ECE445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

OpenAI Glasses for Navigation

Team #3

Jiashen Ren [jiashen3]
Haoyu Zhu [haoyu13]
Jinnan Zhang [jinnanz2]
Darui Xu [daruixu2]

Advisor: Timothy Lee

April 1, 2026

Contents

1. Introduction

1.1 Problem Statement

This senior design project targets a practical assistive-navigation problem: a visually impaired user needs timely and interpretable guidance while walking along tactile paving, approaching a crosswalk, waiting for a safe opportunity to cross, and searching for a requested object. Existing smartphone-based solutions usually solve only one piece of that workflow at a time, such as generic object recognition or turn-by-turn route planning, and they often assume stable framing, a free hand, or a screen-based interface. For a wearable system, those assumptions are weak. The camera view is head-mounted and noisy, the user needs hands-free audio feedback, and guidance has to remain understandable when the scene switches rapidly between sidewalk, intersection, and near-field object manipulation.

The OpenAI Glasses for Navigation senior design project addresses this by integrating a wearable camera and microphone with a server-side real-time perception stack. The design combines three concrete capabilities in one assistive workflow: blind-path following with obstacle avoidance, crosswalk alignment with traffic-light gating, and voice-triggered item search with hand guidance. The challenge is therefore not only detection accuracy, but also state coordination. The system must choose when to stay in conversational mode, when to follow tactile paving, when to promote a crosswalk cue into a crossing workflow, and when to suspend navigation to help the user search for an object. This document focuses on that coordination and on the hardware and services that support it.

1.2 Solution Overview & Visual Aid

The system is organized around a wearable sensing front end and a centralized perception and orchestration back end. A XIAO ESP32S3 Sense based device captures VGA JPEG video, 16 kHz microphone audio, and IMU data, then sends those streams to an application server. The server buffers the data, serves browser-based monitoring, and forwards each frame into a stateful navigation controller. That controller selects one of several workflows: blind-path navigation, crosswalk approach and crossing, traffic-light waiting, item search, or normal conversation. Spoken feedback is produced either from local prompt audio or from the DashScope/Qwen multimodal service.

At the perception level, the design deliberately mixes task-specific modules instead of forcing one model to solve every scene. Blind-path and crosswalk geometry are obtained from a segmentation model, obstacle filtering is handled by a YOLO-E based detector with a whitelist, traffic-light color is stabilized with majority voting, and object search uses open-vocabulary prompts plus MediaPipe hand landmarks to guide the user toward the target. This division keeps each task explainable and debuggable. Figure 1 shows the full closed loop.

