



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85276>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

EYRO - AI Glasses: A Unified Multimodal Assistive Navigation Prototype

Shravani Mahajan¹, Srushti Nitnaware², Ishant Hirekhan³, Mayank Jorande⁴, Ms. Priya Meshram⁵

Department of Computer Science and Engineering, GH Rasoni University, Amravati, India

Abstract: EYRO is a prototype assistive-navigation system designed to combine visual perception, speech interaction, task management, and audio guidance for people with visual impairment. Unlike systems that concentrate on a single capability, the implementation integrates walkable-area segmentation, obstacle detection, crosswalk and traffic-light assistance, open-vocabulary object search, hand-guided localization, experimental Indian currency recognition, browser-based map navigation, and fail-safe health monitoring under one task-controlled application. The software runs primarily on a Windows laptop and accepts camera and microphone input from local devices or optional network-connected hardware. A NavigationMaster state machine coordinates the active task so that only one primary guidance workflow is executed at a time, while a reduced-rate obstacle safety layer can remain active during selected temporary tasks. The implementation uses SegFormer-B0 for walkable-area segmentation, YOLO11n for obstacle and traffic-related detection, YOLOE for open-vocabulary search and banknote localization, ByteTrack for object tracking, MediaPipe HandLandmarker for hand guidance, faster-whisper for speech recognition, pyttsx3/SAPI5 for speech output, OpenRouteService for map routing, and optional MobileNetV3-Small classifiers for currency denomination and authenticity estimation. The system deliberately avoids unverified safety claims: an unclear traffic light remains UNKNOWN, a nearby-vehicle estimate is treated as a visual proxy rather than distance or speed, and currency output is explicitly an AI estimate. This paper presents the engineering design, module-level methodology, state-machine logic, implementation decisions, comparison with a recent smart-glasses base paper, and an evaluation framework. Because project-specific benchmark measurements and controlled user-study results were not supplied in the source materials, no fabricated accuracy, latency, FPS, or user-performance values are reported.

Keywords: Assistive Technology, AI Glasses, Computer Vision, Object Detection, Navigation Assistance.

I. INTRODUCTION

Visual information is central to mobility, object identification, road interaction, and everyday decision making. For a person with severe visual impairment, these activities can require continuous environmental interpretation rather than a single recognition event. The recent literature shows a movement from specialized aids toward wearable systems that combine perception and audio feedback. The selected 2026 base paper by Kumari and Hammady integrates object detection, distance estimation, OCR, and auditory feedback on Vuzix Blade 2 smart glasses, and reports a university-campus dataset and model comparison. Its emphasis on unified wearable assistance provides a useful reference point for EYRO, but its architecture and experimental results are not transferred to the present prototype. EYRO approaches the problem from a different implementation perspective. Instead of deploying the full processing pipeline on dedicated smart-glass hardware, the current prototype uses a conventional Windows laptop as the main AI processor. A camera and microphone enable sensing and interaction, while optional devices provide additional input. The application is exposed through a FastAPI server and browser interface, while spoken feedback is produced through the Windows SAPI5 speech engine. This arrangement makes the prototype easier to assemble and modify while exposing a practical limitation: the current system is not itself a compact, self-contained wearable computer. The main research problem addressed by this work is therefore not the creation of a single new vision model. It is the integration of heterogeneous assistive functions under a single control architecture while maintaining conservative safety behavior. EYRO combines continuous path assistance with short-lived tasks such as crossing, traffic-light checking, object search, map navigation, and currency recognition. The system must decide which task owns the primary guidance channel, when a temporary task ends, how uncertain perception is communicated, and how failures are prevented from producing stale guidance. These engineering decisions form the central contribution of the prototype. The paper is organized as follows. Section II defines the problem and motivation. Section III reviews related work and the selected base paper. Section IV states the research gap and contributions. Sections V–VIII describe the EYRO architecture, workflow, state machine, and perception modules. Section IX presents implementation and interaction design. Section X defines the evaluation methodology and reports only evidence available from the project materials. Sections XI–XIII discuss the comparison, limitations, and future work, followed by the conclusion and references.

II. PROBLEM STATEMENT, MOTIVATION, AND OBJECTIVES

A. Problem Statement

A practical assistive system must interpret several different visual situations without forcing the user to manually operate separate applications. A walking user may need an obstacle warning, then a crosswalk cue, then a traffic-light check, and later a route instruction or an object-search interaction. The main challenge lies in managing multiple detection tasks together and ensuring that users receive only relevant, up-to-date, and consistent guidance without conflicting alerts.

The project materials define EYRO as a prototype rather than a certified or medically approved safety device. The implementation therefore treats perception uncertainty as a first-class condition. In particular, an unclear traffic-light state is retained as UNKNOWN, a nearby-vehicle condition is estimated from bounding-box size rather than physical distance or speed, and currency recognition is presented as an AI estimate rather than a guarantee of authenticity. These constraints are central to the problem definition rather than disclaimers added after implementation.

B. Motivation

The motivation for EYRO comes from the fragmentation visible in assistive-technology research. Previous studies have explored assistive technologies for tasks such as detecting obstacles, reading text, and providing remote assistance. However, limitations related to hardware requirements and processing capabilities remain common challenges across these approaches.

EYRO uses that observation as a design prompt but does not reproduce the base paper architecture. Instead, the prototype emphasizes a software-defined task manager capable of combining several assistive functions with shared camera, speech, and feedback infrastructure. A second motivation is affordability and development accessibility. The project can run on a standard Windows laptop with a webcam and microphone, while optional ESP32-CAM, network microphone, and button components can be added. This arrangement is useful for research prototyping because models and control logic can be changed without redesigning a dedicated wearable computer. The trade-off is that the resulting system is not yet an on-body standalone product.

C. Objectives

The primary objective of EYRO is to develop an integrated assistive prototype that combines environmental perception, object identification, navigation, and voice-based interaction within a common software architecture. The system aims to identify walkable regions and surrounding obstacles, recognize crosswalks, and interpret traffic-light conditions using conservative decision logic. It also supports user-directed object searching through open-vocabulary detection, tracking, and hand-landmark-based spatial guidance. In addition, the prototype explores Indian currency recognition by communicating AI-generated predictions with appropriate uncertainty, rather than claiming guaranteed authenticity. EYRO further aims to provide destination-based navigation through browser geolocation and OpenRouteService while supporting interaction through voice commands, browser controls, API communication, and optional physical buttons. A shared task manager coordinates these functions and helps prevent competing modules from delivering conflicting instructions through the guidance channel. Particular attention is given to handling failed detections, uncertain predictions, and outdated results so that unreliable information is not presented as confirmed guidance. Collectively, these objectives establish a modular foundation for evaluating the prototype's functionality, identifying its limitations, and guiding future improvements in accessible assistive technology.

TABLE I. FUNCTIONAL SCOPE OF EYRO

Module	Primary input	Core processing	User-facing output
Blind path	Camera	SegFormer-B0 + YOLO11n + optical flow	Path/obstacle audio cues
Crossing	Camera	YOLOE crosswalk + YOLO11n light/vehicle + safety logic	Cautionary crossing guidance
Traffic check	Camera	YOLO11n + ROI colour analysis	Red/yellow/green/unknown announcement
Object search	Voice/typed target + camera	YOLOE + ByteTrack + HandLandmarker	Direction/hand guidance
Map route	Destination + browser GPS	Local landmarks + OpenRouteService	Turn-by-turn speech
Currency	Camera	YOLOE banknote + optional MobileNetV3-Small	Denomination/authenticity estimate or uncertainty
Chat	Unmatched text/voice	Local or optional cloud LLM	Conversational reply when no task is active

III. RELATED WORK AND LITERATURE REVIEW

A. Assistive Navigation and Wearable Systems

Assistive navigation research spans depth-camera systems, sonar or haptic aids, smartphone applications, smart glasses, and audio scene encoders.

The base paper surveys examples such as NAVI, Smart Cane, OrCam MyEye, Seeing AI, Aira, and IrisVision and argues that many solutions concentrate on individual capabilities. Earlier Kinect-based systems demonstrated the value of depth perception for navigation but introduced hardware bulk and environmental constraints. These approaches remain relevant because they show that assistive perception must be interpreted in the context of the user's movement rather than treated as an isolated object-recognition benchmark.

Wearable systems also expose a trade-off between computational resources and usability. A device that performs more inference locally can reduce network dependence but must operate within constraints on power, memory, thermal load, and interaction bandwidth. Conversely, cloud-assisted systems can provide stronger processing resources but introduce connectivity and privacy considerations. EYRO chooses a third prototype-oriented compromise: the laptop performs the main computation locally, while only map routing and optional chat can depend on external services. This is an architectural choice, not a claim that laptop-based processing is superior for a final wearable product.

