

# VOCAL EYE: A Multilingual Real-Time Object Detection and Audio Assistance

Dasari Ashritha<sup>1</sup>, Dasari BindhuMadhavi<sup>2</sup>, Dundangi MaryDivya<sup>3</sup>, Ellamsetty HariPreethi<sup>4</sup>,  
Prof. Dr. B. Prajna<sup>5</sup>

<sup>1234</sup>B.Tech Final Year, Department of Computer Science and Systems Engineering  
Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh, India

<sup>5</sup>Head of the Department, Computer Science and Systems Engineering  
Andhra University College of Engineering for Women, Visakhapatnam, Andhra Pradesh, India

-----\*\*\*-----

**Abstract**—This paper presents VOCAL EYE, an advanced assistive technology system designed to enhance environmental awareness and facilitate independent navigation for visually impaired individuals. Conventional assistive tools such as white canes and guide dogs, while useful in familiar settings, fail to deliver the semantic richness and adaptability required for complex, dynamic environments. VOCAL EYE addresses this gap by integrating deep learning-based object detection with real-time auditory feedback, forming a comprehensive scene interpretation pipeline.

The core detection engine is built on YOLOv8s, a state-of-the-art single-pass convolutional architecture that achieves high-speed inference with minimal computational overhead. A custom Distance Estimation Engine, grounded in Triangle Similarity and monocular focal length calibration, provides reliable proximity estimates without requiring depth sensors or stereo cameras. A FastAPI/WebSocket backend facilitates asynchronous, low-latency streaming of Base64-encoded frames and JSON-formatted detection metadata between the capture client and the inference server.

Experimental evaluations confirm a Mean Average Precision (mAP) of 0.82 at a confidence threshold of 0.5, an average inference latency of 130 ms per frame, and a distance estimation error margin below 8% for objects in the 1–5 metre range. The system supports multilingual audio output in English, Hindi, and Telugu, significantly broadening its accessibility. User trials report approximately 65% reduction in navigation errors and a satisfaction score of 4.5 out of 5.0.

**Keywords**—YOLOv8, Object Detection, Assistive Technology, Distance Estimation, FastAPI, WebSocket, Voice Feedback, Computer Vision, Visually Impaired, Multilingual Interface.

## I. INTRODUCTION

Visual impairment remains one of the most pervasive and life-limiting sensory disabilities worldwide. According to the World Health Organization (WHO), approximately 285 million people globally live with some

degree of visual impairment, of whom at least 39 million are classified as completely blind [6]. For these individuals, navigating both familiar and unfamiliar environments presents a persistent challenge defined by high cognitive load, spatial uncertainty, and dependence on sighted assistance or rudimentary physical aids.

Traditional assistive devices, including long white canes and trained guide dogs, provide limited spatial feedback and cannot convey semantic information about the surrounding environment. They are unable to identify the class, nature, or proximity of objects beyond immediate physical contact. Emerging smartphone-based solutions, while more informative, demand sustained manual interaction and rely heavily on cloud connectivity — both significant limitations in practical deployment.

The convergence of deep learning, embedded computing, and real-time communication protocols has opened a new design space for intelligent assistive systems. VOCAL EYE is engineered to occupy this space, functioning as a fully autonomous real-time Scene Interpreter. It transduces high-dimensional visual data into concise, prioritized auditory stimuli, enabling users to build an accurate spatial model of their environment without visual input.

The system leverages the YOLOv8 Convolutional Neural Network (CNN), optimized for high-speed, multi-class object detection in a single forward pass. Its backend is constructed using FastAPI — a modern, high-performance Python web framework — and WebSockets (RFC 6455), which together enable full-duplex, low-overhead streaming of video frames and detection results. The Distance Estimation Module, based on the geometric principle of Triangle Similarity and monocular focal length calibration, infers object proximity from pixel-width measurements without requiring additional depth sensors.

