conda create -n tensorflow pip python=3.6

When the necessary installation is done then the tensors are active by using the following command:

activate tensorflow

Now the require tensor flow is installed in your system.

If the required tensorflow is not that much supportive then you can use the following command to upgrade it.

pip install --ignore-installed --upgrade tensorflow==1.15.0

The version that we use in our model is 1.15.0.

* **Pre-Requisites Installation**

Table 4: Pre-Requisites Installation

This table shows the pre-requisites required to install

|  |  |
| --- | --- |
| Name | Version Build |
| pillow | 5.4.1-py36hdc69c19\_0 |
| lxml | 4.3.1-py36h1350720\_0 |
| jupyter | 1.0.0-py36\_7 |
| matplotlib | 3.0.2-py36hc8f65d3\_0 |
| Open cv | 3.4.2-py36h40b0b35\_0 |

* **Tensor flow Model**

The tensor flow model repository is downloaded and the object detection model is used to get the appropriate results for detection.

* **Protobufs Installation**

The Tensor flow Object Detection API uses Protobufs to configure model and training parameters. Before the framework can be used, the Protobuf libraries must be downloaded and compiled.

protoc object\_detection/protos/\*.proto --python\_out=.

The version that we use is protoc 3.5.1.

* **COCO Dataset API download**

The COCO API is downloaded and used with mirror changes like holes in road, pole etc. are added through custom frozen files. The link of COCO API is as follow:

https://github.com/philferriere/cocoapi

* **Test Installation**

The installation is tested using the following command on Anaconda:

jupyter notebook

* **Configuration:**

The configuration files used in this case are YOLO v3 with dark net configuration. The file contains the number of classes which is 90 in this case of COCO API.

Running on Jupyter notebook through Google collab.

The configuration file runs and the following steps are created:



Figure 20: Configuration File

This image shows the steps created when configuration file is generated

Then the folder of coco object detection API is created with inference graph. The inference graph contains the inferences of object detection with optimal graphs of tensors. The folder created in this case is as follow:

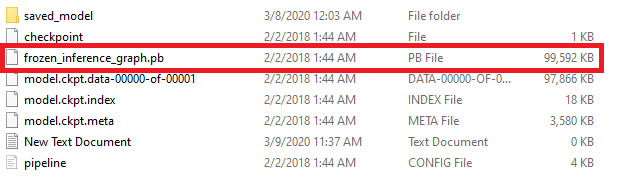


Figure 21: Inference Files

This image shows inference files created

This frozen\_infrence\_graph.pb file run on python IDE and the camera is opened with OPENCV code written by us that take frozen graph and detect the respective images.

* **Tfrecord File Creation:**

In order to read data efficiently, it is often helpful to store it in a set of files of size 100-200MB that can be read linearly. This is done by using the tfrecord format. The tfrecord format is a simple format for storing a sequence of binary records.

* **Protobuf and .Pbtxt files:**

After the creation of tfrecord files, we will be needing protobuf and .pbtxt files for our coco dataset. In order to create these two files, we run the following command in our anaconda command prompt

python object\_detection/export\_tflite\_ssd\_graph

Here the parameters given to run the python file are:

1. Pipeline\_config: Path to the config file. In our case it was darknet yolov3.cfg
2. Trained\_checkpoint: Path to the model.ckpt file
3. Output directory where we want to create the file

Once the code is run, we will get two files namely tflite\_graph.pb and tflite\_graph.pbtxt. We will be using tflite\_graph.pb file to create the tflite file.

* **TfLite File Creation**

In order to run our trained model in any android application, tensorflow gives a very good API called tensor flow lite. Once we have converted our model in tflite file, we can use it on our mobile application for detection purposes. Refer to ("see Appendix B") for the code of tflite file creation.

* **Integration**

The tf lite file generated by this entire process is integrated in android device to start detection.

So far, we have implemented common object detection for blind people. Now our next target is to implement belonging object detection by creating a custom dataset for each blind person.

## 4.2 Test case Design and description

Our application informs the blind person of the impending obstacle. The users can login, create an account, select day mode or night mode and will be informed of the detected obstacle.