B. Real-Time Object Detection

YOLO-family detectors are widely used when a system must localize objects quickly. The base paper evaluates YOLOv8 variants and selects a balance between accuracy and resource use for its Vuzix platform. EYRO instead uses YOLO11n for its configured obstacle and traffic-light/vehicle detection. The official Ultralytics documentation identifies YOLO11n as a supported detection checkpoint and notes that YOLO11 is available across detection, segmentation, classification, pose, and related tasks. Because EYRO does not provide a project-specific benchmark, the paper does not transfer the model's public benchmark numbers into system performance claims.

C. Semantic Segmentation for Walkable Areas

Blind-path navigation benefits from a representation of the scene that is richer than independent obstacle boxes. EYRO uses the SegFormer-B0 checkpoint fine-tuned on ADE20K for semantic segmentation. SegFormer combines a hierarchical Transformer encoder with a lightweight multilayer-perceptron decoder and was designed as an efficient semantic-segmentation framework. In EYRO, the segmentation output is interpreted as a walkable-area cue and combined with obstacle detection and optical-flow information before a spoken guidance decision is produced. This combination is an implementation choice of the prototype rather than a claim of novel segmentation architecture.

D. Tracking and Hand-Guided Search

Object search is different from obstacle warning because the user supplies a semantic target, such as keys or a phone. EYRO combines YOLOE to identify a wide range of objects, EYRO integrates YOLOE for recognizing different objects, ByteTrack for tracking them across successive video frames, and MediaPipe HandLandmarker for providing guidance based on hand movements. ByteTrack maintains object identities between consecutive frames, helping the system track movement over time. MediaPipe Hands provides an on-device hand-tracking pipeline based on palm detection and hand landmark estimation. Together, these components let EYRO maintain a target across frames and express its relative direction to the user's hand without introducing a separate object-specific classifier for every search target.

E. Speech and Route Interaction

Voice commands are transcribed with faster-whisper, while pyttsx3/SAPI5 provides local speech output. Whisper is a large-scale weakly supervised speech-recognition system released by OpenAI; the project uses the smaller base configuration through faster-whisper rather than reproducing Whisper training. Route guidance uses browser geolocation and OpenRouteService. The ORS documentation exposes directions services that accept route coordinates and return route representations, including GeoJSON. EYRO applies its own destination-resolution checks before requesting a route and performs off-route confirmation and rerouting in the navigation layer.

IV. RESEARCH GAP AND BASE PAPER ANALYSIS

A. Research Gap

The literature indicates a recurring separation between perception functions and user-facing task management. A system may recognize obstacles, read text, or provide route guidance, but combining heterogeneous functions creates additional engineering questions: which task owns the audio channel, how temporary tasks return to a default navigation state, how uncertainty is represented, and what happens when perception becomes stale. The selected base paper demonstrates integrated wearable detection, OCR, and audio, yet it focuses on a Vuzix Blade 2 platform, YOLOv8 variants, OCR, and a university-campus dataset. EYRO addresses a narrower but complementary gap: a single software state machine coordinating multiple assistive workflows that extend beyond object detection and OCR. The prototype includes walkable-area segmentation, obstacle detection, crosswalk and traffic-light assistance, open-vocabulary object search with tracking and hand guidance, map navigation, experimental currency recognition, multi-input control, and a health monitor.

An additional challenge is ensuring that these functions operate consistently when tasks overlap or individual modules fail. EYRO focuses on managing different functions smoothly, delivering understandable guidance, and avoiding misleading feedback when the system is unsure of a result. Its contribution lies in exploring how diverse assistive capabilities can be organized within one maintainable software framework, rather than claiming novelty for any individual detection model.

B. Selected Base Paper

The selected base paper is “Assisting Blind People with AI and Audio Using Smart Glasses: System Design with YOLOv8 Variants Comparisons” by Priyanka Kumari and Ramy Hammady, published in *Multimedia Systems* in 2026. The study employs Vuzix Blade 2 smart glasses and integrates YOLOv8-based object detection, distance estimation, optical character recognition (OCR), and audio feedback. It uses a university-campus dataset containing 15,951 annotated images and evaluates YOLOv8-N, YOLOv8-S, and YOLOv8-M using metrics such as precision, recall, mean average precision (mAP), and inference time. The study also compares its approach with other assistive technologies.

This paper provides a useful foundation for EYRO by demonstrating how visual recognition and audio feedback can be combined to support people with visual impairments. It also highlights opportunities to explore additional functions, including route navigation, hands-free interaction, changing lighting conditions, and broader scene understanding. EYRO extends this direction by incorporating map navigation, voice commands, path segmentation, and coordinated task management. At present, EYRO operates primarily on a laptop, and additional experiments are needed to evaluate its reliability in various real-life environments.

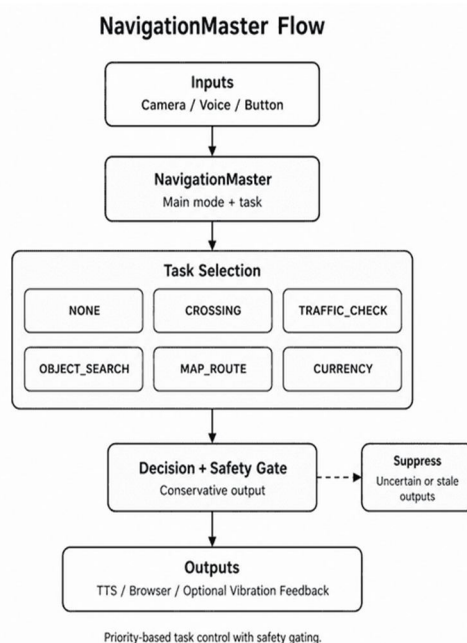


Fig. 4. NavigationMaster state machine coordinating the primary assistive task and routing outputs through a conservative decision and safety gate.

TABLE II. BASE PAPER AND EYRO COMPARISON

Aspect	Selected base paper	EYRO prototype	Interpretation
Primary platform	Vuzix Blade 2 smart glasses	Windows laptop + camera/mic; optional ESP32/phone inputs	EYRO prioritizes prototyping flexibility over standalone wearable deployment
Core detection	YOLOv8 variants	YOLO11n + YOLOE	Different detector generation and task allocation
Walkable-area segmentation	Not the central reported module	SegFormer-B0	Adds explicit semantic walkable-area reasoning
OCR	Azure-based OCR	Not implemented as an EYRO feature	Avoids claiming an unsupported OCR capability
Object tracking	SORT in reported architecture	ByteTrack	Temporal object association for search
Hand guidance	Not central to the reported pipeline	MediaPipe HandLandmarker	Adds hand-oriented search guidance
Navigation	No full point-to-point route navigation in the reported limitation	Browser GPS + OpenRouteService	Addresses a capability identified as future work in [1]
Currency	Not a core reported function	Experimental Indian banknote mode	Adds a localized everyday task
Task control	Feature pipeline around wearable application	NavigationMaster two-level state machine	Makes task ownership and transitions explicit
Safety behavior	Object/distance alerts and auditory feedback	Conservative crossing logic + health monitor + background safety	Adds explicit uncertainty and failure handling
Evaluation	Dataset and quantitative model comparison reported	No project-specific benchmark supplied in source materials	EYRO evaluation must remain qualitative/traceable until measurements are collected

V. EYRO PROTOTYPE AND SYSTEM ARCHITECTURE



Fig. 1. EYRO prototype with camera/electronics assembly and cabling.

Architecture interpretation. The figure separates input acquisition, task ownership, model execution, decision logic, and feedback. This separation allows a camera or microphone implementation to change without changing the NavigationMaster contract, while optional hardware remains outside the core decision layer.

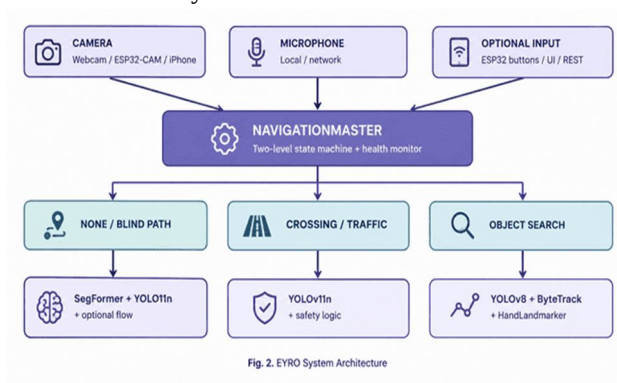


Fig. 2. EYRO System Architecture

The architecture is intentionally asymmetric: perception modules produce observations, while NavigationMaster determines which observations are relevant to the current task. This prevents individual modules from independently taking control of the user interaction channel.

End-to-end workflow. The same control boundary is reused for every task, while the health layer can suppress unreliable output.

