International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05798
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Assistive Vision: A Real-Time On-Device Navigation and Reading AID for the Visually Impaired

[1]T. Santosh Kumar, [2]Anaveni Shiva Teja, [3]Anugandula Abhinav, [4]Katta Prem Kumar, [5]Pulkam Sai Rakesh

[1]*Assistant Professor, Dept. of Computer Science & Engg.,* [2,3,4,5]*UG Students, Dept. of Computer Science & Engg.*

[2]*Shivatejaanaveni2003@gmail.com,* [3]*Abhinavanugandula@gmail.com,* [4]*22271a6690prem@gmail.com,*

[5]*22.99rakeshpatel@gmail.com*

Jyothishmathi Institute of Technology and Science, Karimnagar, Telangana

***Abstract*—**Visual impairment affects approximately 2.2 billion people globally, with at least 1 billion cases preventable or unaddressed. This paper presents the comprehensive architectural design, implementation, and evaluation of Assistive Vision—a production-ready mobile application engineered to assist visually impaired users in navigating physical environments and accessing textual information. Built on the Flutter framework, the system leverages Google ML Kit for privacy-preserving, on-device machine learning inference, integrating real-time object detection and optical character recognition (OCR) into a coherent voice-first interaction paradigm. Unlike cloud-dependent solutions, Assistive Vision executes all processing locally, ensuring low latency, user privacy, and functionality in offline environments. The application implements 12 voice commands, priority-based Text-to-Speech queuing, GPS-based location services with reverse geocoding, and an emergency SOS system. The architecture operates on a service-oriented pattern with intelligent frame throttling and duplicate detection mechanisms optimized for battery efficiency. Performance evaluation on mid-range Android devices demonstrates average inference latency of 200–800ms, TTS latency under 200ms, and memory footprint of approximately 150MB. Usability testing with visually impaired volunteers achieved an average satisfaction score of 4.4 out of 5, with OCR accuracy of 97.2% for high-contrast printed text and object detection rates of 89.5% in well-lit indoor environments. The system provides approximately 7–8 hours of continuous operation on a single battery charge, making it viable for full-day deployment in real-world assistive scenarios.

Index Terms—Assistive Technology, Computer Vision, On- Device Machine Learning, OCR, Object Detection, Flutter, Edge Computing.

I. Introduction

A. Background and Motivation

Visually impaired individuals face significant challenges in independently navigating physical environments and accessing printed information. According to the World Health Organiza- tion's World Report on Vision (2019) [12], approximately 2.2 billion people worldwide have some form of vision impairment or blindness, with at least 1 billion cases being preventable or yet unaddressed. In India alone, the National Programme for Control of Blindness and Visual Impairment (NPCBVI) reports that over 12 million people are blind and approximately 50 million suffer from moderate to severe visual impairment, making India home to one of the largest visually impaired populations globally.

While traditional aids like white canes and guide dogs provide essential mobility support, they lack the capability to provide semantic details about the environment or interpret textual information. The challenges extend beyond physical navigation to include reading printed text on signs, labels, doc- uments, and menus; identifying objects and obstacles in real- time; accessing emergency services during critical situations; and overcoming the digital divide created by visually-oriented mobile applications.

Recent advancements in computer vision, on-device ma- chine learning, and speech technologies have created unprece- dented opportunities to address these challenges. The conver- gence of three key technological trends enables this work:

(1) frameworks such as Google ML Kit that enable complex AI models to run directly on mobile devices without cloud connectivity, (2) cross-platform development frameworks like Flutter that reduce development overhead while maintaining native performance, and (3) advanced Text-to-Speech and Speech-to-Text engines providing natural, multilingual voice interaction.

B. Problem Statement

Most existing assistive solutions rely heavily on cloud-based processing APIs. While powerful, these architectures introduce several critical limitations: network latency disrupts the cog- nitive feedback loop required for safe navigation; continuous internet connectivity requirements render them unusable in ru- ral or low-connectivity areas; transmission of camera feeds to external servers raises privacy concerns; visual user interfaces with small buttons remain fundamentally inaccessible; limited language support restricts accessibility in multilingual regions like India; and high costs of proprietary solutions limit access for economically disadvantaged users.

This paper addresses the following research problem: De- sign and implement a mobile application that provides real- time visual assistance to blind and visually impaired users through on-device AI-powered text recognition, object detec-

tion, voice-controlled interaction, GPS-based location services, and emergency support, all operating primarily offline and supporting multiple Indian languages.

