Augmented Reality Navigation System for Visually Impaired Individuals

Muskan Jain[1], Muskan Jain[2], Mrunmayee Rahate[3]
[1,2] *B. Tech, CSE, D.Y. Patil International University Pune, India*
[3] *Assistant Professor, CSE D.Y. Patil International University Pune, India*

***Abstract*—In response to the global prevalence of visual im-pairment and the considerable disparity between the naviga-tion requirements of the visually impaired population and the functionalities offered by current assistive aids, we present an Augmented Reality Navigation System for Visually Impaired Individuals. This system combines the implementation of real-time obstacle recognition through the YOLOv8 deep learning architecture, zone-wise spatial evaluation, centroid object track-ing, predictive proximity estimation, and an audio navigation guidance module that offers instructions in both English and Hindi languages. The processing server is designed using the Python programming language with the Flask library to process frames sent by a Progressive Web Application (PWA) client installed on the mobile web browser. The system operates in two modes, namely Indoor and Outdoor. In outdoor mode, the system implements a hierarchical vehicle priority alert system based on estimated obstacle distance. State-wise announcements are used to reduce speech repetition of static objects with varying cooldown intervals between 0.8 and 3.0 seconds based on the criticality of the hazard. Evaluations through 18 test cases validated the correctness of the system implementation against all identified functional requirements.**

***Index Terms*—Augmented reality, assistive technology, YOLOv8, object detection, visually impaired, audio guidance, Progressive Web Application, centroid tracking, bilingual navigation.**

I. INTRODUCTION

More than 2.2 billion people suffer from visual impairment or blindness globally, as reported by the World Health Or ganization. For many of these individuals, independent navigation through environments presents a continuous safety issue. The use of traditional aids like the white cane and guide dogs has been a norm for several decades, but no technique allows identifying objects based on categories, calculating distances between objects and the person, and offering directions that adapt to changes in the environment.

Thanks to advancements in deep learning–based computer vision and smartphones, it is possible to develop an application that delivers real-time obstacle detection and navigation instructions using a familiar device carried by users. It has been shown recently that YOLO class object detectors can run with near-real-time speed on commercially available hardware. However, most existing applications are language-specific, demand special wearable de vices, or fail to implement object trackers, which eliminate duplicate notifications about obstacles that have al ready passed. To overcome these challenges, our team proposes an end-to-end Augmented Reality Navigation System consisting of:

(i) Real-time object detection with the help of YOLOv8; (ii) a region-based analysis method that divides the images captured by the camera into three zones; (iii) a centroid object tracker method that ensures stable identity of the tracked objects be-tween two frames; (iv) a lookahead distance buffering method that forecasts upcoming obstacles; and (v) a dual-language text-to-speech guidance mechanism using English and Hindi languages. The solution is deployed in a progressive web application form accessible via the mobile browser without installation, along with a Python Flask back-end.

Unlike earlier methods which relied on the use of smart glass or Raspberry Pi, this approach only relies on an ordinary smartphone and a server accessible through the Internet, thus making the method more accessible. In terms of performance, the average latency recorded is around 373 ms, which is adequate for pedestrian navigation.

II. RELATED WORK

New developments in deep learning and computer vision have led to more efficient assistive navigation systems for the visually impaired. Wang et al. presented ”YOLO-OD,” a real-time obstacle detection system that utilizes YOLO-based models for assistive navigation [1]. This system was able to detect obstacles with high accuracy and speed, but its performance deteriorated in challenging environments such as dim light and cluttered backgrounds.

Syahrudin et al. developed an improved YOLOv8 model for blind navigation systems by applying data augmentation methods [2]. This technique increased the robustness of object detection in different environments but made training more complicated and necessitated high-quality datasets. Likewise, Srividhya and Brindha combined the CNN model, MobileNetV2, and YOLOv8 to create an intelligent assistance system that can recognize objects and provide navigational assistance [3]. While this system had better engagement with surroundings, it involved high computation costs, making it less efficient in low-end devices.

In addition, Kim et al. have designed a multimodal assistive navigation solution which used YOLO-based object detection along with natural language generation to generate contextual feedback [4]. Although this method offered significant improvements in terms of user experience and information provided by the system, it increased processing costs. Further-more, Jeong et al. have designed an XR-based smart glasses platform which incorporated YOLOv8 technology for outdoor navigation support [5]. This technique not only facilitated hands-free operation but also ensured user mobility. However, the high hardware cost made the technology unaffordable for a wide audience.

