Trivex: An Intelligent AI-Powered Camera System for Real-Time Object Detection and Navigation with Audio Feedback for the Blind People

Samartha Ashok Koti [1], Suraj Shantaraj Vibhuti[2], Sagar Shivappa Kushalapur [3], Vivek Somesh Pattanakodi[4], Prof. Girish Shivanagudi[5]

[1234] Students, B.E. Computer Science and Engineering , KLS Vishwanathrao Deshpande Institute of Technology, Haliyal

[5] Guide, Department of Computer Science and Engineering, KLS Vishwanathrao Deshpande Institute of Technology, Haliyal

Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, India

[1] 2vd23cs058@klsvdit.edu.in | [2] 2vd23cs078@klsvdit.edu.in

[3] 2vd23cs056@klsvdit.edu.in | [4] 2vd23cs093@klsvdit.edu.in

[5] gns@klsvdit.edu.in

***Abstract*—Blind and visually impaired people face persistent difficulty in identifying nearby objects, reading printed text, determining their location, and summoning help. Existing assistive applications usually address only one of these needs. This paper presents TRIVEX, a native Android application that integrates real-time object detection, scene description, text reading, location assistance, and emergency support behind a voice-first interface. Camera frames are analyzed on the device by a YOLO26n model executed through ONNX Runtime, printed text is recognized by Tesseract, and spoken commands are decoded by Vosk. Results are delivered through Android Text-to-Speech in English, Hindi, Kannada, Marathi, or Telugu. An optional cloud model supplies contextual scene descriptions when connectivity is available, and an SOS module calls a trusted contact and sends the user's location by SMS. Functional testing on a smartphone confirmed the reading, location, emergency, and voice-command workflows. Detection accuracy, response latency, and usability were not measured quantitatively; an evaluation protocol for these properties is outlined.**

Keywords—assistive technology, visual impairment, object detection, YOLO, optical character recognition, speech recognition, text-to-speech

I. Introduction

Visual impairment restricts access to information that sighted people obtain at a glance: which objects are nearby, what a notice or signboard says, where one is standing, and how to reach help. The World Health Organization reports that at least 2.2 billion people globally have a near or distance vision impairment [1]. Blind and visually impaired individuals often depend on other people, a white cane, or a guide dog for activities such as walking through an unfamiliar place or reading a printed page.

Smartphones now combine a camera, microphone, speaker, satellite positioning, and enough computing power to run compact neural networks. They are also devices that many users already own and carry. These properties make the smartphone a practical platform for assistive software that requires no additional hardware, provided that the software can be operated without sight.

Most assistive prototypes reported in the literature concentrate on a single function, typically object or obstacle detection with spoken output. A user who also needs to read text, confirm a location, or raise an alarm must switch between several applications, each with its own interface and often its own connectivity requirement. TRIVEX addresses this fragmentation by placing object detection, scene description, text reading, location assistance, and emergency support in one Android application that is controlled by voice and reports its results as speech.

This paper makes three contributions:

- An architecture in which object detection, text recognition, and voice-command recognition execute on the device, with cloud processing confined to an optional scene-description service;
- a voice-first interaction design that pairs spoken commands and a single large touch target with multilingual speech output in five languages; and
- an account of the implemented modules, their local storage, and their observed behavior, together with the limitations and measurements that remain before performance claims can be made.

The work is an implementation account of a functional prototype rather than a claim of experimentally demonstrated

superiority over existing systems. The remainder of this paper is organized as follows. Section II reviews related work, Section III states the requirements and design goals, Section IV describes the system architecture, Section V presents the assistive processing pipelines, Section VI covers location and emergency support, Section VII discusses local storage and privacy, Section VIII presents the evaluation and its limitations, and Section IX concludes.

II. Related Work

A. Real-Time Object Detection

Single-stage detectors made real-time object recognition practical on modest hardware. Redmon et al. introduced YOLO, which predicts bounding boxes and class probabilities for the whole image in one network evaluation [2]. Because the image is processed once, the method is fast enough for live video, although small objects and crowded scenes are harder for it. Later versions such as YOLOv4 improved the balance between speed and accuracy [3].

