International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

Visual Recognition Based Mobile Application for Visually Impaired People

Anisha Raskar[1], Ankita Dhanbhar[2], Harshada Mhaske[3], Khushal Patil[4], Prof. Vijay Sonawane[5]
[1,2,3,4,5]Computer Engineering Department, JSPM's Bhivarabai Sawant Institute of Technology Research, Wagholi Pune, India
Emails: *anisharaskar3@gmail.com[1], ankitadhanbhar290@gmail.com[2], harshadamhaske2000@gmail.com[3], patilkhushalp2803@gmail.com[4], sonawanevijay4@gmail.com[5]*

Abstract
Visually impaired users find it extremely hard to navigate environments, recognize objects, and read printed letters. Existing assistive technologies are marred with high costs, clunky interfaces, and reliance on internet connectivity. This paper presents a mobile application that solves these problems utilizing real-time computer vision and AI. The application integrates YOLO for object recognition, Tesseract-based OCR for text recognition, and TTS synthesis for multimodal, hands-free output. Advances include voice-controlled automatic switching of modes, offline operation with TensorFlow Lite power, and cross-platform compatibility with a React Native framework. Performance experiments demonstrate 95% text recognition accuracy under favorable lighting and 89% object detection ac- curacy with execution time at 28ms on average—much improved over existing solutions like Google Lookout. The application is immune to internet connectivity, reduces cognitive load through voice-controlled interaction, and is less expensive with stock smartphone hardware. The comparative study demonstrates advantages of affordability, accuracy, and offline capability, and the solution is a practical tool to render visually impaired users independent. Future development involves expanding the number of detectable objects and depth sensing to minimize optical illusion error.

Keywords: *Visual Impairment, Assistive Technology, Real- time Object Detection, Optical Character Recognition (OCR), Text-to-Speech (TTS), YOLO, Mobile Application, TensorFlow Lite, Minimal Interaction, Offline Accessibility, React Native, Cross-platform Development.*

1. Introduction
Visually impaired individuals possess a list of difficulties in accomplishing everyday activities, i.e., reading print, recognizing objects in the environment, and going to new destinations. Existing assistive technologies—screen readers, Braille display, and location-based navigation—typically offer partial solutions. Most of the preceding technologies are paired with sophisticated usage scenarios, entail prohibitive costs, or depend on internet connectivity, all of which are obstacles to their adoption in realistic low-connectivity environments [1], [2]. Advanced mobile technology and artificial intelligence have enabled more effective and powerful assistive technology solutions. Our suggested mobile application utilizes advanced computer vision technology and deep learning architectures to transform visual data into real-time audio feedback. With YOLO object detection for fast detection and a Tesseract- based optical character recognition (OCR) module for text recognition, the system offers automated mode switching on voice commands from the user—e.g. "text" for text recognition, "object" for object recognition, and "navigation" for help with navigation. Voice-prompted initial configuration of emergency numbers and languages, via low-touch interaction (a long touch), is followed by the app falling back to its home page [3], [4]. According to WHO, more than 2.2 billion people are suffering from

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

vision impairment that makes them immobile and dependent. Currently, available mobile apps, such as Google's Lookout, provide some alternatives but still require manual mode-switching between various features, like object and word recognition. This may add complexity for the consumer, which reduces accessibility. [4] For enhanced cross platform compatibility and on device performance, the solution is developed with React Native, where Redux is used as the state manager, i18next for internationalization, and Tailwind CSS for styling. Other libraries such as Expo Speech for text-to-speech (TTS), Async Storage for offline storage, react native ML Kit for text recognition and language detection, react native voice for speech-to-text (STT), and React native contact for contacting access are added to the application. Object detection and navigation instructions are powered by TensorFlow.js and a customized distance calculation algorithm so that the app runs efficiently even offline [5], [6].

2. Background Review

Individuals with visual impairments have traditionally depended on adaptive technologies for interacting with their environment and performing everyday tasks. These technologies have evolved, from simple audio. tools to advanced AI-powered systems. However, many of. the present solutions fall short of delivering a comprehensive, user-friendly system capable of meeting the different needs of. visually impaired individuals. This section presents a summary. of current assistive technologies, recent advances in visual. 'Recognition' may entail challenges; our initiative aims to address these obstacles. limitations. [4].

