Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 24, 2025

VISIONTOOL: A DEEP LEARNING EMBEDDED DEVICE FOR BLIND PEOPLE

Rakhi Bharadwaj	Vishwakarma Institute of Technology, Pune, India	rakhi.bharadwaj@vit.edu
Bhushan Bachewar	Vishwakarma Institute of Technology, Pune, India	bhushan.bachewar221@vit.edu
Shreya Barsude	Vishwakarma Institute of Technology, Pune, India	shreya.barsude221@vit.edu
Harsh Badagandi	Vishwakarma Institute of Technology, Pune, India	harsh.badagandi22@vit.edu
Snehal Darade	Vishwakarma Institute of Technology, Pune, India	snehal.darade22@vit.edu

* Corresponding author

ABSTRACT

Aim/Purpose	This paper presents a proof-of-concept device designed to create an affordable, real-time vision assistance tool for visually impaired individuals. The device detects obstacles, estimates their distance, provides an audio alert, and informs caretakers of the user's live location. It addresses the lack of portable, low-cost, and connected navigation aids for blind users.
Background	This paper addresses the problem by combining computer vision, distance estimation, audio alerts, and real-time caregiver notification through a mobile app, offering a smart solution to enhance safety and independence for visually impaired individuals.
Methodology	The system uses a Raspberry Pi with a camera module running a YOLOv8 object detection model for real-time object recognition, an ultrasonic sensor (or camera depth estimation) for distance measurement, and a Flutter-based mobile app to send live GPS coordinates to caretakers. Testing was done in controlled indoor and outdoor environments.
Contribution	This project contributes a multi-functional assistive tool that not only alerts visually impaired users about nearby obstacles with distance information but also

Accepting Editor Aaron M. Glassman | Received: April 30, 2025 | Revised: July 12, 2025 | Accepted: August 22, 2025.
Cite as: Bharadwaj, R., Bachewar, B., Barsude, S., Badagandi, H., & Darade, S. (2025). VisionTool: A deep learning embedded device for blind people. *Journal of Information Technology Education: Innovations in Practice, 24,* Article 24. https://doi.org/10.28945/5611

	keeps caretakers informed, combining safety, mobility, and communication in a single solution.
Findings	The paper's major findings demonstrate the achievement of real-time object detection with approximately 20-25% mean average precision (mAP) and reliable distance measurements. Audio feedback about both object type and distance significantly improved user navigation. The Flutter app successfully updated caretakers with the user's live location, enhancing safety. The system is portable, battery-operated, and user-friendly.
Recommendations for Practitioners	Calibrate distance sensors carefully for different environments (indoor vs outdoor). Customize audio feedback to include critical distances (e.g., warn more urgently when obstacles are very close). Ensure Flutter app notifications are lightweight and battery-optimized.
Recommendations for Researchers	Explore integrating AI-based path planning alongside object detection. Investigate methods to improve GPS accuracy in indoor environments. Research multi-sensor fusion (camera + ultrasonic + inertial sensors) for better obstacle detection.
Impact on Society	This tool not only increases mobility for visually impaired individuals but also reassures families and caretakers by providing live location tracking, leading to greater independence, confidence, and peace of mind for users and their loved ones.
Future Research	Future work could involve adding features such as automatic emergency alerts if the user remains stationary for too long, voice-command-based control of the device, and integrating AI-driven adaptive navigation suggestions based on user movement patterns. Additionally, adding face recognition helps individuals recognize the person in front of them.
Keywords	vision assistance, object detection, Raspberry Pi, GPS tracking, SSDlite MobileNet V2 model, Flutter app

INTRODUCTION

Individuals with visual impairments often face significant challenges in navigating their daily environment, including difficulties in detecting obstacles, estimating distances, and identifying key objects. These challenges not only hinder their independence but also increase their reliance on others for basic mobility and spatial awareness. Although several assistive technologies have been developed to address these issues, many existing solutions are either prohibitively expensive, lack real-time feedback, or offer limited functionality, making them inadequate for practical, everyday use.

To bridge this gap, we present **VisionTool**, a comprehensive, real-time vision assistance system designed specifically for visually impaired users. The device combines the strengths of embedded computing, deep learning, and cloud connectivity to deliver obstacle detection, distance estimation, GPS tracking, and emergency alerts – all in a single, affordable unit. The core hardware includes a Raspberry Pi 4B paired with a camera and ultrasonic sensor. Under normal conditions, captured images are sent to AWS Rekognition for cloud-based object identification. However, in the absence of connectivity or when latency exceeds 100ms, the system automatically switches to a locally deployed SSDlite MobileNet V2 model on the Raspberry Pi for real-time object detection.

In addition to object identification, VisionTool incorporates spatial awareness using ultrasonic sensors and delivers real-time audio feedback via Bluetooth earphones. A GPS module updates the user's location in Firebase, enabling caregivers to monitor movement in real time through a Flutter-

based mobile application. An integrated emergency button enhances safety by triggering instant alerts and live location sharing.

To handle multiple tasks concurrently – such as image processing, sensor reading, and data transmission – the system uses multiprocessing, ensuring smooth performance without delay. This hybrid, modular, and user-friendly approach sets VisionTool apart from conventional solutions by combining AI, IoT, and embedded systems into a unified assistive tool. To ensure affordability, low latency, and seamless operation, all hardware and software platforms were selected through a comparative analysis of available options, as detailed in the Methodology section. To validate the proposed system, this paper is structured as follows. The next section presents a review of related work. An outline of the methodology and system architecture is then presented, followed by the results and analysis. The paper concludes with future directions.

LITERATURE REVIEW

Several assistive systems have historically leveraged ultrasonic and infrared sensors for obstacle detection. Researchers such as Mugwenhi and Mudawarima (2023), A et al. (2024), Layaraja et al. (2024), and R et al. (2019) employed ultrasonic sensors to detect nearby objects and provide feedback via voice or vibration. These systems were praised for their affordability and simplicity, but often suffered from inaccuracies due to environmental noise. Similarly, Ou et al. (2020) and Khan et al. (2018) introduced infrared and ultrasonic-based devices with auditory and haptic cues, enhancing real-time obstacle mapping but still limited by sensor range and response speed. Mohajeri et al. (2011) and Shahira et al. (2019) explored the combination of ultrasonic sensors with depth estimation or tactile vests to enhance spatial awareness, although they also noted the need for more precise distance readings and clearer feedback.

To overcome the limitations of traditional sensors, researchers have increasingly turned to deep learning. Sanjay and Ahmadinia (2019) introduced MobileNet-Tiny, optimized for real-time detection on embedded systems, achieving a strong balance between accuracy and speed. Khandewale et al. (2020) and Atitallah et al. (2024) employed SSDLite MobileNet V2 and other CNN-based models for real-time object detection using lightweight architectures suited for edge devices like the Raspberry Pi. Islam et al. (2023) and Said et al. (2023) combined deep learning with audio description or tactile feedback to generate rich environmental understanding for blind users.

