

VisionAI: Giving Sight Through Sound – An AI-Powered Real-Time Assistive Navigation System for the Visually Impaired

M.D. MANASVINI¹, AYUSH SURYAWANSHI¹, SHREYAS SRIDHAR²

¹ Department of Computer Science and Engineering, Sir M. Visvesvaraya Institute of Technology, Bangalore, Karnataka, India-562157

² Department of Data Science and Engineering, Sir M. Visvesvaraya Institute of Technology, Bangalore, Karnataka, India-562157

Abstract - Getting around safely is a daily difficulty for people with visual impairment, especially on crowded and uneven streets. This paper describes VisionAI, a wearable navigation aid that runs entirely on a small embedded board. A Sony IMX219 camera fixed to a pair of smart glasses sends video to an NVIDIA Jetson Nano, where a YOLOv8n (You Only Look Once, version 8, nano) detector optimised with TensorRT in half precision (FP16) finds obstacles and OpenCV handles the frames. The distance to each object is estimated from the height of its bounding box. A walking stick with an ultrasonic sensor and an ESP32 board covers the short range and the ground level. A logic controller combines both sources, lets the ultrasonic reading take priority inside 1 m, and turns the result into short spoken directions through bone-conduction earphones and vibration on the stick. A Global Positioning System (GPS) receiver, a Global System for Mobile Communications (GSM) module and an SOS button send the user's position to a chosen contact. The detector was trained on 12,500 frames of Indian street scenes recorded at eye level and reaches a mean average precision (mAP) of 0.875 at an intersection-over-union threshold of 0.5 over four object groups. Ground hazards such as potholes remain the hardest group (0.81).

Key Words: Artificial intelligence, assistive technology, computer vision, edge computing, Jetson Nano, object detection, sensor fusion, YOLOv8

1. INTRODUCTION

For a person with visual impairment, a walk along a busy street means dealing with vehicles, animals, open drains and uneven ground without seeing any of them (Fig. 1) [1]. White canes and guide dogs help, but neither can describe what is around the user or warn about something a few metres ahead [2].

Object detectors such as You Only Look Once (YOLO) [3] now run fast enough for live video, and several assistive devices already use them. Most of these devices stop at naming or flagging an object. They do not decide which object matters most, say where it is in simple terms, or keep working when the camera view is poor. Many also depend on a laptop, a phone or a network link, which makes them harder to carry and to afford.

VisionAI addresses these points with a self-contained wearable. The paper makes four contributions. First, detection and distance estimation run on a Jetson Nano attached to the glasses, so no laptop or cloud service is needed. Second, an ultrasonic walking stick works as a separate short-range channel, and its reading overrides the camera inside 1 m. Third, detections are ranked by class and distance and spoken as clock-position phrases. Fourth, we describe a dataset of Indian street scenes captured at eye level, with classes such as auto-rickshaws, stray animals, potholes and speed breakers. A Global Positioning System (GPS) and Global System for Mobile Communications (GSM) safety module with an SOS button completes the system.

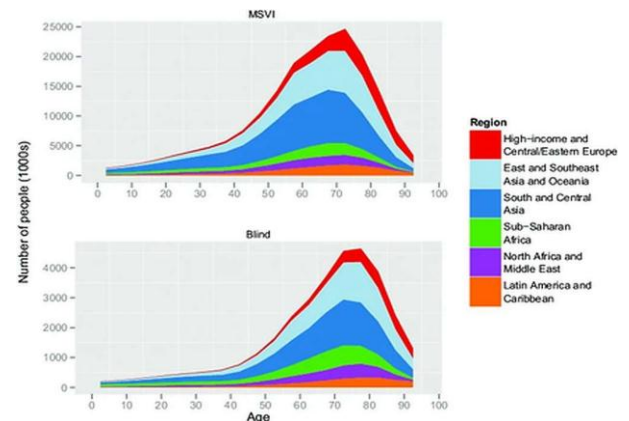


Fig. 1. Number of blind people and people with moderate or severe vision impairment (MSVI), by region and age. Adapted from [1].