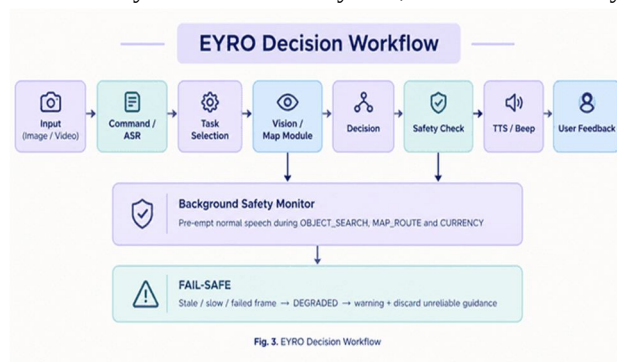


Fig. 3. EYRO Decision Workflow

The workflow separates model inference from user-facing decision logic so that an uncertain observation can be withheld or converted into a cautious message. Figure 2 summarizes the implementation as a software-centered pipeline. The camera and microphone can be local or network-connected; the optional physical button is independent of microphone recognition. NavigationMaster acts as the central controller, managing the selected operating mode and the active task within the system. Each task invokes its workflow and returns status and guidance to the browser interface and speech engine. This architecture keeps mode transitions centralized rather than allowing individual modules to activate each other arbitrarily.

The supplied prototype image in Figure 1 is included as project evidence of the physical glasses assembly. It should not be interpreted as evidence that all processing electronics are currently mounted on the frame. The source documentation explicitly states that the laptop performs the AI processing and that no dedicated glasses hardware is required for the prototype. The optional ESP32 components provide input or network connectivity rather than constituting a complete standalone compute platform.

A. Input Layer

The input layer accepts video, audio, and discrete control events. The camera module can receive video from a locally connected webcam or a network-based source, including an ESP32-CAM or a phone running an IP Webcam application. The audio path can use the laptop microphone or a network WAV stream. Voice commands pass through voice activity detection and faster-whisper, while typed/API commands and browser buttons use the same NavigationMaster entry points. The optional ESP32 button provides short-press and long-press actions and can remain useful if speech recognition is unavailable.

B. Control Layer

NavigationMaster implements a two-level state machine. The main mode is IDLE or ASSISTIVE_NAVIGATION. The task is NONE, CROSSING, TRAFFIC_CHECK, OBJECT_SEARCH, MAP_ROUTE, or CURRENCY. Only one primary task is active at a time. Temporary tasks return to NONE when completed, cancelled, or timed out. This prevents, for example, a map instruction stream and an object-search stream from competing for the same audio channel.

C. Output Layer

The output module manages spoken messages through a priority-based queue using pyttsx3 and SAPI5, while beep sounds notify users when the system switches between modes. Informational speech is throttled, while urgent obstacle, vehicle, degraded-camera, or disconnect messages can pre-empt queued informational items. The browser UI receives annotated frames and status events through WebSockets. The project also contains a vibration-controller seam, but the documentation explicitly identifies it as a stub; the paper therefore does not claim functional haptic feedback.

VI. END-TO-END WORKFLOW AND STATE MANAGEMENT

A. Mode-Transition Rules

At startup, navigation is configured to begin in ASSISTIVE_NAVIGATION by default. Starting a temporary task first ensures that navigation is active. A crosswalk can automatically trigger CROSSING after five consecutive crosswalk detections when auto-detection is enabled. During CROSSING, non-stop voice commands are refused so that a misheard utterance does not unexpectedly switch the guidance context. Traffic checks, object search, map routes, and currency are temporary tasks and are not intended to remain active indefinitely.

B. Voice and Non-Voice Control

Voice is only one control path. Browser buttons, REST commands, and an optional ESP32 button enter the same task methods. This is important for fault tolerance: if the microphone is disabled or unavailable, the user can still interact through the web interface or API. The physical button provides a short-press cycle through default path navigation, object search, and currency, and a long press returns to the default path mode. The current implementation contains a known safety caveat: the physical-button handler does not yet enforce the same crossing mode-lock as voice commands.

VII. METHODOLOGY: PERCEPTION AND ASSISTIVE MODULES

A. Blind-Path Navigation

Blind-path navigation is the default continuous task. The camera frame is processed by SegFormer-B0 to estimate semantic regions associated with the environment. The walkable mask is then considered together with YOLO11n obstacle detections. Optical-flow information is used for motion/stabilisation cues before the navigation logic produces a concise direction or obstacle message. The system does not attempt full autonomous path planning; its role is to provide assistive cues based on current visual evidence.

B. Crosswalk Assistance

Crosswalk assistance combines open-vocabulary crosswalk detection with traffic-light and nearby-vehicle analysis. The crosswalk detector uses a YOLOE prompt for “crosswalk” and falls back to classical OpenCV processing if the YOLO model cannot load. Because the prompt has not been custom-trained or validated in the project, it is treated as an experimental perception component. Traffic lights are detected with YOLO11n and the cropped region is analyzed with classical colour rules. Temporal confirmation is used so a single ambiguous frame does not immediately determine the state.

The crossing-safety decision layer deliberately avoids the phrase “safe to cross”. A green light and detected crosswalk are not sufficient for an unconditional proceed instruction. A nearby-vehicle proxy can override the green state, while an uncertain light remains UNKNOWN. This design reflects the implementation’s conservative rule: the system assists the user in checking the scene but does not claim to establish physical safety.

C. Traffic-Light Check

The standalone traffic-light task is intentionally shorter than CROSSING. It searches for a traffic light, announces the detected colour once, and returns to the default task. If a light cannot be confirmed within the configured window, the system asks whether to continue checking. Unanswered or negative responses contribute to a maximum prompt count. The state UNKNOWN is preserved when colour evidence is insufficient.

D. Object Search and Hand Guidance

Object search begins with a semantic target extracted from the command. YOLOE supports open-vocabulary detection, allowing the same search pipeline to accept different target descriptions without a dedicated classifier for every object. ByteTrack maintains temporal association between detections. MediaPipe HandLandmarker estimates hand landmarks, allowing the system to express coarse spatial guidance for grasping the target. The interaction is intentionally concise and rate-limited; the user can confirm that the object has been found or allow the task to time out.

Crossing and search interpretation. The two workflows share camera infrastructure but have different decision semantics. Crossing prioritizes conservative caution, while object search prioritizes target continuity and spatial guidance.

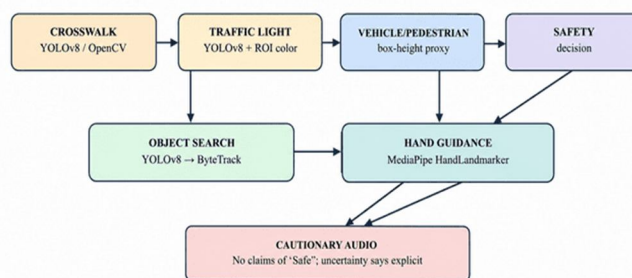


Fig. 6. Combined crossing and object-search design.

Fig. 6. Combined crossing and object-search design.

The distinction is important for evaluation: crossing should be measured for safety-relevant false positives and false negatives, whereas object search should be measured for target acquisition, tracking continuity, and understandable hand-direction guidance.

TABLE VI. Decision Differences Between Crossing And Object Search

Stage	Crossing	Object search
Primary target	Crosswalk / traffic context	User-specified object
Perception	YOLOE crosswalk + YOLO11n light/vehicle	YOLOE target detection
Temporal handling	Confirm light state / crosswalk evidence	ByteTrack target continuity
Additional cue	Vehicle-near box-size proxy	MediaPipe hand landmarks
Decision priority	Caution and uncertainty	Target direction and grasp guidance
Unsafe assumption avoided	Green light is never treated as proof of safety	Unknown target is never treated as found
Evaluation focus	False warnings, missed hazards, state uncertainty	Search success, tracking continuity, guidance clarity

E. Currency Recognition

Currency mode is experimental. YOLOE first locates a banknote using the “banknote” prompt. If the optional MobileNetV3-Small denomination and authenticity checkpoints load and the confidence threshold is met, the system can provide an estimate. Otherwise it states that the note cannot be identified clearly. The software carries an explicit disclaimer that the output is an AI estimate and not a guarantee of authenticity. This distinction is important because the project does not provide a validated financial-recognition benchmark.

F. Map Navigation

Map navigation uses browser geolocation for the current position and OpenRouteService for routing. The project first checks a curated local-landmark file and then uses ORS geocoding with confidence, distance, and regional checks. An optional language-model correction can adjust a destination string, but the corrected string must pass the geocoder again. Route guidance uses two-stage cueing, an arrival radius, and off-route confirmation before rerouting. The mapping feature cannot operate without a valid ORS API key, so it remains disabled when the key is not configured.



Fig. 7. AI glasses navigation and assistive guidance interface.