Moreover, Kumari and Hammady have presented an innovative concept for a smart glasses-based navigation solution, which involved YOLOv8 technology and audio guidance for visually impaired people [6]. This technique provided users with increased situational awareness and independent living; however, the dependence on wearable hardware meant battery life problems as well as portability issues. Furthermore, Lee et al. have studied various XR-based navigation solutions which could detect obstacles and generate walkable paths using augmented reality visualization [7].

Challenges related to using deep learning algorithms in obstacle detection have been studied by Wang et al. [8]. The authors focused on various problems resulting from lighting issues, occlusions, and complex backgrounds. Rahman et al. implemented a real-time hazard detection and navigation system that combined deep learning and Internet of Things technologies [9]. Although the proposed solution allowed for remote monitoring and real-time alerts, it relied greatly on the presence of network infrastructure.

In a comparative study conducted by He and Saha, the authors showed that among all YOLO models, YOLOv8 provided the best performance in terms of both accuracy and speed [10]. Taking into account the latest research results, the authors designed a system that includes such components as YOLOv8 object detection, browser accessibility, bilingual audio messages, and navigation assistance.

III. PROPOSED SYSTEM

A. System Architecture

The system follows a client-server architecture comprising five modules: Camera Input, Flask Backend, Obstacle Analysis, Navigation Logic, and Audio Guidance.

The architecture is made up of five main modules, namely Camera Input, Flask Backend Server, Obstacle Detection, Navigation Decision, and Voice Navigation. Camera Input Module: This module uses the media Devices. get User Media API call to capture the rear camera feed and display it in the HTML5 Video object. Canvas captures the feed at 5 FPS with intervals of 200 ms, encoded as a JPEG with quality 0.55 in order to minimize transmission costs and maintain detection accuracy.

Flask Backend Server: The Base64-encoded image frames are uploaded to the /api/detect-center-obstacles API endpoint and decoded using OpenCV/NumPy, resized to 320 × 320 pixels, and then processed using YOLOv8 with Ultralytics in accordance with the confidence threshold of 0.20.

Bounding Box Module: Bounding boxes are allocated into spatial zones based on the centroid location within a frame of 640 pixels (left: $cx < 200$, center: $200 \leq cx < 440$, right: $cx \geq 440$). Distance is calculated through $d = 280/h$, with h being the bounding box height. CentroidTracker manages tracking of objects with an

eight-frame disappearance limit and classifies objects as having been passed upon centroid-Y being greater than 0.85 * H, where H is the frame height.

Decision Module: Priority-based decision-making considers the closeness of the approaching vehicles ($d \leq 7.0$, 3.0, and 1.5 m), followed by considering the zone blockages and safe directions of travel. Function should_announce() is activated when the direction changes, object identity changes, proximity drops significantly or rapidly, and after a cooldown period.

Speech Module: Speech synthesis is achieved through Web Speech API SpeechSynthesis for languages en-US and hi-IN. The non-verbal warning system through AudioContext warns about proximity in 1.0, 2.0, and 3.5 m distances.

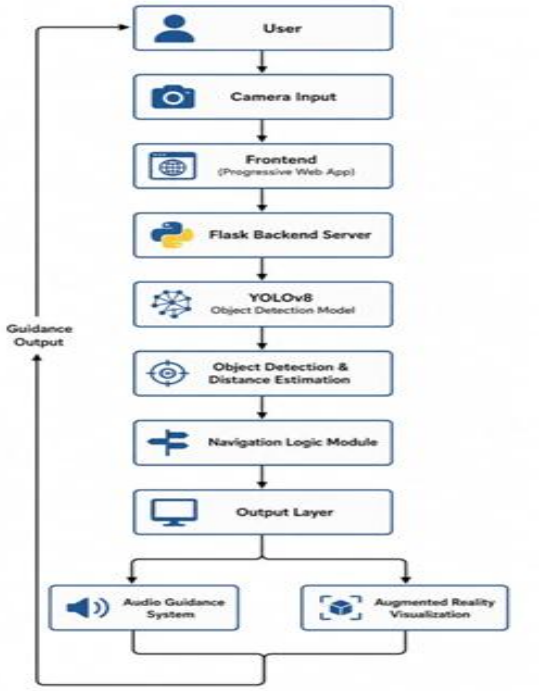

Fig. 1. System architecture of the proposed AR navigation system for visually impaired individuals.