### 4.2.1 User Login

The following test case is about the user login functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Login** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.1* | **QA Test Engineer:** | | *Muneeb Akhtar* |
| **Test case Version:** | | *1* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.2* |
| **Revision History:** | |  | | | |
| **Objective** | | *Users must be able to log into the app by providing username and password* | | | |
| **Product/Ver/Module:** | | *User Login Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must have valid username and password* | | | |
| **Prerequisite:** | | *No Prerequisites* | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User enters a valid username and password through audio input and speaks login* | | | *System will now check whether the input is valid or not. If valid, redirects the user to the home page.* | |
| **2** | *User will be directed to the home page.* | | |  | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed* *☐Failed* *☐Not Executed* | | | | | |

### 

### 4.2.2 User Sign Up

The following test case is about the user sign-up functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Sign Up** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.2* | **QA Test Engineer:** | | *Hamna Faisal* |
| **Test case Version:** | | *1* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.1* |
| **Revision History:** | |  | | | |
| **Objective** | | *Users must be able to create their account.* | | | |
| **Product/Ver/Module:** | | *User Sign Up Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | |  | | | |
| **Pre-Requisite:** | | *No Pre-Requisites* | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User speaks sign up and sign up button is clicked* | | | *System will ask the user to enter account information* | |
| **2** | *User enters valid name and email address through audio input* | | | *System will authenticate, identify user and redirect user to particular homepage after creating the account* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.3 Select Day Mode

The following test case is about the selection of day mode functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Select Day Mode** | | | | | |
| **2** | | | | | |
| **Test Case ID:** | | *1.1.3* | **QA Test Engineer:** | | *Zain Ghazanfar* |
| **Test case Version:** | | *1* | **Reviewed By:** | | *Hamna Faisal* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.3* |
| **Revision History:** | |  | | | |
| **Objective** | | *The system should take a stimulus from the user in the form of one tap on the screen and produce a response in the form of activating light mode of the application.* | | | |
| **Product/Ver/Module:** | | *Day Mode Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system* | | | |
| **Pre-Requisite:** | | *The user must not touch for too long that the set time of release is out.* | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *The user activates the day mode by a single tap.* | | | *System open the camera and start capturing photos in burst mode* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.4 Select Night Mode

The following test case is about the selection of the night mode functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Select Night Mode** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.4* | **QA Test Engineer:** | | *Zain Ghazanfar* |
| **Test case Version:** | | *1* | **Reviewed By:** | | *Hamna Faisal* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.4* |
| **Revision History:** | |  | | | |
| **Objective** | | *The system should take a stimulus from the user in the form of two taps on the screen and produce a response in the form of activating night mode of the application.* | | | |
| **Product/Ver/Module:** | | *Night Mode Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system* | | | |
| **Pre-Requisite:** | | The user must not touch for too long that the set time of release is out. | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *The user activates the night mode by double tap* | | | *System open the camera with a flash and start capturing photos in burst mode* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.5 Login with incorrect credentials

The following test case is about the user login with incorrect credentails functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Login** | | | | | |
| **2** | | | | | |
| **Test Case ID:** | | *1.1.5* | **QA Test Engineer:** | | *Hamna Faisal* |
| **Test case Version:** | | *2* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.2* |
| **Revision History:** | | *The last version was revised by adding by verification in this version.* | | | |
| **Objective** | | *Users must not be able to log into the application by providing incorrect username or password* | | | |
| **Product/Ver/Module:** | | *User Login Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must enter invalid username or password.* | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User enters an invalid username or password and clicks on login* | | | *System will now check whether the input is valid or not. If invalid, prompts an error that incorrect username or password has been entered.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.6 Obstacle Detection Label Output

The following test case is about the obstacle detection label output functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Obstacle Detection** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.6* | **QA Test Engineer:** | | *Muneeb Akhtar* |
| **Test case Version:** | | *2* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.5* |
| **Revision History:** | | *This module’s testing was completed in version 2.* | | | |
| **Objective** | | *The system will detect the obstacles by the images captured by camera, compare it with the dataset of objects and will inform the blind person about the type of obstacle detected.* | | | |
| **Product/Ver/Module:** | | *Obstacle Detection Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system.* | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User taps once to open camera in day mode or twice to open the camera in night mode* | | | *System opens the camera and starts taking photos in burst mode until an obstacle is detected and informs the blind through audio output about the type of obstacle detected.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.7 Obstacle Detection Position Output

