IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 28, Issue 6, Ser. I (Nov. – Dec. 2026), PP 07-13
www.iosrjournals.org

AI Based Smart Navigation System for Visually Impaired Individuals

K Vinutha, Lakshmi Priya K S, Sahana, Shilpa Sannamani
Department of Artificial Intelligence & Data Science
K. S. School of Engineering and Management, Bengaluru, India

Abstract-*Visual impairment can make independent travel difficult because pedestrians, vehicles, uneven surfaces, and other obstacles may not be identified early enough. This paper presents a proposed AI-based smart navigation system intended to support visually impaired individuals through camera-based scene understanding and spoken guidance. The proposed workflow captures frames from a camera, preprocesses the images, applies a YOLO-based object detector, estimates the relative position of detected objects, and converts selected detections into concise text-to-speech alerts. The design is organized as modular stages so that the detection model, position-estimation logic, and speech component can be developed and improved independently. The proposed prototype can first be evaluated using recorded video and then tested with a live camera in controlled conditions. The paper also describes an implementation plan and an evaluation framework based on detection quality, end-to-end response time, and alert usability. As the project has not yet been implemented or field-tested, no measured accuracy, latency, or user-study results are claimed. The system is intended to complement, not replace, a white cane, guide dog, or established mobility practices.*

Keywords-*Assistive Technology, Visually Impaired, Computer Vision, YOLO, Object Detection, Voice Guidance, Text-to-Speech.*

Date of Submission: 02-11-2026 Date of Acceptance: 12-11-2026

I. INTRODUCTION

Safe and independent mobility is a daily challenge for many people with visual impairments. A white cane is valuable for detecting some nearby obstacles, but it does not describe the wider scene or identify objects such as vehicles, signboards, or people. Computer vision offers a way to interpret camera images and communicate selected information through audio. This project proposes an integrated navigation assistant that captures visual information, detects relevant objects, estimates their position in the scene, and provides brief spoken alerts. The intention is to provide concise environmental cues through an accessible software pipeline that may later be deployed on a smartphone or lightweight wearable device. The proposed system is supplementary assistance and is not intended to replace established mobility aids or training.

II. RELATED WORK

Prior research has explored computer-vision navigation, wearable cameras, optical character recognition (OCR), distance estimation, and audio feedback for assistive mobility. NaVIP investigates camera-based indoor navigation and relocalization without additional infrastructure [1]. Research on dynamic crosswalk scene understanding combines object recognition and scene cues to support safer road crossing [2]. Low-cost assistive devices have integrated cameras, OCR, and speech output in wearable designs [3]. Other systems have investigated voice-guided navigation and sensor feedback in controlled indoor settings [4]. Pothole detection using YOLO-based computer vision has also been explored to warn users about road hazards [5]. Deep-learning virtual-assistance frameworks combine object detection, position estimation, and scene descriptions to provide contextual information [6].

These studies demonstrate the value of combining perception with accessible feedback, while highlighting challenges such as changing illumination, distance-estimation errors, limited computing resources, and the need for real-world evaluation. The proposed project focuses on a clear detection-to-audio pipeline: detect supported objects, estimate their coarse horizontal position, and announce selected information using short speech messages. More advanced features, including dependable distance measurement and OCR, are treated as possible extensions rather than completed functions.

These studies demonstrate the value of combining perception with accessible feedback, while also highlighting challenges such as changing illumination, distance-estimation errors, compute limitations, and the need for real-world evaluation. The proposed project therefore prioritizes a clear detection-to-audio pipeline and

treats additional functions as modular extensions.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system is organized into five principal stages: camera input, frame preprocessing, object detection, relative-position and alert selection, and voice-alert generation. A camera supplies frames from the user's surroundings. The preprocessing stage converts each frame into the size and format expected by the selected YOLO model. The detector returns object labels, confidence scores, and bounding boxes for supported classes. The position module uses the horizontal location of each bounding box to assign a coarse left, centre, or right region. An alert-selection stage filters detections and creates a short message, and a text-to-speech component converts that message into audio. The architecture is modular, allowing the detector, position logic, and speech engine to be optimized separately. The exact computing platform and deployment configuration will be finalized during implementation.

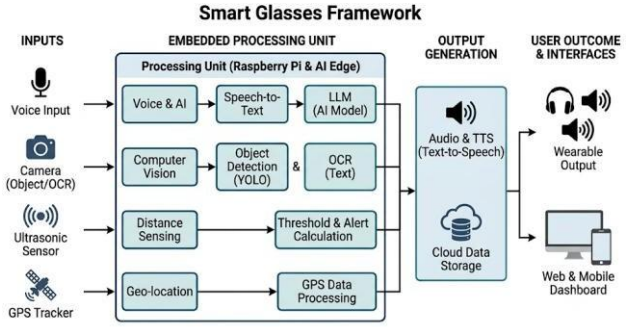


Fig. 1. Proposed processing pipeline for camera-based object detection and voice guidance.