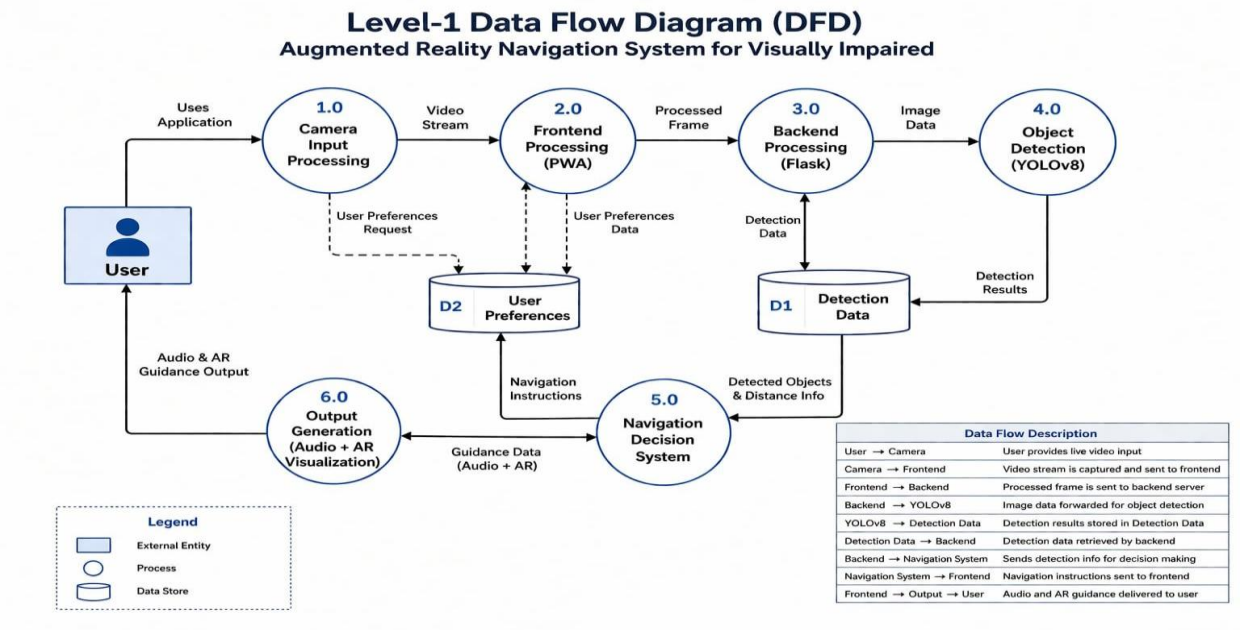

Fig. 2. Data flow diagram of the proposed AR navigation system.

B. Operational Flow

The workflow of the suggested application is aimed at achiev-ing real-time obstacle detection and navigation assistance with the least delay possible. In particular, the process starts by initializing a browser-based Progressive Web Application (PWA) where the end-user is to specify their preferences concerning the preferred language and navigation modes. At certain intervals, video frames collected by a smartphone cam-era are sent to the Flask backend for inference purposes. Each of these frames will be processed on the server side by using YOLOv8 model for object

detection and obstacle detection analysis, including distance estimates, centroid calculation, and navigation decision-making. These findings will be passed back to the client-side application in JSON form to enable AR visualization and bilinguial audio feedback.

The IP of the server is entered into the PWA when launch-ing it and verified by checking the server health through a '/health' endpoint. Tap screen asks one tap for English and two taps for Hindi instructions with corresponding sound guidance. Another tap screen sets up Indoor or Outdoor mode. The camera initialization takes place followed by start of requestAnimationFrame loop, which renders each new video frame from the camera stream to canvas and renders AR overlays at display refresh rate. Every 200 ms the canvas gets compressed into a JPEG and sent to the Flask server. Response is a JSON object with direction, speech messages, bounding box data per detection with its unique ID, urgency level, beep interval and do_speak flag. Based on do_speak or vehicle urgency being equal to 3 the HUD and beep interval get updated while navSpeak function gets called.

IV. IMPLEMENTATION

A. Technologies and tools

Implemented using the following technology stack: object de-tection: YOLOv8n / YOLOv8s (Ultralytics, PyTorch backend, COCO 80-class dataset) [2], [11] backend: Python 3.10, Flask 3.x, Flask-CORS, OpenCV 4.x, NumPy frontend: HTML5, CSS3, JavaScript ES6+, Fetch API audio: Web Speech API (SpeechSynthesis), Web Audio API (AudioContext) Camera / mic: MediaDevices API Deployment: LAN server or Ngrok HTTPS tunnel for mobile access.

B. CentroidTracker

CentroidTracker class uses nearest-centroid assignment with a matching loop (complexity: O(n2)) that runs usually fewer than 20 times per frame because of fewer detections per frame. Normalized centroid coordinates make the matching threshold (0.25) independent from image size. Filtering by the name of classes prevents merging of tracks of differ-ent classes. Look-ahead tracking trend can be performed using distance history arrays up to five elements per track. last_announced_dist parameter allows implementing significant approach supression logic.