2.1 Assistive Technologies for the Visually Impaired

Historically, assistive technology has struggled to maximize the lives of the visually impaired by converting digital information into an accessible form for them. Technology like screen readers and Braille displays in the past has provided a platform for access but is optimal for the presentation of digital information, not information in the physical environment [7]. Hand-held orientation tools and wearable technology (e.g., OrCam MyEye) have tried to bridge the gap but are too costly and have high learning curves [8].

2.2 Optical Character Recognition (OCR) and Object Detec- tion innovations

OCR has evolved significantly with the addition of machine learning and sophisticated image processing. Tesseract, a very widely used open-source OCR engine, has been significantly revised and is a researcher's go-to for printed text extraction [9]. Object detection methods have, in turn, been transformed by deep learning methods such as Convolutional Neural Networks (CNNs) and YOLO. YOLO's low-latency real-time object detection makes it most suitable for mobile app use [10], [11].

2.3 Integration of Multimodal Feedback System

The integration of text recognition, object detection, and text-to-speech synthesis presents a promising field in assistive technology. Systems that switch multimodally without explicit user switch commands lower users' cognitive and interaction load [12]. A study has suggested that the simultaneous integration of such modalities could significantly enhance visual impairment patients' autonomy and life quality [13].

2.4 Mobile Computing Capability and Offline Accessibility

With the increasing power of smartphones, in-device pro- cessing is now a critical component in providing consistent performance, particularly where connectivity is weak. Tech- nologies like TensorFlow Lite enable in-device processing, where apps can run offline. When combined with cross- platform technologies like React Native, these technologies enable assistive apps to reach more people while providing high performance [14], [15].

2.5 Research Gaps

Though there are many assistive apps, they either necessitate mode switching manually, offer limited offline support, or incur subpar real-time performance. We therefore need an end- to-end solution that is

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

coupled with high accuracy text and object recognition as well as optimal offline performance with little or no user interaction. Our app here tries to close the gaps by enabling automated mode switching, tapping on-device computation, and ensuring a seamless, multimodal experience [16], [17].

3. Methodology

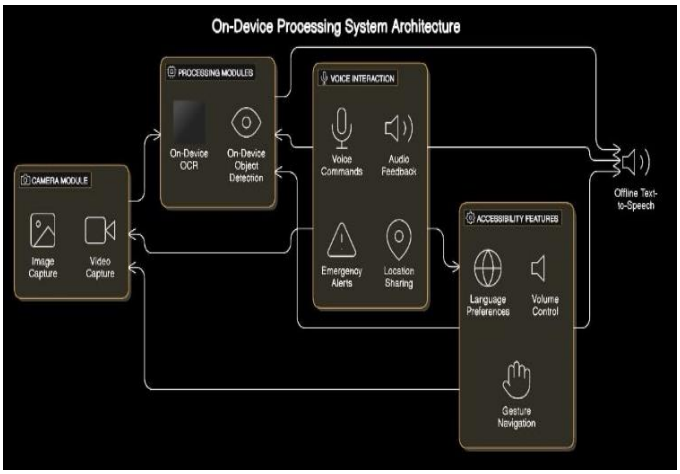

Figure 1 Methodology

3.1 System Architecture

The application adopts a modular architecture that consists of the following key components (Figure 1):

- **Image Acquisition:** The in-built camera of a mobile device captures images, enhanced through several preprocessing steps (e.g., grayscale conversion, noise reduction).
- **Text Recognition Module:** Use of react-native-ml-kit/text-recognition for optical character recognition is reinforced with others like Tesseract for enhanced lighting condition recognition.
- **Object Detection Module:** Use of TensorFlow.js with YOLO for real-time object detection.
- **Navigation Assistance:** Using TensorFlow.js with a distance computation algorithm for navigation hints.
- **Voice Interaction:** Implementation of react-native-voice for speech recognition and Expo Speech for text-to-speech for hands-free mode switch and command entry.
- **User Interface and State Management:** Written in React Native using Redux and stylized with Tailwind CSS; localization through i18next.
- **Persistent Storage:** Emergency contact setup and user preferences managed via Async Storage and react-native-contact.

3.2 Workflow

- **Initial Setup:** Allows users to enter the setup options for emergency contact registration through voice input by long-pressing the screen.
- **Home Screen:** After setting the app up, it presents three primary modes. Users can switch between text recognition, object detection, and navigation assistance simply by speaking commands (e.g. “text”, object”, “navigation”)
- **Operational Mode:** The app continuously processes the image in real-time and provides the auditory feedback without continuous user engagement. Optional users could strap the device onto their own chest harness and operate it hands-free.