The YOLO26 family is designed to produce final predictions without a separate non-maximum suppression step and omits distribution focal loss, which simplifies export to runtimes such as ONNX. Its nano variant targets low-power edge devices [4]. These properties motivate the use of a nano-scale YOLO model for on-device detection in TRIVEX, where the processor, memory, and battery of a handset are the limiting resources.

B. Assistive Systems for Visually Impaired Users

Several assistive systems build on these detectors. Rachburee and Punlumjeak proposed an obstacle detection model based on YOLOv3 for visually impaired people [5]. Anbananthen et al. developed The Eye, a lightweight mobile application that uses an improved YOLOv5l algorithm to recognize objects for visually challenged users [6]. Bamane et al. combined YOLO with OpenCV for real-time object detection and direction prediction [7], and Joshi et al. presented multi-object detection with smart navigation support [8]. Aralikatti et al. added face recognition to real-time object detection [9], while Jawaid et al. coupled an object detector with a narrator [10].

Text access is a separate line of work. Tesseract is a widely used open-source optical character recognition (OCR) engine for printed text [11], and assistive tools commonly pair such an engine with speech synthesis to read documents aloud. Recognition quality in this setting depends strongly on lighting, focus, and framing, which a blind user cannot verify visually. speech recognition has likewise become feasible on handsets through compact models built on toolkits such as Kaldi [12].

C. Comparative Summary

Table I summarizes the principal function of each cited system as stated in its publication and places TRIVEX beside them. The comparison is one of scope only. No system was re-implemented or benchmarked for this work, so the table does not rank the systems by accuracy or speed.

I. Principal Scope of Related Assistive Systems

System	Principal reported scope
Rachburee and Punlumjeak [5]	Obstacle detection with YOLOv3
The Eye [6]	Mobile object recognition with improved YOLOv5l
Bamane et al. [7]	Object detection and direction prediction
Joshi et al. [8]	Multi-object detection and smart navigation
Aralikatti et al. [9]	Object detection and face recognition
Jawaid et al. [10]	Object detection with narration
TRIVEX (this work)	Detection, scene description, text reading, voice control, location, and SOS

The systems above demonstrate that camera-based detection with audio feedback is feasible, but each concentrates on detection, navigation, or reading in isolation, and several assume particular hardware or continuous connectivity. TRIVEX does not propose a new detection or recognition algorithm. Its contribution is the integration of detection, scene description, reading, voice control, location assistance, and emergency signalling in one application whose core functions remain available without a network connection.

III. Requirements and Design Goals

A. Design Objectives

The project set six objectives: to assist visually impaired individuals in daily activities through an AI-powered application; to identify nearby objects in real time with the mobile camera; to describe the surrounding scene through short audio messages; to read printed text from books, documents, and signboards; to provide voice commands for hands-free control; and to convert all detected information into speech. Location assistance and emergency support were added to the scope so that the application also covers orientation and safety.

B. Functional Requirements

Table II maps the functional requirements to their realization in the application. Each requirement is stated from the point of view of a user who cannot see the screen, which is why every function has both a spoken trigger or a large touch control and a spoken result.

II. Functional Requirements and Their Realization

Requirement	Realization in TRIVEX
Accessible interface	One large touch area, four buttons, and voice prompts
Object detection	YOLO26n applied to live camera frames
Scene understanding	Short spoken description from an optional online service
Voice command control	phrases such as “open camera” and “sos call”
Text reading	Tesseract OCR in live and snapshot modes
Audio feedback	Text-to-Speech for objects, text, and location
Language selection	English, Hindi, Kannada, Marathi, or Telugu
Voice speed control	Adjustable speaking rate
Current location	Position announced as a readable address
Home location	Saved once and recalled on request
Emergency SOS	Trusted contact, emergency call, and location SMS