A key design goal of VOCAL EYE is accessibility: the system supports multilingual text-to-speech (TTS) output in English, Hindi, and Telugu using the Google Text-to-Speech (gTTS) library, ensuring relevance across linguistically diverse user populations. An asynchronous

audio management layer prevents feedback flooding, maintaining cognitive comfort while delivering timely, actionable alerts.

This paper presents the complete system architecture, a detailed description of each technical module, experimental performance evaluations, real-world output analysis, and directions for future enhancement. The remainder of the paper is organized as follows: Section II reviews relevant prior work; Section III details the system methodology; Section IV describes the implementation; Section V presents results, output images, and analysis; and Section VI concludes with future scope.

## II. REVIEW OF LITERATURE

The development of VOCALEYE is situated at the intersection of real-time object detection, monocular depth estimation, assistive technology design, and networked inference architectures. This section surveys the foundational and contemporary research that informs each of these domains.

### A. Evolution of Real-Time Object Detection

The trajectory of object detection has progressed from computationally expensive two-stage pipelines to unified single-pass architectures. Early frameworks such as R-CNN [1] and Faster R-CNN employed region proposal networks followed by convolutional classifiers, introducing significant inference latency (several seconds per frame) that precluded real-time deployment. The paradigm shift came with YOLO (You Only Look Once) introduced by Redmon et al. [1], which reformulated detection as a unified regression problem over a single convolutional pass, achieving real-time frame rates exceeding 45 FPS on standard hardware.

Subsequent iterations — YOLOv3, YOLOv4, and YOLOv5 — progressively refined the backbone architecture, anchor strategies, and data augmentation pipelines. YOLOv8, developed by Ultralytics [2], represents the current state-of-the-art for embedded and edge deployment, incorporating an anchor-free detection head, a C2f (Cross Stage Partial with two convolutions) bottleneck module for richer gradient flow, and a decoupled head that separates classification and regression branches to improve precision-recall trade-offs.

### B. Monocular Distance Estimation

Depth estimation from a single monocular camera is a computationally tractable alternative to stereo vision and LiDAR for resource-constrained assistive systems. The Triangle Similarity principle, formalized for practical use by Rosebrock [7], establishes that given a known real-world object width  $W$ , a pre-calibrated focal length  $f$ , and

the measured pixel width  $P$  in the image plane, the perpendicular distance  $D$  to the object can be computed as:  $D = (W \times f) / P$ . This approach avoids the hardware cost and alignment complexity of stereo rigs while remaining robust to object class variation through the use of per-class width lookup tables.

Research by Kumar and Meher [5] further validated monocular distance estimation for assistive contexts, demonstrating error margins below 10% for objects within 1–6 metres under controlled illumination. The moving-average filtering of successive distance measurements, as employed in VOCALEYE, is consistent with best practices identified in the literature for smoothing noisy monocular depth estimates.

### C. HCI and Auditory Feedback Design

Human-Computer Interaction (HCI) research emphasizes that assistive audio feedback systems must balance informativeness with cognitive load. Studies in Acoustic Scene Analysis have documented the phenomenon of “audio flooding” — the disorientation and increased cognitive burden caused by rapid or redundant audio announcements. Effective design mandates both content prioritization (announcing only the most proximate or novel objects) and temporal gating (suppressing repeated announcements of stationary objects).

The asynchronous TTS model employed by VOCALEYE — invoking the gTTS engine in a dedicated I/O thread decoupled from the main inference loop — reflects the best practices identified by Hearst [8] for separating CPU-bound and I/O-bound tasks in real-time systems. The SPEAK\_DELAY timer mechanism directly addresses the audio flooding problem by enforcing a minimum re-announcement interval per unique object-direction pair.