The processing pipeline is designed to support repeatable development. Each stage can be checked independently before integration: camera frames can be saved for inspection, detector outputs can be compared with labelled examples, position assignments can be checked against bounding-box coordinates, and generated alert text can be tested with the speech engine. This staged approach helps isolate errors and reduces the difficulty of debugging the complete application.

The following depicts an end-to-end latency threshold ($\leq$ 300 ms) to make the wearable safe for real-time walking speeds (1.2 m/s):

a. Frame Capture & Preprocessing: 20 ms
b. Model Inference (NPU/CPU): 80 ms
c. Spatial Mapping & Logic: 10 ms
d. Audio Generation (TTS): 150 ms

IV. PROPOSED METHODOLOGY

A. Data Acquisition and Preprocessing

A camera captures live frames or sample video from the surrounding environment. During initial development, recorded clips are useful because the same scenes can be replayed while tuning and debugging the pipeline. Frames are resized and formatted according to the selected detection model's input requirements. Where supported by the implementation library, image conversion and normalization are performed consistently with the model's expected input. The system should handle unavailable camera input and frame-read errors without terminating unexpectedly. The test set should include indoor and outdoor scenes, different lighting conditions, partially occluded objects, and scenes with multiple objects.

$$\mathbf{I}_{\text{out}} = \text{Resize}\left(\frac{\mathbf{I}_{\text{in}} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, (640,640)\right)$$

Where:

$\mathbf{I}_{\text{in}} \in \mathbb{R}^{H\times W\times 3}$ is the incoming RGB frame.

$\boldsymbol{\mu} = [0.485,0.456,0.406]$ and $\boldsymbol{\sigma} = [0.229,0.224,0.225]$ are ImageNet mean and standard deviation normalization parameters.

B. Object Detection

A pretrained YOLO-family model is proposed for detecting common supported classes such as people, cars, bicycles, and other objects available in the selected model. For each frame, the detector produces class labels, confidence scores, and bounding boxes. A confidence threshold will be selected and tuned using validation examples rather than chosen solely by visual inspection. If the pretrained classes do not adequately represent hazards relevant to the intended use, fine-tuning or a separate hazard detector may be considered using a suitable labelled dataset. The initial implementation should record the model version, class list, input size, confidence threshold, and inference environment so that evaluation is repeatable.

$$\mathrm{IoU}(B_1, B_2) = \frac{\mathrm{Area}(B_1 \cap B_2)}{\mathrm{Area}(B_1 \cup B_2)}$$

Confidence Filtering: A detection vector $d_i = (x_1, y_1, x_2, y_2, c_i, p_i)$ is retained only if $c_i \cdot p_i \geq \theta_{\text{conf}}$, where $\theta_{\text{conf}} = 0.50$.

NMS Metric: Overlapping bounding boxes for identical classes are pruned using Intersection over Union (IoU ≥ 0.45).

C. Relative Position Estimation

For each detected bounding box, the horizontal centre can be compared with the frame width. Let the frame width be W and the bounding-box horizontal limits be x1 and x2; the box centre is calculated as (x1 + x2) / 2. The frame can then be divided into three horizontal regions. A centre in the left region produces a left-side cue, a centre in the middle region produces a centre/ahead cue, and a centre in the right region produces a right-side cue. The region boundaries must be documented and adjusted during testing. This method provides only a coarse directional cue: image position does not provide a reliable physical distance and should not be described as such. Distance estimation requires an appropriate depth sensor, a calibrated camera method, or a validated monocular-estimation approach.

Spatial regions are assigned using boundaries defined by normalized horizontal splits ($\alpha_1 = 0.33, \alpha_2 = 0.66$):

$$\text{Region}(x_c) = \begin{cases} \text{Left}, & 0 \leq x_c < \alpha_1 W \\ \text{Center}, & \alpha_1 W \leq x_c \leq \alpha_2 W \\ \text{Right}, & \alpha_2 W < x_c \leq W \end{cases}$$

To add technical rigor to distance estimation without extra depth sensors, incorporate pinhole camera focal length geometry:

$$D = \frac{f \cdot H_{\text{real}}}{h_{\text{bbox}}}$$

D. Alert Selection and Speech

The alert module converts selected object labels and their relative regions into short messages, for example, “Person ahead” or “Obstacle on your left.” Only supported detections above the chosen confidence threshold should be considered for announcements. Repeated detections of the same object across adjacent frames may produce excessive speech, so the implementation should use a cooldown period, duplicate-message suppression, or a priority rule. The speech component converts the selected text into audio using the chosen text-to-speech library or operating-system service. Message length, volume, speech rate, and alert timing should be evaluated for clarity and distraction. Any priority scheme should be treated as provisional until tested in controlled conditions.

$$\text{Score}(i) = w_1 \cdot \left(\frac{1}{D_i}\right) + w_2 \cdot C_i + w_3 \cdot S_{\text{center}}$$

Where:

D_i is object distance.