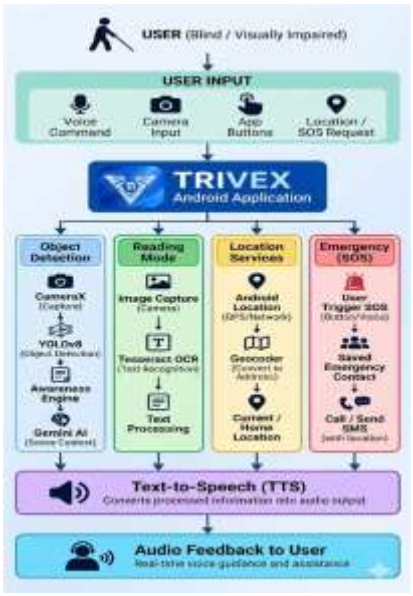
C. Non-Functional Goals

The non-functional goals concern how the application behaves. Performance requires that detections be announced with minimal delay. Availability requires that the main functions work without continuous internet access. Usability and accessibility require operation through audio and voice without dependence on visual information. Portability and cost effectiveness require that the application run on supported Android smartphones using only the built-in camera, microphone, speaker, and positioning receiver.

Further goals are reliability, security of the saved contact and location, multilingual output, battery efficiency, and a structure that allows features and languages to be added. Several of these goals shaped the architecture directly: availability and performance led to on-device processing, and cost effectiveness ruled out external sensors. Others, notably performance, reliability, and battery efficiency, are stated as targets whose attainment has not yet been measured, as discussed in Section VIII.

IV. System Architecture

TRIVEX is a native Android application written in Java 17 and built with Gradle in Android Studio. It follows a client-heavy model: the computationally demanding stages run on the smartphone so that the response does not depend on network round trips. Fig. 1 shows the workflow from user input to audio feedback.



1. Workflow of the TRIVEX application from user input to audio feedback. The detection stage labelled YOLOv8 in the diagram corresponds to the YOLO26n model of the reported configuration.

A. Input, Processing, and Output Layers

The input layer accepts four kinds of request: spoken commands, live camera frames supplied by CameraX, touch on the application buttons, and location or SOS requests. Each request is routed to one of four processing paths. The object detection path passes camera frames to the YOLO detector, an awareness engine, and, optionally, a cloud model for scene context. The reading path captures an image, applies Tesseract OCR, and cleans the recognized text. The location path queries Android location services and converts coordinates to an address with the Geocoder. The emergency path contacts a saved trusted person by call or SMS.

The output layer is common to all paths. Every result, whether a detected object, a passage of text, an address, or a status message, is converted to speech by the Android Text-to-Speech (TTS) engine. Routing all output through one channel keeps the interaction consistent and removes any need to read the screen.

B. Software Module Design

Fig. 2 shows the five software modules and the data flow between them. The user module represents interactive input and audio feedback. The frontend module contains the screens, layouts, and buttons, which are built with XML layouts, Material Components, and AndroidX libraries and are shown in Fig. 3. The backend module holds the core logic for speech processing, OCR text extraction, and location processing. The external services module groups maps and positioning, AI scene analysis, the object detection model, and SMS and telephony interfaces. The database module stores settings locally.

Table III lists each function with its enabling technology. Only scene description and the download of map tiles require internet access; calls and SMS require cellular service. All other functions operate on the device.