The following test case is about the obstacle detection position output functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Obstacle Detection** | | | | | |
| **2** | | | | | |
| **Test Case ID:** | | *1.1.7* | **QA Test Engineer:** | | *Muneeb Akhtar* |
| **Test case Version:** | | *2* | **Reviewed By:** | | *Hamna Faisal* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.5* |
| **Revision History:** | | *This module’s testing was completed in version 2.* | | | |
| **Objective** | | *The system will detect the obstacles by the images captured by camera and will inform the blind person about the position of obstacle detected.* | | | |
| **Product/Ver/Module:** | | *Obstacle Detection Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system.* | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User taps once to open camera in day mode or twice to open the camera in night mode* | | | *System opens the camera and starts taking photos in burst mode until an obstacle is detected and informs the blind through audio output about the position of obstacle detected.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.8 Sign Out

The following test case is about the sign out functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sign Out** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.8* | **QA Test Engineer:** | | *Hamna Faisal* |
| **Test case Version:** | | *2* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | | *UC 3.6.6* |
| **Revision History:** | | *This module’s testing was completed in version 2.* | | | |
| **Objective** | | *When user swipes left, the application is signed out.* | | | |
| **Product/Ver/Module:** | | *Sign Out Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system.* | | | |
| **Pre-Requisite:** | | *User must be logged into the system.* | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User wants to logout and swipes left.* | | | *System destroys the session of the user and the user is signed out of the application.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.9 Usability Check

The following test case is about the usability check non functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Usability Non-Functional Requirement** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.9* | **QA Test Engineer:** | | *Muneeb Akhtar* |
| **Test case Version:** | | *1* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | |  |
| **Revision History:** | |  | | | |
| **Objective** | | *The type of obstacles detected and informed to the user in simple English Language should be easy to comprehend for the blind.* | | | |
| **Product/Ver/Module:** | | *Usability Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system.* | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User has single tapped to open the camera in day mode.* | | | *System recognizes the obstacles and in simple English outputs the label of the obstacle detected through audio.* | |
| **Comments:** *User is able to comprehend the audio output by the system.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.10 Performance Check

The following test case is about the performance check non functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Performance Non-Functional Requirement** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.10* | **QA Test Engineer:** | | *Muneeb Akhtar* |
| **Test case Version:** | | *2* | **Reviewed By:** | | *Hamna Faisal* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | |  |
| **Revision History:** | | *Its testing was completed in version 2.* | | | |
| **Objective** | | *The application shall be optimized to reduce the time for obstacle detection and response will get back within 2ms* | | | |
| **Product/Ver/Module:** | | *Performance Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system.* | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User has single tapped to open the camera in day mode.* | | | *System recognizes the obstacles and informs the blind within 5ms about the obstacle detected.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.11 Security Check

The following test case is about the security check non functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Security Non-Functional Requirement** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.11* | **QA Test Engineer:** | | *Zain Ghazanfar* |
| **Test case Version:** | | *3* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | |  |
| **Revision History:** | | *This module’s testing was completed in version 3.* | | | |
| **Objective** | | *The application shall provide security about the belongings of a blind person which will be only accessible to them and not to any other user.* | | | |
| **Product/Ver/Module:** | | *Security Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | | *User must be logged into the system.* | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User has single tapped to open the camera in day mode.* | | | *System recognizes the obstacles and informs the blind about his belongings only that are stored in their database.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| *☐Passed ☐Failed* **☒***Not Executed* | | | | | |

### 4.2.12 Compatibility Check

The following test case is about the compatibility check non functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Compatibility Non-Functional Requirement** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.12* | **QA Test Engineer:** | | *Muneeb Akhtar* |
| **Test case Version:** | | *1* | **Reviewed By:** | | *Hamna Faisal* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | |  |
| **Revision History:** | |  | | | |
| **Objective** | | *The application is Android based so it will be compatible with minimum Android version 6.0 and API Level 23* | | | |
| **Product/Ver/Module:** | | *Compatibility Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | |  | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *The application was installed in Android version 5.* | | | *The application did not launch in that version.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| **☒***Passed ☐Failed ☐Not Executed* | | | | | |