VIII. TECHNOLOGY STACK AND IMPLEMENTATION

TABLE III. EYRO TECHNOLOGY STACK

Layer	Technology	Role in EYRO
Backend	Python, FastAPI, Uvicorn, WebSockets	Application server, REST endpoints, status/video streaming
Vision	OpenCV, NumPy, Pillow	Frame capture, image processing, fallback CV
Segmentation	SegFormer-B0 / Transformers	Walkable-area semantic segmentation
Detection	YOLO11n / Ultralytics	Obstacles, traffic light, vehicles
Open vocabulary	YOLOE	Object search, crosswalk prompt, banknote locator
Tracking	ByteTrack	Temporal object association
Hands	MediaPipe HandLandmarker	Hand position and guidance cues
Speech input	faster-whisper base, sounddevice	Speech-to-text and microphone input
Speech output	pyttsx3 / Windows SAPI5, winsound	TTS and audio cues
Maps	Browser Geolocation, OpenRouteService, Leaflet/OSM	Location, routing, browser map
Chat	llama.cpp / Ollama / optional OpenAI	Free-form chat when no task is active
Optional hardware	ESP32-CAM, INMP441, ESP32 button	Remote camera/audio/control inputs

A. Backend and Browser Interface

FastAPI provides the application surface. REST endpoints expose command and mode actions, location updates, TTS voice selection, and recording controls. WebSockets stream annotated JPEG frames and status events to the browser. The frontend is intentionally lightweight HTML/CSS/JavaScript rather than a large framework. A Leaflet/OpenStreetMap map is used for route display. This separation allows the perception loop to remain in Python while the browser provides visualization and manual controls.

B. Model Loading and Fallbacks

The project is designed to tolerate optional component failures. If a YOLO model is unavailable, OpenCV fallbacks are available for crosswalk and traffic-light processing. If YOLOE is unavailable for object search, the configured obstacle model can be used as a fallback. If currency checkpoints are absent, the system can still locate a note but does not fabricate a denomination. If the microphone is disabled, browser/API control remains available. These fallbacks are part of the architecture and should be preserved in future revisions.

C. Speech Timing and Interaction

Continuous speech would create a cognitive burden and would also interfere with command recognition. EYRO therefore applies per-category throttling, wider minimum intervals for blind-path and search guidance, and priority levels for urgent messages. The microphone is ignored while TTS is speaking and for a short echo-cooldown interval. When a task changes, the TTS queue is cleared so that stale instructions from the previous task are not spoken after the transition.

D. Configuration and Reproducibility

Configuration is centralized in config.py and environment variables rather than hard-coded into individual modules. This includes camera source, microphone source, model paths, confidence thresholds, task timeouts, health-monitor settings, and map parameters. The project documentation also records known mismatches between older documentation and the current code configuration, including the transition from an older YOLOv8n obstacle reference to the yolo11n.pt default. For reproducible reporting, the current configured model should be treated as the source of truth.

IX. ALGORITHMIC DECISION LOGIC

A. Task Selection

EYRO uses deterministic regular-expression intent routing for fixed commands. This is a deliberate engineering choice because fixed task commands should not depend on generative interpretation. A fuzzy matching layer is available only after regex rules fail and requires explicit confirmation. If no task command is recognized and the current task is NONE, the utterance can be routed to the optional chat backend. During an active task, unmatched speech receives a short stop-oriented prompt rather than a potentially distracting chat response.

Algorithm 1: Command Routing: receive transcript, normalize text, test ordered intent patterns; if a pattern matches, activate the corresponding task, otherwise attempt fuzzy matching, if the fuzzy match is above threshold, request yes/no confirmation, if confirmed, activate the proposed task, if no task is active and no command is recognized, route to chat, otherwise retain the active task and request stop/cancel if necessary.

B. Crossing Decision

The crossing decision is intentionally not a binary “safe/unsafe” classifier. The perception layer supplies crosswalk evidence, a traffic-light state, and a vehicle-near proxy. The decision layer maps these observations into caution levels such as NONE, WAIT, PROCEED_WITH_CAUTION, or UNCERTAIN. The most important invariant is that a green light never produces an unconditional safety claim, and an unknown light is never silently converted to a known colour.

C. Health Monitoring

Health monitoring is a temporal rule rather than a single-frame check. Workflow exceptions, stale frames, and slow processing are accumulated across consecutive frames. Once the configured threshold is reached, the system enters DEGRADED and discards unreliable perception. Recovery requires multiple consecutive good frames. This reduces the risk that a transient timing anomaly immediately changes the user experience while still preventing prolonged processing failure from producing normal guidance.

D. Background Safety

During object search, map navigation, and currency recognition, EYRO performs obstacle checks at a reduced frequency. If a nearby obstacle is detected, an urgent warning can pre-empt routine information in the text-to-speech (TTS) queue, ensuring that important safety alerts receive priority.

This background mechanism is disabled during CROSSING and TRAFFIC_CHECK to preserve deterministic state logic and prevent competing workflows from interfering with crossing-related guidance. It serves as a supplementary safety aid and does not replace the user's judgement or independent precautions during real-world navigation

X. EXPERIMENTAL SETUP AND EVALUATION

A. What Can Be Evaluated From the Supplied Materials

The available project documentation describes the implementation and its intended behavior but does not provide a controlled dataset, confusion matrices, precision/recall measurements, mAP values, FPS logs, latency measurements, or a human-participant study for EYRO. The paper therefore does not invent quantitative results. The correct academic treatment is to report the implemented functions and define a reproducible evaluation protocol that can be executed before a future publication version.

B. Functional Verification Plan

The initial evaluation should examine whether each EYRO function behaves as expected under different conditions. Walkable-path testing should cover clear surfaces, stationary obstacles, moving objects, and changes in camera position. Crossing assistance should be examined under different crosswalk visibility conditions and traffic-light states, including red, yellow, green, and unknown. Nearby vehicles should also be included to examine how the system responds to changing surroundings. Object-search testing should cover several target categories and situations in which an object becomes temporarily hidden or the user's hand enters the camera view.

Currency recognition should be checked using supported Indian denominations under different lighting conditions and orientations, including cases where the note is difficult to identify. Navigation testing should cover recognised landmarks, unsuccessful geocoding requests, movement away from the suggested route, and arrival at the destination. Each test should record the expected response and the system's actual output. Repeating these tests would help reveal missed detections, incorrect warnings, and situations requiring more reliable handling.

C. Quantitative Evaluation Plan

A quantitative study should evaluate each major function using measurements suited to its purpose. Object-detection performance can be assessed using precision, recall, and mean average precision (mAP) on a labelled test dataset. Tracking performance can be measured using an established tracking metric when annotated video sequences are available.

For walkable-area segmentation, an appropriate mask-based metric can be used with manually labelled or otherwise validated reference data.

Responsiveness should also be measured at two levels: model inference time and total response time. The latter includes image capture, task processing, state transitions, communication between components, and delivery of audio guidance. Recording these measurements separately would help identify whether delays originate in the vision models or in the wider system. Testing should document the hardware, software configuration, dataset, and environmental conditions so that the procedure can be repeated. The present work distinguishes implemented functionality from performance that remains to be verified. No quantitative result is presented as established unless it has been measured.

Future testing should also examine warning reliability, conflicting audio messages, and behaviour when individual modules fail or return uncertain outputs.

TABLE IV. EVALUATION MATRIX FOR A FUTURE CONTROLLED STUDY

Subsystem	Suggested metric	Ground truth / test artifact	Current status in supplied materials
Blind-path segmentation	IoU / mIoU	Annotated walkable masks	Not supplied
Obstacle detection	Precision, recall, mAP	Frame-level object boxes	Not supplied
Crosswalk detection	Precision, recall	Annotated crosswalk frames	Not validated; no project benchmark supplied
Traffic-light colour	Per-state accuracy / confusion matrix	Annotated light states	Not supplied
Object tracking	IDF1 / HOTA / MOTA where appropriate	Annotated video sequences	Not supplied
Object search	Task success rate + time-to-find	Repeated target-search trials	Not supplied
Currency	Denomination/authenticity accuracy	Labeled banknote set	Not validated
Map navigation	Route success / reroute rate	Known routes + controlled deviations	Not supplied
End-to-end responsiveness	Frame-to-guidance and command-to-action latency	Timestamped event logs	Not supplied
Usability	Task completion, workload, satisfaction	Ethics-approved participant study	Not supplied

D. Reproducibility and Testability

The repository includes unit and workflow-oriented tests covering navigation state management, intent matching, health checks, map navigation, traffic safety, TTS priority, blind-path workflow, and currency workflow. These tests are valuable because they verify deterministic control logic even when quantitative model benchmarks are unavailable. A future paper should report test counts, pass/fail outcomes, test environment, and commit/version information so that software behavior can be reproduced.

