Available online at https://sej.journals.ekb.eg/
FACULTY OF ENGINEERING – SOHAG UNIVERSITY
Sohag Engineering Journal (SEJ) VOL. 3, NO. 1, March 2023

Camera-Based Navigation System for Blind and Visually Impaired People

Islam Mohamed*, Mostafa Salah, Ahmed Farghal

Dept. of Electrical Engineering, Faculty of Engineering, Sohag University, Sohag 82524, Egypt

Abstract

One of the essential aspects of our life as humans is vision, the ability to see and describe the world with our eyes. Therefore, blinds, even visually impaired people, have plenty of hardships in their daily life activities. They may find it difficult to recognize objects or people in front of them. This paper presents an emphasis on developing a real-time system to aid these people using a Raspberry Pi to process pieces of information provided via camera and ultrasonic sensors with a simple feedback mechanism to inform the user through a headphone. Using this system, the blind could walk without the white cane, and most importantly, it reduces the blind's dependence on other people, thus increasing their quality of life. The proposed system works in different scenarios at both indoor or outdoor environment and has multiple modes of operation to help the blind in different situations with minimum hardware to get an affordable device with a satisfactory offline real-time performance. The system mainly utilizes the technology of deep learning with computer vision to efficiently solve common difficulties such as object recognition, face recognition, and reading text in real-time feedback responses. it sends an audio signal for the user informing him about objects with their count and relative position, familiar faces, or text recognized in the captured frame.

 DOI: **10.21608/SEJ.2022.155927.1018**

Keywords: Blind; Visually impaired; Navigation; Travel aid; Assistive system; Raspberry Pi; Ultrasonic sensor; Camera; Computer vision;Deep learning; Object detection; Face recognition; OCR.

1. INTRODUCTION:

1.1. Problem statement:

Visual impairments can discourage one from performing his usual activities. It hinders his work, studies, travel, and overall health. Moreover, navigation becomes a real challenge. Unfortunately, the World Health Organization (WHO) derived that about 2.2 billion people of all ages worldwide have a visual impairment or blindness, and at least half of them have a visual impairment that can be prevented [1]. We are concerned with cases where there is no available cure or wearing glasses solutions. The number of these cases is increasing every year according to Vision Loss Expert Group estimations [2] for the global number of people who suffer from visual impairment or blindness over time as shown in Table 1.

TABLE 1. ESTIMATION BY VISION LOSS EXPERT GROUP FOR THE WORLDWIDE NUMBER OF PEOPLE WHO ARE BLIND OR VISUALLY IMPAIRED OVER TIME

Year	Global number affected over all ages (millions)		
	Blind	Visually Impaired	Total
1990	31	160	191
2000	32	176	208
2010	34	199	233
2015	36	217	253
2020	39	237	276
2030	55	330	385
2040	80	451	531
2050	115	588	703

* corresponding author: islammohamedkamal25@gmail.com

1.2. Related work and available solutions:

There are several attempts developed for blind and visually impaired (BVI) people to help them independently navigate safely and efficiently. Convenient navigation aids such as white canes or guide dogs are cheaper and more cost-effective; however, they require full attention by the user so, they are extremely useful for near-ground obstacle avoidance especially when there is no crowd in safer outdoor environments. With the growth of technologies, smart electronic travel aids (ETAs) have been developed that further help reduces accidents and facilitate movement, thus improving the traveling experience in unfamiliar places in general [3,4]. ETAs can be represented in subcategories such as robotic navigation aids (RNA), smartphone-based systems, and embedded systems in wearable attachments. An RNA example is mostly a hardware-based system as shown in [5,6]. RNAs mainly use friendly blind interfaces in form of smart canes. These are useful as they can be used passively as a normal white cane. These canes are equipped with various sensors such as 3D cameras, ultrasonic or LiDAR sensors, fire or water sensors, global positioning system (GPS) modules, etc. Their limitations are mainly due to their compact size which affects the cost of developing such systems. This is because they should contain several sensing elements to convey sufficient obstacle information for pathfinding and navigation purposes in such a small size. A useful hardware-based product [7] is available but unluckily, it is expensive at 599$ per unit. Unfortunately, most of the existing systems are either hard to adapt, costly, difficult to carry, or sophisticated to use developed features. This reduces the usefulness of the majority of hardware solutions. Another type of ETAs is utilizing the smartphone in order to develop a reliable device not a bulky one with less hardware and hence, relatively low cost. This allows internet of things (IoT) and cloud computation to be the dominant fields of study for developing such systems. According to [8-10] which uses IoT via a Bluetooth assistance application in a smartphone. Since the smartphone is the main computing device, such systems are limited to smartphone sensors. An additional sensor must communicate with the smartphone with some sort of Radio Frequency Identification (RFID) or beacon or external server. This allows easier firmware and software to be updated via the internet. That reveals the major disadvantage for these systems which is the total dependence on beacon and internet signals for communication between outer sensors, the cloud, and the smartphone. Hence, there might be inapplicable for real-time performance. There are software-based available solutions [11-14] with no hardware but a smartphone such as KNFB Reader (one step reader), Tap Tap See, Cash Reader, and Seeing AI. These apps mainly utilize the camera sensor of the smartphone applying some sort of computer vision techniques to process the image/frame and then give feedback to the user describing the image/frame captured. Ignoring the fact that some of these applications are pay-to-use apps, the major disadvantage lies in there is still a need for visual interaction for selecting the app and features from the selection menu provided like that in seeing AI. This makes them perfect solutions for the visually impaired; however, they are not very useful when it comes to blind people. A better approach can be achieved by integrating smart systems into one or more wearable attachments with lightweight sensors to help the blind with common activities. According to [15,16], these systems provide real-time performance and immediate feedback since they are worn by the user. These can contain multiple sensors in different attachments like belts, gloves, glasses, jackets, shoes, etc. in order to acquire different information about the obstacle. Hence, they are suitable for aiding in various cases.

The proposed project is wearable glasses that is basically a deep learning computer vision-based system. The challenge developing such an assistive system is to increase the precision for a deep learning convolutional neural network (CNN), you have to use more complex architecture which actually affects the speed. A deep learning architecture determines the number of parameters need to be calculated before the classification. Different object detection architectures are tested; however, for real-time situations, the smaller architecture, the better it is. Thus, a single shot detector (SSD) object detection architecture [17] is used. It gives an optimal balance between speed and accuracy. This proposed system will aid BVI people in three functions: object recognition with relative localization from user with distance threshold calculating, face recognition for family and friends, and reading text which utilizes the optical character recognizer (OCR).