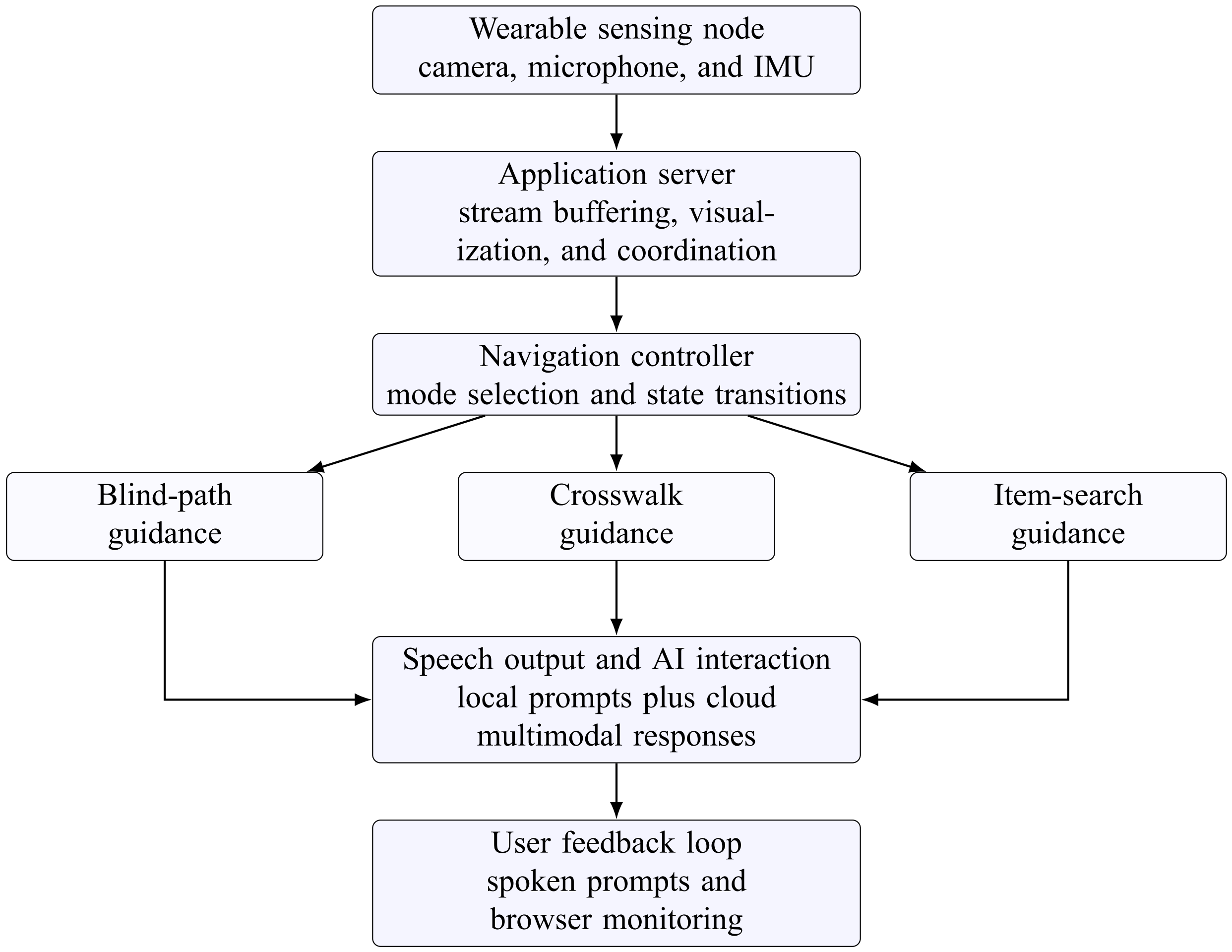
Wearable sensing node
camera, microphone, and IMU
Application server
stream buffering, visualization, and coordination
Navigation controller
mode selection and state transitions
Blind-path guidance
Crosswalk guidance
Item-search guidance
Speech output and AI interaction
local prompts plus cloud multimodal responses
User feedback loop
spoken prompts and browser monitoring

Figure 1: Overall solution overview for the navigation glasses system.

1.3 High-Level Requirements List

1. **Hands-free navigation requirement:** the system shall accept live wearable video and audio, detect tactile paving or crosswalk structure from the incoming scene, and produce spoken guidance frequently enough to be useful during walking. In the current design this is reflected by 16 kHz, 20 ms audio chunking, blind-path guidance intervals of 4.0 s for straight motion and 3.0 s for corrective direction prompts, and frame-by-frame state orchestration.

2. **Safe crossing requirement:** the system shall promote a crosswalk cue into a crossing workflow only after repeated evidence that the user is aligned and at the curb, and it shall not announce crossing permission on a single unstable traffic-light observation. The design implements this through staged crosswalk thresholds, angle/offset alignment checks, and a five-frame stable green-light requirement before crossing guidance is issued.

3. **Interactive assistance requirement:** the system shall support voice-triggered item search and multimodal conversation without forcing the user to stop wearing the device or use a touchscreen. The design supports this through streaming ASR, multimodal speech replies, hotword-based interruption, and an item-search workflow that combines open-vocabulary object detection with MediaPipe hand landmarks and grasp confirmation.

2. Design

2.1 Block Diagram

The design uses a layered architecture rather than a monolithic application. The wearable node is intentionally simple: it captures sensor data, performs lightweight compression and streaming, and leaves heavy inference to the host. On the host side, the application server acts as the integration point for transport, buffering, recording, web visualization, and orchestration. Below that service, the design separates state control from perception. The navigation controller owns the transition logic, while the blind-path, crosswalk, and item-search blocks each implement their own task-specific perception heuristics. Figure 2 summarizes these layers and their major interfaces.