2. RELATED WORK

2.1 Sensor-Based Assistive Systems

Ultrasonic smart canes are the most common electronic travel aid [2]. They measure the distance to the nearest obstacle and warn the user by vibration or sound, and some add GPS and GSM units for tracking and emergency calls. They are cheap and easy to build, but they only sense a short range and cannot tell what the obstacle is.

2.2 Vision-Based Systems

YOLO [3] and the Single Shot MultiBox Detector (SSD) [4] showed that many objects in a frame can be found in a single pass, which made real-time detection practical. OrCam MyEye [5] is a commercial wearable camera that reads text and recognises faces and products, but it is expensive and is not designed around continuous obstacle warning. Drishti [6] combined a wearable computer, GPS and voice output for outdoor guidance, but it appeared before modern deep-learning detectors.

2.3 Hybrid Systems

Some products mix several sensing methods. WeWALK [7] adds a smartphone link to an ultrasonic cane, which improves everyday use, although its warnings are still mostly reactive and it has no camera-based understanding of the scene. Table - 1 compares these systems with VisionAI.

Table - 1: Comparison with representative assistive systems

System	Sensing / Output	Gap addressed by VisionAI
Ultrasonic smart cane [2]	Ultrasonic ranging; vibration or buzzer	Cannot identify objects or give context
WeWALK [7]	Ultrasonic cane and smartphone navigation	Reactive; no visual object recognition or hazard ranking
OrCam MyEye [5]	Wearable camera; speech	High cost; built for recognition, not continuous obstacle guidance
Drishti [6]	Wearable computer, GPS, voice	No on-device visual detection
VisionAI (this work)	Camera with YOLOv8 on Jetson Nano, ultrasonic stick, GPS/GSM	Prioritised clock-position guidance, fail-safe haptics and SOS

2.4 Research Gap

The systems above either detect obstacles, recognise objects or give simple alerts. We found few that combine on-device detection, distance estimation, a second short-range sensor and ranked spoken guidance in one low-cost wearable. VisionAI is built to fill this gap for crowded Indian streets.

3. DATASET AND PREPROCESSING

The model was trained on 12,500 annotated frames. Pre-trained weights and base categories come from Common Objects in Context (COCO) [8]. We added frames collected on Indian roads and in indoor spaces, and used BDD100K [9] for extra road-scene variety.

3.1 Classes

Objects are placed in four groups according to how they affect a walking user (Table - 2). The grouping was defined by the authors for this application.

Table - 2: Dataset taxonomy and operational significance

Class Category	Object Labels	Operational Significance
Dynamic Obstacles	Person, Car, Motorcycle, Auto-rickshaw, Stray Animal	Tracking to avoid collisions with moving objects
Infrastructure	Traffic Light, Zebra Crossing, Signpost, Bench	Cues for crossing and following the street
Ground Hazards	Pothole, Open Drain, Speed Breaker	Prevent falls and injuries to the legs
Indoor Utilities	Chair, Table, Door, Stairs, Elevator	Movement inside offices and homes

3.2 Collection Conditions

Frames were recorded in bright sunlight, under overcast skies and at night. The camera was held at 1.5–1.7 m, the height of glasses worn by an adult, so the training views match the real device. Locations included crowded markets, narrow lanes and unpaved residential roads.

3.3 Augmentation

During training we rotated images by up to 15°, flipped them horizontally and rescaled them. Hue, saturation and brightness were varied to imitate different times of day and artificial light. Gaussian and motion blur were added to imitate the shake of a walking wearer, which also affects distance estimation.

3.4 Annotation and Split

Images were labelled in the Computer Vision Annotation Tool (CVAT) [10] with a 2D bounding box and a class ID from Table - 2. The data were split into 8,750 training images (70%), 2,500 validation images (20%) and 1,250 test images (10%).

4. SYSTEM ARCHITECTURE

VisionAI has three parts: perception, decision making, and feedback through sound and touch. They work as a continuous loop of sensing, planning and acting (Fig. 2).