C_i is detection confidence score.

$S_{\text{center}} = 1$ if the object is in the Center lane, else 0.5.

E. Integration and Safety

The software modules will be connected into a single application and tested first with recorded inputs, then with a live camera in controlled environments. Integration should verify that frames reach the detector, detections are converted into position labels, and selected alerts are spoken without unnecessary repetition. Error handling should cover missing camera input, model-loading failures, and speech-engine errors. The prototype is intended as supplementary assistance; it must not be presented as a replacement for a white cane, guide dog, or professional mobility training. Field use should only be considered after appropriate technical and usability testing.

V. IMPLEMENTATION PLAN AND EVALUATION

A. Software Modules and Data Flow

The planned software implementation can be divided into independent modules: (1) camera acquisition, responsible for reading live frames or video files; (2) preprocessing, responsible for preparing frames for model input; (3) YOLO inference, responsible for returning object classes, confidence values, and bounding boxes; (4) position estimation, responsible for mapping bounding-box centres to left, centre, or right regions; (5) alert generation, responsible for creating short messages and suppressing duplicates; and (6) text-to-speech output, responsible for playing the selected message. A main control loop will coordinate these modules and handle camera, model, and speech errors.

B. Proposed Development Sequence

Development should begin by confirming the camera and software environment, then running the selected pretrained YOLO model on recorded sample frames. The next step is to inspect bounding boxes and labels, implement the horizontal-region calculation, and test message generation separately from speech playback. The speech engine can then be connected to the alert module. After each module works independently, the full pipeline should be integrated and tested with recorded clips before moving to a live camera in a controlled setting. The final implementation record should identify the programming language and library versions, model file, input resolution, confidence threshold, hardware used, and configuration changes.

C. Planned Evaluation Measures

The evaluation should report detection precision, recall, and mAP when annotated test data is available. Runtime performance should be measured over a defined set of frames and summarized using average response time. Functional checks should record whether the system correctly produces left, centre, and right cues for selected bounding-box locations and whether repeated detections are handled appropriately. Results should include the number and type of test scenes, test conditions, and known failure cases. Until these tests are conducted, values for accuracy, speed, and usability remain unknown and should not be estimated or presented as achieved results.

The figures in this paper illustrate the intended system arrangement and possible hardware configuration. They are design references, not evidence of a completed implementation. The proposed implementation will be assessed after the software pipeline is built and tested. Until then, the figures and example messages should be interpreted as illustrative outputs rather than measured system results.

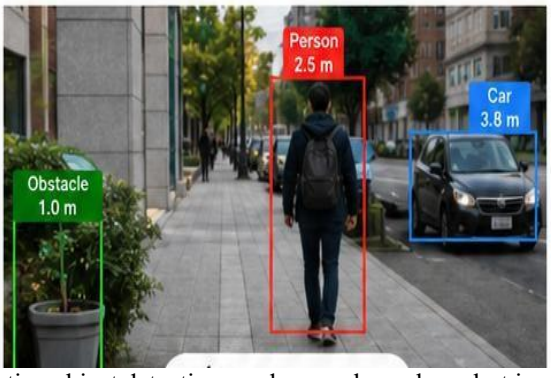


Fig. 2. Illustrative object detections and example spoken alert in a street scene.

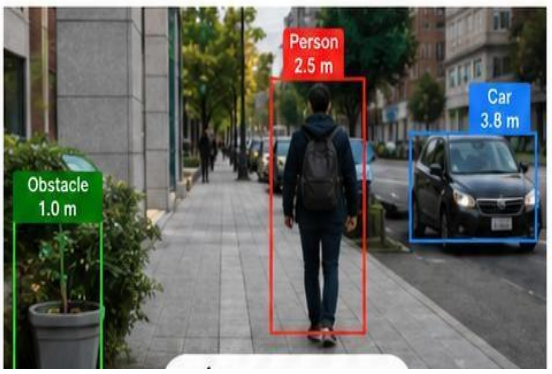


Fig. 3. Illustrative voice-guidance message; this is not a measured system result.