III. Functional Modules and Enabling Technologies

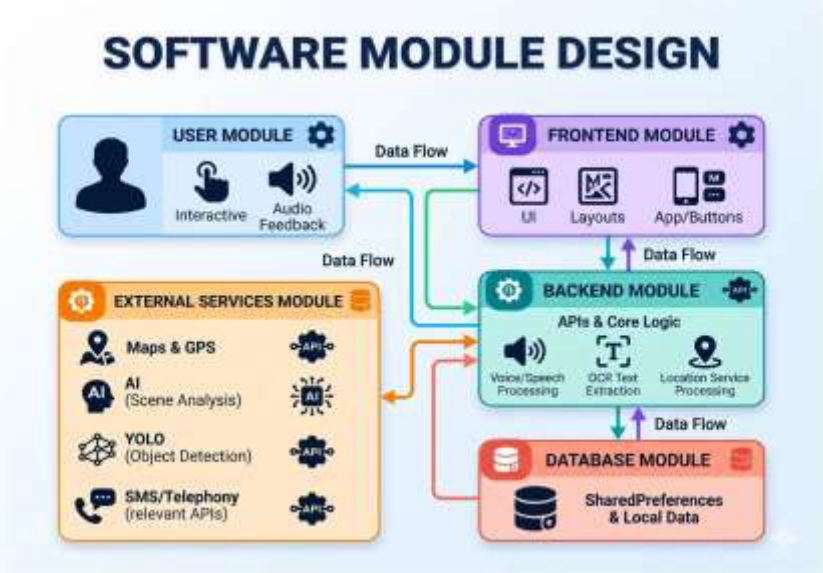
Function	Enabling technology
Object detection	CameraX frames; YOLO26n model on ONNX Runtime
Scene description	Gemini Live API over OkHttp WebSocket (online, optional)
Reading mode	Tesseract OCR on live or captured images
Voice commands	Vosk speech recognition
Audio feedback	Android Text-to-Speech in five languages
Location	LocationManager, Geocoder, OpenStreetMap
Emergency SOS	Android telephony call and SmsManager
Settings storage	Android SharedPreferences

The interface is deliberately minimal. The home screen in Fig. 3(a) is dominated by one touch area that opens the camera on a double tap or on the spoken command “open camera”, with four buttons for Settings, SOS, Location, and Auto On beneath it. High contrast and large controls serve users with residual vision, while voice prompts serve users with none.

C. Implementation Environment

Development used Android Studio with Gradle for dependency management and builds, JUnit for component tests, and Git with GitHub for version control. At run time the application needs an Android smartphone with a rear camera, microphone, speaker, and satellite positioning receiver. A SIM card with network service is needed only for emergency calls and messages, and an internet connection only for scene description and map tiles.

On-device processing moves the cost of the system from the network to the handset. Internal storage must hold the detection model, the OCR language data for each supported language, and the speech recognition model, and the processor and memory must sustain neural inference on live frames. The minimum processor, memory, and storage for acceptable operation have not been established and form part of the evaluation in Section VIII.



2. Software module design of TRIVEX, showing the user, frontend, backend, external services, and database modules and the data flow between them.



3. TRIVEX application screens: (a) home screen, (b) settings with language, voice speed, and voice commands, (c) SOS screen, and (d) reading mode with the camera directed at a handwritten page.

V. Assistive Processing Pipelines

A. Object Detection and Scene Understanding

When the camera is opened, CameraX delivers live frames to the detection module. A YOLO26n model, exported to the ONNX format and executed by ONNX Runtime, identifies everyday objects in each frame and returns their classes and bounding boxes. Because inference runs locally, detection continues without an internet connection. An awareness engine sits between the detector and the speech output and governs which detections are announced; a quantitative characterization of its announcement policy is left to future work.

The announcement policy matters as much as the detector. A camera produces many frames each second, and a literal announcement of every detection in every frame would overwhelm the listener. Speech is also serial: a spoken sentence occupies the audio channel for longer than the scene may take to change. An assistive detector must therefore decide what is worth saying and when, and the application provides a spoken command to mute alerts.

Object labels alone do not convey how a scene is arranged. When connectivity is available, TRIVEX can therefore send camera input to the Gemini Live API, using the Gemini 3.1 Flash Live model in the reported configuration, over an OkHttp WebSocket connection. The model returns a short natural-language description that relates the objects in view to one another. This service is optional, and the application remains usable for detection, reading, and voice control when it is unreachable.

The two mechanisms offer different assurances. The on-device detector is predictable in availability but limited to the object classes on which it was trained. The cloud description is richer but depends on the network and on a generative model whose statements cannot be verified by the user. Such descriptions should consequently be treated as supporting context and not as a basis for safety-critical decisions such as crossing a road.

B. Reading Mode

Reading mode recognizes printed text in books, documents, notices, and signboards. The user can scan with the live camera view or capture a snapshot. The image is passed to Tesseract, and the recognized text is cleaned into structured strings before being spoken. Spoken or on-screen controls start and stop the reading.

