Navigation Enabling Application for Blind People

Atindra Nath Sarkar, Jayita Saha, Pamela Das, Ritu Kumari, Tulika Chakraborty, and Naiwrita Dey

Abstract This paper proposes an assistive navigation application system to help visually impaired persons with independent travel in indoor. The proposed system is implemented in two parts: first, the obstacle detection is done by object identification using image processing and the images are stored in cloud further. You only look once (YOLO) version 3 which is a fully convolutional neural network (FCNN) is used for object identification. Twenty classes are considered with 2600 number of image samples. The application assembly is developed in android studio in Java platform. The application is named as "NETRA". The blind person has to be logged in to the application beforehand. Later with text-to-speech conversion, the blind individual will be navigated. In the present work, the object identification using image processing and the overall application assembly is shown and discussed.

Keywords Navigation · Image processing · YOLO · Cloud · Application

1 Introduction

According to WHO, the aggregate of visually disabled individuals is approximately 285 million worldwide, of whom 39 million are totally sightless. A large part of these people lives by themselves due to their children residing abroad for their jobs. Some of the major problem faced by them in their daily life is navigating alone, which may arise due to streets having heavy traffics, potholes and stairs in the path and collision with other pedestrians. The traditional solution for a long time was to use the white cane or walking with the guide dogs. With the advancement of technologies, many new methods are coming up every day, which are more reliable and less embarrassing. J. S. Sierra and J. S. R. Togores designed an application with special controls (buttons, sliders, tables, etc.) provided with Low Vision Mobile

A. N. Sarkar · Jayita Saha · P. Das · R. Kumari · T. Chakraborty
Department of AEIE, RCCIIT, Kolkata, India

N. Dey (✉)
Department of ECE, RCCIIT, Kolkata, India

S. Bhaumik et al. (eds.), *Proceedings of International Conference on Industrial Instrumentation and Control*, Lecture Notes in Electrical Engineering 815,
https://doi.org/10.1007/978-981-16-7011-4_43

App Portal for aged and visually impaired users to access mobile applications on their smartphones. Though this application would help the users to navigate on the smartphone, they will not be of any help in the real-world navigation [1]. A face detection framework has been developed by P. Viola and M. Jones, called robust real-time face detection, which is able to process images swiftly while gaining high detection rates [2]. Various algorithms such as AdaBoost and methods for merging succeeding more composite classifiers in a cascade form which enhances the pace of the detector by concentrating on optimistic areas of the image have also been used in this work. S. K. Chadalawada has done real-time object detection and recognition—using deep learning methods, which identifies, evaluates and compares suitable and highly efficient deep learning models for real-time object recognition and tracking, concluding YOLOv3 as the best algorithm [3]. Route tracking on Android device is done by J. Samualby building a monitoring Android application using object GPS devices to determine its present and previous locations at particular intervals. Unlike other tracking systems, this will facilitate its users to create bookmarks of present locations and to return back from anywhere using Google Maps APIs in case they forget the previous location [4]. In this work, text-to-speech and speech-to-text systems having various steps (speech recognition, feature extraction, language translation, etc.) have been used along with various analysis models and algorithms to process the data and produce the desired outputs [5]. Glass-type wearable terminal using an ultrasonic sensor with a pair of earphones has been designed by the authors which aids not only the user, but also the other pedestrians from collision. Multi agent simulation study has validated the efficiency of the proposed system [6]. A wearable electronic travel assistance for visually impaired persons using HC-SR04, Raspberry Pi 2 computer for controlling the ultrasonic sensors and translating the data from the output to instructions heard by the sightless user using Bluetooth headphone, has been developed in this project [7]. A smart cap for helping the sightless to navigate freely using real-time object detection and identification using Raspberry Pi-3 with a pre-trained convolutional neural network (CNN) model has been developed in this project using TensorFlow using NoIR camera [8]. The authors of this paper have studied and reviewed the ETAS for people who are visually impaired, and a new wearable system CASBliP has been described in this paper [9]. Object detection with R-CNN using category-independent region proposals, convolutional neural network that extracts a predetermined length feature vector from each area and set of class-specific linear SVMs, has been done in this project [10]. Real-time-based object detection, using SSD algorithms, mAP and FPS as standard parameters for object detection, has been done in this work [11]. Along with this study, a novel method of navigation assistance tools for blind people has been proposed in this work.