### 4.2.13 Reliability Check

The following test case is about the reliability check non functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Reliability Non-Functional Requirement** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.13* | **QA Test Engineer:** | | *Zain Ghazanfar* |
| **Test case Version:** | | *3* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | |  |
| **Revision History:** | | *This module’s testing was completed in version 3.* | | | |
| **Objective** | | *The accuracy of the obstacle detection by the application is 80%.* | | | |
| **Product/Ver/Module:** | | *Reliability Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | |  | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *User has single tapped to open the camera in day mode.* | | | *The application doses not detects the obstacles with an accuracy of 80%.* | |
| **Comments:** *System is not working according to our needs.* | | | | | |
| *☐Passed* **☒***Failed ☐Not Executed* | | | | | |

### 4.2.14 Modularity Check

The following test case is about the modularity check non functionality and the reproduce steps in order to execute it.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modularity Non-Functional Requirement** | | | | | |
| **1** | | | | | |
| **Test Case ID:** | | *1.1.14* | **QA Test Engineer:** | | *Hamna Faisal* |
| **Test case Version:** | | *1* | **Reviewed By:** | | *Roshan Ul Haq* |
| **Test Date:** | | *01/01/2020* | **Use Case Reference(s):** | |  |
| **Revision History:** | | *This module’s testing was completed in version 3.* | | | |
| **Objective** | | *The application is divided into 3 modules for easier maintenance.* | | | |
| **Product/Ver/Module:** | | *Modularity Module* | | | |
| **Environment:** | | *Software: Android Studio*  *Language: Java and Python*  *OS: Android* | | | |
| **Assumptions:** | |  | | | |
| **Pre-Requisite:** | |  | | | |
| **Step No.** | **Execution description** | | | **Procedure result** | |
| **1** | *The code hierarchy could be seen clearly divided into 3 modules* | | | *The application is easier to maintain.* | |
| **Comments:** *System is working according to our needs.* | | | | | |
| *☐Passed ☐Failed* **☒***Not Executed* | | | | | |

## 4.3 Test Metrics

The below summarizes the total number of test cases executed that were passed or failed.

|  |  |
| --- | --- |
| **Metric:** | **Value** |
| **Number of Test Cases:** | 14 |
| **Number of Test Cases Passed:** | 14 |
| **Number of Test Cases Failed:** | 1 |
| **Test Case Defect Density:** | 100 |
| **Test Case Effectiveness:** | 0 |
| **Traceability Matrix:** | - |

So far, we have created test cases for every use case and non-functional requirement and further the information provided in the above tables perfectly depicts how thoroughly we have tested our application.

# Chapter 5: Experimental Results and Analysis

During the project**,** different experiments were done with different mode**ls** developed and used for obstacle detection and recognition. The results were analyzed and from among all models the best model was chosen which gave us comparatively better results. The following details will further elaborate about experiment and results analysis.

Since lots of models were used with different hyper-parameters so it was crucial for us to come up with some metric to see that which model performs better. Human evaluation was also done but it was only based on personal observation of each of the group member. The details of the selection of the model is given below.

## 5.1 Obstacle Detection Experimental Setup

We had daily life pictures from their surroundings and their corresponding label. The dataset was split into three parts which were training, validation and testing. Models were trained on the training data, validated on validation and were tested on the test data. Usually test data is the real-world unseen data so we decided to make just two folds, i.e. training and validation. Validation data was used to fine tune the hyper-parameters and to evaluate the model whereas training data was used to fit the model on the captions to recognize the obstacles.

### 5.1.1 Comparison of different models

Mean average precision score and frames per second score is used to analyze and classify that how good the results of the models are. Although, our application is detecting obstacles in real time so we were more concerned about the speed of detection rather than the accuracy. For our application, speed of obstacle detection is more important than the accuracy performance. See Table 5 for the comparison between different models.