The rest of this paper is organized as follows: Firstly, we will be exploring the proposed system overview in Section 2, followed by methodology and results in Section 3. Eventually, the conclusion and ideas for future work will be discussed in Section 4.

2. SYSTEM OVERVIEW:

2.1. Proposed prototype:

The prototype in its simplest form is an assistive system for the visually impaired and blind people through wearable attachments. It is agreed to pick up 3D designed glasses as wearable attachment. However, multiple intelligent attachments can be deployed for different purposes, as stated before. These systems should be interfaced and integrated together with fast sensors to react in the real-time manner. Also, they should help in preventing the user from dangerous indoor or outdoor situations to allow some kind of path planning for BVI people with a cost-effective way. In the proposed system, shown in Fig. 1, we are considering two main

subsystems with three different modes of operation to provide the sufficient aid for the BVI people. Although we are considering this system mostly to be used in an indoor environment, it can still be used under outdoor environment. We will firstly begin discussing the functions of each sub system then go through more details for how the three different modes work.

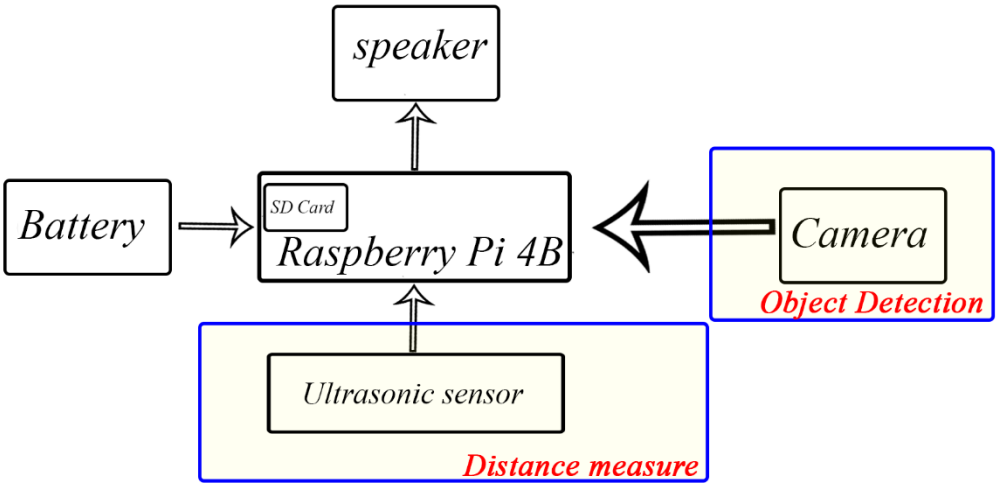


Fig. 1. Proposed system block diagram.

2.2. Subsystems and some information about the hardware used in these subsystems:

We are using a single camera module rev 1.3 with a cable that is designed specifically for raspberry pi. This module provides 5MP. It supports capturing video in 480p @ 60/90, 720p @ 60fps, and 1080p @ 30fps. However, the frames per second drop due to processing done on each frame. This sensor gives information signals needed for object detection. Also, we are using a single hc-sr04 basic ultrasonic sensor which gives the necessary echo signal to calculate the distance based on the ultrasonic wave triggered. I keep saying a single unit as several units in multiple wearable attachments can give multiple signals for different purposes, please refer to ideas for future work in section 4.

- *Object detection system:* Utilizing a camera that takes a live video (or frame images) and sends them to the Raspberry Pi, then the Raspberry Pi can process these frames by applying a deep learning object detection model on the captured frames. And therefore, the camera system is used to detect each object in each frame. Now, we have got the information from the image signal. We will consider a simple feedback mechanism for this sub-system. So, the feedback output is done via converting the recognized object name to speech that BVI users can hear audible phrases through a speaker (hand free).

- *Distance measuring system:* Utilizing an ultrasonic sensor to provide the required signals for near objects and obstacles distance measuring. Those signals can be processed by the Raspberry Pi and translated to indicate the relative position of the obstacle from the user by giving meaningful warning feedback. As we have stated previously that in our scenario case, we are considering helping BVI to navigate and recognize objects in indoor (enclosed) environments. Although the first and second modes can be used in an outdoor environment for simple object detection and face recognition, this system can be improved and developed further for including safer outdoor environment navigation.

2.3. Modes of operation:

The proposed system has three main modes of operation, as represented in Fig. 2. The BVI user can toggle between the three different modes using a single pushbutton. The distance measuring subsystem, via ultrasonic sensors, that allows the system to provide audible stop warning alerts beyond certain threshold. This subsystem mainly used in mode 0; however, it could be useful to be carried over to mode 1, too.

- *Mode 0:* This is a general-purpose mode in which the Raspberry Pi focuses on detecting different objects with the help of two different models that are combined together, namely, 'model A' which is a quantized pre-trained model on the Microsoft common objects in context (COCO) dataset [18] to detect 80 different classes of objects, and 'model B' which is a custom object detection model based on SSD architecture used to classify classes that don't exist in the pre-trained model increasing the number of classes with a total of 96 classes.

And since SSD architecture is state-of-the-art when it comes to speed and accuracy, the real-time performance is minimally affected by having two models that run on the captured frame.

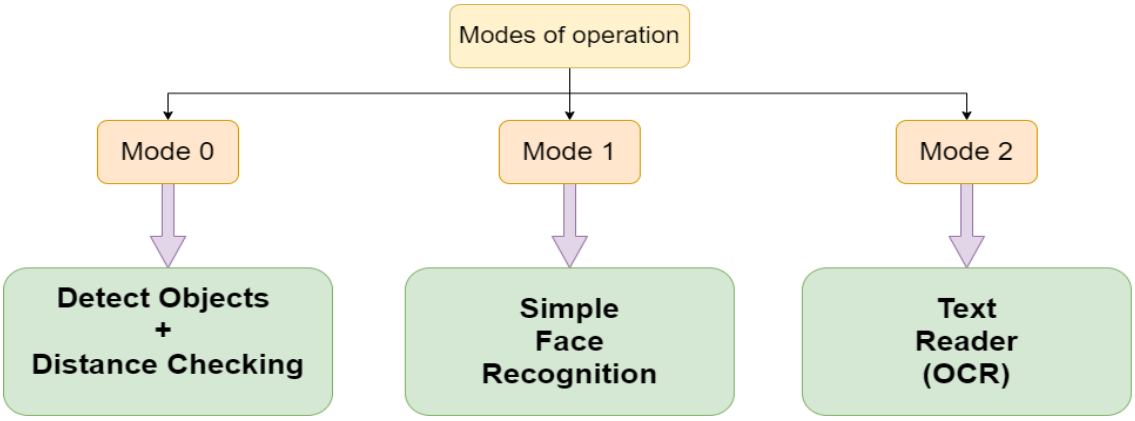


Fig. 2. Modes of operation.