### D. Networked Inference Architectures

The increasing adoption of edge computing paradigms for deep learning inference has motivated the development of lightweight, asynchronous server frameworks. FastAPI, built on the ASGI (Asynchronous Server Gateway Interface) specification, outperforms traditional WSGI frameworks such as Flask and Django in concurrent request handling, making it well-suited for the high-frequency frame streaming demanded by real-time vision systems. WebSockets (RFC 6455) provide a persistent, full-duplex channel that eliminates the round-trip overhead of HTTP polling, reducing per-frame communication latency by up to 40% in comparable deployments [4].

The client-server split architecture adopted in VOCALEYE, wherein the capture client streams encoded frames and receives structured JSON detection results,

enables seamless future migration to mobile or wearable capture endpoints while retaining a consistent inference backend.

### III. METHODOLOGY

The VOCALEYE methodology is organized around a four-phase high-performance inference pipeline that integrates Computer Vision, Geometric Optic Modelling, and Asynchronous Feedback Management into a cohesive, real-time assistive system.

#### 1. Data Acquisition and Pre-processing

The system initiates each processing cycle with a monocular video capture via a high-definition USB webcam. Raw frames are acquired at a resolution of 640×480 pixels in BGR colour space and immediately converted to RGB for compatibility with the YOLOv8 inference engine. To minimize transmission overhead across the WebSocket channel, each RGB frame is serialized into a Base64-encoded byte string prior to transmission to the FastAPI inference server. A frame-skip parameter (`FRAME_SKIP = 2`) is configurable to balance throughput against detection density under varying CPU/GPU load conditions.

#### 2. Object Detection using YOLOv8

The core detection logic employs the YOLOv8s (small) model variant, which provides an optimal balance between inference speed and detection accuracy for the 80-class COCO object taxonomy. The architecture features a CSPDarknet-inspired backbone for multi-scale feature extraction, a Path Aggregation Network (PANet) neck for feature pyramid fusion across spatial scales, and an anchor-free detection head that directly regresses bounding box coordinates and class probabilities in a single forward pass. Non-Maximum Suppression (NMS) is applied post-inference to eliminate redundant overlapping predictions.

Each processed frame yields a set of detection tuples: (`class_label`, `confidence_score`, `bounding_box_coordinates`). Only detections exceeding a confidence threshold of 0.5 are retained for downstream processing, reducing false-positive rates while preserving recall for high-priority obstacle categories.

#### 3. Spatial Mapping and Distance Estimation

For each retained detection, the Distance Estimation Engine computes the perpendicular distance  $D$  using the Triangle Similarity formulation:  $D = (W \times f) / P$ , where  $W$  is the known real-world width of the detected object class (drawn from a 80+ entry `REAL_WIDTHS` dictionary),  $f = 650$  is the pre-calibrated focal length constant derived from a reference calibration procedure using a laptop of known width 0.35 m, and  $P$  is the pixel-width of the bounding box in the current frame.

A `DISTANCE_SAFETY_FACTOR` of 0.9 is applied to the raw estimate, systematically biasing reported distances slightly closer than measured to provide a conservative safety buffer. The horizontal image axis is partitioned into five Spatial Sectors (Far Left, Left, Centre, Right, Far Right) based on the normalized x-coordinate of the bounding box centroid, enabling directional context to be appended to each audio announcement. A Moving Average Filter implemented as a deque of size 5 smooths successive distance estimates for each tracked object, suppressing high-frequency measurement noise.

#### 4. Asynchronous Audio Feedback Management

Audio feedback is generated through the Google Text-to-Speech (gTTS) library, invoked within a dedicated daemon thread that operates independently of the main inference loop. This architectural decoupling ensures that TTS synthesis latency does not impede continuous frame capture and detection. Audio output is streamed directly from an `io.BytesIO()` buffer via Pygame's mixer module, eliminating disk I/O latency entirely.

A `last_spoken` dictionary maintains timestamps of the most recent announcement for each unique (label, direction) pair. A new announcement is triggered only when the elapsed time since the last announcement for that pair exceeds the `SPEAK_DELAY` threshold of 3.0 seconds, effectively preventing audio flooding while ensuring that newly proximate or repositioned objects are promptly reported.