Despite their success, most of these models were constrained either by hardware limitations or the need for stable internet connectivity for cloud-based inference. Our VisionTool system enhances reliability by implementing a hybrid detection mechanism – using AWS Rekognition for high-accuracy detection when online and SSDLite MobileNet V2 via TensorFlow Lite for offline use, ensuring continuity in varying network conditions.

Embedded platforms play a critical role in assistive technology development. Maksimović et al. (2014) and David et al. (2021) assessed Raspberry Pi and TensorFlow Lite Micro (TFLM) for low-power, real-time inference in IoT applications. Compared to alternatives like Arduino (lacking processing power) and NVIDIA Jetson (higher cost and power demands), Raspberry Pi offered a cost-effective middle ground for processing camera input, running AI models, and integrating peripheral sensors. S et al. (2016) also demonstrated successful integration of GPS modules with Raspberry Pi for spatial awareness, although GPS accuracy remained a challenge.

Our work extends this approach by employing multiprocessing on Raspberry Pi to concurrently manage object detection, distance estimation, GPS tracking, and real-time audio output, ensuring efficient and uninterrupted performance even with limited hardware resources. Feedback mechanisms are essential for blind users to interact with assistive devices effectively. Buchs et al. (2017) and Texeira et al. (2023) explored haptic interfaces like the EyeCane and wearable vibrating systems to enhance obstacle detection. While these systems improved navigation over traditional white canes, they

often required significant user training and failed to convey rich contextual information. Voice feedback, on the other hand, as demonstrated by Chavan et al. (2024) and Mugwenhi and Mudawarima (2023) proved more intuitive but suffered from delays or a lack of spatial detail in some implementations. Our approach builds on these insights by integrating real-time audio alerts using a text-to-speech system that reports both the object type and its estimated distance (e.g., "Chair, 1.5 meters ahead"). This dual-feedback model combines the immediacy of auditory information with the spatial precision of sensor data.

Robust training datasets are foundational for any vision-based system. Lin et al. (2014) introduced the Microsoft COCO dataset, which has become the standard for object detection benchmarks, with over 80 object categories and high-quality annotations. This dataset has been widely adopted in works like Sanjay and Ahmadinia (2019) and Atitallah et al. (2024), enabling generalization across varied environments. Our model also leverages the COCO dataset for training, ensuring compatibility with commonly encountered real-world objects. This helps VisionTool provide meaningful alerts to users in everyday scenarios, improving its practical effectiveness.

In summary, while prior research has made significant strides in vision and sensor-based navigation aids, existing systems often lack the integration, scalability, or adaptability to function under variable network and environmental conditions. By synthesizing the best of embedded computing, hybrid AI inference, and multimodal feedback, VisionTool offers a unified solution that advances the current state of assistive navigation systems for visually impaired users.

These works collectively highlight both the promise and limitations of current assistive technologies. In the following section, we present the design and implementation of VisionTool, which integrates these ideas into a cohesive, real-time assistive device.

METHODOLOGY

OVERVIEW OF SYSTEM ARCHITECTURE

The system architecture for the proposed vision assistance device is shown in Figure 1. The development process follows a well-organized and thorough approach to ensure the device is functional, reliable, and user-friendly. This methodology combines both hardware and software components to deliver real-time object detection, distance measurement, and GPS tracking, specifically designed to assist blind individuals.

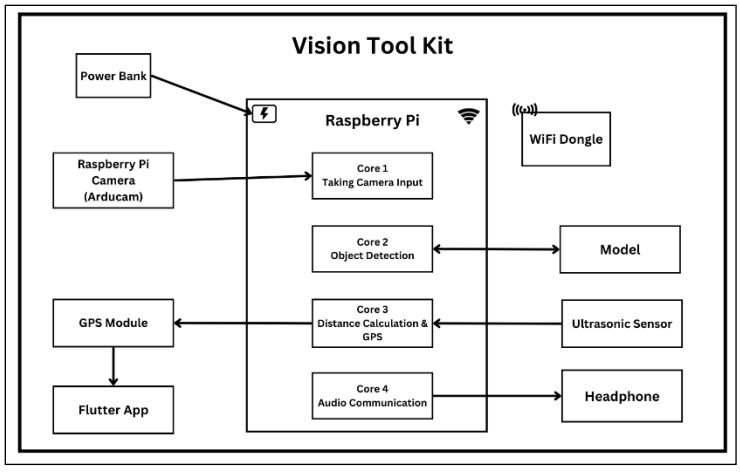


Figure 1. System architecture

HARDWARE SELECTION AND JUSTIFICATION

The Raspberry Pi 4B was selected due to its balance between processing power, GPIO flexibility, and affordability. Compared to alternatives like Arduino (limited processing) or NVIDIA Jetson Nano (higher cost and power consumption), the Raspberry Pi provides an ideal middle ground for real-time AI applications with peripheral support. The HC-SR04 ultrasonic sensor was chosen for its accuracy in short to medium ranges and ease of integration with Raspberry Pi GPIO pins. The Arducam camera was used for its plug-and-play compatibility and CSI support, unlike USB-based webcams that introduce latency.

To enhance user safety and enable caregiver monitoring, a GPS module is included. It continuously transmits the user's real-time location to a Firebase database, allowing remote tracking. Audio feedback, a vital feature for blind users, is provided through Bluetooth earphones, ensuring a hands-free experience. The system is powered by a portable battery, making it both lightweight and mobile. Additionally, AWS Image Recognition has been integrated to improve the accuracy of object detection and classification.

The circuit diagram in Figure 2 illustrates how the various components are connected to the Raspberry Pi 4B, enabling functionalities like distance measurement, GPS tracking, audio output, AWS image recognition, and emergency alerts. Each connection is carefully detailed to ensure smooth interaction between hardware and software.