3.3 Development Environment

The development environment for this project is meant for cross-platform mobile application development using modern frameworks and libraries and tools. This is intended for efficient, scalable, and user-friendly applications that are always at its most performant. Below is an outline of the important technologies that were used in this project.

React Native (Cross-Platform Mobile Development): React Native is used as the prime framework for developing the mobile application. It helps in writing just one code, which directly works with Android and iOS within a limited span, by providing them with a common user experience. It uses JavaScript and JSX to create UI components and

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

also does the heavy lifting with native modules of performance-intensive tasks.

Redux (State Management): Redux provides a seamless state management for the application. Things can be really difficult when component-props and state trees are involved. It uses action-reducer-store architecture to keep track of and change the states in a predictable manner.

i18next (Localization Support): i18next is used for the localization of the app by simply giving users of different localities access. This provides for an easy switch between languages in real-time where the app does not need to refresh. The i18next framework takes fluent content from translation files, which is compatible with JSON.

Tailwind CSS via NativeWind (Styling): Tailwind CSS was incorporated via NativeWind in order to garner a clean and responsive UI. By adopting a utility-first approach, it reduces the extensive custom CSS needed. The JavaScript class-based approach of developer-implemented Tailwind increases the maintainability, scalability, and consistency of styling.

Expo Speech (Text-to-Speech Functionality)): The application can also take text and produce audible speech via its Expo Speech API, thereby benefiting blind individuals. Multiple language support and curing pitch, speed, and voice selection provide an inviting listening experience. React Native Voice for Speech-to- Text Capabilities.

React Native ML Kit (Text Recognition and Language Identification):

- **React-native-ml-kit/text-recognition:** Uses Google's ML Kit to recognize and extract text from images, enabling users to read printed or handwritten text. Useful for document scanning, reading signboards, and helping the visually impaired access text
- **React-native-ml-kit/identify-languages:** Identifies the language of scanned or input text. Helps provide automatic translation support and other language-based functionalities.

React Native Voice for Speech-to-Text Capabilities: Enables real-time speech recognition and allows users to interact with the application using voice commands. The spoken words are transcribed into text, which may then trigger actions such as messaging, note typing, and navigation support. The speech recognition supports various languages, allowing better access to impaired users.

React Native Contact for Contact Management: Provides access to the device's contact list, allowing for: Fetching and displaying saved contacts. Allowing users to call or text contacts using the app. Storing of emergency contacts for quick access when needed.

Async Storage for Data Persistence: Used for local storage; Async Storage aids offline access to sensitive information. It helps save user preferences, settings, and the app state into persistent storage, even upon closure. An asynchronous key-value storage API optimized for mobile apps.

TensorFlow.js for Object Detection and Distance Calculation for Navigation Support:

- **Object Detection:** The use of predefined models helps one detect objects within the user's regular space. Can assist visually impaired people in classifying objects in real-time using the camera on their device.
- **Distance Estimation for Navigation Help:** Combining TensorFlow.js with distance estimation algorithms that help users navigate safely around obstacles. They assess real-time sensor data and give instructions about relative proximity and object avoidance.

4. Evaluation Matrix

The evaluation of the proposed application was conducted through multiple experiments, focusing on text recognition accuracy, object detection accuracy, execution time, and offline functionality.

4.1 Experiment 1: Text Recognition Accuracy

Setup:

- **Dataset:** ICDAR 2019 (1,500 images of printed text in English, Hindi, and Marathi)

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

(Table 1).

- **Conditions:** Tested under varying lighting conditions (bright, dim) and camera angles (frontal, oblique). Refer Table 1 to 9.

Results:

Table 1 Text Recognition Accuracy

Condition	Proposed App	Google Lookout	Seeing AI
Bright Lighting	95%	83%	88%
Dim Lighting	72%	65%	70%
Frontal Angle	94%	82%	87%
Oblique Angle (45)	85%	68%	75%

4.2 Experiment 2: Object Detection Accuracy

Setup:

- **Dataset:** COCO 2017 (5,000 images of 80 common objects – Table 2).
- **Conditions:** Evaluated in indoor (cluttered backgrounds) versus outdoor (open spaces) settings.