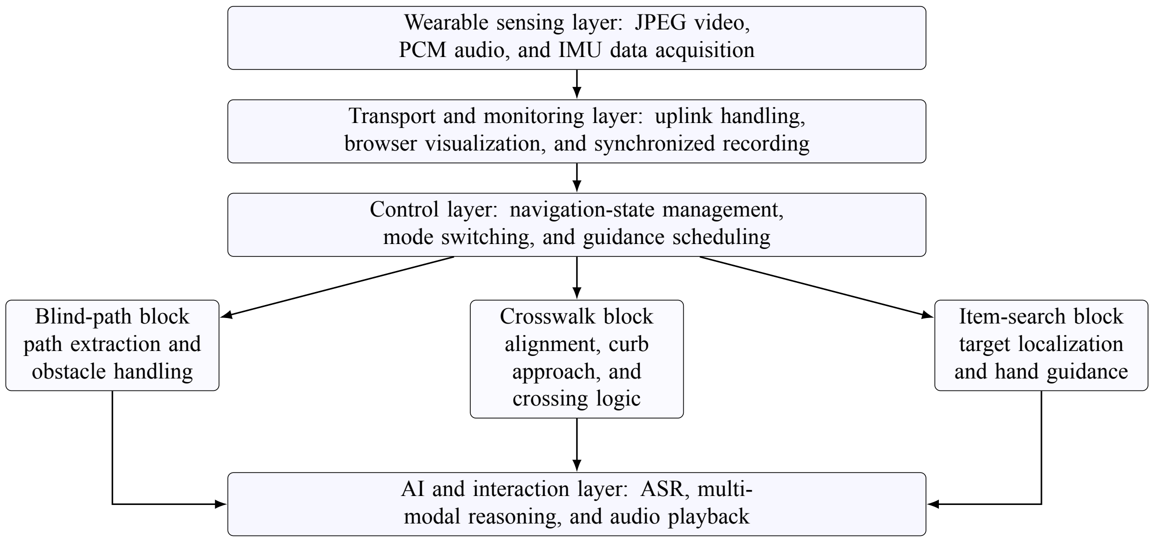
Wearable sensing layer: JPEG video, PCM audio, and IMU data acquisition
Transport and monitoring layer: uplink handling, browser visualization, and synchronized recording
Control layer: navigation-state management, mode switching, and guidance scheduling
Blind-path block path extraction and obstacle handling
Crosswalk block alignment, curb approach, and crossing logic
Item-search block target localization and hand guidance
AI and interaction layer: ASR, multi-modal reasoning, and audio playback

Figure 2: Top-level block diagram of the implemented system.

2.2 Subsystem Description 1: Wearable Sensing and Streaming

2.2.1 Hardware Support

The wearable front end is built around the XIAO ESP32S3 Sense family. The device configures the camera for VGA JPEG capture, uses a 16 kHz PDM microphone with 20 ms audio chunks, and drives an I2S speaker/amplifier chain for playback. The same hardware also streams IMU measurements from an ICM42688 over SPI to the host through UDP. On the server side, the system provides separate channels for camera uplink, audio uplink, browser monitoring, and mixed audio playback. This split keeps the wearable lightweight while allowing a laptop or GPU workstation to run the heavier models.

2.2.2 Functional Logic

The sensing path uses producer-consumer style queues for captured JPEG frames and PCM audio chunks. The host mirrors that structure with a thread-safe bridge and a synchronized recorder, which lets the application serve multiple browser viewers while still preserving raw input and processed output for debugging. This is a strong design choice for a senior design project because it reduces coupling between transport and inference. If a browser disconnects, the navigation loop

can continue; if model inference stalls, the transport layer can still detect queue pressure and drop old frames rather than freezing. The same structure also makes the data plane explicit: the wearable sends raw sensing data upstream while the browser receives processed monitoring output.
The audio path is similarly segmented. Real-time ASR consumes 16 kHz PCM, the AI response path streams text and audio from a multimodal service, and the local audio player can mix pre-recorded prompts with service-generated speech. This prevents the common failure mode where every subsystem tries to own the speaker at the same time. The design therefore includes a hard reset path and playback-state checks.