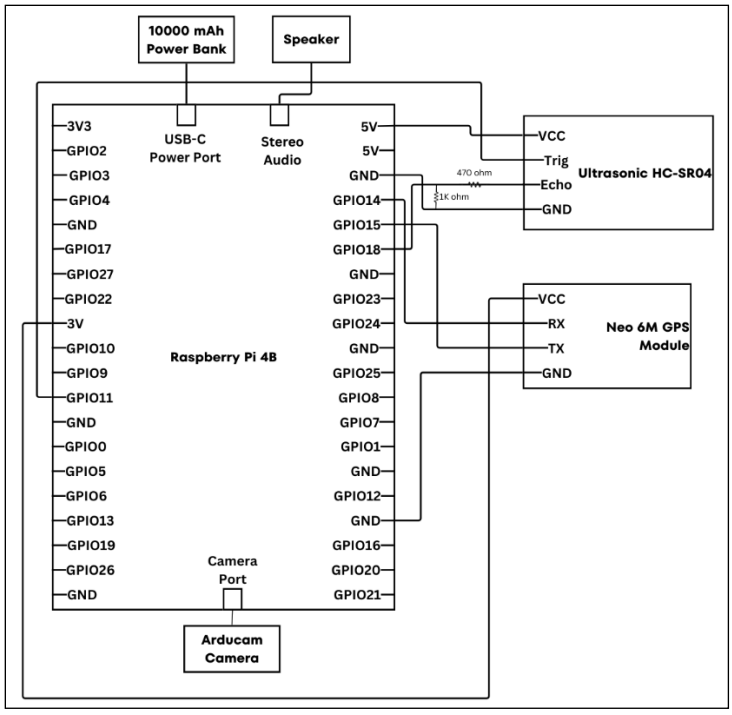


Figure 2. System circuit diagram

A key feature of the VisionTool is the emergency button. When pressed, the Raspberry Pi sends an emergency alert to Firebase, triggering an immediate notification on the Flutter app. This alerts caregivers or family members about the user's emergency, providing real-time assistance. This hands-free solution significantly improves safety, making the device even more effective for blind users.

The HC-SR04 ultrasonic sensor is linked to the Raspberry Pi for measuring distances. Its VCC pin connects to the 5V power output on the Raspberry Pi, supplying the necessary voltage for operation. The GND pin is attached to a ground (GND) pin on the Raspberry Pi, completing the power circuit. The Trig pin, responsible for emitting ultrasonic pulses, is connected to GPIO14, while the Echo

pin, which detects the reflected pulses, is connected to GPIO15. To reduce the 5V signal to a safe 3.3V level for the Raspberry Pi's GPIO, a voltage divider circuit using a 470-ohm resistor and a 1k-ohm resistor is implemented between the Echo pin and GPIO15. An Arducam camera is connected to the Raspberry Pi's dedicated Camera Serial Interface (CSI) port. The CSI connection ensures high-speed data transfer between the camera and the Raspberry Pi.

CAMERA CALIBRATION

Camera images are not perfect; lenses introduce distortions. The first crucial step in working with the camera and getting the real-world objects and their coordinates involves camera calibration, in which the 2D (two-dimensional) objects are mapped to 3D (three-dimensional) real-world objects. The pin-hole camera model is the foundational concept behind how we mathematically model cameras in computer vision.

Figure 3 illustrates the pinhole camera model, which explains how a 3D point in space P (X, Y, Z) is projected onto a 2D image plane as pixel coordinates (u,v). Given an image point(u,v), the corresponding direction in camera coordinates is calculated using the camera's intrinsic parameters:

$$x = \frac{u - c_x}{f_x}, \quad y = \frac{v - c_y}{f_y}$$

where (f_x) and (f_y) are the focal lengths in pixel units and (c_x, c_y) is the principal point. These normalized coordinates define a ray from the camera center through the image plane into the 3D world. To determine the actual 3D coordinates (X, Y, Z), the depth Z of the point must be known. Once Z is available, the real-world coordinates can be recovered as:

$$[X = x \cdot Z, \quad Y = y \cdot Z, \quad Z = Z]$$

This coordinate mapping helps in knowing the distance between the detected objects.

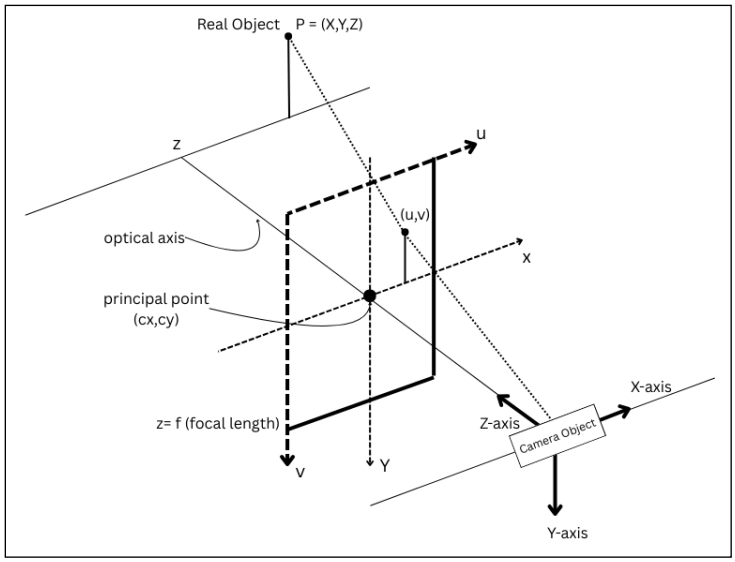


Figure 3. Pinhole camera model – geometric interpretation

Image acquisition

A total of 30-40 images of a 9×6 chessboard pattern were captured from different angles and positions using the Raspberry Pi camera. The physical size of each square on the chessboard was 20 mm. These images form the basis for finding the mapping between 3D world points and their 2D image projections.

Defining 3D-2D point correspondences

For each image, a set of 3D object points was generated, assuming the chessboard lies on the Z=0 plane. These were of the form:

$$[\text{Object Points:}(X, Y, 0) \quad \text{where} \quad X, Y \in \{0,20,40, \dots\} \text{ mm}]$$

The corresponding 2D image points were detected using OpenCV's corner detection functions. These points are refined using subpixel accuracy.

Final calibration

Using all object–image point pairs, the intrinsic and distortion parameters were estimated using the pinhole camera model:

$$\begin{bmatrix} s \cdot u \\ s \cdot v \\ s \end{bmatrix} = K \cdot [\, R \quad | \quad t \,] \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where (X, Y, Z) are 3D coordinates, (u, v) are 2D image coordinates, s is the scaling factor, K is the camera intrinsic matrix, and $[R \mid t]$ are rotation and translation (extrinsic parameters).

The intrinsic matrix K is of the form:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

The distortion coefficients were computed as:

$$Distortion = [k_1, k_2, p_1, p_2, k_3]$$

where k_1, k_2, k_3 are radial distortion coefficients. They correct the "barrel" or "pincushion" distortion where straight lines appear curved, especially at the edges of the image, and p_1, p_2 are tangential distortion coefficients. They correct distortions caused by a slight misalignment of the lens with the image sensor, resulting in points being displaced in a tangential direction.

Object Detection System

COCO dataset

The Common Objects in Context (COCO) dataset is one of the most widely used and comprehensive datasets for object detection, segmentation, and image captioning tasks. Microsoft created it to advance research in computer vision and provide a rich set of images with detailed annotations. The dataset contains over 330,000 images with more than 80 object categories, covering a wide range of real-world objects commonly found in daily life, such as people, animals, vehicles, furniture, and household items.