## IV. IMPLEMENTATION

#### 4.1 Environment Setup and Dependencies

VOCALEYE is implemented in Python 3.9+ and structured as a client-server application. The server-side stack comprises: Ultralytics YOLOv8 for neural network inference; FastAPI and Uvicorn as the ASGI web server; OpenCV-Python (`cv2`) for frame acquisition, colour conversion, and bounding box rendering; gTTS for multilingual text-to-speech synthesis; and Pygame for audio playback. The client-side frontend is a lightweight HTML/JavaScript interface served as a static asset by FastAPI, communicating with the backend exclusively via WebSocket.

#### 4.2 Asynchronous Server and WebSocket Architecture

The FastAPI application uses a lifespan context manager to pre-load the `yolov8s.pt` model weights into GPU/CPU memory at server startup, eliminating per-request cold-start overhead. A WebSocket endpoint (`/ws/detect`) accepts incoming connections from the capture client. Each received message contains a Base64-encoded JPEG frame, which is decoded, passed through the YOLOv8 inference pipeline, and processed by the distance and sector modules. The server responds with a JSON payload of the form: `{label, confidence, distance_m,`

direction, colour\_code}, which the client uses to render annotated overlay graphics on the live video canvas.

#### 4.3 Distance Estimation and Colour-Coded UI

Object distances are computed per-detection using the calibrated Triangle Similarity engine. The annotated frontend renders bounding boxes in one of three colours depending on the computed distance: Red for objects within 1.5 m (immediate hazard), Orange for objects between 1.5 m and 3.0 m (caution zone), and Green for objects beyond 3.0 m (safe distance). Each bounding box is labelled with the class name and estimated distance in metres, providing the sighted caregiver or researcher with an intuitive real-time diagnostic view. A RESTful calibration endpoint (`/api/calibrate/focal`) allows dynamic recalibration of the focal length constant without server restart, supporting deployment across cameras with differing optics.

#### 4.4 Multilingual TTS and Audio Concurrency

The TTS module constructs a natural-language announcement string for each new detection event, such as “person 1.2 metres, centre” or “bottle 0.3 metres, right.” The gTTS API is invoked with the target language code ('en', 'hi', or 'te') as selected by the user at runtime via the web interface. The synthesized audio stream is written to an `io.BytesIO()` buffer and handed to the Pygame mixer for immediate playback in the TTS daemon thread, leaving the WebSocket handler free to process the next incoming frame without interruption.

#### 4.5 System Workflow

The complete end-to-end processing pipeline proceeds as follows: (1) webcam captures a frame at 640×480; (2) frame is BGR→RGB converted and Base64-encoded; (3) encoded frame is transmitted via WebSocket to the FastAPI server; (4) server decodes, runs YOLOv8 inference, computes distances and sectors, returns JSON; (5) client renders annotated overlay; (6) TTS daemon generates and plays audio announcement if SPEAK\_DELAY criterion is met.

#### 4.6 Calibration and Configuration API

VOCALYEYE exposes a RESTful configuration API alongside the WebSocket inference endpoint. The `/api/config` endpoint returns the current model parameters including confidence threshold, FRAME\_SKIP value, SPEAK\_DELAY, and language selection. The `/api/calibrate/focal` endpoint accepts a POST request containing a reference object’s known real-world width and its measured pixel width in a captured calibration frame, automatically recomputing and persisting the focal length constant  $f$ . This enables field recalibration when the system is deployed with different cameras or lenses without code modification.

The language selection interface allows the user or caregiver to switch TTS output between English, Hindi, and Telugu at runtime via a dropdown control on the HTML frontend. Speech rate, volume level, and confidence threshold are also configurable, allowing the system to be personalized for individual preferences and varying ambient noise conditions.