- *Mode 1:* This a specific-purpose mode at which the Raspberry Pi focuses on recognizing relevant humans for the visually impaired user. With this mode, The BVI people can recognize specific men who are standing in front of themselves. The main objective is to allow users to identify known family members, or friends which will help them to be secure and social. The main issue with this mode is that we have to get datasets for faces in order to be detected later. There is also a challenge in the hardware side which is the camera sensor resolution which can be further improved via image processing for the frame.

- *Mode 2:* This is also a specific-purpose mode in which the Raspberry Pi focuses on reading plain text. This text is scanned from a frame by the camera via an optical character recognizer (OCR). Therefore, OCR scans and reads printed documents and signboards. The main challenge in that mode, in my opinion, is the requirement for on-board computation on the wearable platform as images observed, by a moving camera, are subjected to motion or out-of-focus blur. Thus, OCR should be applied after preprocessing the frame.

3. METHODOLOGY AND RESULTS:

3.1. The 3D glasses design:

The assistive device is a simple 3D glasses that are designed through CAD software called Fusion360. This software is very good for creating and designing efficient mechanical structures in STL (or Standard Tessellation Language) file format. The material used for realizing the STL design is called PLA+ (high concentration of polylactic acid). The final design of the 3D glasses, shown in Fig. 3, consists of specific parts which can support the camera and ultrasonic sensors and two boxes for raspberry pi and battery holding. However, these boxes can be detached from the glasses structure. Noting that is better realized if the boxes all together away from the glasses so, for further details on this idea please refer to section 4.

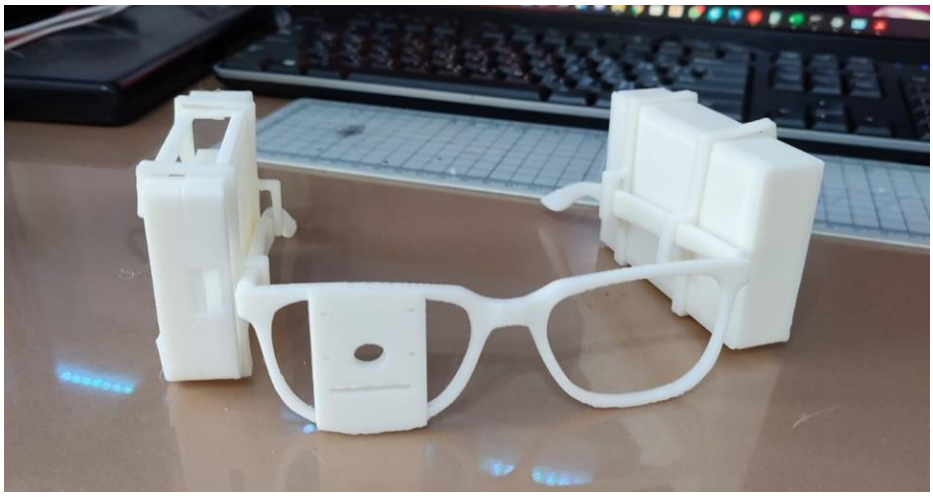


Fig. 3. The final design of 3D glasses parts.

3.2. Procedures and implementations with results:

The entire processes will be explained in this section. The proposed methodology as block diagrams is inspired from [19] with additional functionalities shown in Fig. 4.