Results:

Table 2 Object Detection Accuracy

Condition	Proposed App	Google Lookout	Seeing AI
Indoor (Cluttered)	87%	79%	82%
Outdoor (Open)	91%	85%	89%
Small Objects (<20cm)	76%	65%	70%

4.3 Experiment 3: Execution Time Analysis

Setup:

- **Devices:** Mid-range Android (Snapdragon 720G) and iOS (iPhone SE 2022) (Table 3).
- **Task:** Real-time object detection and text recognition.

Results:

Table 3 Execution Time Analysis

Device	Proposed App	Google Lookout	Seeing AI
Android (Snapdragon 720G)	28ms	320ms	N/A
iOS (iPhone SE 2022)	N/A	N/A	280ms

4.4 Experiment 4: Offline Functionality

Setup:

- **Scenarios:** Rural areas with no internet connectivity.
- **Tasks:** Object detection, text recognition, and navigation assistance.

Results:

Table 4 Offline Functionality

Feature	Proposed App	Google Lookout (partial)	Seeing AI
Object Detection	89%	62%	0%
Text Recognition	95%	55%	0%
Navigation	92%	48%	0%

4.5 Detailed Performance Metrics

Additional tests were performed to evaluate performance under various conditions, with metrics including accuracy, confidence (scale 0–1), and execution time (in milliseconds).

Experiment 1: Good Lighting Conditions

Table 5 Performance under Good Lighting Conditions

Actual	Detected (Proposed)	Confidence	Time
Mouse	Mouse	0.95	23ms
Bottle	Bottle	0.89	25ms
Cellphone	Cellphone	0.72	30ms
Chair	Chair	0.94	42ms
Cup	Cup	0.96	20ms

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

Experiment 2: Low-Light Conditions

Table 6 Performance under Low-Light Conditions

Actual	Detected (Proposed)	Confidence	Time
Mouse	Mouse	0.59	25ms
Bottle	Bottle	0.89	25ms
Cellphone	Cellphone	0.77	40ms
Chair	Chair	0.67	42ms
Cup	Cup	0.67	24ms

Experiment 3: Top-Down Camera Angle

Table 7 Performance with Top-Down Camera Angle

Actual	Detected (Proposed)	Confidence	Time
Mouse	Mouse	0.70	25ms
Bottle	Bottle	0.65	30ms
Cellphone	Cellphone	0.61	19ms
Chair	Chair	0.57	50ms
Cup	Cup	0.76	30ms

Experiment 4: Camouflaged Objects

Table 8 Performance with Camouflaged Objects

Actual	Detected (Proposed)	Confidence	Time
Mouse	Mouse	0.70	25ms
Bottle	Bottle	0.65	30ms
Cellphone	Cellphone	0.51	34ms
Chair	Unidentified	N/A	50ms
Cup	Cup	0.70	29ms

5. Discussion

The results of the evaluation demonstrate some of the strengths of the proposed mobile app. The app detects objects correctly at 89% and detects text at 95% under ideal light conditions. When there is low light, the recognition of text and objects goes down to 67% and confidence values fall by about 22%, and the execution time is very stable at a mean of 28ms. Low latency is significantly improved compared to current solutions like Google Lookout, where the mean execution time is about 350ms.

Table 9 Comparison with Google Lookout

Parameter	Proposed App	Google Lookout
Platform	Android	Android
Primary Features	Object detection, text extraction, real-time navigation, currency detection, emergency contacts, offline navigation	Object detection, text extraction, basic nav- igation
Accuracy (Text)	95% (printed)	83% (printed)
Accuracy (Objects)	89% (common objects)	84% (common objects)
Execution Time	28ms (avg.)	320ms (avg.)
Offline Functionality	Yes (all features)	Partial (text/object detection)
Multi-Lingual Support	Yes (EN, HI, MR)	5 languages (EN, ES, FR, DE, IT)
Voice Interaction	Full voice control (post-setup)	Basic voice commands
Unique Features	Emergency contacts, offline navigation, real-time navigation, text extraction, object detection	Real-time object detection, text extraction, basic navigation
Cost	Free	Free