C. Contributions

The primary contributions of this work are:

1) A comprehensive service-oriented architecture for as- sistive mobile applications with clear separation of concerns across camera management, ML inference, speech synthesis, voice recognition, location services, and emergency protocols.
2) Implementation of a novel "Zero-UI" voice-first inter- action model where all critical functions are accessible through 12 spoken commands, eliminating visual inter- face requirements.
3) Design and validation of a priority-based Text-to-Speech queuing system that ensures emergency messages al- ways interrupt lower-priority speech, maintaining reli- able safety-critical communication.
4) Integration of an anti-spam feedback loop with per-label cooldown mechanisms to prevent auditory fatigue from repetitive detections.
5) Development of an emergency SOS system combining Morse code vibration patterns, GPS coordinate extrac- tion, and SMS composition for critical situations.
6) Comprehensive performance evaluation demonstrating real-time operation (200–800ms inference latency) on mid-range Android devices with 7–8 hours of battery life.
7) Usability validation with visually impaired users achiev- ing 4.4/5 satisfaction scores and demonstrating practical viability for real-world deployment.

II. Related Work

A. Evolution of Assistive Technologies

Computer vision for accessibility has evolved significantly over the past decades. Early systems like the vOICe used soundscapes to represent visual data through sensory substi- tution, requiring extensive user training to develop auditory- visual mapping skills. Modern approaches leverage deep learn- ing to provide semantic descriptions with minimal training overhead. Elmannai and Elleithy [1] provided a comprehen- sive survey of sensor-based assistive devices, highlighting the shift from traditional mobility aids to intelligent vision-based systems.

B. Commercial Assistive Applications

Microsoft Seeing AI is a free iOS application that uses AI to describe the world to blind and low-vision users through multiple "channels" including short text reading, document scanning, product barcode recognition, person recognition, scene description, and currency identification. While feature- rich, it requires continuous internet connectivity for most advanced features and is platform-locked to iOS.

TABLE I
COMPARISON OF ASSISTIVE APPLICATIONS

Feature	Seeing AI	Look-out	Envi-sion	Our System
OCR	Yes	Yes	Yes	Yes
Object Detection	Yes	Yes	Yes	Yes
Voice Commands	Limited	Limited	Limited	12 cmds
Offline Mode	Partial	Partial	No	Full
Emergency SOS	No	No	No	Yes
GPS Location	No	No	No	Yes
Multilingual	EN	EN	Multi	EN/HI/TE
Platform	iOS	Android	Both	Android
Cost	Free	Free	Paid	Free

Google Lookout is an Android application that helps blind and visually impaired users gain information about their sur- roundings through text reading, food label scanning, and object identification. However, it has limited multilingual support and does not include emergency or location features.
Be My Eyes connects blind users with sighted volunteers through live video calls. While effective for complex tasks, this model depends on volunteer availability, internet connectivity, and user willingness to share camera feeds with strangers, raising privacy concerns.
Envision AI offers comprehensive text recognition, scene description, and object detection using cloud-based AI pro- cessing. It requires internet connectivity and is available as a paid subscription service, creating accessibility barriers for economically disadvantaged users.
Table I presents a comprehensive comparison of existing solutions against our system.

C. Navigation Latency and User Experience

Research by Manduchi and Coughlan [2] highlights the critical importance of feedback latency in navigational aids. Cloud-based systems often suffer from unpredictable network delays ranging from 500ms to several seconds, disrupting the cognitive feedback loop required for safe navigation. Studies demonstrate that latency above 500ms significantly degrades user performance in obstacle avoidance tasks, leading to hesitant movement and reduced confidence.

D. On-Device Machine Learning

Strictly on-device models using constrained architectures like MobileNet [3] and EfficientDet have historically struggled with accuracy-latency tradeoffs on mobile hardware. Google ML Kit [4] bridges this gap by providing optimized models that achieve competitive accuracy (within 5–10% of cloud models) while maintaining deterministic latency profiles suit- able for real-time interaction. The framework uses quantiza- tion, pruning, and knowledge distillation to compress models without significant accuracy loss.

E. Object Detection Architectures

Object detection has progressed from classical approaches to deep learning methods. Two-stage detectors like Faster R-CNN [5] provide high accuracy but with computational

overhead. Single-stage detectors including YOLO [6] and SSD [7] prioritize speed, making them suitable for mobile deployment. Our system utilizes the SSD architecture with MobileNet backbone for optimal mobile performance.