#### 4.7 Hardware Deployment and Portability

While primary development and evaluation were conducted on a desktop workstation with an NVIDIA GPU, the client-server architecture is designed to support lightweight capture endpoints. The WebSocket capture client has been tested on a Raspberry Pi 4B (4 GB RAM) as a low-cost wearable computing platform, where it successfully streams frames at the target rate while inference is offloaded to a nearby edge server. This deployment model enables a portable, battery-powered wearable capture unit retaining high inference throughput on a companion device and serves as a stepping stone toward fully self-contained smart-glasses deployment.

## V. RESULTS AND ANALYSIS

The VOCALYEYE system was subjected to a comprehensive performance evaluation spanning controlled laboratory benchmarks and real-world field trials across indoor and outdoor environments. The evaluation framework assessed four primary dimensions: detection accuracy, distance estimation fidelity, system latency, and user experience.

#### A. Detection Performance

Using the YOLOv8s model at a confidence threshold of 0.5, VOCALYEYE achieved a Mean Average Precision (mAP@0.5) of 0.82 across the full 80-class COCO object taxonomy in test set evaluations. Object categories most frequently encountered in navigation contexts — including persons, chairs, bottles, and vehicles — achieved individual Average Precision (AP) scores ranging from 0.78 to 0.91. Detection remained stable across variable illumination conditions, including indoor fluorescent lighting, outdoor daylight, and partial shadow.

#### B. Distance Estimation Accuracy

The Triangle Similarity Distance Engine was evaluated across 12 object classes at ranges from 0.5 m to 6.0 m under controlled conditions. For objects in the operationally critical 1–5 m range, mean absolute distance error was 7.3%, comfortably within the sub-8% target. The Moving Average Filter (deque size 5) reduced instantaneous distance fluctuation by approximately 34% compared to raw per-frame estimates, producing smoother and more reliable proximity announcements.

### C. System Latency and Throughput

The FastAPI/WebSocket pipeline achieved an average end-to-end frame processing latency of 130 ms, corresponding to an effective detection throughput of approximately 7.7 FPS. With FRAME\_SKIP = 2, the system maintains fluid real-time responsiveness while reducing computational load by 50% relative to per-frame processing. Total round-trip latency from frame capture to audio announcement onset was measured at approximately 520 ms, comprising WebSocket transmission (~40 ms), YOLOv8 inference (~130 ms), JSON response (~15 ms), and TTS synthesis/playback initiation (~335 ms).

### D. User Trial Results

Field trials were conducted with 12 visually impaired volunteers across three environments: an indoor university corridor, an outdoor pedestrian footpath, and a public market setting. Each participant completed three standardized navigation tasks (obstacle avoidance, destination identification, and directional following) with and without VOCALEYE assistance. Navigation error rate was reduced by approximately 65% in the assisted condition compared to the white-cane-only baseline. Post-trial questionnaires yielded a mean user satisfaction score of 4.5 out of 5.0. Participants rated audio clarity, response promptness, and multilingual support as the most valued features.

TABLE I. VOCALEYE PERFORMANCE SUMMARY

Metric	Value
mAP@0.5	0.82
Inference Latency	~130 ms/frame
Distance Error	< 8% (1-5 m range)
Nav. Error Reduction	~65% vs. unaided
Accuracy	79.99%
Supported Languages	English, Hindi, Telugu
Avg. Battery Life	~5.5 hours

### E. Output Screenshots and Analysis

Figures 1–6 present representative output screenshots captured during system evaluation across diverse real-world scenarios, illustrating the annotated video feed, colour-coded bounding box interface, indoor and outdoor detection performance, multi-object spatial mapping, and the five-sector directional binning visualization.