A blind user cannot judge whether a page is properly framed, so the application gives spoken guidance during capture. Fig. 3(d) shows one such prompt, which reports that the text is near the edge of the view and asks the user to move back slightly. The reading language is chosen in the settings and determines the language of both recognition and speech.

Fig. 3(d) also illustrates a boundary of the present implementation. The camera in that screen is directed at a handwritten page, and the string shown in the output panel is not meaningful. This is consistent with Tesseract being designed for printed text [11]. Reading mode should therefore be regarded as a printed-text facility; handwriting recognition is listed as future work in Section VIII.

C. Voice Command Interface and Audio Feedback

Spoken commands are recognized on the device by the Vosk toolkit, which builds on Kaldi [12], so the principal controls work without connectivity. Table IV lists the supported phrases by function. Several phrasings are accepted for the same action so that the user need not recall one exact wording.

IV. Supported Voice Commands

Function	Spoken commands
Start detection	“open camera”, “start camera”, “open object detection mode”
Stop detection	“pause camera”, “close camera”, “stop detection”
Open reading	“open reading mode”, “start reading mode”, “reading mode”
Control reading	“start reading”, “stop reading”
Settings and alerts	“open settings”, “mute alerts”
Language	“change the language to Hindi”
Emergency	“sos call”

As Fig. 3(b) shows, voice activation can be enabled or disabled, commands are accepted while the application is visible, and listening pauses while the assistant is speaking so that it does not respond to its own output. The settings screen also lists the available commands as a reference during setup.

Speech output supports English, Hindi, Kannada, Marathi, and Telugu, and the speaking rate is adjustable. Both settings can be changed by voice. Two constraints apply. Command phrases are currently recognized in English only, and speech in a regional language requires the matching voice to be installed in the Android speech settings. A user who speaks no English can thus hear results in a familiar language but must still learn a small set of English commands.

VI. Location and Emergency Support

A. *Location Assistance*

The location module obtains the current position from the Android LocationManager, which draws on satellite positioning and available network-based sources. The Geocoder converts the latitude and longitude into a readable address that includes the locality and PIN code, and the address is announced through TTS. The position is also drawn on an OpenStreetMap view for sighted companions and users with partial vision.

A home location can be saved by tapping the map or by voice from the current position. The stored coordinates allow the application to report the home address on request. The map component is a display aid and not a navigation system: it offers no routing, and without previously downloaded tiles it shows only an coordinate grid. The reported position carries the measured accuracy of the device sensors and should not be presented to the user as exact.

B. *Emergency SOS*

The SOS screen in Fig. 3(c) lets the user save the phone number of one trusted contact, with country code, and provides three large actions. SOS Call dials the contact through the Android telephony service. SOS Location sends an SMS containing the current location through SmsManager. Emergency raises a general alert. The call can also be started with the spoken command “sos call”, so that help can be requested without locating a button.

This module depends on resources outside the application. Calls and messages require a working SIM card and carrier coverage and may incur standard charges. The interface validates the number format, as the prompt in Fig. 3(c) shows, but a syntactically valid number is not evidence that the contact is reachable. Delivery of an SMS is also not confirmation that it has been read. The feature therefore complements, and does not replace, established emergency services.

VII. Local Storage and Privacy

TRIVEX keeps no user account and no remote database. Its persistent state consists of five values held in Android SharedPreferences, a lightweight key-value store, as listed in Table V. This choice suits an application whose data are a handful of settings, and it keeps the configuration available.

V. Local Key-Value Storage Schema

Key	Type and purpose
Emergency_Contact	String; phone number of the trusted contact
Home_Lat	Double; latitude of the saved home location
Home_Long	Double; longitude of the saved home location
App_Language	String; language for announcements and reading
Voice_Speed	Float; speaking rate of the assistant

Although small, this store is sensitive: it contains a personal phone number and the coordinates of the user’s home. SharedPreferences files reside in application-private storage, which separates them from other applications on an unmodified device, but their contents are not encrypted by default. Protecting these values with encryption backed by the Android Keystore is a recommended hardening step that the present prototype does not implement.