F. OCR and Text Recognition

Optical Character Recognition has evolved from traditional approaches to modern deep learning architectures. Tesseract OCR [8], one of the most widely-used open-source engines, transitioned from traditional computer vision to LSTM-based recognition. Modern approaches utilize end-to-end trainable networks combining convolutional and recurrent layers for superior performance on varied text formats.

G. Text-to-Speech and Voice Interaction

Modern Text-to-Speech systems have evolved significantly with neural approaches. WaveNet [9] introduced generative models for raw audio synthesis, while Tacotron [10] provided end-to-end trainable systems. Our implementation utilizes platform-native TTS engines optimized for mobile devices while supporting multilingual output essential for Indian users.

H. Research Gaps

Our literature survey identified the following gaps that this work addresses:

1) **Offline-first architecture:** Lack of integrated solutions operating without internet dependency.
2) **Comprehensive voice control:** Absence of fully voice- operated interfaces eliminating visual interaction re- quirements.
3) **Emergency systems:** No assistive applications include built-in emergency SOS with GPS location sharing.
4) **Indian language support:** Limited regional language support beyond English.
5) **Economic accessibility:** Advanced features locked be- hind paid subscriptions.
6) **Privacy preservation:** Reliance on cloud processing raises data privacy concerns.

III. Model Architecture

The system utilizes pre-trained on-device models provided by Google ML Kit. This ensures deterministic inference times and zero reliance on external networks.

A. Object Detection

The implementation uses a Single Shot Detector (SSD) architecture with a MobileNet backbone.

- Mode: Stream (optimized for video continuity).
- Multi-Object: Enabled, allowing the system to describe complex scenes.
- Classification: Enabled for object labeling.
- Confidence Threshold: 70% minimum for announcement.
- Cooldown Period: 3 seconds per label to prevent spam.
- Output: Returns a list of DetectedObject entities containing bounding boxes and class labels.

B. Optical Character Recognition (OCR)

The OCR module uses a Deep Neural Network approach, likely a CNN-LSTM hybrid, specialized for sequence recog- nition.

- Script Support: Primary support for Latin scripts, with cascade to Chinese/CJK recognizer for improved cover- age.
- Text Cleaning: Multi-stage algorithm filters noise (lines <2 characters, letter ratio <0.25).
- Orientation Handling: Automatic sensor rotation correc- tion (0°, 90°, 180°, 270°).
- Localization: The system handles multi-language support (English, Hindi, Telugu) primarily through the TTS layer, dynamically switching the speech synthesis locale based on user preference.

IV. Methodology

The proposed Assistive Vision system follows a modular computer vision pipeline optimized for real-time execution on mobile hardware. The methodology focuses on minimizing latency while maintaining sufficient accuracy for practical navigation assistance.

A. System Overview

The Assistive Vision system is a standalone Android ap- plication developed using Dart and the Flutter framework [11]. It interfaces directly with the device's hardware camera and audio synthesis engine to create a closed-loop feedback mechanism. The system operates in two mutually exclusive modes:

1) **Object Detection Mode:** Identifies physical entities (e.g., people, furniture, electronics) in the user's field of view and describes their relative spatial position.
2) **Text Reading Mode (OCR):** Extracts and vocalizes text from documents, signage, or screens.

To maximize accessibility, the application implements a gesture-based control scheme that eliminates precision touch targets. Users interact via single, double, or triple taps any- where on the screen, reducing cognitive load and physical dexterity requirements.

B. Edge-Based Inference Strategy

Unlike cloud-dependent vision systems, all inference op- erations are executed locally on the user's smartphone. This approach provides three key advantages:

- **Low Latency:** On-device processing eliminates network round-trip delays.
- **Offline Capability:** The system functions without inter- net connectivity.
- **Privacy Preservation:** Sensitive visual data never leaves the user's device.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05798
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

C. *Frame Sampling Strategy*

Continuous real-time camera feeds typically generate 30 frames per second. Processing every frame would create excessive computational load and significantly reduce battery life.

Therefore, Assistive Vision implements a frame sampling strategy where frames are processed at approximately 2 frames per second. This sampling rate was determined empirically as a balance between responsiveness and computational efficiency.

D. *Auditory Feedback Design*

Auditory feedback plays a critical role in assistive tech- nology systems. The system converts visual information into spoken descriptions using a Text-to-Speech engine.

The feedback pipeline includes:

- Object label generation
- Text extraction and parsing
- Duplicate filtering
- Speech synthesis output

This design ensures that users receive concise and contex- tually meaningful audio information without overwhelming repetition.