The dataset is split into training, validation, and test sets, with over 200,000 labeled images in the training set alone. COCO has high-quality annotations and challenges, realistic scenarios, making it a preferable dataset for training models that should perform well in practical, real-world settings. For object detection, the COCO dataset has bounding box annotations for each object in the image, providing models to both classify and locate objects within the image. This makes it particularly useful for projects like the VisionTool, where real-time object detection is essential for identifying and locating day-to-day life objects to aid navigation and provide assistance. The wide range of categories and diverse scenes in COCO help models trained on it to generalize well to new environments and various object types, making it a crucial resource for developing reliable vision-based systems.

AWS Rekognition

Amazon Rekognition is an image and video analysis service offered by Amazon Web Services (AWS). It uses deep learning techniques and allows developers to build applications that can analyze visual content and accurately detect and recognize objects, people, text, scenes, and activities. This service makes it much easier to integrate advanced computer vision features into applications, enabling real-time visual analysis without the need to build complex models from scratch.

In the VisionTool project, Amazon Rekognition has been used to enhance the accuracy of object detection and classification. By using Rekognition's pre-trained machine learning models, the system can efficiently process the visual data captured by the Raspberry Pi's camera, apply object detection, and send the detected objects to the Raspberry Pi. Using Amazon Rekognition enables VisionTool to handle complex vision-based tasks with greater precision and reliability when a stable WiFi connection is available. In this case, VisionTool utilizes the local object detection model of SSD MobileNetV2. This is particularly important for assistive technology applications, where accurately identifying objects and understanding spatial relationships are critical for providing meaningful feedback and support to blind or visually impaired users.

Model Selection

The SSDLite MobileNet v2 model is a lightweight and highly efficient version of the popular MobileNet v2 architecture, specifically made for real-time object detection on devices with limited computational resources, like the Raspberry Pi used in the VisionTool project. One of the main features of SSDLite MobileNet v2 is its small model size and low computational complexity. Traditional deep learning models are often too large and resource-hungry for real-time use on embedded devices. In contrast, SSDLite MobileNet v2 has been optimized to operate smoothly even with limited power and memory, allowing VisionTool to perform object detection in real time without relying on a powerful server or desktop.

When compared to other models like the standard MobileNet v2 or larger deep learning models such as Faster R-CNN or YOLO, SSDLite MobileNet v2 offers a much better balance of speed, size, and accuracy for edge devices like the Raspberry Pi. While larger models might give slightly higher accuracy, their heavy computational demands make them less practical for real-time applications on resource-constrained platforms. SSDLite MobileNet v2, provides a practical solution, delivering a smaller, faster model that still meets the needs of the VisionTool's object detection.

Table 1 presents a comparison of different models based on detection time and efficiency. Detection time refers to the duration it takes for a model to identify an object; thus, shorter detection times are preferred for lower latency. Additionally, higher efficiency is essential for optimal performance. We evaluated the models listed in the table, as both detection time and efficiency are important for the proposed system. While SSD MobileNet and CenterNet with MobileNetV2 were among the more favorable options, we finally chose SSD MobileNet due to its greater compatibility with Raspberry Pi, and CenterNet was found to be less compatible with Raspberry Pi because of the Detection Time.

Table 1. Comparison of different models

Model name	Detection time	Efficiency
SSD MobileNet	19	20
EfficientDet	95	45
CentreNet with ResNet50	30	29
CenterNet with Hourglass	211	64
CenterNet with MobileNetV2	6	23
Faster R-CNN with ResNet50s	65	31
Faster R-CNN with InceptionResNetV2	206	37

Among various object detection models, SSDlite MobileNet V2 was selected for its lightweight architecture, optimized performance on edge devices, and compatibility with TensorFlow Lite. While more accurate models like YOLOv5 or Faster R-CNN were considered, they demand high computational power and memory, making them impractical for real-time execution on Raspberry Pi. CenterNet models were tested but showed lower compatibility and higher inference latency. SSDlite MobileNet V2 strikes the right trade-off between detection speed, model size, and accuracy for the intended application.

Object Detection Pipeline

The device's core functionality is powered by Google's SSDlite MobileNet V2 model, which processes video frames captured by the Raspberry Pi Camera. This lightweight deep learning model identifies objects in the user's environment and labels them with their respective names. To ensure real-time performance on the Raspberry Pi, TensorFlow Lite is used to optimize the model, reducing inference latency while maintaining accuracy. The model operates continuously, detecting and recognizing objects even in dynamic or cluttered environments.

Input pre-processing

The detection pipeline begins with pre-processing the input image. Each image is resized to a fixed dimension of 300×300. Pixel values are normalized to maintain consistency in the input distribution. This standardized image is then passed through the detection architecture.

Feature extraction using a lightweight convolutional network

The feature extraction backbone used is a lightweight convolutional network designed for efficient computation. It utilizes depthwise separable convolutions, which divide the standard convolution into two operations: a depthwise convolution followed by a pointwise convolution. This greatly reduces computational complexity while preserving spatial information. The backbone produces a series of multi-scale feature maps, with varying resolutions, which are crucial for detecting objects of different sizes. These feature maps serve as the basis for bounding box and class predictions.

Default boxes and multi-scale predictions

At each cell location within the feature maps, the model defines a set of default boxes (also known as anchor boxes) with various aspect ratios and scales. These default boxes act as reference points for predicting the final bounding boxes. For each default box, the model predicts the class confidence scores across all predefined object classes. The offsets needed to adjust the default box to better match the actual object in the image.

Bounding box regression

Bounding box regression is performed by predicting offsets between the default boxes and the actual object locations. These offsets are defined as:

$$t_x = \frac{g_x - d_x}{d_w}, \quad t_y = \frac{g_y - d_y}{d_h}$$

$$t_w = \log\left(\frac{g_w}{d_w}\right), \quad t_h = \log\left(\frac{g_h}{d_h}\right)$$

where $\mathrm{g_x}, \mathrm{g_y}, \mathrm{g_w}, \mathrm{g_h}$ are the ground truth box center coordinates and dimensions, and d_x, d_y, d_w, d_h are the default box parameters.

During inference, the predicted offsets are applied to decode the final bounding boxes:

$$l_x = d_x + t_x \cdot d_w, \quad l_y = d_y + t_y \cdot d_h$$

$$l_w = d_w \cdot \exp(t_w), \quad l_h = d_h \cdot \exp(t_h)$$

Confidence score computation

For each default box, the network simultaneously predicts a set of class confidence scores, one for each object class, including the background. The class with the highest score is selected as the predicted class for that box. The classification loss is computed using softmax cross-entropy, applied only to the matched positive and selected negative default boxes:

$$L_{\text{conf}} = -\sum_{i \in \text{Pos}} \log\left(\widehat{c_i^p}\right) - \sum_{i \in \text{Neg}} \log\left(\widehat{c_i^0}\right)$$

where $\widehat{c_i^p}$ is the predicted confidence score for the true class p, and $\widehat{c_i^0}$ is the predicted confidence score for the background class.