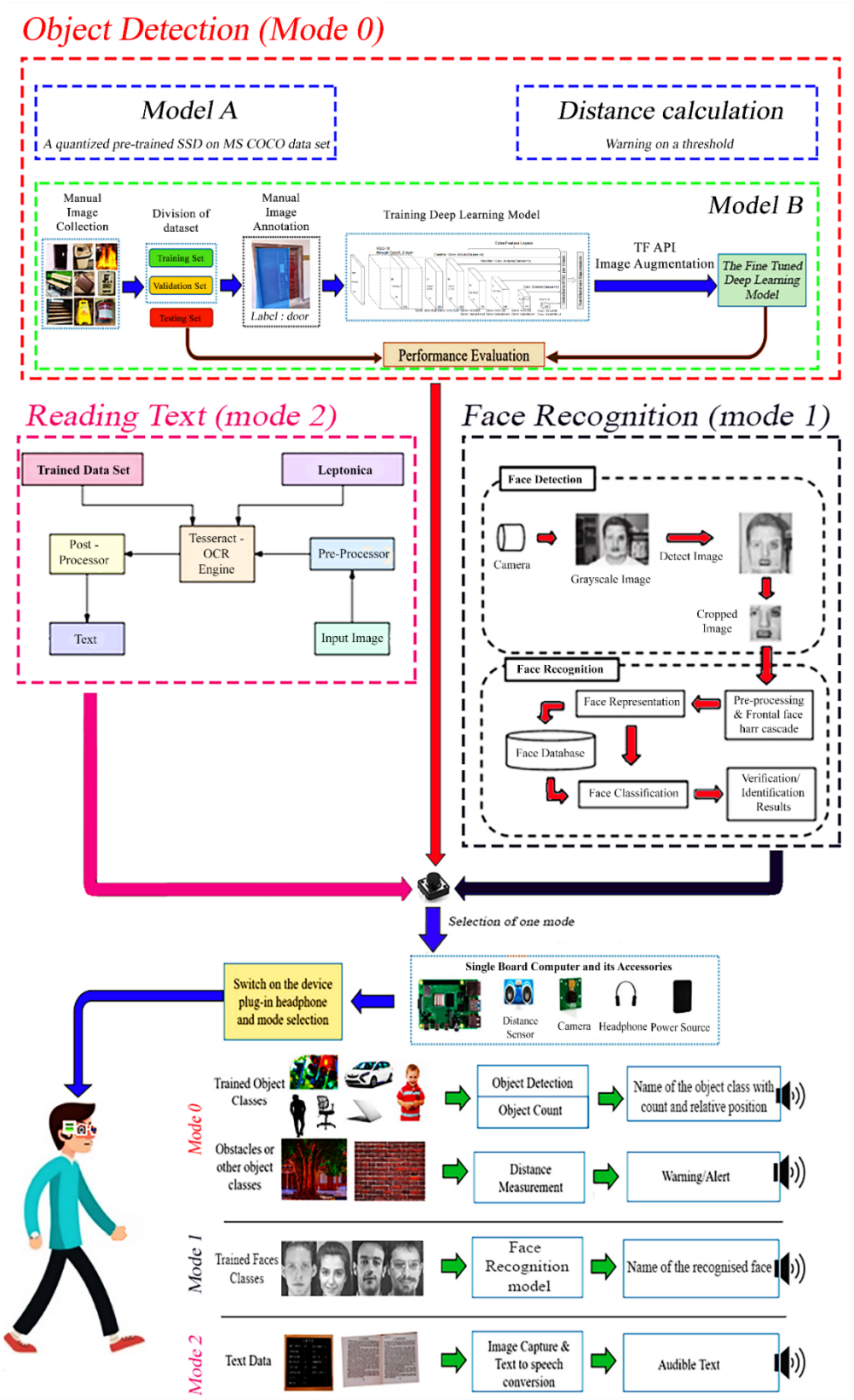


Fig. 4. Proposed methodology as block diagrams.

Considering the first part which is the object detection mode which is mode 0. Mode 0 can be divided into distance calculation for comparing it to a certain threshold defined in the script, and a combination of two models inspired by ensemble learning; however, unlike ensemble learning, the two models have different outputs, predicting different categories.

Distance estimation is done by ultrasonic sensor. Calculations are done every multiple frames for lower power consumption. However, they are much faster (higher calculation frequency per frame) when there is no object presented on the captured frame by the camera. Some factors may affect distance estimation. This is due to the theory of operation of the distance-estimation sensor. Ultrasonic depends on the speed of sound for distance calculation which is actually a variable due to temperature, humidity, and other weather conditions. This variation can affect results, especially in extreme environments; however, actually not taken into consideration developing the distance measurements in the proposed project instead the average sound velocity is considered constant $v_{ultrasonic\ wave} \sim 34300\ cm/s$. And hence, the distance can be calculated, by measuring the round-trip time for the triggered wave, as follows $distance = (timeElapsed * v_{ultrasonic\ wave})/2$. If the distance is less than a threshold value, i.e., 30 centimeters, a stop alert is fired. Therefore, obstacle avoidance is basically the aim of this distance measuring system and that is all about this block.

In this mode, deep learning is utilized for object detection. Considering firstly, the pre-trained SSD model A. This model is quantized for faster processing, and since it is pre-trained on a relatively large database, it has a decent generalization error. Meaning the model will perform well in different environmental conditions. In fact, generalization error reduction is the actual goal of any deep learning model. During training, an algorithm, called an optimizer, adjusts some weights and biases on a neural network architecture to obtain the minimum error at training. However, generalization error cannot be controlled. You have to measure it on a test set. Therefore, after some point, reducing training error results in an increment in generalization error. That is the main challenge which is famously called overfitting. Having a small dataset makes your model more prone to overfitting in a specific network. Unfortunately, a huge dataset should be used for training a network architecture from scratch but luckily, we can use some knowledge from the pre-trained model as we do not have a lot of images for each class. This is a type of transfer learning called fine-tuning. By freezing some of the earlier network parameters which contain low-level features and only concentrating on learning higher-level features, we can achieve a decent generalization. Hence, we can use our small manually gathered image set to let the optimizer improve a smaller number of parameters. The bad news is that we need to include the pre-trained classes' original set with our custom image set during training, or this fine-tuning will significantly affect the generalization for the pre-trained classes. Hence, a way to come around this in the proposed system is to integrate two different object detection models. Model A will be a quantized pre-trained model identifying 80 different common object classes [20], shown in Fig. 5, with good generalization in different scenarios, and Model B will be a custom object detection model to serve for other objects that are not existing in the COCO labels including some hazards detection shown in Fig. 6.

Fig. 5. The Microsoft COCO labeled objects for model A.

Fig. 5. Custom class labels for model B.

After defining the labels needed for model B, it is time for gathering images. There are several datasets for object detection such as the CIFAR-100 [21], CALTECH-101 [22], PASCAL VOC [23], ImageNet [24], and a lot more on Kaggle [25]. Since deep learning models extract all features (representations) and patterns from the raw data, the quality of the data matters. Thus, bad data implies a bad-performing deep learning model. Images taken via mobile should be preprocessed to 1024 by 1024 JPEG images to remain containing the object details with relatively lower size as this will be better for minimizing the training time. The preprocessing software, we have used, to resize images and reduce the size of images is named Caesium [26]. There is also a website for jpg image compression called tiny jpg [27]. When collecting images using a mobile camera, we have tried to take them from various angles and lighting conditions so, that the model can generalize features for different conditions.

Taking into consideration the data imbalance problem. Therefore, for model B, we took 60 images per class both gathered from fair-use google images and manually captured images via mobile phone. From which, we take 7 images as validation data. With a validation ratio of 11.67% of the total data at hand. Total images at hand divided like that 172 currency-class from Kaggle [28] (5Egp, 10Egp, 20Egp, 50Egp, 100Egp, 200Egp) + 60 bench + 60 door + 60 fire + 60 fire extinguisher + 60 recycle bin + 120 stairs (Up + Down) + 60 toggle switch + 60 wallet + 60 wet Floor Sign = 772 images. Later added 25 more fire images. This is a deliberate data imbalance for the fire class as it is the only class that performs poorly. So, it ends up with ~ 800 images for training, validation, and testing.

After splitting, we created annotation files via a python software (tool) developed by Tzutalin called LabelImg [29]. Gathered images were manually annotated with this tool to produce extensible markup language, or shortly XML, annotation files. These files contain the answer to a particular image for the supervised training. They include image size, path, object names, and bounding boxes' position. While annotating, we considered keeping the bounding box as tight as possible, therefore, the model can learn the feature of that specific object, not the surroundings.