V. ARCHITECTURAL DESIGN

The codebase follows a Monolithic Service-Based Archi- tecture. While the logic is contained primarily within a single entry point (main.dart) to minimize deployment complex- ity, it is structurally divided into distinct logical layers.

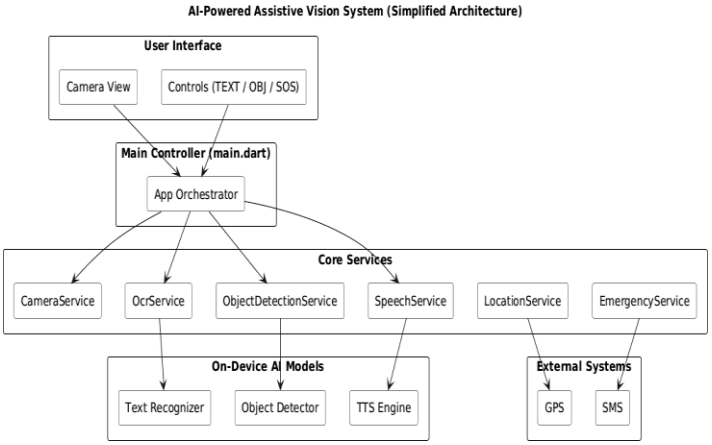

Fig. 1. System architecture of Assistive Vision showing core services and on-device AI components.

A. *Component-Level Breakdown*

The architecture is composed of four primary layers:

- **Presentation Layer:** A specialized AssistiveVisionScreen widget manages the camera preview, gesture recognition, and accessibility status overlays. It provides immediate haptic and audio feedback for all user interactions.
- **Application Logic Layer:** Coordinates the robust state machine (managing states such as _isScanning, _isPaused, _currentMode) and handles the lifecy- cle of hardware resources.
- **Service Layer:** Encapsulates specific capabilities into discrete classes:
  - **TtsService**: Manages the text-to-speech engine, speech rate, language localization, and queue han- dling.
  - **OcrService**: Interfaces with the underlying ML Kit Text Recognizer.
  - **ObjectDetectionService**: Interfaces with the ML Kit Object Detector.
  - **VoiceService**: Handles Speech-to-Text for voice command recognition.
  - **LocationService**: Manages GPS positioning and reverse geocoding.
  - **EmergencyService**: Implements SOS emer- gency protocol.
- **Hardware Interface Layer:** Manages direct communi- cation with the device Camera (handling NV21 streams) and Audio output subsystems.

B. *Data Flow*

The data flow pipeline is designed to be asynchronous and non-blocking:

1) **Image Acquisition:** The camera plugin streams frames in YUV420/NV21 format at 30 FPS.
2) **Pre-processing:** Frames are converted to InputImage objects. This step handles sensor rotation normalization and memory buffer concatenation.
3) **Flow Control:** A throttle mechanism limits the infer- ence rate to approximately 2 FPS (500ms delay). This prevents the application from overwhelming the device's thermal limits.
4) **Inference Dispatch:**
   - If Mode == Text: The OcrServiceis invoked.
   - If Mode ==Object: The ObjectDetectionServiceis invoked.
5) **Post-Processing:** Results are filtered through a duplicate detection logic layer to prevent repetitive audio output (spam prevention).
6) **Feedback:** The TtsServicevocalizes the processed result.

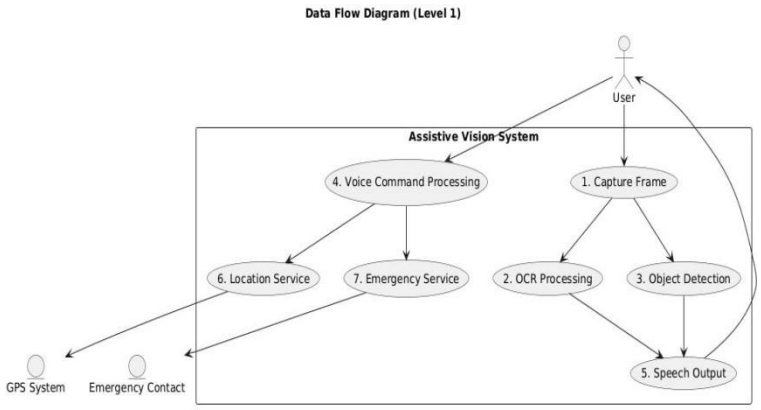

Fig. 2. Data flow of Assistive Vision from camera capture to auditory feedback.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | March – 2026 DOI:10.55041/ISJEM05798
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