Localization loss and total objective function

To refine the bounding box predictions, the model uses a Smooth L1 Loss between the predicted and actual box offsets:

$$L_{\text{loc}} = \sum_{i \in \text{Pos}} \sum_{m \in \{x,y,w,h\}} \text{Smooth}_{L_1}(l_i^m - g_i^m)$$

with:

$$Smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

The total loss is a weighted sum of the classification and localization losses:

$$L = \frac{1}{N}(L_{\text{conf}} + \alpha L_{\text{loc}})$$

where N is the number of matched (positive) default boxes and alpha is a balancing weight.

Matching strategy and hard negative mining

Each ground-truth object is matched to the default box with the highest Intersection over Union (IoU) score:

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where A is the predicted bounding box and B is the ground truth box.

Boxes with IoU above a certain threshold (typically 0.5) are considered positive matches, while others are negative. To prevent class imbalance, a technique called hard negative mining selects the most challenging negative examples for training based on their confidence scores.

Post-processing with non-maximum suppression

After predictions are made, multiple overlapping bounding boxes may correspond to the same object. Non-Maximum Suppression (NMS) is applied to remove redundant boxes and keep only the ones with the highest confidence scores. NMS ensures that only one bounding box is output per object.

Distance Measurement

In parallel with object detection, ultrasonic sensors measure the distance to the detected objects. The device synchronizes this distance data with the object detection results, providing precise spatial

awareness. For example, if the camera detects a "table," the corresponding distance from the ultrasonic sensor is calculated and associated with the detection. This helps the device send meaningful spatial information to the user.

To ensure correct distance measurements and reduce the noise in the sensor data, we have implemented the Kalman Filter. The Kalman Filter is an algorithm based on recursion, which is used to estimate the true state of a dynamic system based on a series of noisy measurements. It is primarily used in control systems, robotics, and sensor data fusion due to its efficiency and ability to handle uncertainty in both process and measurement data. It has two primary phases: prediction and update. In the prediction step, the filter estimates the current state and error covariance using the previous estimates:

$$\hat{x}_{k \mid k-1} = \hat{x}_{k-1 \mid k-1}$$

$$P_{k|k-1} = P_{k-1|k-1} + Q$$

Here, $\hat{x}_{k|k-1}$ is the predicted state (distance), $P_{k|k-1}$is the predicted error covariance, and Q is the process noise covariance representing uncertainty in the model. In the update step, the predicted values are corrected using the actual sensor measurement:

$$K_k = \frac{P_{k|k-1}}{P_{k|k-1} + R}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - \hat{x}_{k|k-1})$$

$$P_{k|k} = (1 - K_k)P_{k|k-1}$$

where K_k is the Kalman Gain, z_k is the measured distance from the ultrasonic sensor, $\hat{x}_{k|k}$ is the updated state estimate, $P_{k|k}$ k is the updated error covariance, and R is the measurement noise covariance.

By recursively applying the prediction and update steps, the Kalman Filter yields stable and accurate distance readings, even in environments with high measurement noise.

Audio Feedback

Audio feedback is a critical feature for blind users, enabling them to understand their surroundings through speech. A text-to-speech (TTS) engine is employed to convert the detected objects and their distances into clear audio messages. These messages, such as "Chair, 1.5 meters ahead," are transmitted to the user via Bluetooth earphones. This communication helps users to safely and confidently navigate their environment.

To enhance safety and enable remote monitoring, the device uses Firebase Realtime Database for location tracking. The GPS module on the Raspberry Pi continuously captures the user's latitude and longitude, then uploads this data to Firebase. A Flutter-based mobile application retrieves this information and displays the user's live location on a map. This feature allows caregivers or family members to track the user in real time, offering reassurance and an added layer of safety.

Firebase Integration for GPS Tracking

Figure 4 shows an image of the realtime database, which stores updated latitude and longitude values from the Raspberry Pi. The GPS module on the Raspberry Pi continuously locates the user's latitude and longitude, and provides this to the Raspberry Pi, which in turn sends these details to the Firebase Realtime Database. A Flutter-based mobile application retrieves this information and displays the user's live location on a map. This feature allows caregivers or family members to track the user in real time, providing reassurance and an added layer of safety. Also, when the emergency button is pressed, Raspberry Pi sends a trigger to the Firebase, which triggers the Flutter app and sends an email to the email addresses provided in the profile section of the Flutter app.

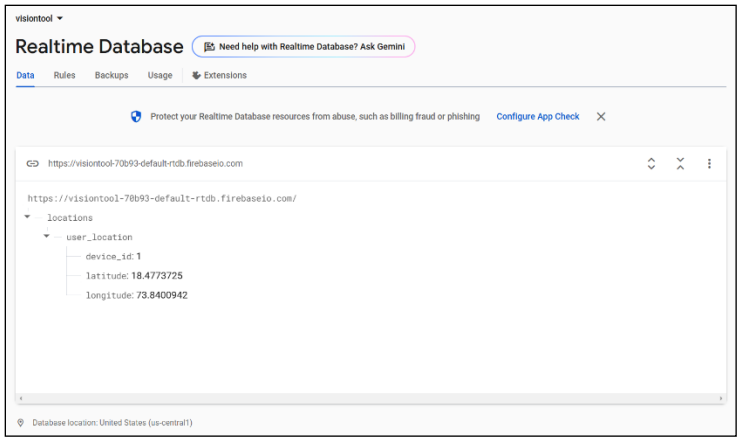


Figure 4. Firebase realtime database

Emergency Button Feature

The VisionTool device features an emergency button as a safety measure, providing users with a quick and reliable way to notify caregivers when help is needed. When the emergency button is pressed, the Raspberry Pi immediately sends an alert to a Firebase database, along with the user's live location. This triggers a notification on the Flutter application, ensuring that caregivers or family members are instantly informed of the user's emergency and location. This solution improves usability, particularly for blind users, by providing a simple and intuitive method to request assistance without relying on visual input.

Flutter App

Features

The Flutter application is a monitoring and safety app for the device user's caregivers. It is integrated with Firebase to provide real-time location updates. It retrieves latitude and longitude data from Firebase, sent by the Raspberry Pi, and displays the user's live location on Maps. For displaying maps, we have used the OpenStreetMap API. This feature allows caregivers to track the user in real time, ensuring their safety and offering navigation assistance when needed.