After annotating images, it is time to input them into a particular architecture to learn from the labeled images. The main reference, from which implementation for custom object detection (OD) models from TensorFlow 2 (TF2) detection model ZOO [30], was Nicholas Renotte from the GitHub repository [31] based on TensorFlow application programming interface (TF API) for object detection. Fine-tuning the dense output classification layer of a CNN is utilized due to dataset size limitations. If training is done from scratch, it is very likely for overfitting to take place. To further eliminate (or resist) this problem, image augmentation is used during training in the TF API to increase the number of images hence, improving generalization. We first created the label map which is a text file in (. pbtxt format) that contains all objects labeled in images. Then prepare the input file for the TF API for OD which is called TF records. TF records are just binary files for annotation and images you will feed the model with this API. Generating training and validation records, writing their paths in the configuration file, updating the number of classes, selecting the batch size for the optimizer, and adding augmentation options needed are all we did in the pipeline configuration file. The selected pre-trained model was chosen to be the ssd_mobilenet_v2_fpnlite_320x320_coco17. FPN lite is a lite Feature Pyramid Network that is a feature extractor designed with the feature pyramid concept for feature sharing to improve accuracy and speed. And the $320x320$ refers to the input image size that is accepted by the CNN of the model. SSD architecture, shown in Fig. 7, is a deep fully convolution network. SSD utilizes a set of default (fixed-size) anchor boxes with different aspect ratios and sizes, as shown in Fig. 8, to discretize (regress and classify) the output space of default bounding boxes for every convolutional layer output for handling objects with various sizes. The output of the convolutional layer is called a feature map. Thus, Higher-resolution feature maps are responsible for detecting smaller objects and vice versa. For every default bounding box, SSD predicts both the box offsets $\Delta(cx, cy, w, h)$ and the confidence scores for all object categories $[(c1, c2, \cdots, cp)]$. Hence, the SSD loss is a weighted sum of localization losses and confidence losses. The last layer after the detection from multiple feature maps is followed by a non-maximum suppression layer to produce the final detections. Non-maximum suppression (NMS) layer is used to overcome multiple detections for the same object which deletes output boxes with low confidence score values if there are two or more intersecting rectangles with the help of IoU (Intersection over Union).

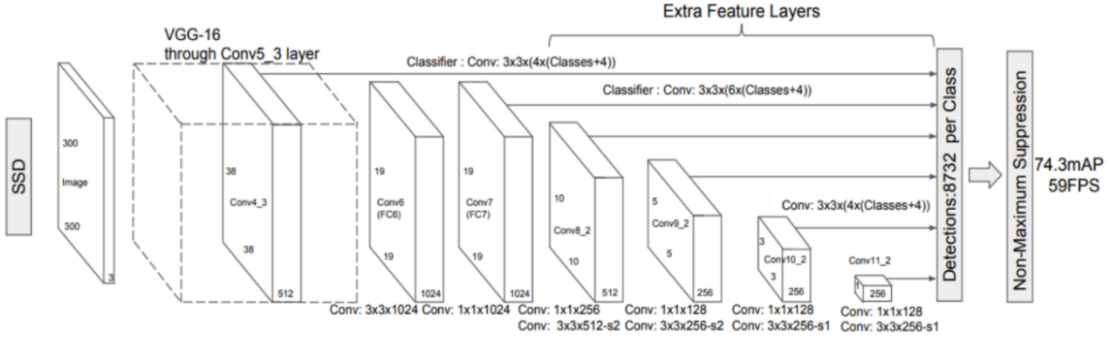


Fig. 6. SSD object detection architecture [17].

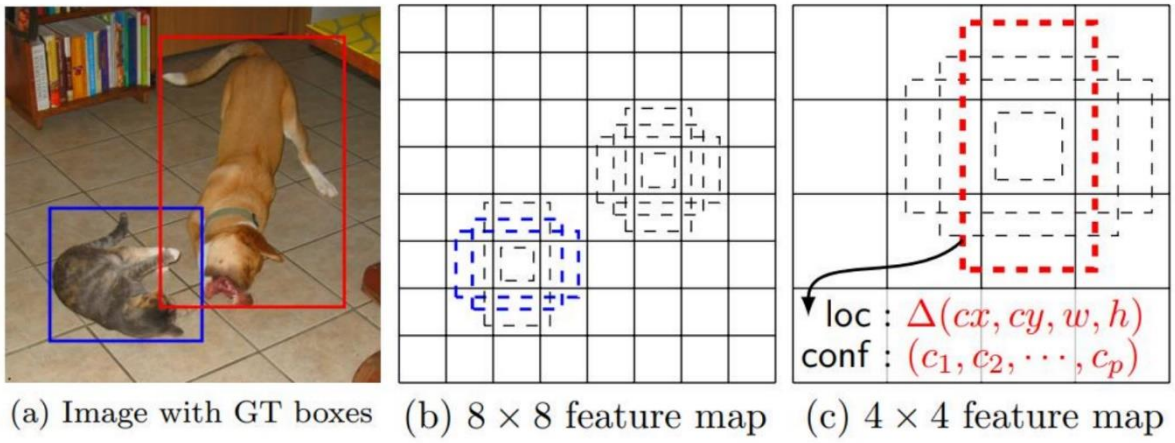


Fig. 7. How predictions are made in SSD architecture [17].

Training is done after setting up the configuration file. With augmentation options are set to be a random horizontal flip, image cropping, image scaling, brightness, and jpeg quality. And with the optimizer's learning rate decaying from an initial value of 0.08 till zero at maximum 50K training steps, and momentum of 0.9. The training process was done on Google Colaboratory (Google Colab) mainly utilizing the free GPU access for minimum training time and saving checkpoints and results in the drive. TensorBoard is used for visually monitoring metrics during training and evaluation. It is decided to write the metrics for evaluation in the log file every 1K training steps. There are multiple models deployed and tested with different hyperparameters; however, we will show the latest model trained. Some of our models trained and deployed, before that final one, suffer from information leakage due to changing the configuration which contains hyperparameters for the optimization algorithm and the architecture for enhancing the evaluation metrics. Unfortunately, by doing so you indirectly sent a feedback signal to the training algorithm, and if this is done multiple times you may be overfitting the validation set. Hence, we use a test set that is separate and different from the validation set for measuring the generalization.

The main metrics for training is the total loss, which consists of the classification loss, localization loss, and regularization, presented in Fig. 9. And the evaluation metrics used for measuring performance are mean average precision and average recall shown in Fig. 10. We have considered the recall calculated over all IoU values with a maximum of 1,10, and 100 detections; however, AR@1 detection is shown only in Fig.10 (b). Noting that the model used was early stopped at 27K; however, training is continued further than that but the generalization error seemed to be increasing and the architecture begins to overfit the train set. Some results for model B detections on the test set shown in Fig. 11.