XI. RESULTS, DISCUSSION, AND COMPARATIVE ANALYSIS

A. Results Available From the Current Project

The supplied materials support implementation-level conclusions: EYRO contains the described task workflows, uses the specified model stack, centralizes task control in NavigationMaster, and implements explicit uncertainty and health-monitor behavior. They do not support numerical claims about detection accuracy, precision, recall, mAP, FPS, latency, user satisfaction, or navigation success. Accordingly, the present manuscript reports no project-specific performance number as an experimental result.

This differs from the base paper, which reports a formal evaluation using 15,951 annotated campus images and quantitative comparisons among YOLOv8 variants [1]. The base paper's numerical results must not be copied or relabeled as EYRO results. They establish the methodological context for what a complete evaluation could look like, while EYRO still requires its own dataset and measurement protocol.

B. Engineering Discussion

The strongest aspect of EYRO is architectural integration. The same camera and feedback infrastructure can support multiple tasks, while the state machine prevents uncontrolled concurrency. This is especially important for speech-driven systems because multiple modules producing simultaneous audio would be difficult to interpret. The explicit return-to-NONE rule also simplifies recovery from temporary tasks.

A second strength is the use of uncertainty-aware logic. The project does not treat every model output as equally trustworthy. Traffic-light uncertainty remains UNKNOWN, currency uncertainty can suppress identification, and health degradation blocks stale perception from reaching the user. These choices are particularly appropriate for assistive prototypes because a confident but wrong message can be more harmful than an explicit limitation statement. A third strength is modularity. YOLOE, YOLO11n, SegFormer, ByteTrack, MediaPipe, speech recognition, TTS, and routing are separated into workflow or service modules. The browser interface is also decoupled from the processing loop. This allows individual models to be replaced or benchmarked without rewriting the entire application, which is useful for future work on smaller on-device hardware.

C. Comparison With Existing Work

In comparison with the selected base paper, EYRO introduces additional functions such as path segmentation, object searching through hand guidance, currency recognition, and route navigation. These features are combined within a different processing structure designed to manage multiple tasks through a common control system. Unlike assistive applications that focus mainly on one specific function, EYRO aims to bring several useful capabilities together in a single platform. This approach reduces the need to switch between separate tools for different activities. However, combining multiple functions also creates technical challenges. Each task requires proper testing, and errors in one module may affect the overall user experience. The system must also handle task switching carefully to avoid repeated, outdated, or conflicting voice instructions. Adding more features does not automatically make a system more reliable or effective. Its usefulness depends on how accurately the individual modules work together and how safely the system responds to different situations. Therefore, EYRO focuses on a structured integration approach, coordinated task execution, and clearly defined safety limitations. These aspects represent the main contribution of the prototype while also highlighting the need for further testing and improvement before practical use.

TABLE V. MODULE-LEVEL COMPARISON

Capability	Base paper [1]	EYRO	Status / significance
Obstacle detection	YOLOv8-based	YOLO11n	Implemented; benchmark not supplied
Distance estimation	Reported	Not implemented as physical distance estimation	EYRO uses obstacle geometry/decision cues instead
OCR	Azure AI Vision	Not implemented	Intentionally excluded from EYRO claims
Walkable segmentation	Not central	SegFormer-B0	Adds semantic scene-region cue
Open-vocabulary search	Not central	YOLOE	Supports target-driven search
Tracking	SORT reported	ByteTrack	Temporal association for search
Hand guidance	Not central	MediaPipe HandLandmarker	Adds grasp-oriented spatial cue
Crosswalk	Not core	YOLOE prompt + OpenCV fallback	Experimental, unvalidated
Traffic-light state	Detection + application logic	YOLO11n + classical ROI colour	UNKNOWN preserved
Map navigation	Identified as future scope	Browser GPS + ORS	Implemented in prototype
Currency	Not core	YOLOE + optional MobileNetV3-Small	Experimental
Task state machine	Application pipeline	NavigationMaster	Central EYRO design contribution
Health/fail-safe	Not central	HealthMonitor + degraded state	Prevents stale guidance

XII. LIMITATIONS AND THREATS TO VALIDITY

A. Perception Limitations

The crosswalk detector is an open-vocabulary YOLOE prompt and has not been validated or custom-trained for the target environment. Traffic-light colour is based on classical ROI analysis rather than a trained state classifier. The nearby-vehicle condition is estimated from bounding-box size, so it is not a measurement of actual distance, velocity, or collision time. These limitations make the corresponding outputs assistive cues rather than safety guarantees.

B. Hardware and Deployment Limitations

The main processor is a Windows laptop, which is practical for development but not equivalent to a compact wearable compute unit. A future wearable implementation would need to address power, weight, thermal behavior, camera placement, audio output, and connectivity. The supplied physical prototype images document the glasses assembly, but they do not establish that all processing is performed on the frame.

C. Evaluation Limitations

No controlled EYRO benchmark dataset or participant study was supplied. The absence of quantitative validation limits claims about model performance and usability. In particular, no accuracy number should be inferred from the existence of pretrained checkpoints. Public benchmark performance of a model is not the same as performance of that model within this application, camera, environment, and task logic.

D. Interaction Limitations

Speech output cannot be interrupted once pytsx3/SAPI5 is actively speaking because of the current engine behavior. Voice recognition can be affected by ambient noise, while the browser interface depends on external frontend assets for some UI features. The physical button also has a known crossing-lock caveat. These issues should be resolved before a safety-oriented user study.

E. Threats to Validity

The current evaluation evidence is implementation-centric and therefore vulnerable to confirmation bias: successful demonstrations do not estimate failure rates in diverse real environments. Lighting, camera viewpoint, object scale, occlusion, road geometry, traffic density, speech accents, and network conditions can all alter performance. A rigorous publication study should predefine test cases and success criteria, record failures, and report confidence intervals where appropriate.

XIII. FUTURE WORK

The next stage should prioritize validation before adding more features. First, the team should build representative datasets for crosswalks, traffic lights, walkable regions, object search, and Indian currency. The data should include variation in illumination, camera angle, object scale, occlusion, and background clutter. Second, the perception modules should be benchmarked independently and as an end-to-end system so that model limitations can be distinguished from control or communication delays.

A second direction is hardware migration. The current laptop-centered architecture can serve as the reference implementation while selected components are moved to a smaller on-body processor. Model quantization, inference scheduling, and selective frame processing can be investigated, but any claimed improvement should be measured rather than assumed. A real vibration motor and compact audio hardware could also be integrated, replacing the current vibration stub.

A third direction is safety and human-centered evaluation. The physical-button mode-lock should be aligned with voice behavior. Crossing assistance should be tested against ambiguous and adversarial conditions, including partially visible lights, glare, occlusion, and nearby vehicles. A formal usability study with visually impaired participants should then assess learnability, cognitive load, task completion, and perceived usefulness under approved ethical procedures. The base paper also identifies the value of future longitudinal user studies, making this an appropriate point of comparison.

Finally, the software architecture can be extended with stronger scene understanding and route context while preserving the existing task-state principle. Any new capability should have an explicit owner, failure mode, feedback policy, and test coverage. The goal should be controlled integration rather than uncontrolled feature accumulation.

XIV. CONCLUSION

EYRO presents a unified assistive-navigation prototype that brings several computer-vision and interaction tasks under one task-controlled software architecture. Its implementation combines SegFormer-based walkable-area understanding, YOLO11n obstacle and traffic-related detection, YOLOE open-vocabulary search, ByteTrack tracking, MediaPipe hand landmarks, experimental currency recognition, browser GPS routing, faster-whisper speech recognition, local text-to-speech, and explicit health monitoring. The central engineering idea is the NavigationMaster state machine, which prevents multiple primary guidance tasks from competing and provides deterministic transitions between default and temporary assistance modes.

The system also demonstrates a conservative approach to uncertain perception. Crossing assistance never labels a green light as proof of safety, unclear traffic-light states remain UNKNOWN, currency output is presented as an estimate, and degraded camera conditions can suppress unreliable guidance. These behaviors are important design properties for an assistive prototype and distinguish system integration from a simple collection of independent AI models.

The comparison with the selected base paper shows that EYRO should be understood as a complementary extension rather than a reproduction. The base paper provides a strong reference for wearable object detection, OCR, auditory feedback, and quantitative evaluation on Vuzix Blade 2. EYRO expands the task-management and navigation scope while using a laptop-centered prototype architecture and a different model stack. The remaining research challenge is empirical validation: the next publication stage should measure each module, quantify end-to-end responsiveness, and evaluate usability with appropriate participants. Until those measurements exist, the scientifically correct position is to report the implementation and its limitations without inventing performance claims. EYRO provides a foundation for future assistive-technology research through further testing and real-world validation.

XV. DETAILED MODULE SPECIFICATION

A. Camera Capture and Frame Handoff

The camera subsystem isolates device-specific acquisition from the perception workflows. It supports a local camera index or a network URL, and it can apply configured flip, rotation, and reconnection behavior. The captured frame is passed through a thread-safe bridge so that the processing loop and browser streaming path do not directly compete for the camera resource. Connection events are surfaced to the user because a disconnected camera invalidates every visual task.