The paper is ordered as follows. In Sect. 2, the overall methodology of the work is explained. In Sect. 3, object detection and identification are discussed. Image storage in cloud is discussed in Sect. 4. In Sect. 5, the real-time application assembly is presented followed by a conclusive statement along with future extension of this work. References are defined at the end.

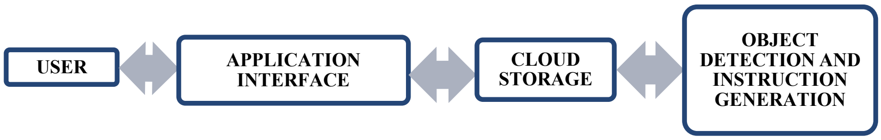


Fig. 1 Schematic overview of the methodology used

2 Methodology

The proposed system will include a mobile application that can interact with the user and send the data to the cloud storage where object detection is performed and required results are returned back to the user. At first, the user needs to enter the login credentials. Once the authentication is verified, the application, enabled with real-time location tracking, activates speech recognizer for converting speech to text. Then, the application can be used to capture images simultaneously and upload to the cloud storage. A fully convolutional neural network, called YOLOv3, is used to detect the obstacles or objects in the images. For this, a custom dataset comprising of 2600 images is created which is used to train the model. The images obtained from the mobile application are passed through the trained model, and output in the form of text is stored back to the cloud. This information is transferred to the mobile application which converts the text to speech and gives the required instruction to the user. Figure 1 shows the overall block diagram of the system.

3 Object Detection and Identification

Object detection is a computer technology that is associated with image processing and computer vision. Instances of the interpreted objects of a definite class present in a digital image or in videos are detected in this process. It comprises two parts—object classification (what objects?) and object localization (where are the objects located?). Well-researched fields of the object detection accommodate face detection in addition to pedestrian detection. The two major methods of object detection are machine learning construct and deep learning approaches.

In machine learning, it is necessary to define prior traits and then use a method like support vector machine (SVM) for the categorization. On the contrary, deep learning methods can detect entities without even explicitly defining features and are constructed on convolutional neural networks (CNNs). G. Shrestha et al. had earlier used CNN-based method for agricultural productivity through detection of plant diseases [12].

YOLOv3 is used in this proposed model for image processing. Figure 2 sums up the entire process.

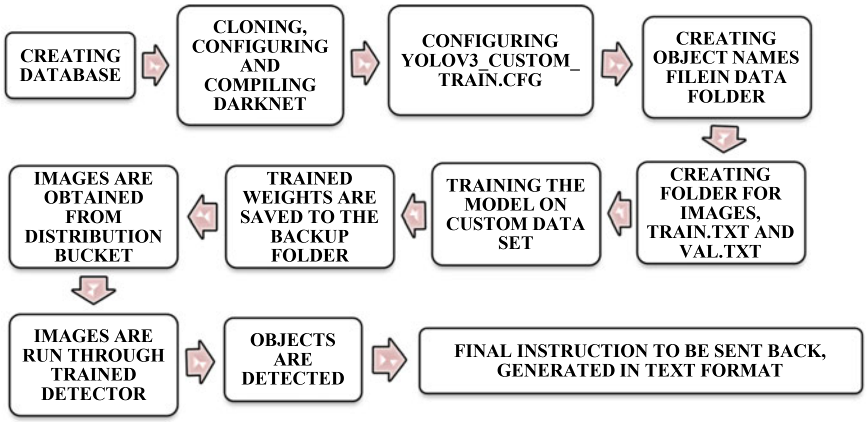

CREATING DATABASE
CLONING, CONFIGURING AND COMPILING DARKNET
CONFIGURING YOLOV3_CUSTOM_ TRAIN.CFG
CREATING OBJECT NAMES FILEIN DATA FOLDER
CREATING FOLDER FOR IMAGES, TRAIN.TXT AND VAL.TXT
TRAINING THE MODEL ON CUSTOM DATA SET
TRAINED WEIGHTS ARE SAVED TO THE BACKUP FOLDER
IMAGES ARE OBTAINED FROM DISTRIBUTION BUCKET
IMAGES ARE RUN THROUGH TRAINED DETECTOR
OBJECTS ARE DETECTED
FINAL INSTRUCTION TO BE SENT BACK, GENERATED IN TEXT FORMAT