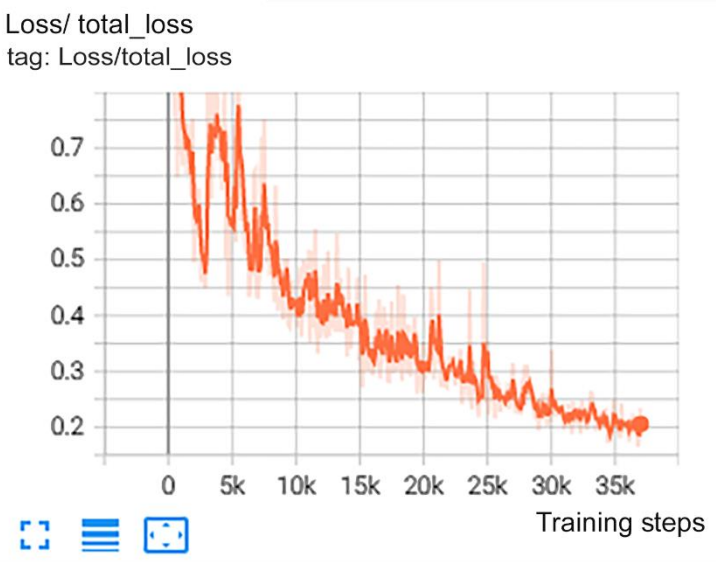
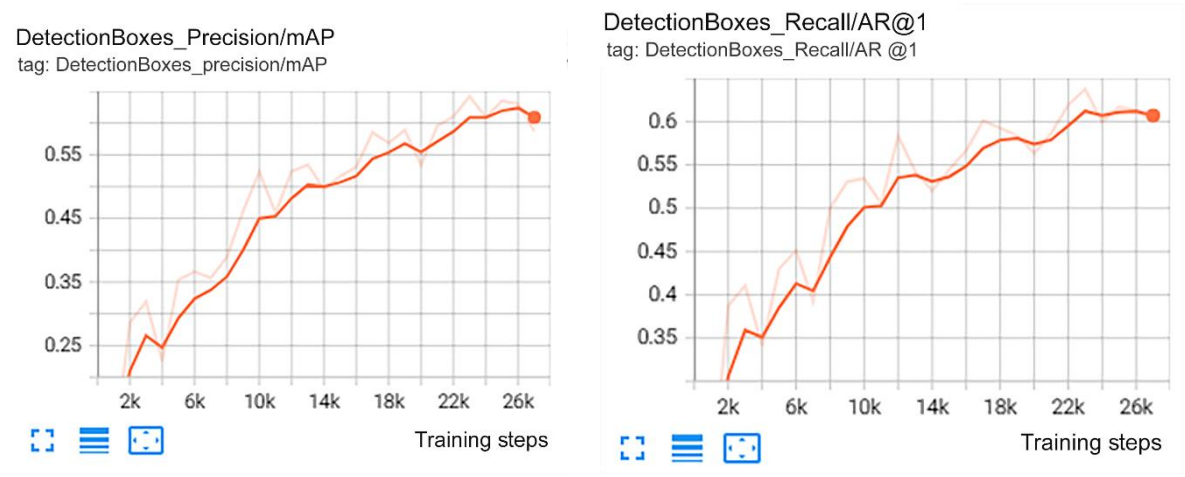


Fig. 8. Total loss over time during training. The training metrics for model B.



(a) (b)

Fig. 100. Evaluation metrics for model B vs training steps: (a) Mean average precision (b) average recall on one detection per image.

Fig. 10. Few results for model B detections on the test set.

The currency dataset was from Kaggle, not manually captured, which not very similar to real-world ones. And hence, currency classes can be improved by adding high quality images. However, when relatively good results on the test set are acquired, the model is saved and frozen for making it ready for exporting to TFlite format to be deployed on Raspberry Pi. Noting that a simple TFlite conversion was used. One further optimization can be done to quantize the TFlite file which will give a faster performance. The conversion outputs the detect.tflite file which is the needed file for Raspberry Pi to run model B. Then we run both models A and B together for the final result shown in Fig. 12.

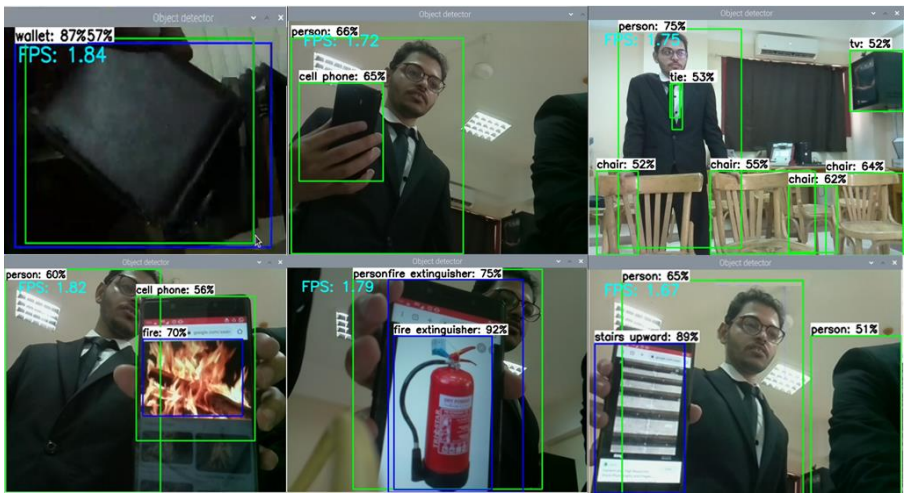


Fig. 11. Model A and B running on Raspberry Pi.

Noting that, the code for detection A has two features. The first one is counting the number of objects of the same class presented in the decision frame. The second is to give relative position information from the object detected. By dividing the image into three-thirds, the user can know whether the detected object is left, right, or in his front. Meanwhile, the code for detection B is just informing the user that the object does exist. For decreasing information to the user and conveying higher confidence score objects, we make a detection limit for the detections to be 3 detections for model A and another 3 for model B. There is one decision frame every multiple frames to run both A and B detections. Firstly, the audio was conveyed via several mp3 files containing the labels; however, this becomes messy for developing for a lot of classes so, we tried the offline text-to-speech python library pyttsx3. The pyttsx3 engine uses strings which is more flexible for further developing or changing class names.

Some optimization to utilize the resources was done such as using the multi-threading to improve the frames captured by the camera, counting the object decreasing time instead of hearing the same object name multiple times, the user hears the number of the objects, ultrasonic code is optimized to work with higher frequency if there is no object detected by either of models in the captured frame, and finally, minor optimizations replacing some logical statements with a single arithmetic operation which is huge for the logical statements repeated every frame.