After implementation, evaluation should use a labelled set of indoor and outdoor scenes. Where ground-truth annotations are available, object detection can be evaluated using precision, recall, and mean average precision (mAP). Precision measures the proportion of predicted detections that are correct, while recall measures the proportion of relevant objects that are detected. End-to-end response time should be measured from frame capture to audible alert, with the test hardware and software configuration recorded. Testing should include different lighting conditions, crowded scenes, partially occluded objects, multiple simultaneous detections, and repeated detections across frames. Position-estimation checks should compare the assigned left, centre, or right region with the bounding-box location. Usability assessment should consider whether messages are clear, timely, understandable, and distracting.

Mean Average Precision (mAP@0.5):

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N}\int_{0}^{1} P_i\,(R)\,dR$$

Where P represents Precision, R represents Recall, and N is the number of target classes.

Frame Processing Throughput:

$$\text{FPS} = \frac{1}{T_{\text{preprocess}} + T_{\text{inference}} + T_{\text{postprocess}}}$$



Fig. 4. Camera module proposed for capturing the user's surroundings.



Fig. 5. Glasses frame with a camera mounting bracket



Fig. 6. USB adapter shown as a possible connection accessory; compatibility must be confirmed.



Fig. 7. Wired earphones that may be used for spoken guidance (illustrative audio component).



Fig. 8. USB cable shown as a possible connection accessory; required connectors depend on the selected hardware.



Fig. 9. Example of a camera mounted on glasses; this illustrates a possible arrangement, not a tested prototype.

VI. LIMITATIONS AND FUTURE ENHANCEMENTS

The proposed approach depends on camera visibility, the object classes supported by the selected model, available computing resources, and the reliability of detections. Low light, motion blur, occlusion, and crowded scenes may reduce detection quality. A bounding box's image position provides only a rough directional cue and must not be treated as a verified distance measurement. Frequent spoken alerts may also distract users or obscure environmental sounds. The system's practical usefulness and safety therefore require controlled technical evaluation and user-centred assessment before deployment.

Future work may include validated distance estimation, OCR for reading signs and printed text, voice commands, emergency SOS functionality, and integration with wearable glasses. Additional hazard classes may be incorporated if suitable data and a validated model are available. A mobile or wearable deployment may be explored after the core software has been evaluated.

VII. CONCLUSION

This paper has presented a proposed AI-based smart navigation system for visually impaired individuals using camera input, YOLO-based object detection, coarse relative-position estimation, and text-to-speech guidance. The design aims to provide concise environmental cues through a modular and potentially low-cost platform. The next stage is to implement the pipeline, verify the interaction between its modules, and evaluate detection quality, response time, and alert usability under controlled conditions. Since implementation and testing have not yet been completed, the paper does not claim measured accuracy or real-world effectiveness. The system is intended to supplement existing mobility aids rather than replace them.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to K S School of Engineering and Management and the Department of Artificial Intelligence and Data Science for their continuous support, encouragement, and for providing the necessary resources to carry out this work.

REFERENCES

[1] J. Yu, Y. Yang, and V. Namboodiri, "NaVIP: A Computer Vision-Based Indoor Navigation Solution for Visually Impaired Persons," arXiv preprint arXiv:2410.18109, 2024.doi: 10.48550/arXiv.2410.18109.Available: https://arxiv.org/abs/2410.18109.

[2] S. Tian, M. Zheng, W. Zou, X. Li, and L. Zhang, "Dynamic Crosswalk Scene Understanding for the Visually Impaired," IEEE Access, 2022. Full publication details and DOI should be verified against the original paper before submission.

[3] R. Král, P. Jacko, and T. Vince, "Low-Cost Multifunctional Assistive Device for Visually Impaired Individuals," 2023. Full publication details should be verified against the original paper before submission.

[4] E. Malaekah et al., "Sound-based Navigation System for Visually Impaired Individuals," 2026. Full publication details should be verified against the original paper before submission.

[5] A. Paramarthalingam, J. Sivaraman, P. Theerthagiri, B. Vijayakumar, and V. Baskaran, "A Deep Learning Model to Assist Visually Impaired in Pothole Detection Using Computer Vision," Decision Analytics Journal, vol. 12, Art. no. 100507, 2024. doi: 10.1016/j.dajour.2024.100507.

[6] F. Ahmad, M. Z. Ansari, M. Tayyab, S. Shuja, and M. Zaki, "Virtual Assistance System for Visually Impaired People Using Deep Learning," Procedia Computer Science, vol. 258, pp. 67–75, 2025. doi: 10.1016/j.procs.2025.04.196.