Fig. 2 Object detection process

3.1 YOLOv3

You only look once (YOLO) version 3 is an algorithm used for object detection. It is a fully convolutional neural network (FCNN) that can be used in real-time detection purposes without too much loss in accuracy. The whole image is being forwarded by YOLOv3 only once through the network. Features provided by YOLOv3:

- Ease to integrate with an OpenCV application.
- OpenCV CPU version is 9$\times$ faster.
- Supports Python.

Features of the model:

- It comprises 75 convolutional layers, including up-sampling and skip connection layers.
- It uses Darknet53 framework (the layers are shown in Table 1) to train neural networks. In detection process, another 53 layers are stacked onto the original 53 layers, making it a 106-layer fully convolutional network.
- Leaky ReLU is chosen as activation function. It is given by $f(x) = \max(0.01x, x)$.
- 1×1 detection kernels are applied on feature maps of three dissimilar sizes at three separate positions in the network. The detection kernel form is given by $1 \times 1 \times (B \times (5 + C))$ where B represents the count of bounding boxes a cell on the feature map can extract and C is the count of object classes. Figure 3 contains the details of the same.

If anyone uses 1×1 convolutions, the dimensions of the prediction map will be completely the proportion of the feature map. In YOLOv3, prediction map is explicated for each cell to predict a definite number of bounding boxes. It can be

Table 1 Layers of Darknet53

	Type	Filters	Size	Output
	Convolution	32	3 × 3	256 × 256
	Convolution	64	3 × 3/2	128 × 128
	Convolution	32	1 × 1	
1 ×	Convolution	64	3 × 3	
	Residual			128 × 128
	Convolution	128	3 × 3/2	64 × 64
	Convolution	64	1 × 1	
2 ×	Convolution	128	3 × 3	
	Residual			64 × 64
	Convolution	256	3 × 3/2	32 × 32
	Convolution	128	1 × 1	
8 ×	Convolution	265	3 × 3	
	Residual			32 × 32
	Convolution	512	3 × 3/2	16 × 16
	Convolution	256	1 × 1	
8 ×	Convolution	512	3 × 3	
	Residual			16 × 16
	Convolution	1024	3 × 3/2	8 × 8
	Convolution	512	1 × 1	
4 ×	Convolution	1024	3 × 3	

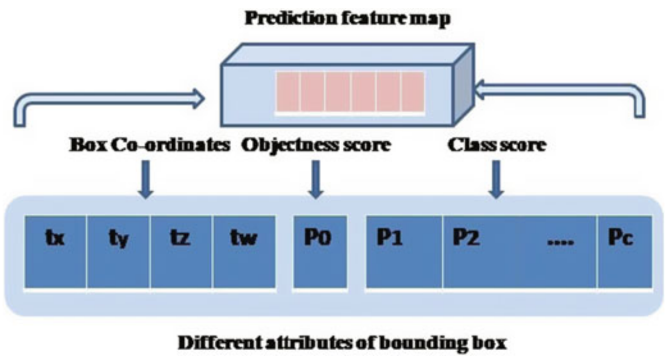


Fig. 3 Object detection process

assumed that depth-wise, there are (B x (5 + C)) appearances in a fixed feature map and the number of bounding boxes that every cell may predict is presented by B. Therefore, every one of these B bounding boxes will be restricted in determining a particular type of entity. Every bounding box has 5 + C traits, which are explained as the dimensions, centre coordinates, the objectness score and class confidences for each bounding box. Three bounding boxes for each cell will be prognosticated by YOLOv3. User can anticipate that every cell of that feature map will identify an entity with one of the bounding boxes supposing that the centre of that entity resides well in the area of the cell. For example, if the input image is of 416 × 416, with network stride 32, the feature map's dimensional size will be of 13 × 13. The inserted image can be split into cells of 13 × 13.