Considering the second block, mode 1, the simple face recognition mode. This mode is used to match human faces in the input video frame to a collection of defined pre-trained faces. Face detection includes separating faces from the background or clutter. It involves pre-processing for grayscale conversion and some filters help for front faces classification and then localizing the position of a bounding box of the faces detected in the image. A face recognition module is some method to identify or verify the identity of the detected face. Verification is done by testing the detected faces with known faces in the collected database.

There is a dataset collected for three persons namely, Karam, Badran, and Fareed. The dataset is collected from the same raspberry pi camera module for training on similar conditions of operations via a python script. About eight images for each person are acquired which is relatively small but this gives an acceptable result. The training process is based on the python script provided in [32]. After training, the face recognition model is tested and the results are shown in Fig. 13.

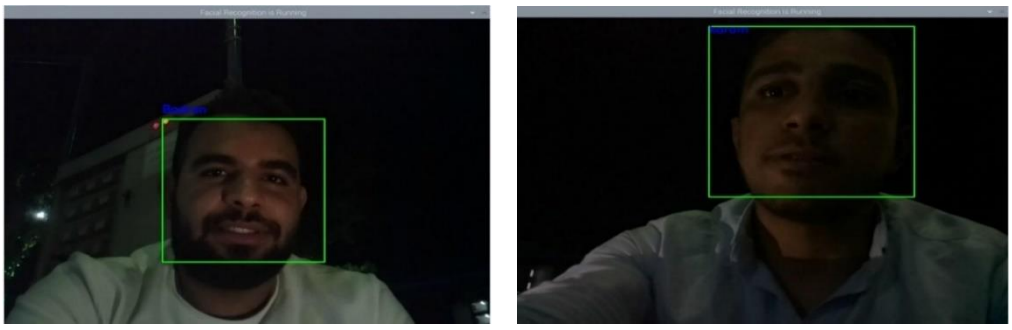


Fig. 12. Results for face recognition model on the Raspberry Pi.

Now, consider the third, mode 2, the reading text mode. This mode is essentially based on OCR which is sometimes referred to as text recognition. Concisely, using the open-source tesseract engine. Python-tesseract, or Pytesseract, is a tool in python which is a wrapper for the Tesseract-OCR Engine. This is the OCR tool used in the proposed aiding system. Utilizing this tool, an image or a video frame captured is analyzed returning a list of strings containing the words detected. Some pre-processing techniques may be used to improve the capability of the ORC of detecting words such as converting the image to grayscale, removing noise by adding some blur, dilation, erosion, applying canny edge detection, and skew correction. But unfortunately, we didn't deploy these

pre-processing techniques hence, the results on the video frame are not good enough on smaller text. Another factor that affects reading smaller text is the camera resolution. However, testing the mode on an image gives satisfactory results for both large and small texts as shown in Fig. 14.

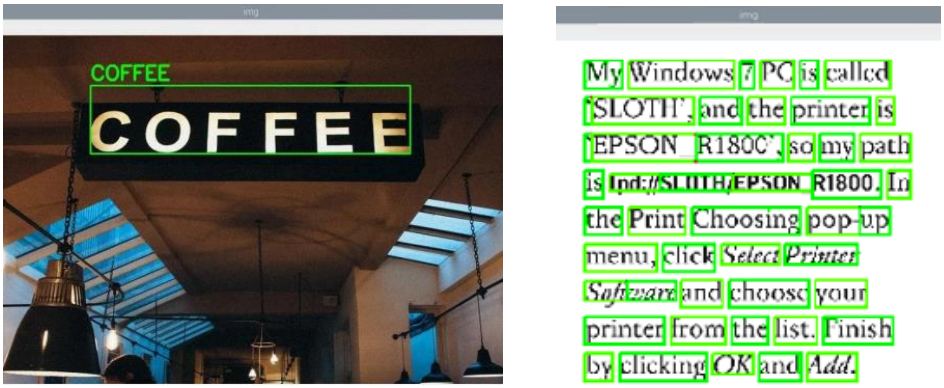


Fig. 13. Results for reading text mode on the Raspberry Pi.

4. CONCLUSION AND FUTURE WORK:

In this paper, we have presented an offline real-time wearable attachment that is a glasses system for aiding blind and visually impaired people. This system includes a camera, an ultrasonic sensor, an embedded Raspberry Pi, a headset, and a battery. It is impossible to give vision to the blind; however, using computer vision and deep learning, the BVI can obtain information through a headset, either indoor or outdoor, about obstacles and situations in front of him during navigation. The BVI user can also recognize his family and friends, or read text and signboards. This helps BVI to be self-dependent for doing many activities safely and comfortably.

Unfortunately, the system had not been experimented by users. However, some further future enhancements and recommendations can make this system more reliable. Fortunately, the system is configurable. Therefore, a lot of work can be done for the product to be released. Firstly, the detection subsystem, the reading text mode can have better software in scanning video frames utilizing image processing for several frames, to obtain good and accurate text detections. For face recognition mode, utilizing a mobile application connected with some sort of server, to store images for people to be included for recognition then the system can update over the air whenever it is connected to the internet. For object detection mode, improve model B, generally, by adding a large number of examples therefore, the model generalizes better. Another technique may be considered that is called Generative Adversarial Networks (or GANs) proposed by Ian Goodfellow in 2014 to synthesize or invent data. Training on new architectures for object detection should be considered and you may consider multi-frame SSD [33] for video object detection. Also, model B could be tuned via transfer learning utilizing the model maker API which does the training with an already quantized efficientDet lite model. Quantized weights will give good results than the proposed TF API training with double32 weights and then converting them to float16 or int8.

For a reliable distance measuring system, use multiple ultrasonic sensors to create some sort of 3D visualization by involving the depth into the situation. Or you may consider using an RGB-D camera and train your models to estimate depth for every labeled object but this RGB-D camera sensor may increase the cost significantly. Raspberry Pi has a lot of spare I/O pins and multiple protocols for interfacing. Therefore, improving the system to communicate with other systems can be possible consider utilizing any kind of helpful sensor and using multiple feedback options. Decrease the weight of the glasses is a major concern so, the detaching boxes from the glasses' sides and using a communication protocol for wiring. In a real-time manner, dynamic channels (wireless communication) shouldn't be used. So, in my opinion, there should be wired channels between the different wearable attachments and processing units. These channels and devices should be either waterproof or can be detached from the wearables and hence, it can these wearables preferable to be treated as ordinary wearables. For harsh environments, there are special types of ultrasonic which can handle a lot of harder circumstances, especially for outdoor environments.