2.2.3 Verification Plan

This subsystem should be verified with transport-focused tests before any perception test is considered valid. First, the wearable shall stream VGA JPEG frames continuously to the host for at least 10 minutes without exhausting the frame queue or dropping the WebSocket connection. Second, the audio path shall preserve the expected 20 ms chunk structure, which corresponds to 640 bytes of 16-bit mono PCM at 16 kHz. Third, the browser monitor shall show processed frames and IMU updates concurrently, demonstrating that the visualization path is not blocking navigation. Finally, synchronized audio/video recording should be checked by replaying a short walking session and confirming that speech prompts still line up with visible state changes.

2.3 Subsystem Description 2: Navigation Perception and State Orchestration

2.3.1 Hardware Support

This subsystem executes on the host machine, which is assumed to run PyTorch, Ultralytics YOLO models, OpenCV, and optionally CUDA. The main model assets are a blind-path/crosswalk segmentation model, a YOLO-E open-vocabulary model for obstacles, and a traffic-light detector with HSV fallback logic. Around those assets sits a navigation controller, a blind-path perception block, a crosswalk perception block, and a crosswalk-awareness helper path for early voice cues.

2.3.2 Functional Logic

The key architectural decision here is to separate *state selection* from *task-specific perception*. A central controller owns the global conversation, blind-path, crosswalk, waiting, crossing, recovery, and item-search states. Each frame is dispatched to the appropriate workflow, and transitions are guarded by counters and cooldown timers so that the user does not hear contradictory commands from one unstable frame.
Inside the blind-path block, the system combines segmentation masks, Lucas-Kanade optical-flow stabilization, geometric centerline extraction, obstacle filtering, and staged crosswalk awareness. Straight-ahead guidance is intentionally rate-limited to 4.0 s, while corrective commands are allowed every 3.0 s. The blind-path workflow also defines explicit onboarding and navigation thresholds: an orientation threshold of 10° and a lateral offset threshold of 15% of image width. These are practical engineering thresholds for a head-mounted camera because they tolerate moderate head motion without forcing constant turn commands.
The crosswalk block is stricter as the user approaches the curb. The design uses a looser alignment window during the seeking phase (15° angle and 0.20 normalized offset) and a tighter one during crossing guidance (5° and 0.08). It also uses geometric size checks to decide when the crosswalk is actually near: at least 30% image area, a bottom edge beyond 80% of image height, and a height

ratio above 35%. This is a sound design because it converts a noisy segmentation output into a user-relevant event: "the crosswalk is now close enough that I should align precisely." When the user reaches the curb, the controller waits for a stable green-light classification before issuing crossing guidance. The traffic-light logic uses majority voting over recent observations, and the crosswalk workflow requires five stable green frames before prompting motion.

2.3.3 Verification Plan

Verification for this subsystem should be divided into scenario tests. For blind-path navigation, pre-recorded and live videos should be used to confirm that the system can move from onboarding into stable navigation and produce left/right/straight cues that agree with the visible path. For obstacle handling, test clips with people, bicycles, and parked vehicles should verify that the whitelist-based detector only announces relevant objects that overlap the drivable path region. For crosswalk behavior, the verification sequence should explicitly check three transitions: blind path to seeking crosswalk, seeking crosswalk to waiting for light, and waiting for light to crossing. The measured outputs are state transitions, spoken guidance, and the annotated frame. Because the thresholds are implemented directly in code, pass/fail can be based on whether the internal counters and thresholds trigger in the intended order rather than on a subjective judgment alone.

2.4 Subsystem Description 3: Multimodal Interaction and Item Search

2.4.1 Hardware Support

This subsystem uses both local and cloud resources. Locally, the host runs MediaPipe hand landmarking, YOLO-based open-vocabulary inference, and the shared audio player. Externally, the system relies on DashScope Paraformer for streaming ASR and Qwen-Omni-Turbo for multimodal response generation. Functionally, this subsystem contains a speech-recognition stage, a label-extraction stage, an audio playback stage, and a near-field item-search stage.

2.4.2 Functional Logic

The voice path is designed around *interruptibility* and *mode isolation*. The ASR stage distinguishes partial text, final text, and hotword interrupts. The hotword list includes stop phrases and triggers a full reset instead of letting stale speech continue into the language model. That design choice is particularly important for an assistive device because the user needs a simple verbal way to silence or reset the system while walking.