On the input image, the cell which accommodates the centre of the ground truth box of the object is accountable for predicting the object. User can allocate the 7th cell of 7th row on the feature map accountable for distinguishing the object. Now, by this cell three bounding boxes can be predicted.

- Pooling layers are not present; instead, a stride 2 convolutional layer is deployed to facilitate down-sampling.
- This aids in ceasing the low-level loss of features that arises due to pooling. In case of YOLO, the prediction is made by using a convolutional layer which uses 1 × 1 convolutions. It uses 3 anchor boxes for each scale, thus 9 in total. For example, for an image of size of 416 × 416, YOLOv3 predicts ((52 × 52) + (26 × 26) + (13 × 13)) = 10647 bounding boxes.

A group of preset default bounding boxes of a certain height and width are called anchor boxes, which can predict the height and width of a bounding box, whereas actually, during training, it can conduct to unstable gradients. In spite of that, log-space transforms are predicted by almost all modern object detectors. Later on, these transforms are put to the anchor boxes for acquiring the output. In YOLOv3, there are 3 anchors. Among them there are 3 bounding boxes per cell for outcome prediction purpose. The bounding box which will be accountable for detection of the required entity will have the highest IoU with respect to the ground truth box. The below-mentioned formulae explain how the output of the network is transfigured to acquire predictions of the bounding box.

$$b_x = \sigma(t_x) + c_x, by = \sigma(t_y) + c_y, b_z = \sigma(t_z) + c_z \tag{1}$$

$$b_w = p_w e^{tw}, b_h = p_h e^{th} \tag{2}$$

where b_x, b_y are the x, y centre coordinates, and b_w, b_h are the width and height of prediction. T_x, t_y, t_w, t_h are the results of the network output. c_x, c_y are the top-left grid coordinates. p_w, p_h are size of the anchors of the boxes. Generally, absolute entire coordinates of a bounding box's centre are not perceived by YOLOv3. Rather, offsets are predicted by it which:

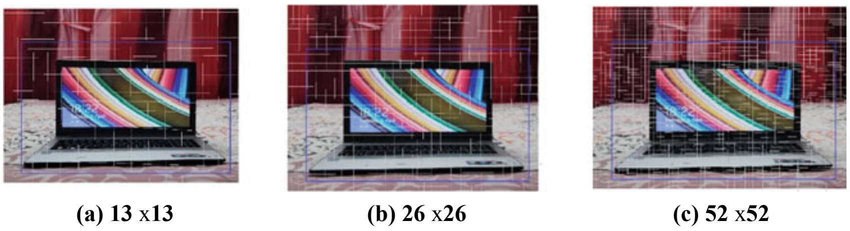

(a) 13 x13 (b) 26 x26 (c) 52 x52

Fig. 4 Prediction feature maps using different scales

- are relative to the grid cell's top-left corner that predicts the entity.
- are normalized by the size of the feature map's cell, that is, 1.

For instance, if the prediction for centre in an image is (0.4, 0.7), as the top-left coordinates of the bounding cell are, is (6, 6), then the centre lies at (6.4, 6.7), on the 13 × 13 feature map. The expectation of an entity that is accommodated within a bounding box is represented by object score. It is mostly one for the centre and its neighbouring grids, but almost zero for those corner grids.

The objectness score then goes through the sigmoid, after which it is explicated as probability. The probabilities of the entities detected that belongs to definite classes (person, dog, car, plant, etc.) are represented by class confidences. Across three different scales, YOLOv3 can make prediction. Feature maps with strides of 32, 16 and 8 are chosen for making detection, which implies, for an input of 416 × 416, user can train to detect in order of 13 × 13, 26 × 26 and 52 × 52 (see Fig. 4).

Down-sampling is performed on the inserted image by the network till first detection layer. Here, stride 32 feature maps are used. Later on, up-sampling by an element of 2 is done on the layers and joined with feature maps of the preceding layers which have alike feature map magnitudes. Repetitions occur at layers with stride 16 and finally with stride 8. At every step, three bounding boxes are estimated by every cell with three anchors, so the entire number of anchors exerted is nine.

- The finest bounding box for an entity is determined by deploying non-max suppression.