This separation is important for reproducibility. The same perception workflows can be exercised with a laptop webcam, ESP32-CAM stream, or phone IP Webcam without changing the downstream task logic. Future experiments should record the camera source and resolution because the same model can behave differently under different field-of-view and compression conditions.

B. Audio Acquisition and Recognition

The audio subsystem accepts input from a local or network microphone and uses voice-activity detection to identify speech segments. Recognised commands are processed using faster-whisper in its base-model configuration with int8 computation. To reduce false activations, microphone input is ignored while the text-to-speech engine is speaking, followed by a short cooldown period to limit acoustic echo.

The architecture also separates command recognition from free-form conversational processing. This allows predefined commands to control assistive tasks through explicit application logic rather than allowing a generative model to determine which operational state should be activated. This separation supports more predictable interaction and helps keep task selection under the control of the main application.

C. Text-to-Speech and Priority Queue

The text-to-speech subsystem uses a priority queue to organise spoken messages according to their urgency. The SAPI5 speech engine is recreated for each utterance, while routine information is rate-limited to prevent repetitive announcements. When an urgent message is received, lower-priority queued information can be cleared so that the warning is delivered sooner.

This queue-management strategy is an important part of the user interaction design. Continuous vision processing may generate repeated descriptions of the same obstacle or walkable path, which could otherwise make audio guidance difficult to follow. By limiting unnecessary narration and prioritising urgent messages, EYRO aims to provide clearer and less distracting feedback. The effectiveness of this approach should be confirmed through latency measurements and usability testing.

D. Browser and REST Boundary

The browser is a presentation and control surface, not the authoritative task manager. REST actions and WebSocket status messages are routed through the same application state as voice and physical-button inputs. This avoids creating a second, inconsistent mode system in the frontend. The browser can show the annotated camera stream, current task, guidance text, microphone status, and map information, while the backend remains responsible for task transitions and safety decisions.

Table VII. Module Responsibility Matrix

Module	Owens	Must not own
camera_capture.py	Camera acquisition, reconnect, transforms	Task selection or safety decisions
asr_core.py	Microphone, VAD, speech-to-text	Direct task execution
navigation_master.py	State, intent routing, task transitions	Low-level image processing
workflow_blindpath.py	Segmentation, obstacles, optical-flow cues	Map routing
workflow_crossstreet.py	Crossing state and traffic interaction	General chat
yolomedia.py	YOLOE, tracking, hand-guidance services	Global task transitions
workflow_currency.py	Banknote locate/classification workflow	Navigation state ownership
map_navigation.py	Geolocation, geocoding, route guidance	Visual obstacle decisions
health_check.py	Frame/workflow health status	Model inference itself
tts_engine.py	Speech queue, priority, beeps	Command interpretation

XVI. SOFTWARE STATE MACHINE: DETAILED BEHAVIOUR

A. Main Mode and Task Separation

The two-level state model is one of the most important design decisions in EYRO. Main mode indicates whether assistance is inactive or active, while task identifies the current guidance workflow. This separation prevents a temporary task from being confused with the overall navigation service. For example, entering OBJECT_SEARCH does not mean that assistance has been stopped it means that the primary task has changed temporarily.

Each task has explicit entry and exit behavior. Starting a task first ensures that assistive navigation is active. Completing, timing out, or cancelling a temporary task returns the task state to NONE. The default NONE state is associated with blind-path processing, which means the system naturally resumes general path assistance after a short task. This reduces the need for the user to repeatedly restart the main navigation service.

B. Mode-Lock During Crossing

Crossing is intentionally different from other temporary tasks. Once CROSSING is active, voice commands other than stop or cancel are refused. The reason is not that voice recognition is inherently unsafe; rather, a mistaken command should not change the context while the user is attempting to cross a road. The implementation therefore prioritizes state continuity over convenience. The current physical-button path does not yet enforce the same lock, which is a known engineering issue for future revision.

C. Automatic Crosswalk Entry

When enabled, five consecutive crosswalk detections can trigger CROSSING automatically. Temporal persistence is used because a single frame may contain a false positive or a partially visible crosswalk. The auto-entry threshold is configurable. Because the crosswalk detector itself is not validated on a project-specific dataset, the auto-entry mechanism should be evaluated separately from manual crossing commands.

D. Temporary Task Recovery

Temporary tasks use bounded retry and confirmation loops. Traffic checking asks whether to continue if a light cannot be confirmed. Object search asks whether to keep trying and can end after a configured timeout. Currency mode similarly avoids repeated guessing when confidence is insufficient. Map navigation can reroute after repeated off-route confirmation. These patterns give the user a predictable recovery path without leaving a task active indefinitely.

E. State-Machine Invariants

The main invariants are: one primary task at a time; task changes clear stale speech, crossing accepts only stop/cancel voice commands, background obstacle checking is restricted to selected tasks; unknown perception is never silently converted to a known class; and degraded health suppresses unreliable guidance.

These invariants can be converted directly into automated tests and should be treated as regression requirements whenever the application is modified.

XVII. SAFETY-ORIENTED DECISION DESIGN

A. Why Conservative Output Matters

Assistive computer vision differs from ordinary object-detection demonstrations because the user may act on the spoken result without independently seeing the scene. A model confidence score is therefore not equivalent to a safety certificate. EYRO addresses this by keeping perception and decision logic separate. Detection modules report observations; the decision layer chooses whether an observation is actionable and how it should be phrased.

The crossing module provides the clearest example. The presence of a crosswalk and a green light is not interpreted as permission to cross. A vehicle-near estimate can override the green state, while an uncertain light remains UNKNOWN. The user is instructed to check traffic rather than being told that the road is safe. This design intentionally limits the semantic strength of the output to what the available sensors and models can support.

B. Stale-Data Protection

The health monitor protects against a less visible failure mode: stale perception. If a camera stalls or frame processing falls behind, the system may otherwise continue to display or speak a result that describes an earlier scene. EYRO enters a DEGRADED state after sustained evidence of stale, slow, or failed processing and discards unreliable results. Recovery requires multiple good frames. This policy is especially relevant to dynamic environments where an obstacle or vehicle can move between frames.

C. Failure Containment

The architecture isolates optional components so that their failure does not necessarily terminate the application. A missing microphone does not prevent browser control. A missing ORS key disables map navigation rather than the entire application. A missing currency checkpoint prevents confident denomination output but does not require the rest of the system to stop. The same principle applies to model fallbacks for crosswalk and traffic-light perception.

D. Human Responsibility and Scope

EYRO should be described as an assistive prototype intended to provide cues, not as a replacement for human judgment or as a certified mobility device. This boundary is essential to the research claims. The prototype demonstrates a software architecture for multimodal assistance; it does not establish clinical effectiveness, legal compliance, or real-world safety certification.

TABLE VIII. SAFETY INVARIANTS AND FAILURE RESPONSE

Condition	EYRO response	Reason
Traffic light unclear	Keep UNKNOWN; do not guess	Prevents unsupported state conversion
Green + crosswalk	Caution; check traffic	No unconditional safety claim
Nearby vehicle proxy	Wait/caution overrides green	Prioritizes observed vehicle evidence
Currency below confidence	Do not identify confidently	Avoids fabricated denomination/authenticity
Camera stale/slow	DEGRADED + warning + discard frame	Prevents stale guidance
YOLO unavailable	Use documented fallback where supported	Contains optional model failure
No microphone	Use UI/API controls	Preserves basic operation
Crossing + new voice command	Refuse non-stop command	Prevents accidental task switching

XVIII. DATA, MODEL, AND DEPLOYMENT CONSIDERATIONS

A. Model Provenance

The EYRO implementation combines pretrained and application-level components rather than presenting a new neural network architecture. SegFormer-B0 is used through the Transformers ecosystem; YOLO11n and YOLOE are loaded through the Ultralytics model stack; ByteTrack provides tracking; MediaPipe supplies hand landmarks; faster-whisper supplies speech recognition; and optional MobileNetV3-Small checkpoints support the experimental currency task. The paper cites the relevant sources so that the origin of each major component is clear.

This model composition creates a validation requirement: each component may have been trained or benchmarked under assumptions that differ from the EYRO deployment environment. Public model benchmarks cannot establish the performance of the complete pipeline. A publication-quality evaluation must therefore measure the assembled system with the actual camera, hardware, lighting, speech environment, and decision thresholds used in the experiment.

B. Domain Shift

The project is intended for everyday environments that can vary substantially from model-training distributions. Road markings, traffic-light shapes, banknote orientation, indoor clutter, and object appearance can change across locations. The open-vocabulary crosswalk prompt is particularly exposed to domain shift because the project documentation explicitly identifies it as unvalidated and not custom-trained. Future work should collect geographically representative data, especially for Indian road scenes and currency conditions.