4.1 Hardware

Our first prototype used a laptop or Raspberry Pi for processing. The final version replaces it with an NVIDIA Jetson Nano [11], which has a quad-core ARM Cortex-A57 central processing unit (CPU) and a 128-core Maxwell graphics processing unit (GPU) and is small enough to be worn. It runs the pipeline of camera, OpenCV, YOLOv8, distance estimation, decision and voice or haptic output, and it also handles speech synthesis and voice commands. A Sony IMX219 8-MP camera [12] is mounted on the front of the glasses so that the view follows the direction the user faces. The stick carries an ultrasonic sensor, a vibration motor, a buzzer and an ESP32 board [13]. The ESP32 reads the sensor and drives the motor and buzzer on its own, so a nearby obstacle is reported even if the vision pipeline is busy. Bone-conduction earphones give spoken output while leaving ambient sound audible, and a microphone takes voice commands. The safety module holds a GPS receiver, a 4G/GSM modem and an SOS button. A rechargeable lithium-ion (Li-ion) or lithium-polymer (Li-Po) battery with voltage regulation, USB-C charging and a level indicator powers the system. Table - 3 lists all components.

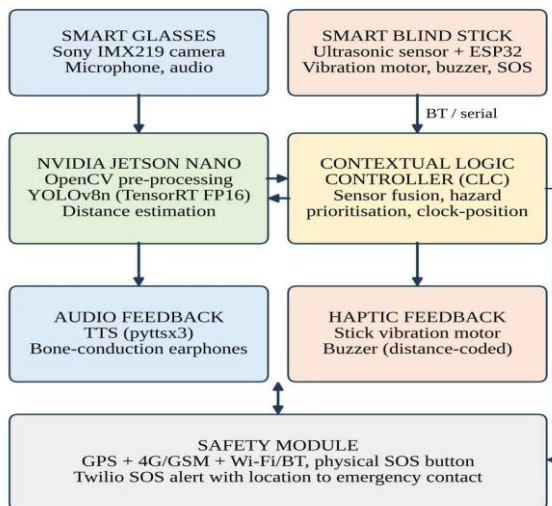


Fig. 2. Hardware architecture of the final VisionAI system.

4.2 Software

The system runs on Ubuntu with NVIDIA JetPack, and the code is written in Python (Table - 4). YOLOv8n [14] was chosen because it is light enough for an embedded board.

Inference is sped up with a TensorRT [15] half-precision (FP16) engine and Compute Unified Device Architecture (CUDA), and video is captured through GStreamer.

4.3 Detection and Distance Estimation

YOLOv8 finds all objects in a frame in a single pass, which keeps the frame rate high. We fine-tuned it on the four groups in Table - 2. Distance is estimated from the height of the bounding box, which avoids the cost of stereo cameras or LiDAR (light detection and ranging):

$$d \approx (f_{px} \cdot H_c) / h_{px} \quad (1)$$

where h_{px} is the box height in pixels, H_c is a typical real height for the class (for example an average adult) and f_{px} is the focal length in pixels, found by calibrating the IMX219 once. The horizontal position of the box centre, together with the camera's field of view, gives the direction, which is converted to a clock position.

Table - 3: Hardware components and purpose

Component	Purpose
Smart glasses frame	Wearable base for camera and audio
Sony IMX219 8-MP camera	Real-time image capture
NVIDIA Jetson Nano	Edge AI processing
Smart blind stick	Ground-level and close-range sensing
Ultrasonic sensor and ESP32	Distance measurement and stick control
Vibration motor and buzzer	Haptic and audible proximity alerts
Microphone; bone-conduction earphones	Voice commands; spoken guidance
GPS module	Location and navigation
4G/GSM, Wi-Fi, Bluetooth	SOS messages and stick link
SOS button	Manual emergency trigger
Li-ion/Li-Po battery and regulator	Portable, stable power

Table - 4: Software components and role

Software	Role
Ubuntu, JetPack, Python	Operating environment and control code
OpenCV [16], GStreamer	Frame capture and pre-processing
YOLOv8 (Ultralytics, PyTorch)	Object detection
CUDA, TensorRT, FP16	GPU acceleration of inference

Software	Role
Speech Recognition library	Voice-command input
pyttsx3	Text-to-speech output
GPS / navigation API	Location and direction logic
Twilio API [17]	SOS text messages
ESP32 firmware; serial or Bluetooth	Stick sensing and link to Jetson