3.2 Dataset Used

A custom image dataset is prepared. Dataset details has been listed in Table 2. The process of image annotation is shown in Fig. 5.

Table 2 Custom dataset details

Number of object classes	20
Total number of images used	2600
Number of images in train.txt (70% of total images)	1820
Number of images in val.txt (30% of total images)	780

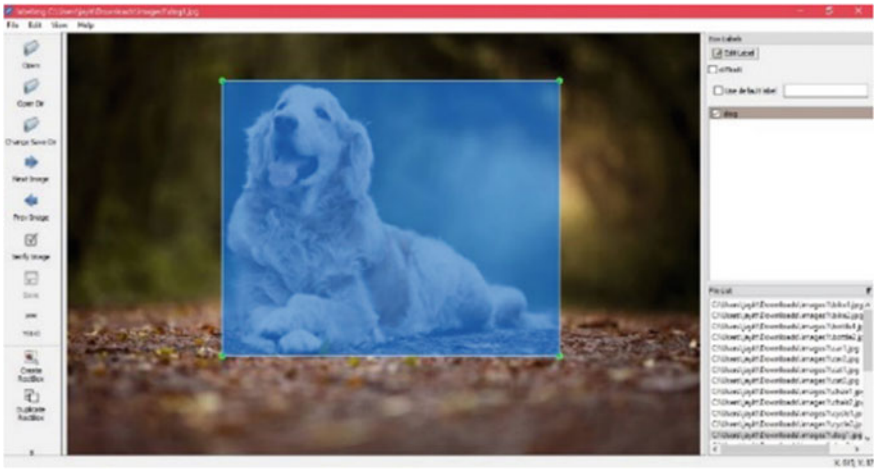


Fig. 5 Object marking using labelImg for annotation

The training steps for object detection are creating dataset, cloning, configuring and compiling Darknet, configuring Yolov3.cfg file, creating object.names file inside the data folder (it contains count of entities, the address of train.txt, test.txt, object.names files and the address where the YOLO weights are to be stored), creating folder for images and creating train.txt file in data folder. After the model is trained, it will be able to detect random objects. (A pre-trained COCO dataset can also be used instead, which contains 80 labels including people, cycles, dogs, etc.)

Generating instruction for navigation guidance. Based on the output of the depth estimation model, final text outputs such as "obstacle ahead" or "<detected object> is on your left" will be generated.

4 Image Storage Cloud

Cloud storage is used to reserve data, files, images, etc., in an external area which can be retrieved through the Internet. After receiving the request from the user, the images are uploaded to the specific area of cloud storage. The V4 signing process is used to create Cloud Storage RSA key signed URLs into the workflow of the app for uploading images to the Google Cloud. After validation, the image files is sent to the Google Colab for image processing. A service hosted is implemented on Google Cloud platform. Using this, a user can upload images constantly. In the present work, a serverless architecture is used. Using this architecture, an unspecified number of user can access cloud resources seamlessly. The requirements involved to implement this service are—managed service, authenticated users, validation of the contents. After receiving request from the user, a signed URL is generated. Signed URL allows

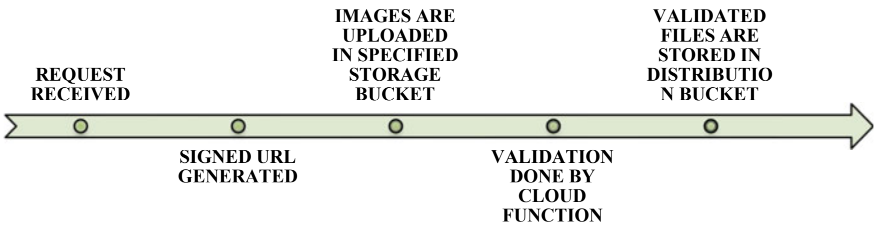

REQUEST RECEIVED
SIGNED URL GENERATED
IMAGES ARE UPLOADED IN SPECIFIED STORAGE BUCKET
VALIDATION DONE BY CLOUD FUNCTION
VALIDATED FILES ARE STORED IN DISTRIBUTIO N BUCKET

Fig. 6 Cloud storage workflow

PUT request that can be executed for a specific cloud storage bucket. Images can be uploaded in the specific bucket. After uploading images, validation is done by cloud function. Next, the validated files are uploaded in the distribution bucket. Figure 6 shows the workflow.