C. Model Scheduling

Not every model needs to run at every frame. The configuration contains per-workflow scheduling controls such as every-N-frame execution for segmentation, obstacle detection, traffic processing, and background safety. This reduces computational load and creates predictable opportunities for the health monitor to observe processing time. A future performance study should measure the effect of these schedules on both model quality and end-to-end responsiveness.

D. Deployment Migration

The present laptop-centered design can serve as a reference implementation for future edge deployment. Migration should be incremental: first profile the current pipeline, then identify the most expensive stages, then test quantized or smaller alternatives, and finally compare end-to-end behavior. A smaller model is not automatically better if its false-negative rate increases in situations where the user depends on the output. Deployment optimization must therefore be evaluated jointly with task reliability.

XIX. INTERACTION DESIGN AND USER EXPERIENCE

A. Cognitive Load

Audio is the primary feedback channel, so the interaction design avoids continuous narration. Guidance is short, rate-limited, and prioritized. The user should receive a message when it changes a decision or provides actionable context rather than hearing every frame described. This principle is reflected in separate timing policies for blind-path guidance, object search, currency, and urgent messages.

B. Command Simplicity

Fixed commands are intentionally short and predictable: “start navigation,” “help me cross,” “check the light,” “find my keys,” “read currency,” and destination-oriented map phrases. The implementation also permits typed/API commands and browser buttons. This multimodal control design is useful for research because it separates perception failures from speech-recognition failures: if voice is unavailable, the same task can still be activated through another input path.

C. Feedback Consistency

Every successful mode switch produces a beep and spoken confirmation. This is important for a blind user because a silent mode transition would make it difficult to know which workflow owns the camera. When a task changes, stale queued speech is cleared. This prevents a route instruction from being spoken after the system has already entered object search, or a search prompt from appearing after a cancellation.

D. Human Factors Evaluation

The current project does not include a formal participant study. A future study should measure task completion, error recovery, workload, learning time, and subjective trust. The study should distinguish between perceived usefulness and actual task performance. Participants should also be allowed to report when a message is too frequent, too late, too ambiguous, or difficult to act upon. These observations can inform thresholds and phrasing without changing the core state-machine architecture.

E. Accessibility Beyond Vision

The prototype is primarily designed around visual impairment, but interaction choices also affect users with hearing or speech constraints. The browser buttons and typed/API path provide alternatives to voice input, while the visual UI can display current guidance. Future hardware work could add physical controls and validated haptic feedback, but the current vibration controller remains a logging stub and should not be represented as functional haptics.

XX. SOFTWARE QUALITY, TESTING, AND MAINTAINABILITY

A. Test Coverage Strategy

The project repository contains tests for NavigationMaster, intent matching, health checks, map navigation, network audio, traffic safety, TTS priority, blind-path workflow, and currency workflow. These tests are particularly valuable because state transitions and safety invariants can be verified without requiring a full vision model for every test. A future release should add regression cases for every bug found during field testing.

For each new feature, the project architecture recommends changing the task definition, start method, frame-processing branch, API action mapping, browser control, and corresponding test where applicable. This checklist reduces the risk that a new mode exists in one layer but is inaccessible or inconsistent in another. It also preserves the single-source-of-truth principle of NavigationMaster.

B. Logging and Recording

The recorder can store annotated MP4 frames and JSONL event logs containing guidance, main mode, and task information. These artifacts are useful for post-hoc analysis because they can align a visual frame with the system's spoken decision. A future evaluation should add timestamps for model start/end, decision creation, TTS queue insertion, and actual speech start so that end-to-end latency can be decomposed.

C. Regression Risks

The most important regression risks are unintended concurrent tasks, stale TTS messages, broken fallback behavior, loss of crossing mode-lock, changes to command-pattern order, and configuration drift between documentation and code. The project materials already identify several documentation/code mismatches, including older obstacle-model references and incomplete environment-variable documentation. Future releases should treat configuration and model versions as reproducibility artifacts rather than informal notes.

D. Maintainability

The modular structure provides a practical boundary for future research. Perception workflows can be benchmarked independently, NavigationMaster can be tested with synthetic events, and the browser can be changed without rewriting the model code. This separation also makes it easier to compare alternative models while keeping the user interaction and safety logic fixed. For a research project, such controlled substitution is preferable to rewriting the entire application for every experiment.

TABLE IX. RECOMMENDED SOFTWARE RELEASE ARTIFACTS

Artifact	Purpose
Source commit/hash	Identifies exact software version
requirements lock / environment record	Reconstructs dependency environment
Model manifest	Records checkpoint names and versions
Configuration snapshot	Records thresholds and scheduling parameters without secrets
Test report	Records deterministic test outcomes
Evaluation dataset manifest	Defines images/sequences and train/test separation
Annotation protocol	Defines ground-truth labeling rules
Run log	Connects experiment ID to hardware, configuration, and outputs
Failure log	Records false positives, misses, degraded states, and user cancellations

XXI. RESEARCH METHODOLOGY FOR A FOLLOW-UP STUDY

A. Phase 1 — Dataset Construction

The first phase should build separate datasets for the modules that require domain-specific validation. Walkable segmentation needs pixel-level masks or a suitable benchmark proxy. Obstacle detection needs frame-level bounding boxes for relevant categories. Crosswalk and traffic-light data should represent local road conditions, including different sign styles, lighting, occlusion, and camera heights. Object search requires target labels and temporal sequences, while currency recognition requires denomination and authenticity labels if authenticity estimation is to be evaluated.

B. Phase 2 — Module-Level Benchmarking

Each perception component should be evaluated independently before the full application is measured. This makes it possible to answer whether a failure came from the model, threshold selection, state logic, or user interaction. For example, if object search fails, the evaluation should distinguish between missed detection, incorrect tracking, incorrect hand localization, and inappropriate audio phrasing. Such decomposition is more informative than a single end-to-end success percentage.

C. Phase 3 — End-to-End Scenario Tests

The full system should then be tested using scripted scenarios. Each scenario should specify initial state, user command, environmental condition, expected task transitions, acceptable guidance, and termination condition. Scenarios should include normal and failure cases. The latter are especially important: unplugging the camera, denying location permission, removing currency checkpoints, interrupting speech, and changing tasks rapidly should all produce controlled responses rather than undefined behavior.

D. Phase 4 — Human Evaluation

After technical validation, a human study can assess whether the system's guidance is understandable and useful. The study should be conducted with appropriate ethical approval and should not ask participants to rely on the prototype for real road crossing without suitable safeguards. A controlled indoor or supervised environment can be used to measure task completion and interaction burden before any outdoor study.

E. Statistical Reporting

A publication-ready study should report the number of trials, train/test separation, confidence intervals where appropriate, failure categories, and experimental hardware. If several model configurations are compared, the same test set and evaluation protocol should be used. The manuscript should report negative results and failure cases instead of selecting only successful demonstrations. This approach will make EYRO's contribution more credible and easier to compare with future assistive systems.

XXII. DESIGN RATIONALE: WHY THE COMPONENTS ARE COMBINED

A. Segmentation Plus Detection

A walkable-area mask and an obstacle detector answer different questions. Segmentation provides a region-level description of where the user can potentially move, while object detection identifies discrete hazards. Combining them allows the decision layer to reason about path structure and localized obstacles without requiring a full 3-D reconstruction of the environment.

B. Open Vocabulary Plus Tracking

Object search needs semantic flexibility because the user can ask for different objects. Open-vocabulary detection provides the target selection mechanism, while tracking stabilizes that target across consecutive frames. Tracking also reduces the need to speak a completely new direction on every frame. This combination is therefore a user-interaction choice as much as a computer-vision choice.

C. Classical Colour Analysis After Detection

The traffic-light module uses YOLO11n to localize the light and classical image analysis to estimate colour. The separation is deliberate: object localization and colour-state estimation are different subproblems. The project retains an UNKNOWN state when the ROI evidence is insufficient, which is preferable to forcing a classifier output from ambiguous pixels.

D. State Machine Plus Priority TTS

The state machine determines which task is authoritative; the priority queue determines which message is urgent. These responsibilities are related but should not be merged. A state transition can clear stale speech, while an urgent obstacle message can pre-empt normal speech without changing the active task. Keeping these concerns separate makes the system easier to test and reason about.

E. Local Processing Plus Optional Services

Most perception and speech processing is local to the laptop. Map routing uses an external service when configured, and chat can use local or optional cloud backends. This arrangement minimizes mandatory network dependence while retaining capabilities that benefit from external geographic data or conversational models. It also creates a clear boundary for future offline deployment: map and optional chat services can be replaced independently of the core vision pipeline.