Table 5: Comparison between different models

*This is the comparison of following models*

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Dataset | Mean Average Precision (mAP) | Frames Per Second (FPS) |
| YOLO | COCO | 51.5 | 45 |
| SSD Inception | COCO | 46.5 | 19 |
| Retinanet | COCO | 57.5 | 5 |
| Faster RCN | COCO | 59.1 | 6 |

Visualizing the scores above, YOLO model was selected for obstacle detection and recognition. YOLO is orders of magnitude faster (45 frames per second) than other object detection algorithms. Single shot detectors have an impressive frames per second at the cost of accuracy. They are here for faster real time processing. YOLO is a single shot detector but it struggles with detecting the small objects like a flock of birds. However, we are more concerned with the speed of detection and recognition so that the blind person is quickly notified about the type and position of the obstacle in order to prevent him from harming themselves.

The pictures below show the boundary boxes around the detecting objects and their accuracy.

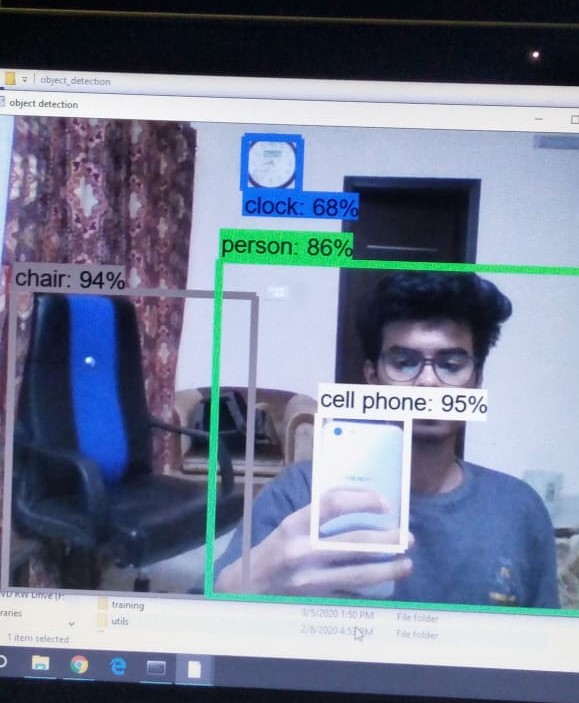


Figure 22: Object Detection through webcam

This image depicts the detection of objects through webcam and detection accuracy