The steps involved in generation of signed URL:

- At first, a new service account is created.
- Then, a necessary string is created for generating a signed URL.
- Next, PUT is chosen for cloud storage.
- At the end, the bucket and object are specified. Storage, buckets.get, storage.objects.create, storage.objects.has are granted for the same.

5 Application Assembly

Entire process of developing application can be summarized in certain steps that has been carried out, and it is given below. Figure 7 shows the block diagram of the same.

Step 1 Creating the app interface in Android studio using Java—In the Main Activity Page, "Intent" object is created and "setOnClickListener" method is used to move to the next page. For Login Activity Page, two "EditText

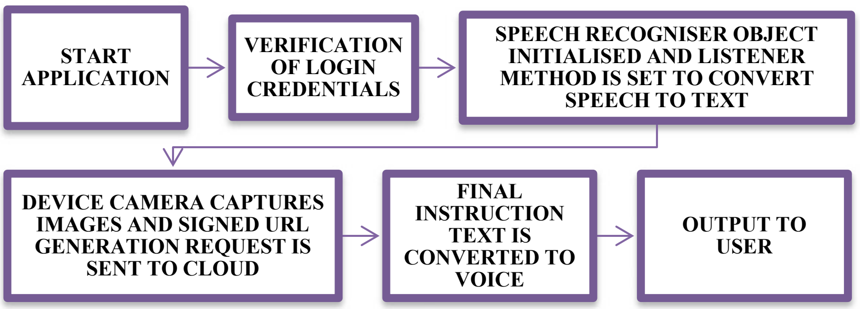

START APPLICATION
VERIFICATION OF LOGIN CREDENTIALS
SPEECH RECOGNISER OBJECT INITIALISED AND LISTENER METHOD IS SET TO CONVERT SPEECH TO TEXT
DEVICE CAMERA CAPTURES IMAGES AND SIGNED URL GENERATION REQUEST IS SENT TO CLOUD
FINAL INSTRUCTION TEXT IS CONVERTED TO VOICE
OUTPUT TO USER

Fig. 7 Mobile application workflow

fields" (username and password) are sent to the cloud using Volley Library API. Authorization success/failure status is collected in JSON Parser file in XML format. The device camera is used to capture one image per second which is simultaneously uploaded to the cloud using the signed URL method.

Step 2 Taking voice commands (speech-to-text)—After initializing the speech recognizer object, the "Intent" object is created and language is evaluated. Next, the listener method is set to convert speech to text.

Step 3 Returning voice information (text-to-speech).