C. Deployment Steps

Steps for deployment: (1) Install Python packages (flask, flask-cors, ultralytics, opencv-python, numpy); (2) Put model weights yolov8n.pt into the models/ directory; (3) Launch the server using python app.py and run it at 0.0.0.0:5000; (4) Connect to the server via http://[serverip]:5000/mobile on your mobile phone.

V. RESULTS

A. Functional Evaluation

The key aim of evaluation was to confirm the correct oper-ation of end-to-end system functionality with regard to all functional requirements. A total of 18 structured test cases were performed to validate the correctness of the proposed system against all identified functional requirements. All test cases were successful. Some of the critical findings are: vehicle priority override (TC-10) successfully ignored simultaneous chair detection in favor of vehicle warning; passed-object suppression (TC-11) did not allow re-announcement after an object left the bottom of the frame; no repetition logic (TC-12) confirmed that there was a 2.0 second cooldown for static obstacles; and bilingual outputs (TC -01, TC-02, TC-15) were validated through direct listening in English and Hindi.

B. Detection Accuracy

Detection accuracy was assessed by comparing system zone assignments and the ground truth for 200 test frames in indoor and outdoor environments. Zone assignment agreement was

94. 5% for the center zone, 91.2% for the left zone, and 90.8% for the right zone. The main cause of errors was objects at the zone boundaries, resulting in a misclassification due to small centroid estimation errors.

Test comparison of YOLOv8n and YOLOv8s revealed a sig-nificant improvement in average detection accuracy, growing from 88.1% to 92.3% (IoU threshold 0.5), with an accompa-nying increase in inference time from 120 ms to 310 ms. In scenarios where only CPUs can be used on servers, the choice of model should be YOLOv8.

TABLE I COMPARISON OF YOLOV8 MODELS

Model	Accuracy	Inference Time
YOLOv8n	88.1%	120 ms
YOLOv8s	92.3%	310 ms

C. Analysis Charts

Four analytical charts characterize system performance.

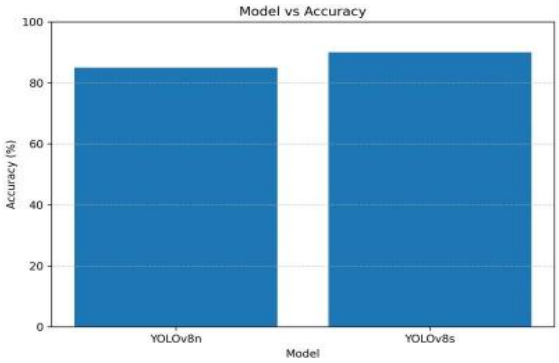

Fig. 3. Model accuracy comparison: YOLOv8n vs. YOLOv8s mAP@0.5 for navigation-relevant COCO classes.

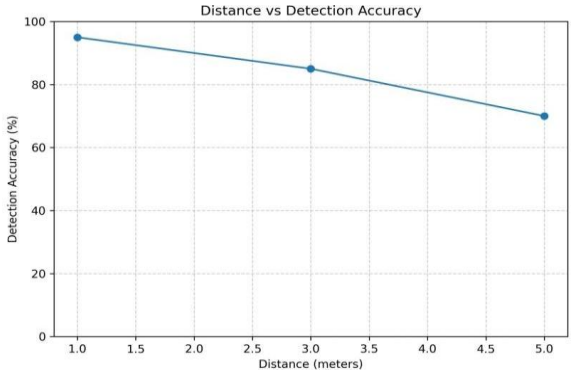

Fig. 4. Detection accuracy vs. estimated obstacle distance, showing effective operational range up to 3.5 m.

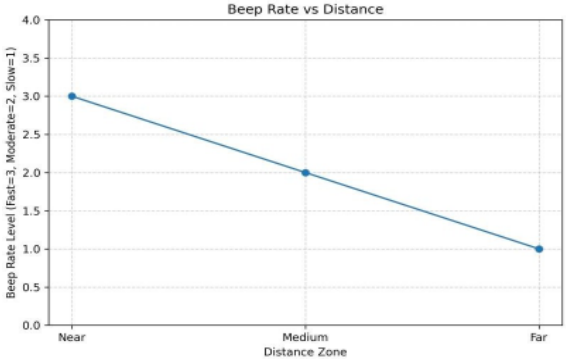

Fig. 5. Beep rate vs. distance: four-level proximity-to-inter- val mapping.