The app also includes a user profile section where caregivers can store important details, such as the name, phone number, and email address of the concerned individuals, to provide emergency updates via email. This information is essential for personalizing the app and facilitating quick communication during emergencies. The app includes an SOS button for emergency situations. When pressed, an email is sent to the registered relative's email ID, containing an emergency message and a Google Maps link to the user's live location. This feature provides an effective way to alert and guide the recipient to assist the user promptly.

Figure 5 displays the image of the Flutter app, which shows the live location of the device. As the image was captured while the device was located on the VIT campus, the app reflects this location. Additionally, the image features the SOS alert button.

Platform justification

Flutter was chosen for its cross-platform development capability, enabling rapid deployment on both Android and iOS with a single codebase. Compared to native development approaches, Flutter significantly reduces development effort and ensures consistent UI across platforms. Firebase was selected as the backend due to its real-time database, cloud messaging capabilities, and seamless integration with Flutter, making it ideal for low-latency GPS tracking and emergency alert functionalities.

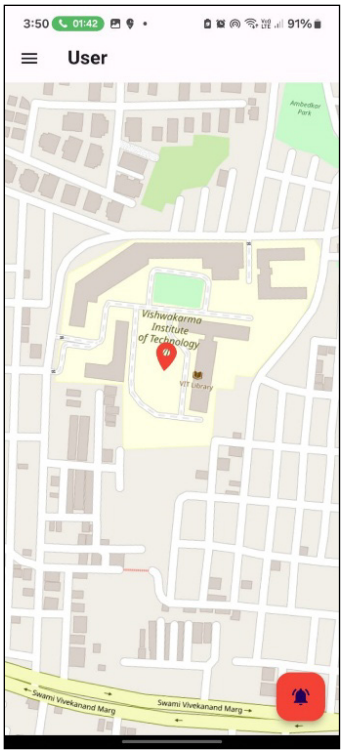


Figure 5. Flutter app

Multiprocessing for Efficiency

To handle multiple tasks concurrently, the device utilizes Python's multiprocessing library. Separate processes manage object detection, distance calculation, GPS data transmission to Firebase, and audio feedback generation. This architecture ensures that each component operates independently without delays or interference. By dividing the workload across processes, the system maintains high responsiveness even during intensive operations.

Working Principle

Figure 6 shows the workflow of the proposed system. The vision assistance device leverages a multiprocessing architecture to efficiently execute multiple real-time tasks simultaneously. This design ensures that critical operations such as object detection, distance measurement, GPS tracking, audio feedback, AWS Rekognition, and the emergency button are handled independently, providing a seamless user experience. The detailed working principle is as follows.

The device uses multiprocessing on a Raspberry Pi to run several tasks simultaneously, ensuring real-time performance. A Pi camera captures video, which is processed by the SSDLite MobileNet V2 model (via TensorFlow Lite) for object detection. Detected objects are labeled and sent to AWS Rekognition for enhanced analysis. In parallel, an ultrasonic sensor measures object distance, synchronized with camera data to provide accurate spatial feedback (e.g., "Chair, 1.5 meters ahead"). A GPS module tracks the user's live location and updates it to Firebase, which a Flutter app uses to display movement in real time for caregivers.

Audio feedback is generated through a text-to-speech engine and delivered via Bluetooth or wired earphones. An emergency button provides quick alerts to caregivers via Firebase. The system is modular and scalable, with processes like object detection, distance sensing, GPS, and audio running simultaneously without compromising performance. This ensures safe and independent navigation for visually impaired users.

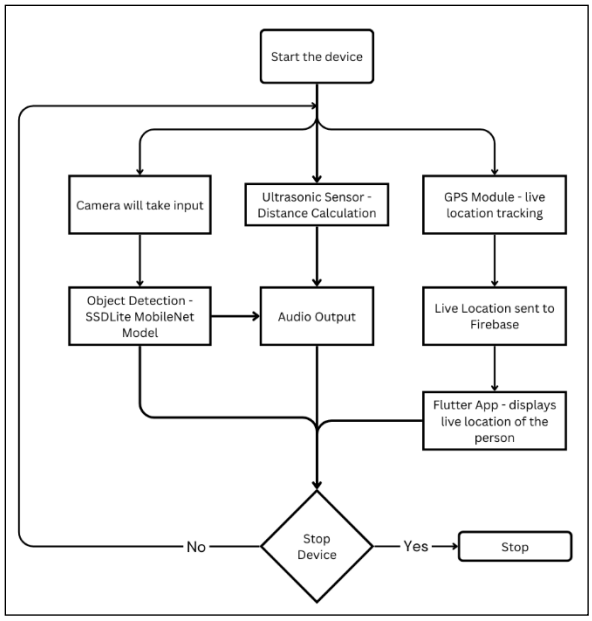


Figure 6. System workflow

RESULTS AND DISCUSSION

The VisionTool project successfully uses several advanced technologies to provide real-time navigation assistance. By utilizing the SSDLite MobileNet v2 model, the system accurately detects objects in the user's environment with low latency. The model's efficiency makes it better for real-time applications, ensuring users receive immediate feedback about nearby objects. The ultrasonic sensor measures the distance to obstacles, enabling the system to calculate the user's distance to them. This feature is crucial for safe navigation, as it provides timely alerts about obstacles.

The evaluation of the MobileNet Single Shot Detector (SSD) model on the Common Objects in Context (COCO) dataset, when deployed on a Raspberry Pi 4B (RPI 4B), demonstrates its performance in terms of accuracy and efficiency. The relevant metrics are presented in Table 2.

Table 2. Result metrics of vision tool device with SSD MobileNet V2

Metric	Value
Framework	TensorFLow Lite
Model Size	20MB
Mean Average Precision (mAP) @ IoU = 0.5	~20-25%
mAP @ IoU = 0.5:0.95	~15–20%
Inference Speed (CPU only)	2–5 FPS
Inference Speed (with TFLite Optimization)	9 FPS
Input Resolution	300 × 300 pixels
Average Latency per Frame (CPU)	200ms
Power Consumption	5W

The inference speed of the Raspberry Pi 4B achieves a processing rate of 2-5 frames per second (FPS) when running the model directly on the Central Processing Unit (CPU). However, more improvements can be made by integrating external hardware accelerators, such as the Coral USB Accelerator, which can boost the inference rate to approximately 30 FPS.

The compact size of the MobileNet SSD model, ranging between 20 Megabytes (MB), makes it well-suited for edge computing scenarios. While the inference speed on the Raspberry Pi 4B is lower than on high-end Graphics Processing Units (GPUs), the system's low power consumption and portability make it an attractive choice for real-time object detection in embedded applications. Table 3 highlights the error reduction and improved stability that the Kalman filter brings to obstacle detection. The ultrasonic sensors have much noise, so the error also varies from 1-3%.