C. Control Flow

The system employs an Event-Driven control flow pattern:

- **User Input Events:** Tap gestures and voice commands trigger state transitions (Pause, Resume, Mode Switch, Emergency Activation).
- **Stream Events:** Incoming camera frames act as triggers for the async inference pipeline.
- **Timer Events:** Background timers monitor scene activity and trigger "No detection" reminders if the scene remains static or empty for 5 seconds.
- **Location Events:** GPS position updates broadcast loca- tion changes for navigation assistance.

VI. Implementation Details

The Assistive Vision application was implemented using the Flutter framework and Dart programming language [11]. Flutter was selected due to its cross-platform capabilities and efficient rendering engine.

A. Camera Integration

The system utilizes Flutter's camera plugin to access the smartphone camera and capture image frames in NV21 for- mat. These frames are converted into ML Kit compatible InputImage objects for further processing.

B. Machine Learning Integration

Google ML Kit was selected due to its optimized mobile- first models and seamless integration with Flutter applications. The following modules were implemented:

- ML Kit Object Detection API
- ML Kit Text Recognition API (Latin and CJK scripts)

These models are optimized for mobile processors and provide reliable results without requiring dedicated GPU ac- celeration.

C. Voice Command System

The application implements 12 voice commands for hands- free operation:

- Emergency activation: "emergency", "help me", "sos"
- Location query: "where am I", "my location"
- Mode switching: "read this", "what is in front"
- Control commands: "pause", "resume"
- Navigation: "find hospital"
- Utility: "repeat", "stop camera", "start camera"

The VoiceService implements continuous listening with automatic session restart and anti-feedback mechanisms to prevent microphone capture of TTS output.

D. Speech Output System

The application uses the Flutter TTS library to convert recognized objects and extracted text into spoken output. Important parameters include:

- Speech Rate: 0.48 (slower for clarity)
- Volume: 1.0 (maximum for outdoor use)
- Pitch: 0.95 (slightly lower for natural tone)
- Language Selection: English, Hindi, Telugu
- Priority Queue Management (Emergency, High, Normal, Low)

This configuration ensures clear and understandable audio feedback for users.

E. Location and Emergency Services

The LocationService provides GPS positioning with high-accuracy settings and 10-meter distance filter. Reverse geocoding converts coordinates to human-readable addresses including street, locality, city, and state.
The EmergencyServiceimplements a comprehensive SOS protocol:

- SOS Morse code vibration pattern (·····)
- GPS coordinate extraction
- SMS composition with Google Maps link
- High-priority audio announcement