Two data flows leave the device. The optional scene-description service transmits camera input to a cloud model, and the SOS function transmits the user’s location by SMS to the saved contact. Both are initiated by the user, but camera frames can include bystanders and private surroundings. A deployment should state clearly when frames are being sent off the device and should allow the cloud feature to be switched off. Object detection, text recognition, and command recognition process their inputs locally and transmit nothing.

VIII. Evaluation and Discussion

A. Functional Testing and Observed Behavior

Testing combined JUnit tests of application components with checks of the complete workflows on an Android smartphone. The plan covered detection and OCR on live CameraX input and on snapshots, the response of the recognizer to commands such as "open camera", "open reading mode", and "sos call", the flow of results from the sensors to the TTS engine, and field checks of the location and SOS modules.

Table VI records the observed outcomes. Location was checked in the field in Dharwad, Karnataka. Spoken English commands moved the application between camera detection, settings, and location services without touch input.

VI. Observed Outcomes of Functional Testing

Workflow	Observed outcome
Reading printed text	Text captured; structured paragraphs read aloud
Current location	Position reported as a readable address
Home location	Saved and retained
Trusted contact	Number stored
SOS call and SMS	Actions triggered from the interface
Voice commands (English)	Hands-free movement between modes
Scene description	Returned when connectivity was available

These observations establish that the workflows function as designed on the test device. They are not a labelled evaluation. No test dataset, ground truth, detection accuracy, OCR error rate, command recognition rate, or timing measurement was recorded, and outcomes for object detection were not logged separately from the demonstration of the camera mode. No study with visually impaired participants has yet been conducted.

B. Performance Considerations and Evaluation Protocol

The architecture is expected to respond quickly because detection, OCR, and command recognition avoid network transit, and the nano-scale detector is intended for low-power processors [4]. This is a design rationale and not a measured result. Frame rate, the delay between capture and announcement, memory use, and battery consumption all depend on the handset and must be measured before any real-time claim is made.

Table VII sets out the measurements required. Detection should be scored on labelled images of indoor and outdoor scenes captured with the application itself, since published benchmark figures do not transfer directly to handheld footage taken by a user who cannot aim the camera. OCR should be scored on printed samples that vary in font size, distance, and illumination, separately for each supported language. Latency should be reported on at least one entry-level and one mid-range handset.

VII. Proposed Measurements for Controlled Evaluation

Property	Metric and condition
Object detection	Precision, recall, and mean average precision by class and lighting
Text recognition	Character and word error rate by language
Voice commands	Recognition accuracy across speakers and noise levels
Responsiveness	Median and 95th-percentile delay from capture to speech
Resource use	Memory, storage, and battery drain in continuous use
Location	Error against surveyed reference points
Usability	Task completion and time with blind and low-vision users

Technical metrics do not by themselves show that the application helps its intended users. A usability study with blind and low-vision participants should record task completion and time for representative tasks such as locating an object, reading a notice, and raising an alert, and should gather feedback on the amount and timing of speech. Excessive or badly timed announcements can be as unhelpful as missing ones.

C. Discussion of the Design

The principal strength of the design is that its core functions do not depend on a network. Detection, reading, and voice control remain available in areas of poor coverage, which are also the places where a visually impaired traveller is least able to seek other help. Local processing also keeps camera images and speech on the handset for these functions, and the use of built-in hardware avoids the cost of dedicated devices.

The same decision imposes costs. Models and language data occupy storage and must be updated with the application, on-device inference draws on the battery during continuous use, and capability is bounded by what a handset can run. The optional cloud description recovers some capability at the price of connectivity and privacy. Stating this trade-off explicitly allows a user or deploying organization to choose which features to enable.

Integration is the second strength. A single interface with one set of commands and one speech channel reduces what the user must learn and removes the need to move between applications. Integration does not, however, imply that each function matches a specialized tool. The location module, for example, reports an address but does not guide the user along a route.