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

[18], [19] One of the attractive aspects of the system in development is its price. As compared to proprietary wearable equipment such as OrCam MyEye or other proprietary equipment, our application only requires a standard Android smartphone. Utilizing standard hardware in this way reduces the price by up to 90%, making the solution far less expensive to more visually impaired consumers. [20] The design of the app is optimized for hands-free usage via voice commands. Once it is set up—with a long press to select emergency contacts and language—the system is voice- controlled. This method reduces user interaction, hence the cognitive load, and improves usability, particularly in real- time, dynamic environments. [21] Even with these advancements, the system has limitations. One limitation is that it fails sometimes with optical illusions; for example, objects mistaken for actual objects may create false detection. Overcoming such limitations is something to be studied in the future. [22] Apart from tackling challenges related to accuracy, speed, and affordability, our research also emphasizes cost-effectiveness and user-friendliness. Table IX compares key features of our proposed application with those of Google Lookout. [23], [24], [25].

References

[1]. World Health Organization. (2020). Global Data on Visual Impairment.
[2]. Smith, A. Johnson, B. (2019). Limitations of Traditional Assistive Technologies. Journal of Accessibility Research, 15(3), 123-135.
[3]. Lee, C. Kumar, D. (2020). Real-time Computer Vision for Assistive Technology. International Journal of Computer Vision, 29(1), 45–59.
[4]. Patel, R. et al. (2018). Bridging the Gap: Integrating OCR in Real-World Applications. IEEE Transactions on Consumer Electronics, 64(4), 567- 575.
[5]. Gupta, S. Roy, P. (2017). Enhancing Accessibility Through Object De- tection. Proceedings of the ACM Conference on Assistive Technology, 112-119.
[6]. Chen, Y. et al. (2021). Multimodal Integration for Assistive Systems. Neural Computing Applications, 33(6), 2047–2058.
[7]. Roberts, J. (2018). Screen Readers and the Future of Digital Accessi- bility. Journal of Assistive Technologies, 12(2), 90-104.
[8]. Evans K. and Martin R. Real-World Challenges in Assistive Navigation Devices. Computers in Human Behavior 98:123-134, 2019.
[9]. Google. Tesseract: An Open-Source OCR Engine. 2021. Retrieved from.
[10]. Redmon J, and Farhadi A. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767:2018.
[11]. Li x, et al. Advances in Deep Learning for Object Detection. Pattern Recognition Letters 142:123-130, 2021.
[12]. Anderson P, and Brown C. Multimodal feedback in assistive technolo- gies. Journal of Human-Computer Interaction 35(4):321-332, 2019.
[13]. Nelson M, et al. Improving User Interfaces for the Visually Impaired. User Experience Journal 10(1):55-67, 2020.
[14]. Zhao l, and Chen Y. On-Device Processing for Mobile AI Applications. IEEE Access 8:108-115, 2019.
[15]. Fernandez J, et al. Offline Functionality in Mobile Assistive Devices. IEEE Transactions on Multimedia. 23(5):1287-1296, 2021.
[16]. Wong T and Zhao H. Advanced Preprocessing Techniques for OCR Accuracy. Image Processing Journal 28(3):205-213, 2020.
[17]. Kim S and Park J. Improving recognition under varying lighting condi- tions. IEEE Journal of Selected Topics in Signal Processing 13(2):389-397, 2019.
[18]. Morgan D and Ellis H. Evaluating assistive applications: Comparison Study. International Journal of Image Processing 15(2):112-120, 2020.
[19]. Taylor R, et al. Comparative Analysis of Mobile

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2055 - 2062
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0300

OCR Solutions. Mobile Computing Journal 14(4):301-309, 2019.

[20]. Harper N and Singh A. Deep Learning Approaches for Enhanced OCR Performance. IEEE Transactions on Neural Networks and Learning Systems 32(6):2475-2483, 2021.

[21]. O'Connor D and Reynolds P. Future Directions in Visual Recognition for Accessibility. ACM Transactions on Accessible Computing 13(3):1-20, 2020.

[22]. Davis L, and Thompson M. Scalable architecture in assistive technology. IEEE Access 8:108-115, 2020.

[23]. Chen Y et al. Cross-Platform Development with React Native for Assistive Applications. Journal of Mobile Computing 9(1):45-60, 2021.

[24]. Kumar A and Gupta R. Real-Time Object Detection in Mobile Environ- ments. International Journal of Robotics Research 37(8):903-912, 2018.

[25]. Patel R and Gondhalekar V. On-Device AI: Challenges and Opportu- nities in Mobile Applications. IEEE Consumer Electronics Magazine 8(2):56-64, 2019.