References

[1] "Blindness and vision impairment," World Health Organization, [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Accessed 14 October 2021].

[2] P. Ackland, S. Resnikoff and B. Rupert, “World Blindness and Visual Impairment: Despite Many Successes, The Problem Is Growing,” *Community Eye Health Journal,* vol. XXX (30), no. 100, pp. 71-73, 2017.

[3] B. Kuriakose, R. Shrestha and E. S. Eika Sandnes, "Tools and Technologies for Blind and Visually Impaired Navigation Support: A Review Article," *IETE Technical Review,* vol. XXXIX (39), no. 1, pp. 3-18, 27

September 2020.

[4] M. Rabani Mohd Romlay, S. Fauziah Toha, A. Mohd Ibrahim and I. Venkat, "Methodologies and Evaluation of Electronic Travel Aids for the Visually Impaired People: A Review," *Bulletin of Electrical Engineering and Informatics,* vol. X (10), no. 3, pp. 1747-1758, June 2021.

[5] C. Ye, S. Hong, X. Qian and W. Wu, "Co-Robotic Cane: A New Robotic Navigation Aid for the Visually Impaired," *IEEE Systems Man and Cybernetics Magazine,* p. 33–42, April 2016.

[6] M. Helmy Abd Wahab, A. A. Talib, H. A. Kadir, A. Johari, A. Noraziah, R. M. Sidek and A. A.Mutalib, "Smart Cane: Assistive Cane for Visually-impaired People," *IJCSI International Journal of Computer Science Issues,* vol. VIII (8), no. 4, July 2011.

[7] "WeWalk smart cane," westminster technologies, [Online]. Available: https://www.westminstertech.com/products/wewalk-smart-cane?variant=31405927923814.

[8] S. B Kallara, M. Raj, R. Raju, N. Mathew, P. V R and D. DS, "Indriya - A Smart Guidance System for the Visually Impaired," *IEEE Xplore Compliant,* pp. 26-29, 23 November 2017.

[9] M. Rahman, M. M. Islam, S. Ahmmed and S. Khan, "Obstacle and Fall Detection to Guide the Visually Impaired People with Real Time Monitoring," *SN Computer Science,* 27 June 2020.

[10] M. A. Rahman and M. Sadi, "IoT Enabled Automated Object Recognition for the Visually Impaired," *Elsevier: Computer Methods and Programs in Biomedicine Update,* vol. I, 21 May 2021.

[11] Sensotec, "OneStep Reader (KNFB READER)," Developed for Android and IOS, 9 October 2015. [Online]. Available: https://apps.apple.com/us/app/onestep-reader/id849732663.

[12] Cloudsight, "TapTapSee," Developed for Android and IOS, 4 April 2014. [Online]. Available: https://play.google.com/store/apps/details?id=com.msearcher.taptapsee.android&hl=ar&gl=US.

[13] M. Douděra and Hayaku, "Cash Reader," Developed for Android and IOS, 26 February 2019. [Online]. Available: https://play.google.com/store/apps/details?id=com.martindoudera.cashreader&hl=ar&gl=US.

[14] Microsoft, "Seeing AI," Develobed only for IOS, 2021. [Online]. Available: https://apps.apple.com/us/app/seeing-ai/id999062298.

[15] Y. Bouteraa, "Design and Development of a Wearable Assistive Device Integrating a Fuzzy Decision Support System for Blind and Visually Impaired People," *Micromachines,* vol. XII (12), no. 9, 7 September 2021.

[16] H.-C. Wang, R. Katzschmann, S. Teng, B. Araki, L. Giarre and D. Rus, "Enabling Independent Navigation for Visually Impaired People through a Wearable Vision-Based Feedback System," *IEEE International Conference on Robotics and Automation (ICRA),* pp. 6533-6540, 29 May 2017.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. Berg, "SSD: Single Shot MultiBox Detector," *arXiv,* 29 December 2016.

[18] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick and H. et al, "Microsoft COCO: Common Objects in Context," *arXiv,* 21 February 2015.

[19] R. Joshi, S. Yadav, M. Dutta and C. Travieso-Gonzalez, "Efficient Multi-Object Detection and Smart Navigation Using Artificial Intelligence for Visually Impaired People," *Entropy,* vol. XXII (22), no. 9, 27 August 2020.

[20] E. EdjeElectronics, "TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi," GitHub, 13 December 2020. [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi/blob/master/Raspberry_Pi_Guide.md.

[21] A. Krizhevsky, V. Nair and G. Hinton, "CIFAR-100 (Canadian Institute for Advanced Research)," [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html.

[22] N. Kamarudin, M. Makhtar, F. Syed Abdullah, M. Mohamad, F. Mohamad and M. F. Abdul Kadir, "Comparison of image classification techniques using caltech 101 dataset," *Journal of Theoretical and Applied Information Technology,* pp. 79-86, September 2015.

[23] M. Everingham, L. Gool, C. K. Williams, J. Winn and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int. J. Comput. Vision,* vol. LXXXVIII (88), no. 2, p. 303–338, June 2010.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 248-255, 2009.

[25] Kaggle, [Online]. Available: https://www.kaggle.com/.

[26] "Caesium Image Compressor - Great Image Compression Tool With High Flexibility," ArtiStudio, 2 November 2021. [Online]. Available: https://wiki.artistudio.xyz/docs/web-development/tools/image-compressor/caesium-image-compressor/.

[27] Tiny JPG, [Online]. Available: https://tinyjpg.com/.

[28] A. E. Egypt-Iris, R. Hisham, S. Tarek and M. ElKarargy, "Egyptian Currency," Kaggle, 1 August 2021. [Online]. Available: https://www.kaggle.com/datasets/egyptiris/egyptian-currency.

[29] Tzutalin, "LabelImg," GitHub, 2015. [Online]. Available: https://github.com/tzutalin/labelImg.

[30] “TensorFlow 2 Detection Model Zoo,” Github, [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md.

[31] N. Renotte, “Tensorflow Object Detection Walkthrough,” GitHub, 3 April 2021. [Online]. Available: https://github.com/nicknochnack/TFODCourse.

[32] Tim, "Face Recognition With Raspberry Pi and OpenCV," Core Electronics, 30 March 2022. [Online]. Available: https://core-electronics.com.au/guides/face-identify-raspberry-pi/.

[33] A. Broad and T.-Y. L. Michael Jones, "Recurrent Multi-frame Single Shot Detector for Video Object Detection," 2018.