4.4 Fusion and Decision Logic

A Contextual Logic Controller (CLC) receives detections from the camera and range values from the stick. Inside 1.0 m the ultrasonic reading takes priority, so the user still gets a warning when the camera is blocked or the light is poor. Vibration is slow for a moderate distance and fast and strong for a very close obstacle. Outside 1.0 m the controller ranks detections by class and distance, with moving obstacles and nearer objects announced first, and converts them into short phrases such as "Person ahead", "Obstacle on the left" or "Obstacle at 10 o'clock". The user can also say "Start navigation", "Stop system", "What is ahead?" or "Emergency".

4.5 Emergency Alerts

When the SOS button is pressed, or the user says "Emergency", the device reads its GPS position and sends a message with the coordinates to a saved contact through the Twilio API over the GSM or internet connection. A physical button is included because speech may not be practical in a real emergency.

5. OPERATIONAL FLOW

The system repeats the following steps while it is switched on:

1. The camera captures a 640×640 frame every 60–80 ms.
2. YOLOv8 detects objects and gives each a confidence score, and the stick reports its ultrasonic range.
3. Detections with confidence above 0.5 that lie in the user's path are given priority. Any ultrasonic reading below 1.0 m overrides them.
4. The instruction is spoken to the user, and the stick vibrates if the obstacle is within 1 m.
5. The user may ask "Where am I?" or "What is in front?" at any time.

6. RESULTS AND DISCUSSION

6.1 Detection Performance

The model was tested on COCO images [8] mixed with our own frames of Indian roads. Table - 5 gives the results for each group. The mean average precision at an intersection-over-union threshold of 0.5 (mAP@0.5) is 0.875 and the mean F1-score is 0.862. Chart - 1 compares the fine-tuned model with the base COCO-trained YOLOv8, which was evaluated on the same test images at the same threshold without fine-tuning, and the biggest improvement is for ground hazards.

Dynamic obstacles score best, since people and vehicles are large and stand out from the background. Ground hazards score lowest, mainly because a pothole has little height and looks different from one road to the next. This is one reason the ultrasonic stick is kept as a second sensor.

Table - 5: Evaluation metrics of the VisionAI detector

Class Category	Precision	Recall	F1-Score	mAP@0.5
Dynamic Obstacles	0.92	0.90	0.91	0.93
Infrastructure	0.88	0.84	0.86	0.87
Ground Hazards	0.82	0.79	0.80	0.81
Indoor Utilities	0.89	0.87	0.88	0.89
Macro Average	0.878	0.850	0.862	0.875

6.2 Processing Speed

A navigation aid must respond quickly, so the whole path from frame capture to spoken output was designed to fit within one frame interval of 60–80 ms. Table - 6 lists the stages and what is done in each to save time. The stick's ESP32 measures distance and drives the motor by itself, so proximity alerts do not wait for the neural network.

Table - 6: Pipeline stages and speed-up measures

Stage	Implementation	Speed-up measure
Frame capture	IMX219 through GStreamer	Hardware-accelerated capture
Pre-processing	OpenCV resize to 640×640	Single resize per frame
Inference	YOLOv8n on the Jetson GPU	TensorRT engine in FP16 with CUDA

Stage	Implementation	Speed-up measure
Distance and logic	Box geometry and CLC	Simple arithmetic on detections
Close-range alerts	ESP32 ultrasonic loop	Independent of the vision pipeline

6.3 Discussion

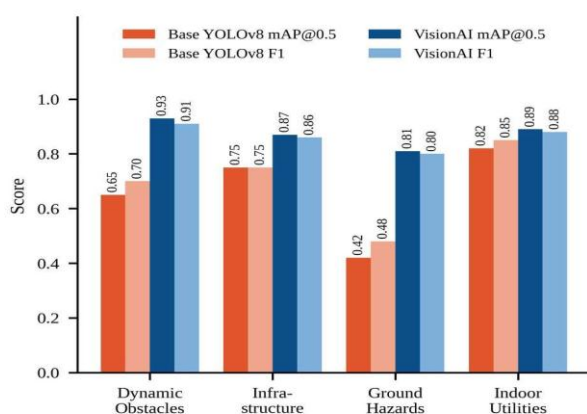
The fine-tuned model finds auto-rickshaws and stray animals more reliably than a generic COCO model, which shows the value of training on local data. Distance estimates are reliable at close range, which matters most for avoiding collisions, but become less accurate further away because the object covers fewer pixels.