Table 3. Readings reduction of ultrasonic sensor using Kalman filter

Actual distance (cm)	Raw sensor reading (cm)	Kalman filtered reading (cm)	Raw error (%)	Filtered error (%)
50	48.9	49.8	2.2	0.4
100	98.7	99.5	1.3	0.5
150	148.5	149.6	0.1	0.3

The average latency, including the image capture, inference, and text-to-speech synthesis, involved approximately 700-900 milliseconds. Figure 7 depicts the VisionTool device connected to a speaker. Any type of speaker can be used, whether large or small, and it can be either Bluetooth or wired. A mobile phone is placed in front of the device, and it is detecting the phone. Although the model achieves a lower mAP compared to high-end models, its performance is suitable for embedded real-time applications where low latency and power efficiency are prioritized over raw detection accuracy.

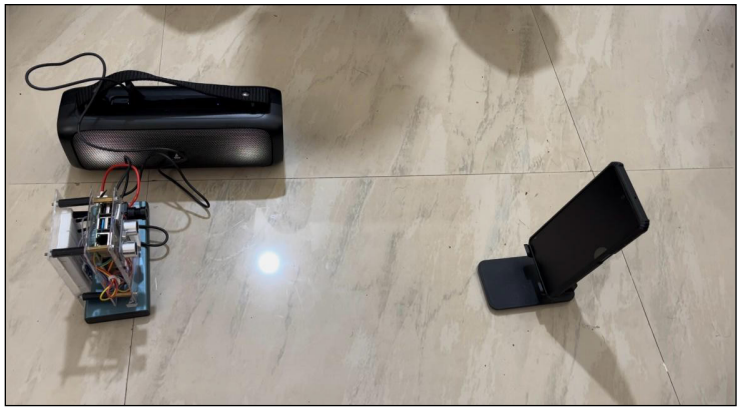


Figure 7. VisionTool device detecting a mobile phone

MARKET POSITION AND COMMERCIAL LANDSCAPE

Several commercial products currently serve the visually impaired community, such as OrCam MyEye, Envision Glasses, Microsoft Seeing AI, Aira, and Orion Glasses. These devices offer features including facial recognition, scene description, text and currency reading, and remote agent support. However, most of them lack integrated real-time obstacle detection, offline functionality, or emergency alert mechanisms.

Table 4 presents a comparative analysis of VisionTool and these commercial alternatives. VisionTool distinguishes itself by offering a unique combination of real-time obstacle detection, offline and online operation, GPS-based live tracking, and an emergency alert system – all within a portable and affordable embedded system. These features collectively enable safe and independent navigation, particularly in resource-limited or low-connectivity environments. "Yes" indicates full support for the feature as of product specifications published in 2024. "No" indicates absence or non-support of the feature.

Table 4. Feature comparison between VisionTool and commercial assistive devices

Feature	Visiontool	OrCam MyEye	Envision glasses	Orion glasses	Microsoft seeing AI	Aira
Real-time obstacle detection	Yes	No	No	Yes	No	No
Live location tracking	Yes	No	No	No	No	Yes
Voice feedback	Yes	Yes	Yes	Yes	Yes	Yes
Offline and online modes	Yes	No	No	No	No	No
Adaptive AI guidance	No	No	No	No	Yes	Yes
Emergency alert feature	Yes	No	No	Yes	Yes	Yes
Facial recognition	No	Yes	Yes	No	Yes	No
Scene description	No	Yes	Yes	No	Yes	Yes
Wearable form factor	No	Yes	Yes	Yes	No	Yes

Currently, VisionTool does not support advanced features such as facial recognition, currency and text reading, scene-level AI description, or wearable form factors – capabilities found in higher-end commercial systems. However, these features are identified as part of the future development roadmap. Subsequent versions of VisionTool aim to incorporate OCR-based text recognition, facial identification for known individuals, scene summarization using advanced computer vision models, and a more compact wearable design to further enhance usability and user comfort.

Despite its current limitations, VisionTool effectively addresses a critical and underserved niche by focusing on essential mobility and safety features with an emphasis on cost-effectiveness and real-time performance. This positions it as a strong, scalable, and accessible alternative in the broader assistive technology ecosystem.

CONCLUSION

VisionTool is an innovative project designed to help people – especially those with visual impairments or mobility challenges – navigate their surroundings more safely and independently. At the heart of the system is a Raspberry Pi 4B, which runs an SSD MobileNet v2 model trained on the COCO dataset to detect objects in real time. Along with object detection, an ultrasonic sensor measures how far obstacles are from the user, giving them critical information to avoid collisions.

To make sure users receive this information in a way that's easy to understand, VisionTool uses Bluetooth earphones to deliver audio updates. The system speaks out, detects objects and their distances, offering real-time guidance without requiring the user to look at a screen – perfect for those who rely heavily on sound cues.

VisionTool also tracks the user's location with an integrated GPS module. This data is sent to Firebase, keeping everything updated and synced with a Flutter mobile app. Family members or caregivers can monitor the user's location anytime through the app, adding an extra layer of security and peace of mind. Thanks to Firebase, all the data – from GPS to object detection – is stored and updated in real time across devices. By combining smart sensing, real-time communication, and accessibility features, VisionTool offers a complete solution for safer, more confident mobility. Looking

ahead, future upgrades could include better sensor accuracy, more features, and an even more user-friendly interface to make it even more practical in everyday life.

Future iterations of VisionTool will explore the integration of facial recognition, scene description, and wearable form factors to expand its functionality and bring it closer to advanced commercial solutions.

REFERENCES

A, M., MS, S., A, M., S, L. R., S, M., & Rana, N. (2024). Implementation of finding the obstacle for blind people using Arduino and ultrasonic sensor. *International Journal of Creative Research Thoughts*, *12*(2), 554-558. https://www.ijcrt.org/papers/IJCRT2402655.pdf

Atitallah, A. B., Said, Y., Atitallah, M. A. B., Albekairi, M., Kaaniche, K., & Boubaker, S. (2024). An effective obstacle detection system using deep learning advantages to aid blind and visually impaired navigation. *Ain Shams Engineering Journal*, *15*(2), 102387. https://doi.org/10.1016/j.asej.2023.102387

Buchs, G., Simon, N., Maidenbaum, S., & Amedi, A. (2017). Waist-up protection for blind individuals using the EyeCane as a primary and secondary mobility aid. *Restorative Neurology and Neuroscience*, *35*(2), 225-235. https://doi.org/10.3233/RNN-160686

Chavan, S. M., Kathane, A. R., Makhare, P. R., & Jadhav, P. K. (2024). Smart belt obstacle detection with voice output for blind person. *International Journal of Advanced Research in Science, Computing and Technology*, *4*(1), 469-475.