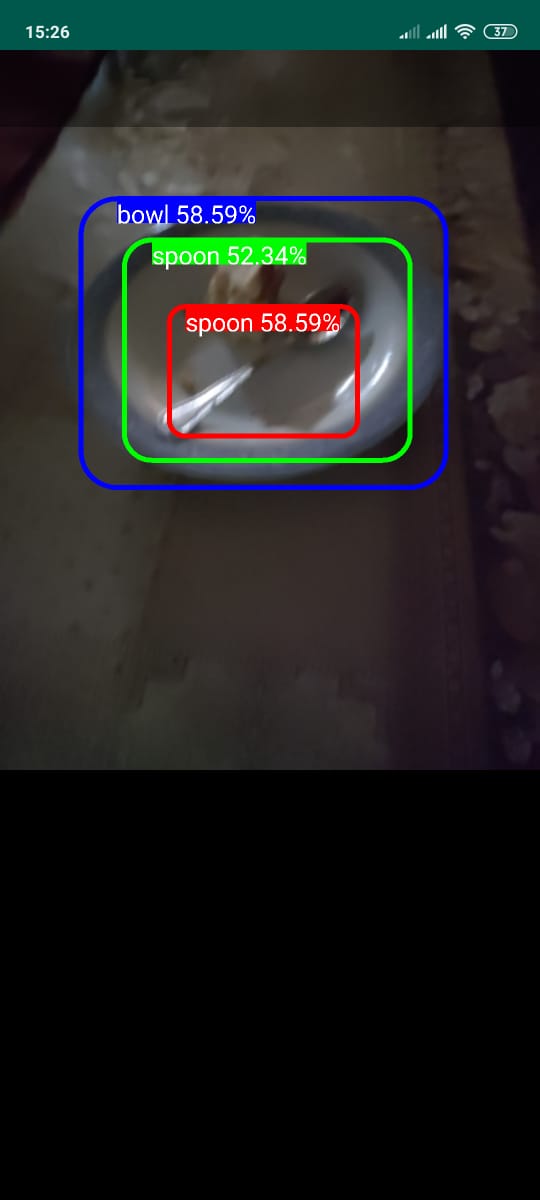


Figure 23: Night Mode Detection with Flash

This image depicts the detection of objects through webcam and detection accuracy in night mode

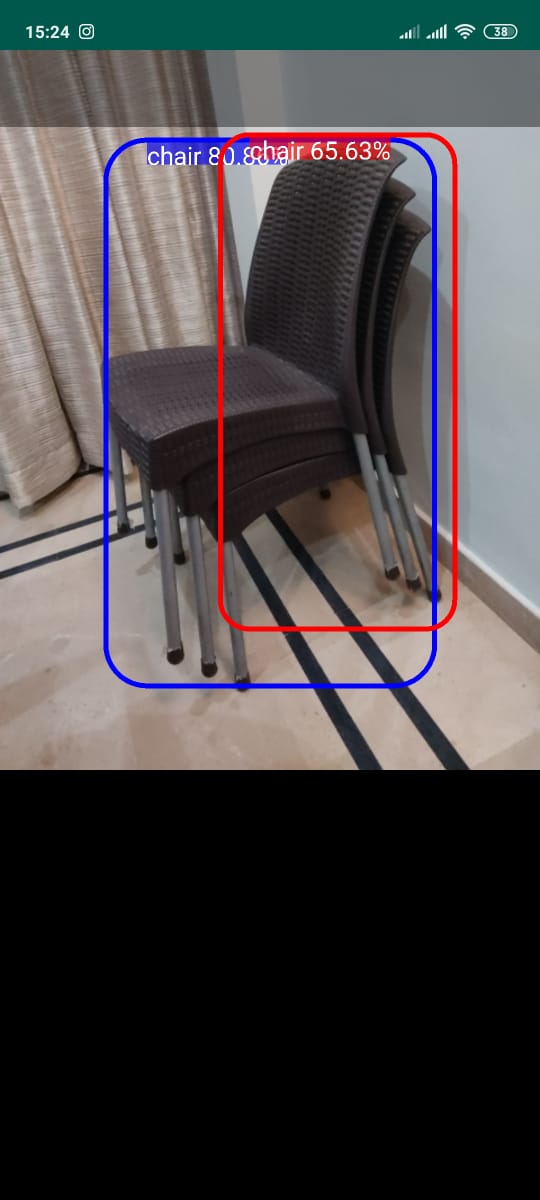


Figure 24: Day Mode Detection

This image depicts the detection of objects through webcam and detection accuracy in night mode

Analyzing the existing work, we concluded that informing the blind through audio, all over interaction with the application through voice, detecting obstacles at night, notifying about the position of the obstacle and the speed of detecting the obstacles in real time has not been dealt in any of the articles which is a major concern for the blind people to reduce their reliance on others. So, we covered this need in our project, by detecting and recognizing the obstacles in real time with a greater speed with YOLO model (45 frames per second), incorporating Google API for voice feedback and provided acoustic interaction with the application. Moreover, our application informs about the position of the obstacles and night detection with camera flash is also incorporated. These additional features in our application will provides a breakthrough in the matter of obstacle detection and recognition that would benefit a great deal to the blind community.

# Chapter 6: Conclusion

Mobility and the safety of the visually impaired is the utmost need for the blind community. All the previously used equipment like the cane stick are obsolete as they do not ensure protection for them. There have been a number of limitations due to which the scope of the project has forcefully narrowed down. This document has put into focus on the development of an android application for the real time detection of obstacles videos and informing the blind through audio feedback. This application will a useful tool for the blind community to reduce their reliance and get notified about the approaching obstacle in no time.