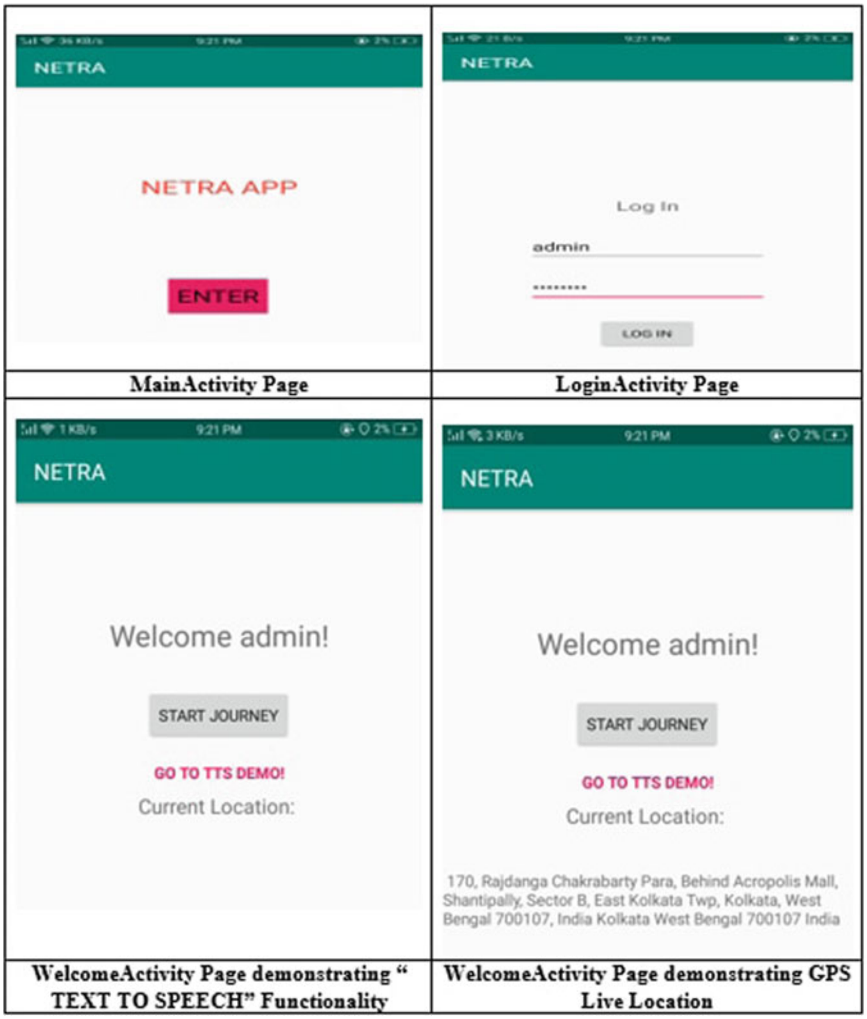
Step 4 Permissions are required for Google Maps, device camera, text, messaging, calls and device microphone.

6 Result

Figure 8 shows the application interface, and Figs. 9 and 10 show the object detection.

7 Conclusion

A mobile application is implemented to assist blind people to be used in indoor as presented here. Obstacle detection and identification are carried out with image processing. The effectiveness of the implemented system is experimented within the room with blindfolded, and the mobile application is worked properly in synchronization with object detection. Further work is to be done to test staircase identification and also for outside locations to make the system more robust and functional.



170, Rajdanga Chakrabarty Para, Behind Acropolis Mall, Shantipally, Sector B, East Kolkata Twp, Kolkata, West Bengal 700107, India Kolkata West Bengal 700107 India