David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., & Warden, P. (2021). TensorFlow Lite Micro: Embedded machine learning on TinyML systems. *Proceedings of the 4th MLSys Conference, San Jose, CA, USA.* https://doi:10.48550/arXiv.2010.08678

Islam, R. B., Akhter, S., Iqbal, F., Rahman, M. S., & Khan, R. (2023). Deep learning based object detection and surrounding environment description for visually impaired people. *Heliyon*, 9(6), e16924. https://doi.org/10.1016/j.heliyon.2023.e16924

Khan, A., Khan, A., & Waleed, M. (2018, November). Wearable navigation assistance system for the blind and visually impaired. *Proceedings of the International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, Sakhier, Bahrain,* 1-6. https://doi.org/10.1109/3ICT.2018.8855778

Khandewale, A. H., Gohokar, V. V., & Nawandar, P. (2020, June). Real time object detection for visually impaired person using TensorFlow Lite. *Proceedings of the 10th International Workshop on Computer Science and Engineering, Shanghai, China,* 150-156. https://doi.org/10.18178/wcse.2020.06.025

Layaraja, T., Sampath, P., Aishwarya, P., & Suraksha, V. (2024). Obstacle detection for blind people. International Journal for Advanced Research in Science and Technology, *14*(2), 154-158.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740-755). Springer. https://doi.org/10.1007/978-3-319-10602-1_48

Maksimović, M., Vujović, V., Davidović, N., Milošević, V., & Perišić, B. (2014, June). Raspberry Pi as Internet of Things hardware: Performances and constraints. *Proceedings of the 1st International Conference on Electrical, Electronic and Computing Engineering, Vrnjačka Banja, Serbia*, 1-6. https://doi.org/10.1109/MIPRO.2014.6859717

Mohajeri, N., Raste, R., & Daneshvar, S. (2011, July). An obstacle detection system for blind people. *Proceedings of the World Congress on Engineering, London, UK.*

Mugwenhi, M., & Mudawarima, T. (2023). *Blind assistance system: Obstacle detection with distance and voice alert.* Technical Report. https://www.researchgate.net/publication/374229722_BLIND_ASSISTANCE_SYSTEM_OBSTACLE_DETECTION_WITH_DISTANCE_AND_VOICE_ALERTS

Ou, S., Park, H., & Lee, J. (2020). Implementation of an obstacle recognition system for the blind. *Applied Sciences*, *10*(1), 282. https://doi.org/10.3390/app10010282

R, S. M., Khan, F., R, G. G., & S, H. (2019, July). Object detection and human identification using Raspberry Pi. *Proceedings of the 1st International Conference on Advances in Information Technology, Chikmagalur, India*, 135-139. https://doi.org/10.1109/ICAIT47043.2019.8987398

S, A., Rani, D. M., D, D., & S, V. (2016). Smart stick for blind using Raspberry Pi. *International Journal of Engineering Research & Technology*, *4*(22), 1-3. https://www.ijert.org/research/smart-stick-for-blind-using-raspberry-pi-IJERTCONV4IS22026.pdf

Said, Y., Atri, M., Albahar, M. A., Ben Atitallah, A., & Alsariera, Y. A. (2023). Obstacle detection system for navigation assistance of visually impaired people based on deep learning techniques. *Sensors*, *23*(11), 5262. https://doi.org/10.3390/s23115262

Sanjay, N. S., & Ahmadinia, A. (2019). MobileNet-Tiny: A deep neural network-based real-time object detection for Raspberry Pi. *Proceedings of the 18th IEEE International Conference on Machine Learning and Applications, Boca Raton, FL, USA*, 647-652. https://doi.org/10.1109/ICMLA.2019.00118

Shahira, K. C., Tripathy, S., & Lijiya, A. (2019, October). Obstacle detection, depth estimation and warning system for visually impaired people. *Proceedings of the IEEE Region 10 Conference, Kochi, India*, 863-868. https://doi.org/10.1109/TENCON.2019.8929334

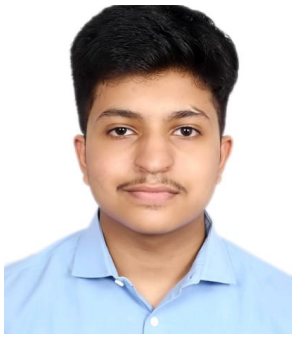
Texeira, C. H. M., Rodrigues, A. A., de Azevedo Costa, A. L. F., & dos Santos, V. R. (2023). Wearable haptic device as mobility aid for blind people: Electronic cane – Wearable device for mobility of blind people. *JOJ Ophthalmology*, *9*(3), 555765. https://doi.org/10.19080/JOJO.2023.09.555765

AUTHORS

Dr Rakhi Joshi Bhardwaj is an Assistant Professor in the Department of Computer Engineering at Vishwakarma Institute of Technology (VIT), Pune. She holds a PhD in Computer Science and Engineering from Koneru Lakshmaiah Education Foundation, Vijayawada, Andhra Pradesh. Her research expertise lies in artificial intelligence, machine learning, data analytics, and multimedia processing. Dr Bhardwaj has authored numerous publications in reputed international journals and conferences and serves as a reviewer for prestigious platforms such as IEEE and Springer. She is actively involved in mentoring students, fostering interdisciplinary research, and contributing to institutional growth.

Bhushan Bachewar is a final-year Bachelor of Technology student in Computer Engineering at Vishwakarma Institute of Technology, Pune. His core interests include artificial intelligence, web development, operating systems, database management, and software design. In addition to his software expertise, Bhushan has significant experience with embedded systems and hardware integration, having developed projects that combine intelligent algorithms with platforms such as Raspberry Pi, Arduino, and ESP32. He is passionate about building real-world solutions that bridge the gap between smart software and responsive hardware, with a focus on impactful, interdisciplinary innovation.

Shreya Barsude is a final-year Bachelor of Technology (BTech) student in Computer Engineering at Vishwakarma Institute of Technology, Pune. Her primary interests include artificial intelligence, generative AI, and web development. She is also passionate about microservices architecture and embedded systems. Shreya actively explores the intersection of intelligent software and hardware, aiming to build scalable, real-world solutions through a combination of modern AI techniques and robust system design.



Harsh Badagandi is currently a student with a Bachelor's degree in the Computer Engineering Department at Vishwakarma Institute of Technology, Pune, where he has been studying since 2022.

Snehal Darade is a third-year Bachelor of Technology student in the Computer Engineering Department at Vishwakarma Institute of Technology, Pune. Her primary areas of interest include Java full-stack development, data science, and software engineering. She is skilled in Java, operating systems, database management, and problem-solving, with a strong grasp of both frontend and backend development. Snehal has contributed to several impactful projects such as financial literacy platforms, responsive web applications, and hospital bed occupancy prediction systems. She is passionate about building user-centric, scalable software solutions that address real-world challenges.