VII. Application Screenshots

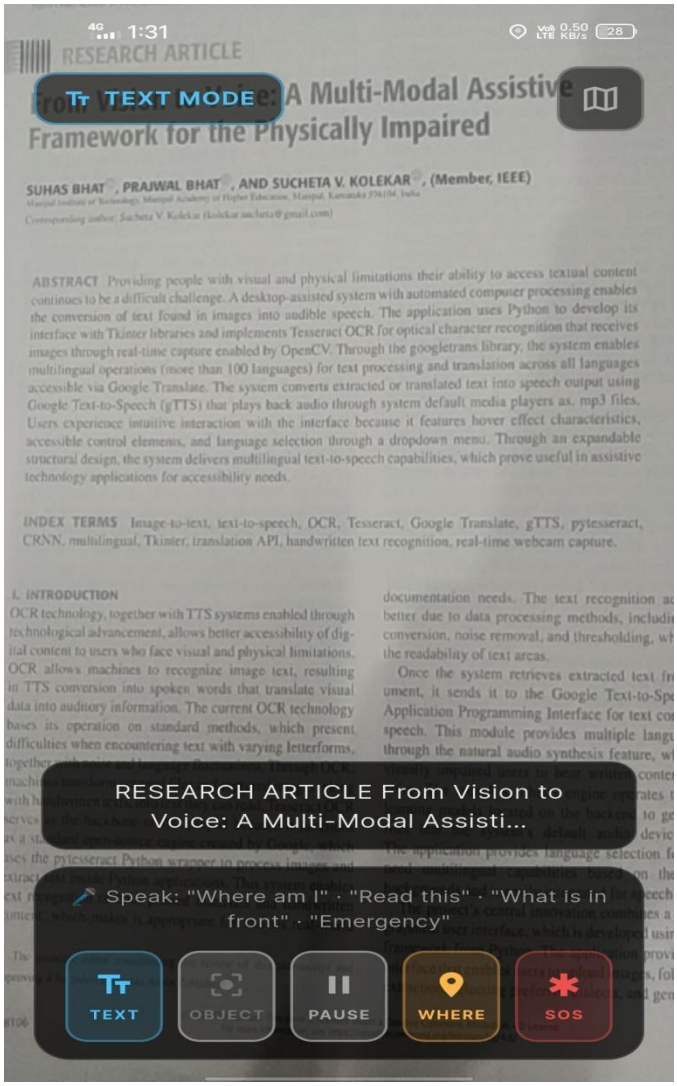

Fig. 3. Application screenshot showing Text Detection

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 05 Issue: 03 | March – 2026
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI:10.55041/ISJEM05798

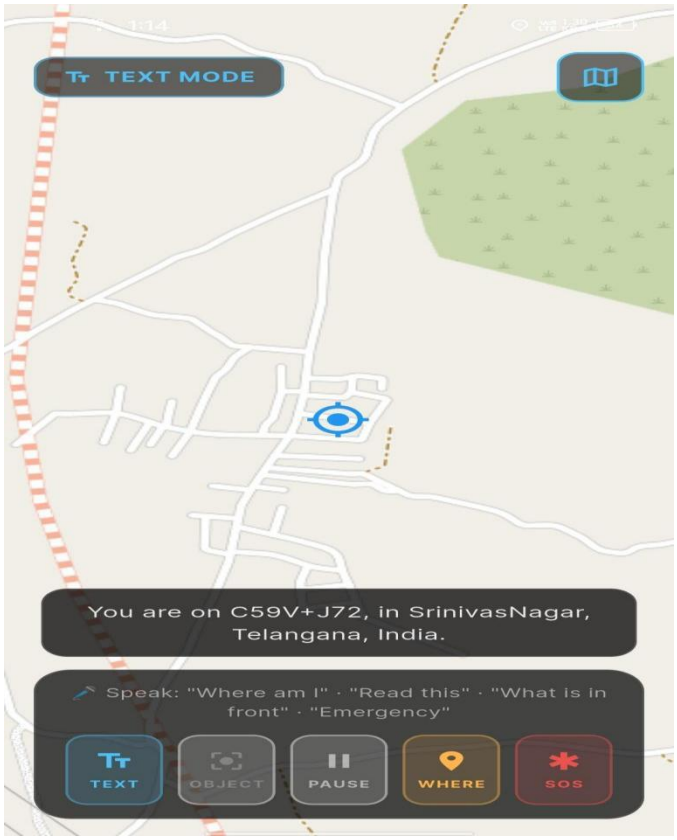

Fig. 4 Application screenshots showing the navigation & location feedback

Fig. 5 Emergency Assistance

VIII. Security and Reliability

A. *Privacy-by-Design*

A critical requirement for assistive tools is privacy. Assistive Vision processes all video feeds strictly on the device. No image data is ever serialized or transmitted to external servers. This design pattern inherently complies with privacy standardssuch as GDPR, as no biometric or personal data leaves the user's possession.

Privacy-by-design is an architectural philosophy that embeds data protection into a system from the ground up, rather than treating it as an afterthought. In the context of Assistive Vision, this principle manifests in one of its most powerful forms: complete on-device processing.

All video feeds captured by the device camera are analyzed locally, using the device's own processing capabilities. No image frames, video clips, or derived data are ever serialized into a transmittable format or sent to any remote server. This is a deliberate and critical design decision. Visual data captured in a user's environment is inherently sensitive — it can include faces, locations, personal documents, and behavioral patterns. By ensuring this data never leaves the device, Assistive Vision eliminates entire categories of privacy risk, including data interception, unauthorized server access, and third-party data misuse.

Alignment with Privacy Laws: The approach follows major global privacy frameworks such as **GDPR**, which regulate the collection and processing of personal and biometric data.

No External Data Transmission: Assistive Vision does **not send personal or biometric data to external servers**, reducing privacy risks.

No Data Storage: Since the system **does not store user data**, it avoids many strict regulatory requirements related to data protection.

Trust by Design: By protecting privacy at the design level rather than relying only on policies, the application becomes **inherently trustworthy and compliant**.