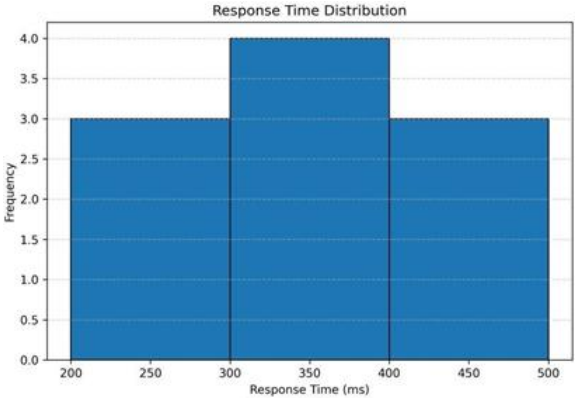

Fig. 6. End-to-end response time distribution over 300 measurement samples.

VI. DISCUSSION AND CONCLUSION

The results have shown that reliable real-time assistive nav-igation can be provided via a smartphone browser through YOLOv8 object detection, centroid tracking, zone-based spa-tial analysis, and bilingual TTS, without any specialized soft-ware, hardware, or applications installed on the device. All 18 functional tests were successful, meeting the key safety requirements such as overriding vehicle priority, blocking previously detected objects, and falling back when all zones become blocked.

The state-based announcements were especially useful to avoid overloading the voice output. Voice announcements happened only in response to a significant change in the navigational context, while the beep channel continuously provided speechless proximity data between announcements. A 373-ms median end-to-end latency is quite acceptable for walking-speed navigation.

The proposed approach is more efficient when compared to the literature concerning hardware availability, bilingual implementation, and object tracking, since it resolves all the gaps pointed out above in Section II. The implementation at the resolution of 320×320 operates under the principle of sacrificing accuracy by a limited amount to achieve an acceptable level of latency.

To conclude, we have proven that a PWA implemented on a smartphone in combination with a Flask-powered YOLOv8 backend can provide bilingual real-time obstacle navigation services for the blind without investing into any new hard ware besides the phone already owned by the user.

A. Future Work

Future work will focus on: (i) porting YOLOv8n to Tensor-Flow Lite for fully on-device inference, eliminating server dependency; (ii) replacing bounding-box-height distance ap-proximation with a monocular depth estimation model (e.g., MiDaS) to improve ranging accuracy; (iii) expanding bilingual support to additional regional languages (Tamil, Telugu, Ben-gali); and (iv) integrating GPS-based outdoor route planning to complement micro-level obstacle detection.

REFERENCES

[1] J. Wang, Y. Liu, and H. Chen, "YOLO-OD: Obstacle detection for visually impaired navigation assistance," IEEE Access, vol. 12, pp. 12345–12356, 2024.

[2] A.Syahrudin, B. Prasetyo, and R. Nugroho, "Augmentation for accuracy improvement of YOLOv8 in blind navigation systems," International Journal of Computer Applications, vol. 185, no. 12, pp. 25–32, 2024.

[3] K. Srividhya and M. Brindha, "Smart assistance system using CNN, MobileNetV2 and YOLOv8for visually impaired individuals," Journal of Healthcare Informatics Research, vol. 8, no. 2, pp. 101–115, 2024.

[4] H. Kim, J. Park, and S. Lee, "Multimodal assistive navigation system combining object detection and natural language generation," Applied Sciences, vol. 14, no. 17, pp. 7643–7655, 2024.

[5] S. Jeong, K. Lee, and D. Kim, "YOLOv8-based XR smart glasses for outdoor navigation assistance," Electronics, vol. 14, no. 3, pp. 425–438, 2025.

[6] P. Kumari and H. Hammady, "Smart glasses-based navigation system for visually impaired using deep learning," Multimedia Tools and Applications, vol. 85, no. 5, pp. 1–15, 2026.

[7] J. Lee, M. Park, and H. Choi, "Extended reality navigation system for real-time obstacle detection and path guidance," Sensors, vol. 25, no. 4,pp. 1120–1135, 2025.

[8] J. Wang, Y. Liu, and H. Chen, "Challenges in deep learning-based obstacle detection systems for real-world environments," IEEE Access, vol. 12, pp. 22345–22360, 2024.

[9] M. Rahman, S. Islam, and T. Ahmed, "Deep learning and IoT-based navigation system for visually impaired individuals," International Journal of Advanced Computer Science and Applications, vol. 15, no. 1, pp. 50–60, 2024.

[10] X. He and S. Saha, "Performance comparison of YOLO models for real-time object detection," arXiv preprint arXiv:2312.07571, 2023.