The ultrasonic stick keeps working in low light or when the camera view is blocked, so the system does not depend on one sensor alone.

6.4 Limitations

Distance from a single camera depends on the assumed real height of each class and on calibration, and it becomes less accurate with range and camera shake. A single ultrasonic sensor covers only a narrow cone. Recall for ground hazards is lower than for the other groups. The Jetson Nano has limited power and cooling, and all results come from one dataset collected in Indian cities. Trials with visually impaired users are still needed.

Chart - 1: mAP@0.5 and F1-score per class group for base YOLOv8 (COCO) and fine-tuned VisionAI. The VisionAI values are those of Table - 5.



7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

This paper presented VisionAI, a wearable navigation aid made of smart glasses with a Sony IMX219 camera, an

NVIDIA Jetson Nano running a TensorRT-optimised YOLOv8n model, and an ESP32-based ultrasonic stick. A logic controller combines the two sensors, ranks what they find and gives spoken and vibration guidance, while a GPS and GSM module handles emergencies. The detector reached a mean mAP@0.5 of 0.875 on four object groups. Letting the ultrasonic reading take priority at close range means the user is still warned when the camera cannot see.

7.2 Future Work

Next we plan to measure latency, frame rate and power use on the Jetson Nano and to test the device with visually impaired users. Longer-term ideas are simultaneous localisation and mapping (SLAM) for path planning, Vision Transformers for wider scene context, smart lenses on biocompatible hydrogel with embedded micro-sensors, and recycled polymer-aluminium laminates for the glasses frame and stick to reduce their environmental cost.

REFERENCES

- 1) S. R. Flaxman et al., "Global causes of blindness and distance vision impairment 1990–2020: a systematic review and meta-analysis," *Lancet Glob. Health*, vol. 5, no. 12, pp. e1221–e1234, 2017.
- 2) J. Sakhardande, P. Pattanayak, and M. Bhowmick, "Smart cane assisted mobility for the visually impaired," *Int. J. Electr., Comput., Energetic, Electron. Commun. Eng.*, vol. 6, no. 10, 2012.
- 3) J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.
- 4) W. Liu et al., "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 21–37.
- 5) OrCam Technologies, "OrCam MyEye." [Online]. Available: <https://www.orcam.com>
- 6) S. Helal, S. Moore, and B. Ramachandran, "Drishti: An integrated navigation system for visually impaired and disabled," in *Proc. 5th Int. Symp. Wearable Comput. (ISWC)*, 2001, pp. 149–156.
- 7) WeWALK, "WeWALK smart cane." [Online]. Available: <https://wewalk.io>
- 8) T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.
- 9) F. Yu et al., "BDD100K: A diverse driving dataset for heterogeneous multitask learning," in *Proc. IEEE/CVF CVPR*, 2020, pp. 2636–2645.
- 10) OpenCV, "CVAT: Computer Vision Annotation Tool." [Online]. Available: <https://github.com/opencv/cvat>

- 11) NVIDIA, "Jetson Nano developer kit and technical specifications." [Online]. Available: <https://developer.nvidia.com/embedded/jetson-nano>
- 12) Raspberry Pi Ltd., "Camera Module 2 (Sony IMX219) datasheet." [Online]. Available: <https://www.raspberrypi.com/documentation>
- 13) Espressif Systems, "ESP32 series datasheet." [Online]. Available: <https://www.espressif.com>
- 14) G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- 15) NVIDIA, "TensorRT developer guide." [Online]. Available: <https://docs.nvidia.com/deeplearning/tensorrt>
- 16) G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 1st ed. Sebastopol, CA, USA: O'Reilly Media, 2008.
- 17) Twilio Inc., "Programmable Messaging API documentation." [Online]. Available: <https://www.twilio.com/docs>