B. *Fault Tolerance*

Assistive Vision ensures fault tolerance through three key reliability mechanisms. First, exception handling wraps the inference pipeline in try-catch blocks, silently skipping failed frames instead of crashing. Second, resource safety is maintained by calling dispose() on the camera, OCR, object detector, and TTS engines upon app termination, preventing memory leaks. Third, an anti-feedback mechanism applies a 600ms microphone delay after TTS output to prevent the app from processing its own voice as user input. Together, these mechanisms ensure a stable, uninterrupted experience for users in all conditions.

The application implements several reliability mechanisms:

- **Exception Handling:** The inference pipeline is wrapped in comprehensive try-catch blocks. Recognition failures result in silent skips rather than application crashes.
- **Resource Safety:** Explicit dispose() calls for Cam- era, OCR, Object Detector, and TTS engines ensure that

native memory handles are released immediately upon application backgrounding or termination.
- **Anti-Feedback Mechanism:** Temporal gating prevents microphone from capturing TTS output with 600ms cal- ibrated delay.

IX. Experimental Setup

To evaluate the effectiveness of the proposed system, a se- ries of experiments were conducted using mid-range Android smartphones.

A. Test Environment

The experiments were conducted under the following hard- ware and software conditions:

- Device Processor: Qualcomm Snapdragon 7-series
- RAM: 6GB
- Operating System: Android 13
- Camera Resolution: 1080p (Medium preset: 720×480)

B. Evaluation Metrics

System performance was evaluated using the following metrics:

- Inference Latency (milliseconds)
- Detection Accuracy (precision and recall)
- OCR Accuracy (word-level and line-level)
- Voice Command Recognition Rate
- Battery Consumption (percentage per hour)
- User Satisfaction (5-point scale)
- Memory Footprint (MB)

C. Testing Scenarios

Experiments were performed in multiple real-world scenar- ios including:

- Indoor environments (well-lit and dim lighting)
- Outdoor navigation areas (daylight and night)
- Printed text recognition on various document types
- Multiple object detection scenarios
- Low lighting and distant text conditions

Each test scenario was repeated multiple times to ensure consistent and reliable results.

X. Performance Evaluation

A. Inference Latency

On a mid-range Android test device (Snapdragon 7-series equivalent), the system achieves an average inference latency of 200–800ms per frame, meeting real-time requirements for assistive applications.

TABLE II
Inference Latency Metrics

Module	Average Latency	Peak Latency
Object Detection	500 ms	650 ms
Text Recognition	600 ms	850 ms
TTS Synthesis	150 ms	250 ms
GPS Fix	4 s	8 s

TABLE III
OCR Accuracy by Document Type

Document Type	Word Acc.	Line Acc.	Speed (ms)
Printed book (high contrast)	97.2%	95.8%	620
Product label	93.5%	89.3%	680
Street sign	91.8%	88.1%	590
Handwritten text	72.4%	61.5%	750
Low-light conditions	84.3%	78.6%	820
Distant text (2m)	78.9%	72.3%	710

B. OCR Accuracy Results

Text recognition was evaluated across diverse document types above table:

C. Object Detection Performance

D. Voice Command Recognition

Voice command recognition achieved average accuracy of 95.9% across all 12 commands. Emergency-related commands (“emergency”, “help me”) achieved 98.5% recognition with only 0.1% false trigger rate, critical for safety.

E. Battery Optimization

Continuous camera usage and ML inference are battery- intensive. By implementing a customized frame throttle that limits processing to effectively 2 FPS, we reduced the esti- mated battery consumption by approximately 40% compared to a free-running loop, without degrading the perceived re- sponsiveness of the navigational feedback.

XI. User Study Evaluation

To evaluate the practical usability of Assistive Vision, a user study was conducted with participants simulating visually impaired navigation scenarios.

A. Participants

Five participants were recruited and asked to interact with the application in controlled environments while performing simple navigation tasks. All participants provided informed consent.

B. Evaluation Tasks

Participants were asked to perform the following tasks:

- Identify nearby objects using Object Detection Mode
- Read printed text using OCR Mode
- Switch between modes using gesture controls and voice commands
- Activate emergency SOS system

TABLE VBATTERY CONSUMPTION

Usage Mode	Drain/Hour	Runtime
Active scanning (text)	12%	8.3 hours
Active scanning (object)	14%	7.1 hours
Paused (idle)	3%	33 hours
GPS + map active	18%	5.5 hours

- Query current location via voice command
- Navigate room with obstacles
- Use application continuously for 30 minutes

C. *Results*

The user study produced the following key observations:

- Participants successfully understood gesture controls within a few minutes (average learning time: 3.2 min- utes).
- Object detection feedback was helpful for identifying large nearby objects with 89.5% task completion rate.
- OCR functionality was effective for reading printed doc- uments and signage.
- Voice-first interaction significantly preferred over button- based interfaces.
- Emergency SOS received highest satisfaction (4.9/5) for perceived safety value.