D. Limitations and Future Work

The present prototype has several limitations. Detection and OCR quality depend on lighting, camera focus, and image stability during capture. Scene description and map tiles require internet access, and the SOS function requires cellular service. Positioning accuracy is bounded by the device sensors, and the map provides no routing. Voice commands are accepted in English only, and reading is limited to printed text.

Future extensions follow from these limits. Integration with smart glasses or a body-mounted camera would free the user from holding and aiming a phone. Indoor positioning with Bluetooth beacons or Wi-Fi would extend location assistance to stations and hospitals. Recognition could be broadened to known faces, currency denominations, barcodes, and handwriting, and reading could add translation and summarization of long documents. A live video link to sighted volunteers would cover cases in which automated description is insufficient, and voice control of smart-home devices would extend the same interface beyond the phone.

IX. Conclusion

TRIVEX brings object detection, scene description, text reading, location assistance, and emergency signalling into a single Android application that is operated by voice and answers in speech. Its design executes the YOLO26n detector, Tesseract, and the Vosk recognizer on the device, reserves cloud processing for optional scene description, and stores only five local settings. Functional testing confirmed the reading, location, emergency, and voice-command workflows on a smartphone. The application uses only the built-in camera, microphone, speaker, and positioning hardware, which keeps it affordable. Its value to visually impaired users nevertheless remains to be demonstrated through quantitative measurement of accuracy and latency and through a usability study with the intended users, which are the next steps of this work.

Acknowledgment

The authors thank Prof. Girish Shivanagudi for his guidance as project guide, Prof. Farjana Nadaf, project coordinator, Dr. Venkatesh Shankar, Head of the Department, Dr. V. A. Kulkarni, Principal, the Department of Computer Science and Engineering, and KLS Vishwanathrao Deshpande Institute of Technology, Haliyal, for their support.

References

1. World Health Organization, “Blindness and vision impairment,” fact sheet, Feb. 10, 2026. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

2. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

3. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” arXiv:2004.10934, 2020.

4. R. Sapkota, R. H. Cheppally, A. Sharda, and M. Karkee, “YOLO26: Key architectural enhancements and performance benchmarking for real-time object detection,” arXiv:2509.25164, 2025.

5. N. Rachburee and W. Punlumjeak, “An assistive model of obstacle detection based on deep learning: YOLOv3 for visually impaired people,” Int. J. Elect. Comput. Eng., vol. 11, no. 4, pp. 3434–3442, Aug. 2021, doi: 10.11591/ijece.v11i4.pp3434-3442.

6. K. S. M. Anbananthen et al., “The Eye: A light weight mobile application for visually challenged people using improved YOLOv5l algorithm,” Emerg. Sci. J., vol. 7, no. 5, pp. 1636–1652, Oct. 2023, doi: 10.28991/ESJ-2023-07-05-011.

7. K. D. Bamane et al., “Enhancement in real time deep learning object detection and direction prediction for visually impaired using YOLO and OpenCV,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 3, pp. 887–892, 2024.

8. R. C. Joshi, S. Yadav, M. K. Dutta, and C. M. Travieso-Gonzalez, “Efficient multi-object detection and smart navigation using artificial intelligence for visually impaired people,” Entropy, vol. 22, no. 9, Art. no. 941, 2020, doi: 10.3390/e22090941.

9. A. Aralikatti, J. Appalla, S. Kushal, G. S. Naveen, S. Lokesh, and B. S. Jayasri, “Real-time object detection and face recognition system to assist the visually impaired,” J. Phys.: Conf. Ser., vol. 1706, no. 1, 2020.

10. N. Jawaid, A. Warsi, A. A. Shaikh, and M. Yahya, “Object detection and narrator for visually impaired people,” in Proc. IEEE 6th Int. Conf. Eng. Technol. Appl. Sci. (ICETAS), 2019.

11. R. Smith, “An overview of the Tesseract OCR engine,” in Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR), 2007, pp. 629–633, doi: 10.1109/ICDAR.2007.4376991.

12. D. Povey et al., “The Kaldi speech recognition toolkit,” in Proc. IEEE Workshop Autom. Speech Recognit. Understanding (ASRU), 2011.