## 6.1 Initial Research

We chose to study further about the work done related to obstacle detection so we read nine papers to get better understanding of what has been done in the field of obstacle detection and recognition. We studied the algorithms like Faster RCNN, Linear Regression, Mean Squared Error and Grid algorithms. To our knowledge no prior work has been done in obstacle detection with a greater speed of detection, whole interaction with the application through audio and night detection. A good model will be used for faster detection and recognition of impending obstacles.

## 6.2 Work Completed

We used COCO dataset that has 80 classes of object from daily life. We have completed the development of the android application. Python is studied in detail which will communicate with the android application through the API and will run the python scripts. YOLO, deep learning model has been employed considering its speed of detection that is 45 frames per second.

At the start, the blind person registers themselves through audio input and then logins into the application. As soon as the blind person logins, home page is displayed, notifying the blind person whether to open the day mode with a single tap or night mode with double tap. By tapping once, camera is opened is day mode. The camera starts detecting the obstacles and generates frames around the objects. Our application will fetch those all frames and assemble them in a form of an audio for the user. It compares the detecting obstacle with the stored images and on recognition informs the blind person about the type and position of the obstacle.

The first challenge faced was the merging of the python backend with android platform. The second challenge was to train the model as it required high end hardware and computational power.

Our project can be further enhanced by increasing the accuracy of obstacle detection and recognition as we were more concerned with the detection speed. The scope of the dataset of objects can be increased by having a different variety of objects. A future recommendation would be to incorporate a navigation system in our application.

# References

|  |  |
| --- | --- |
| [1] | L. Bor-Shing, Cheng-CheLee and C. Pei-Ying, "Simple SmartPhone-Based Guiding System for Visually Impaired," 2017. |
| [2] | S. u. Rahman, U. Sana and U. Sehat, "Obstacle Detection in Indoor Environment for Visually Impaired Using Mobile Camera," 2018. |
| [3] | P. Andra, C. Arthur, O. Florin and N. Sergiu, "Obstacle Detection using stereovision for Android-based mobile devices," September, 2014. |
| [4] | Y. Jason, "Obstacle Detection with voice feedback- YOLOv3+gTTS," December 10, 2018. |
| [5] | C. Alessandro, F. Marco and C. Carlo, "Smartphone-Based Obstacle Detection for the Visually Impaired," September, 2015. |
| [6] | R. Alberto, J. J. Yebes, F. Pablo, M. Luis, A. Javier and C. Andres, "Assisting the Visually Impaired: Obstacle Detection and Warning System by Acoustic Feedback," December 17, 2012. |
| [7] | T. Ruxandra, M. Bogdan, B. Andrei and Z. Titus, "A Smartphone-Based Obstacle Detection and Classification System for Assisting visually impaired people," 2013. |
| [8] | I. Volodymr, C. James, G. William and S. Huiying, "Computer Vision-Based Clear Path Guidance for Blind Wheelchair Users," 2008. |
| [9] | J. Rui, L. Qian and Q. Shuhui, "Let Blind People See: Real-Time Visual Recognition with Results Converted to 3D Audio," March, 2016. |
| [10] | M. Karthikeyan, V. V. P. Fred, A. Malarmannan and K. Baskar, "V-DETECT: An Obstacle Detection System for Visually Challenged People Using Grid Algorithm," 2018. |

# Appendix

## Appendix B: Tflite File Creation Code

Following is the tflite file creation code.

import tensorflow as tf

graph\_def\_file = "E:/Tensorflow/models/research/object\_detection/tflite/tflite\_graph.pb"

input\_arrays = ["normalized\_input\_image\_tensor"]

output\_arrays = ["TFLite\_Detection\_PostProcess", "TFLite\_Detection\_PostProcess:1", "TFLite\_Detection\_PostProcess:2", "TFLite\_Detection\_PostProcess:3"]

converter = tf.lite.TFLiteConverter.from\_frozen\_graph(

graph\_def\_file,

input\_arrays,

output\_arrays,

input\_shapes={'normalized\_input\_image\_tensor':[1, 300, 300, 3]}

)

converter.allow\_custom\_ops = True

tflite\_model = converter.convert()

open("converted\_model.tflite", "wb").write(tflite\_model)