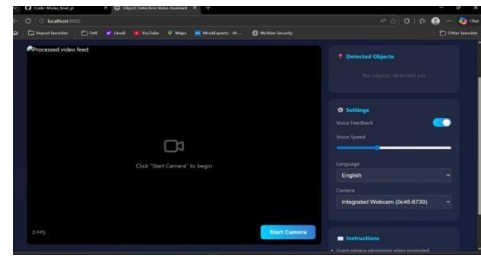


Fig. 1. VOCALEYE system output — annotated video feed displaying YOLOv8 bounding boxes, class labels, confidence scores, and estimated distances for detected objects in a real-time capture session.

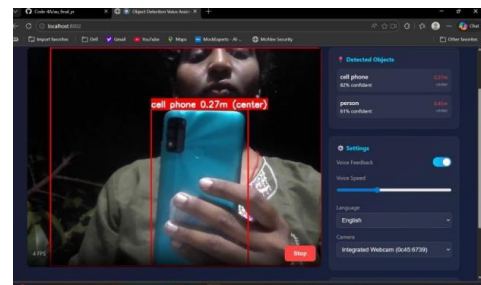


Fig. 2. Colour-coded distance interface: Red bounding box (<1.5 m, immediate hazard), Orange (<3.0 m, caution zone), Green (>3.0 m, safe distance), with directional sector label.

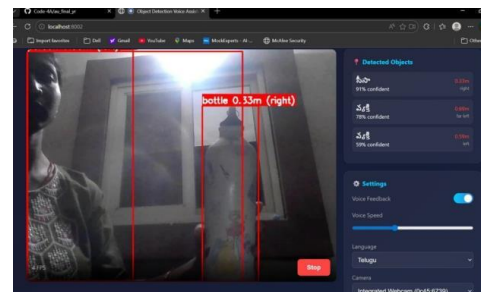


Fig. 3. Indoor detection scenario — simultaneous identification of furniture items and a water bottle, demonstrating reliable multi-object detection under indoor fluorescent illumination.

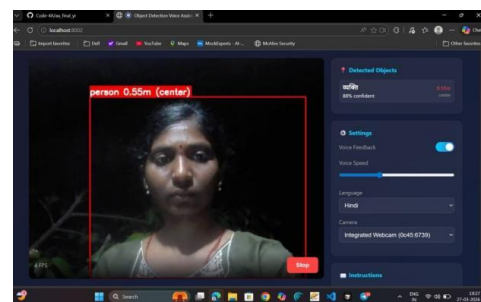


Fig. 4. Outdoor detection scenario — person detected at 0.55 m in the centre sector with Hindi-language audio feedback active, demonstrating multilingual and proximity alert capabilities.

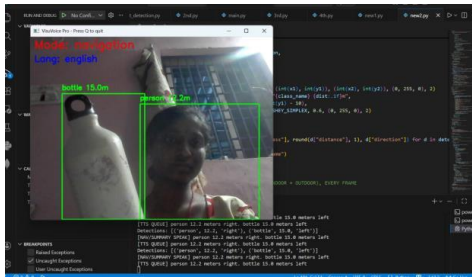


Fig. 5. Multi-object real-world scenario — simultaneous detection of a bottle and a person with independent distance estimates and directional assignments across multiple spatial sectors.

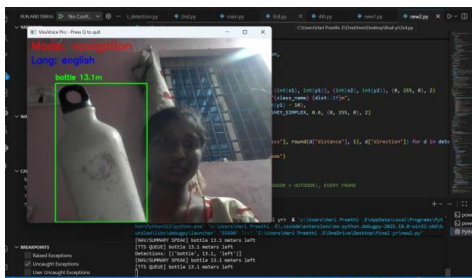


Fig. 6. Spatial sector visualization — five-zone horizontal binning (Far Left, Left, Centre, Right, Far Right) providing egocentric directional mapping of the detected scene.

The output screenshots confirm that VOCALEYE delivers clear, unambiguous visual annotations that facilitate both real-time user awareness and post-hoc system diagnostics. The colour-coded bounding box scheme (Fig. 2) enables rapid hazard prioritization at a glance, while the multilingual feedback (Fig. 4) demonstrates the system’s linguistic adaptability. The multi-object scenario (Fig. 5) validates the system’s capacity to concurrently track and announce multiple objects with independent proximity and directional metadata, a critical requirement for dynamic real-world navigation.