Fig. 8 Real-time application view

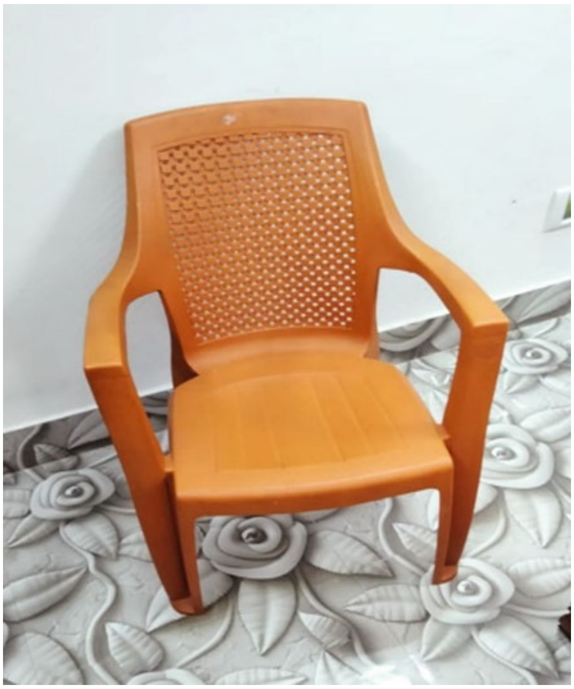


Fig. 9 Image before object detection

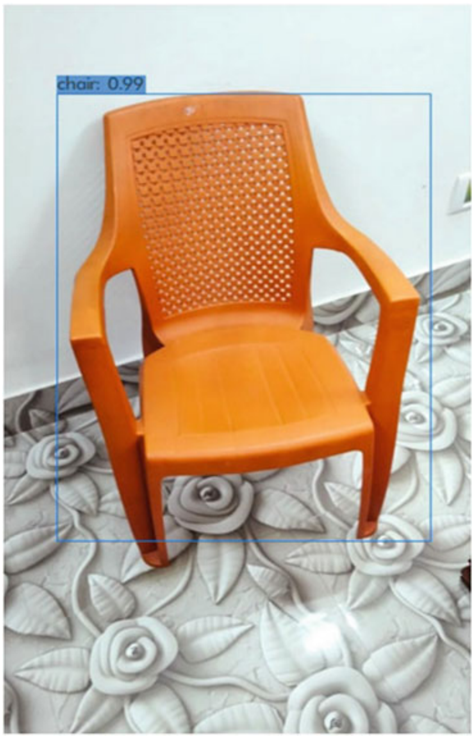


Fig. 10 Object detected by the model

References

1. Sierra, J.S., Togores, J.S.R.: Designing mobile apps for visually impaired and blind users—using touch screen based mobile devices: iPhone/iPad". In: ACHI 2012: The Fifth International Conference on Advances in Computer-Human Interactions, Copyright (c) IARIA, 2012, pp. 47–52, ISBN: 978-1-61208-177-9
2. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vision **57**, 137–154 (2004). https://doi.org/10.1023/B:VISI.0000013087.49260.fb
3. Chadalawada, S.K.: Real Time Object Detection and Recognition—Using Deep Learning Methods, pp. 1–52 (2020)
4. Samual, J.: Implementation of GPS based object location and route tracking on android device. Int. J. Inf. Syst. Eng. **3**(2), 61–72 (2015). ISSN: 2289-7615. https://doi.org/10.24924/ijise/2015.11/v3.iss2/61.72
5. Trivedi, A., Pant, N., Shah, P., Sonik, S., Agarwal, S.: Speech to text and text to speech recognition systems a review. IOSR J. Comput. Eng. (IOSR-JCE) **20**(2), 36–43 (2018). e-ISSN: 2278-0661, p-ISSN: 2278-8727. https://doi.org/10.9790/0661-2002013643
6. Hiroto, K., Katsumi, H.: A Wearable Walking Support System to provide safe direction for the blind. In: 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), pp. 1–4, 2019. Electronic ISBN: 978-1-7281-3271-6, USB ISBN: 978-1-7281-3270-9, ISBN: 978-1-7281-3272-3. https://doi.org/10.1109/ITC-CSCC.2019.8793305

7. Laubhan, K., Trent, M., Root, B., Abdelgawad, A., Yelamarthi, K.: A wearable portable electronic travel aid for blind. In: International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016, pp. 1999–2003. https://doi.org/10.1109/ICEEOT.2016.7755039
8. Nishajith, A., Nivedha, J., Nair, S.S., Shaffi, J.M.: Smart cap—wearable visual guidance system for blind. In: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 275–278, 2018, Electronic ISBN: 978-1-5386-2456-2. ISBN: 978-1-5386-2456-2. https://doi.org/10.1109/ICIRCA.2018.8597327
9. Dunai, L.D., Lengua, I.L., Tortajada, I., Simon, F.B.: Obstacle detectors for visually impaired people. In: 2014 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM), pp. 809–816, 2014, Electronic ISBN: 978-1-4799-5183-3, Print ISSN: 1842-0133. https://doi.org/10.1109/OPTIM.2014.6850903
10. Girshick, R., Donahue, J.: Region-based convolution networks for accurate object detection and segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 142–158 (2018). Print ISSN: 0162-8828, CD: 2160-9292, Electronic ISSN: 1939-3539. https://doi.org/10.1109/TPAMI.2015.2437384
11. Kumar, A., Zhang, Z.J., Lyu, H.: Object detection in real time based on improved single shot multi-box detector algorithm. EURASIP J. Wirel. Commun. Netw. **204**, 1–18 (2020). https://doi.org/10.1186/s13638-020-01826-x
12. Shrestha, G., Deepsikha, D.M., Dey, N.: Plant disease detection using CNN. In: 2020 IEEE Applied Signal Processing Conference (ASPCON), 2020, pp. 109–113. Electronic ISBN: 978-1-7281-6882-1, USB ISBN: 978-1-7281-6881-4, ISBN: 978-1-7281-6883-8. https://doi.org/10.1109/ASPCON49795.2020.927672