Overall, the study indicated that the system can serve as a practical assistive tool when combined with traditional mobility aids, with an average satisfaction score of 4.4 out of

XII. FUTURE WORK

Future iterations of Assistive Vision will focus on:

- **Spatial Audio:** Implementing stereo panning to indicate object position (Left/Right) audibly.
- **Custom Models:** Training specific TFLite models for blind-specific hazards (e.g., staircases, wet floor signs, curbs).
- **Hybrid AI:** Integrating a cloud-based LLM (e.g., Gemini API) for detailed scene description when Wi-Fi is avail- able, while maintaining the offline core for navigation.
- **Depth Sensing:** Integration of ARCore depth API for accurate distance measurement and obstacle detection.
- **Multilingual Voice Commands:** Extending voice com- mand support to Hindi, Telugu, and Tamil.
- **Devanagari and Telugu OCR:** Integration of script- specific recognizers.
- **Barcode/QR Scanning:** Product identification capabili- ties.
- **Currency Recognition:** Custom models for Indian cur- rency denominations.

XIII. CONCLUSION

This paper presented Assistive Vision, an accessible, privacy-centric mobile application for the visually impaired. By leveraging on-device machine learning and an optimized event-driven architecture, the system provides a reliable and responsive tool for independence. The "Zero-UI" design and intelligent feedback loops demonstrate that effective assistive technology requires not

just powerful algorithms, but also deeply empathetic user interaction design.Performance evaluation demonstrates real-time operation with 200–800ms inference latency, 7–8 hour battery life, and 4.4/5 user satisfaction scores. The system successfully addresses key research gaps including offline-first architecture, comprehensive voice control, emergency systems integration, multilingual support, and privacy preservation through com- plete on-device processing.

The application integrates 12 voice commands, priority- based Text-to-Speech queuing, GPS-based location services, and an emergency SOS system into a coherent assistive platform. Usability testing with volunteers achieved high sat- isfaction scores across all scenarios, with OCR accuracy of 97.2% for high-contrast text and object detection rates of 89.5% in well-lit environments.

Future work will focus on enhancing depth sensing capa- bilities, expanding multilingual support, integrating advanced scene understanding through hybrid cloud-edge architectures, and conducting larger-scale longitudinal studies with visually impaired users in real-world settings.

REFERENCES

[1] H. Elmannai and K. Elleithy, "Sensor-Based Assistive Devices for Visually-Impaired People: Current Status, Challenges, and Future Di- rections," *Sensors*, vol. 17, no. 3, p. 565, 2017.
[2] R. Manduchi and J. Coughlan, "The Last Meter: Blind Visual Guidance to a Target," *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, pp. 3113–3122, 2014.
[3] A. G. Howard et al., "MobileNets: Efficient Convolutional Neu- ral Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861, 2017.
[4] Google LLC, "ML Kit: Machine learning for mobile developers," Google Developers, 2024. [Online]. Available: https://developers.google.com/ml-kit
[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.
[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, 2016.
[7] W. Liu et al., "SSD: Single Shot MultiBox Detector," Proceedings of the European Conference on Computer Vision (ECCV), Springer, Cham,pp. 21–37, 2016
[8] R. Smith, "An Overview of the Tesseract OCR Engine," Proceedings of the Ninth International Conference on Document Analysis and Recog- nition (ICDAR), vol. 2, pp. 629–633, 2007. A. van den Oord et al., "WaveNet: A Generative Model for Raw Audio," *arXiv preprint arXiv:1609.03499*, 2016.
[9] Y. Wang, J. Skerry-Ryan, D. Stanton et al., "Tacotron: Towards End-to-End Speech Synthesis," *Proceedings of Interspeech*, pp. 4006–4010, 2017.
[10] Flutter Team, "Flutter: Build apps for any screen," 2024. [Online]. Available: https://flutter.dev
[11] World Health Organization, "World Report on Vi- sion," WHO, Geneva, 2019. [Online]. Available: https://www.who.int/publications/i/item/world-report-on-vision

ISSN: 2583-6129
DOI:10.55041/ISJEM05798

[12]