XXIII. COMPARISON WITH THE SELECTED BASE PAPER: DETAILED DISCUSSION

A. Scope Difference

The base paper concentrates on smart-glass deployment, object detection, distance estimation, OCR, and audio feedback, with quantitative comparison of YOLOv8 variants on a campus dataset. EYRO instead focuses on a broader set of assistive tasks and a unified software controller. The difference is not simply “more features.” EYRO changes the engineering problem from optimizing one wearable perception pipeline to coordinating multiple short-lived and continuous tasks under one interaction policy.

B. Hardware Difference

The base paper uses Vuzix Blade 2 as the execution platform. EYRO uses a Windows laptop as the AI processor and allows camera, microphone, and button inputs to be local or network-connected. This makes EYRO less self-contained but easier to develop, debug, and extend. It also means that claims about wearable portability cannot be transferred from the base paper to EYRO.

C. Method Difference

The base paper evaluates YOLOv8-N, YOLOv8-S, and YOLOv8-M and integrates OCR. EYRO uses YOLO11n for obstacle and traffic-related detection, YOLOE for open-vocabulary functions, SegFormer-B0 for semantic segmentation, ByteTrack for tracking, and MediaPipe HandLandmarker for hand guidance. The model stack therefore reflects a multi-task architecture rather than a single detector comparison.

D. Contribution Difference

The most defensible EYRO contribution is the explicit task-control layer and its integration with several assistive workflows. The system adds map navigation, path segmentation, object search with hand guidance, experimental currency recognition, background obstacle safety, and a health monitor. These contributions should be evaluated empirically in future work, but their presence in the implementation is supported by the project documentation.

E. What EYRO Does Not Claim

EYRO does not claim to outperform the base paper numerically. It does not claim better object-detection accuracy, lower latency, better OCR, or superior wearable hardware. The comparison is architectural and functional. This distinction is necessary because the base paper and EYRO use different datasets, hardware, model stacks, and evaluation conditions.

XXIV. ETHICAL, PRIVACY, AND RESPONSIBLE-USE CONSIDERATIONS

A. Camera and Bystander Privacy

An assistive camera can capture people and private surroundings even when the user is not intentionally recording them. The prototype should therefore be operated with clear local policies regarding recording, storage, and demonstration environments. The recording module should remain disabled unless needed for development or evaluation, and collected videos should be handled as research data rather than casually shared.

B. Audio Privacy

Microphone input can contain bystander speech as well as user commands. The local processing design reduces dependence on cloud speech services, but local storage and event logging can still contain sensitive information. Evaluation protocols should avoid collecting unnecessary conversations and should document retention and deletion procedures.

C. Financial Information

Currency recognition introduces a special responsibility because a wrong authenticity or denomination estimate could affect a financial decision. The current system explicitly labels its output as an AI estimate and suppresses uncertain identification. A future validated system should test both false positives and false negatives and should never imply certification of banknote authenticity without appropriate evidence.

D. Road-Safety Boundaries

The prototype should not be used as the sole basis for crossing a road. The crossing logic itself reflects this boundary by avoiding unconditional “safe” language. Future studies should use supervised environments until the system has been independently validated for the intended application.

E. Responsible Publication

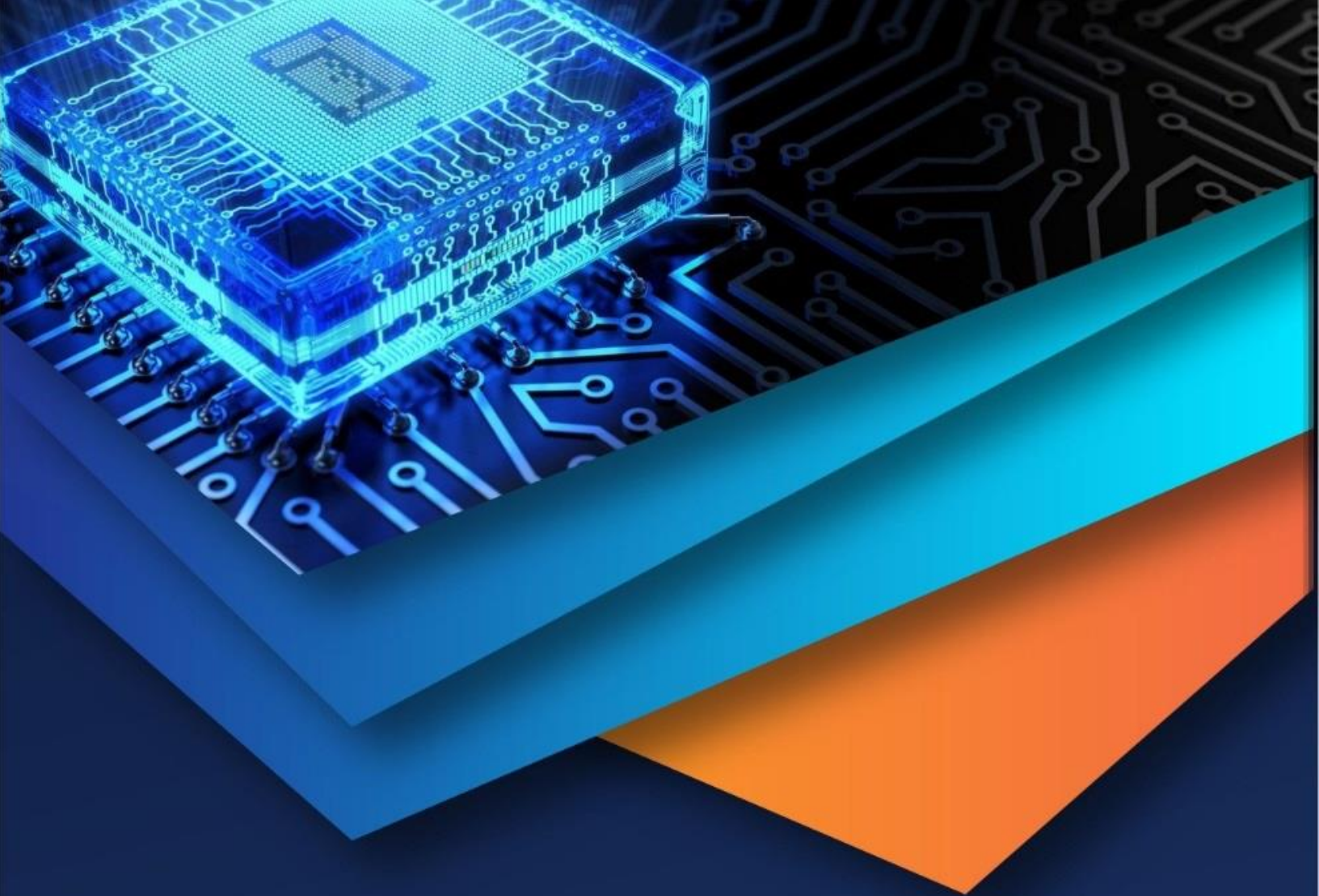
A strong research paper should disclose both capabilities and limitations. Reporting only successful demonstrations would overstate the system. The present manuscript therefore includes the absence of project-specific quantitative results, the unvalidated crosswalk prompt, the box-size vehicle proxy, the currency limitations, and the laptop-centered architecture. These disclosures are part of the scientific contribution because they define the conditions under which the prototype should be interpreted.

XXV. ACKNOWLEDGMENT

The authors thank the project guide, Priya Meshram, for guidance and review during the development of the EYRO prototype. The authors also acknowledge the open research and software ecosystems that provide the models and tools used in the implementation.

REFERENCES

- [1] P. Kumari and R. Hammady, “Assisting blind people with AI and audio using smart glasses: system design with YOLOv8 variants comparisons,” *Multimedia Systems*, vol. 32, art. 73, 2026.
- [2] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.
- [3] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, “ByteTrack: Multi-Object Tracking by Associating Every Detection Box,” in *European Conference on Computer Vision (ECCV)*, 2022.
- [4] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “MediaPipe Hands: On-device Real-time Hand Tracking,” *arXiv:2006.10214*, 2020.
- [5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, “Searching for MobileNetV3,” in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.
- [6] A. Radford et al., “Robust Speech Recognition via Large-Scale Weak Supervision,” in *Proc. 40th International Conference on Machine Learning, PMLR 202*, pp. 28492–28518, 2023.
- [7] A. Wang, L. Liu, H. Chen, Z. Lin, J. Han, and G. Ding, “YOLOE: Real-Time Seeing Anything,” *arXiv:2503.07465*, 2025.
- [8] G. Jocher and J. Qiu, *Ultralytics YOLO11*, Ultralytics, 2024. Official documentation and software release information.
- [9] openrouteservice, “Directions Service and API Documentation,” *openrouteservice API documentation*, accessed 2026.
- [10] M. Zöllner, S. Huber, H.-C. Jetter, and H. Reiterer, “NAVI—A proof-of-concept of a mobile navigational aid for visually impaired based on the Microsoft Kinect,” in *INTERACT 2011, Part IV*, Lisbon, Portugal, 2011.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)