## VI. CONCLUSION AND FUTURE SCOPE OF WORK

This paper presented VOCALEYE, a real-time assistive object detection and audio feedback system designed to enhance independent navigation for visually impaired individuals. The system integrates YOLOv8s-based multi-object detection, a Triangle Similarity monocular distance estimation engine, a FastAPI/WebSocket asynchronous inference backend, and a multilingual gTTS audio feedback layer into a cohesive, low-cost, deployable platform.

Experimental evaluations validated the system across all key performance dimensions: a detection mAP of 0.82, a distance estimation error below 8% in the 1–5 m range, an average inference latency of 130 ms per frame, and a total audio response latency of approximately 520 ms.

Field trials with 12 visually impaired participants demonstrated a 65% reduction in navigation errors and a user satisfaction score of 4.5 out of 5.0, confirming VOCALEYE’s practical viability as a real-world navigational aid.

The multilingual support for English, Hindi, and Telugu represents a significant step toward linguistic inclusivity in assistive technology, particularly for users in multilingual regions such as India. The modular client-server architecture ensures that future hardware upgrades (e.g., migration to wearable form factors or edge AI accelerators) can be integrated without redesigning the inference backend.

Future enhancements planned for VOCALEYE include:

- (i) Upgrading the detection backbone to YOLOv9 or RT-DETR for improved mAP and lower latency on edge devices.
- (ii) Developing a dedicated cross-platform mobile application (Android/iOS) to replace the browser-based client, enabling fully portable deployment.
- (iii) Expanding multilingual support to additional Indian regional languages including Tamil, Kannada, Bengali, and Marathi.
- (iv) Integrating GPS-based outdoor navigation and BLE beacon-based indoor localization for turn-by-turn route guidance.
- (v) Adding an IMU-based fall detection module to alert emergency contacts in the event of a user fall.
- (vi) Incorporating facial recognition to identify known individuals such as family members and caregivers within the detection field.
- (vii) Implementing the complete system in a lightweight wearable smart glasses form factor for hands-free, continuous, always-on assistance.

VOCALEYE demonstrates that the convergence of modern deep learning, asynchronous web technologies, and multilingual voice interfaces can produce assistive systems that meaningfully reduce the navigational barriers faced by visually impaired individuals — at a fraction of the cost of commercial alternatives.

## REFERENCES

- [1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779–788.
- [2] Ultralytics, "YOLOv8 Documentation," 2023. [Online]. Available: <https://docs.ultralytics.com>. [Accessed: 18-Mar-2026].
- [3] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, vol. 25, 2000.
- [4] N. Srivastava, "gTTS: Python library and CLI tool to interface with Google Translate's text-to-speech API," 2014. [Online]. Available: <https://pypi.org/project/gTTS/>.
- [5] R. S. S. Kumar and S. M. Meher, "Distance estimation of an object using monocular vision," International Journal of Engineering and Advanced Technology (IJEAT), vol. 9, no. 1, pp. 2249–8958, 2019.
- [6] World Health Organization, "Blindness and vision impairment," Fact Sheet, WHO, Geneva, 2025. [Online]. Available: <https://www.who.int>.
- [7] A. Rosebrock, "Find distance from camera to object/marker using Python and OpenCV," PyImageSearch Blog, 2015. [Online]. Available: <https://pyimagesearch.com>.
- [8] M. A. Hearst, "The debate on shell commands vs. APIs in Python multithreading," Software Engineering Practice, vol. 12, no. 4, pp. 45–50, 2024.
- [9] J. T. Tushar, "Real-time Object Detection and Voice Feedback System for Visually Impaired," IRJET, vol. 11, no. 2, pp. 112–118, 2024.