The item-search path begins when the user speaks a request such as "help me find a drink." The system extracts or translates an English label for the detector, sets the item-search mode in the controller, and then runs a dedicated near-field guidance pipeline. That pipeline uses segmentation or open-vocabulary detection to localize the object, MediaPipe landmarks to estimate the user's hand position, and optical-flow-based relocking to stabilize the target polygon. Guidance is expressed as alignment cues plus a distance-like score based on the ratio between object area and hand area. The design defines an acceptable interaction band of 1 ± 0.25 for the object-to-hand area ratio; below that band the user is told to move forward, above it the user is told to move back. Once hand overlap and grasp heuristics become stable, the system announces successful acquisition. This is a strong choice for a wearable assistant because it converts raw vision outputs into an action the user can perform without seeing the screen.

2.4.3 Verification Plan

This subsystem should be tested with a set of spoken Chinese commands and everyday objects. Verification steps include: confirming that ASR partial results appear in the UI while only final results trigger a new AI task; checking that stop hotwords interrupt playback and clear the current interaction; verifying that an object request switches the system into item-search mode; and measuring whether the hand-guidance prompt changes correctly as the user moves closer, farther, left, and right relative to the object. A successful demonstration should end with either a stable grasp detection or a clean user-issued cancellation command.

2.5 Supporting Material

Table 1 summarizes the main system interfaces that tie the subsystems together. Listing them explicitly is useful for integration because the senior design system is already large enough that failures often occur at the boundary between subsystems rather than inside a single algorithm.

Interface	`Interface Type`	Purpose
Wearable video uplink	`JPEG frame stream`	Sends forward-facing camera imagery to the host for perception and recording.
Wearable audio uplink	`16 kHz PCM stream`	Sends microphone audio for ASR, logging, and interaction control.
Viewer stream	`Browser monitoring stream`	Broadcasts processed video and status overlays to monitoring clients.
Audio output stream	`Mixed speech channel`	Delivers local prompts and AI-generated speech to the user.
State orchestration	`Controller interface`	Selects the active behavior and guards transitions with counters and cooldowns.
Blind-path perception	`Segmentation guidance block`	Produces centerline, obstacle, and crosswalk-awareness guidance from the live frame.
Crosswalk perception	`Alignment and crossing block`	Computes crosswalk angle/offset, monitors signal state, and determines crossing completion.
Item search	`Near-field guidance block`	Localizes target objects and hand landmarks for near-field assistance.
Voice interaction	`Speech/AI interface`	Performs streaming ASR, hotword reset, and multimodal text/audio response generation.

Table 1: Key system interfaces in the navigation glasses design.

2.6 Tolerance Analysis

The most important tolerances in this design are geometric rather than electrical. The user does not care about pixel coordinates directly; the user cares whether the spoken guidance changes too early, too late, or too often. The software therefore converts pixel geometry into normalized errors and only changes state when those errors remain inside a usable band.

For blind-path alignment, the relevant error is the normalized horizontal displacement between the desired centerline target and the image center:

$$e_{\text{path}} = \frac{|x_{\text{target}} - W/2|}{W}. \tag{1}$$

The current implementation accepts blind-path alignment when $e_{\text{path}} < 0.15$. At 600-pixel image width, this means the target may deviate by about 90 pixels before a correction is required. This is intentionally tolerant because a head-mounted camera naturally oscillates during walking.
For crosswalk alignment, the design uses both angular and lateral tolerance:

$$|\theta_{\text{seek}}| < 15^\circ, \quad e_{\text{seek}} < 0.20 \tag{2}$$

during the approach phase, and

$$|\theta_{\text{cross}}| < 5^\circ, \quad e_{\text{cross}} < 0.08 \tag{3}$$

when the user is close enough to cross. The tighter second threshold is necessary because a few degrees of heading error can translate into a large lateral drift over the width of a road.
For item search, the near-field tolerance is based on the ratio between object area and hand area:

$$s_{\text{range}} = 1 - \min\left(1, \frac{\left|\frac{A_o}{A_h} - 1\right|}{0.25}\right), \tag{4}$$

where A_o is the object area and A_h is the hand area. A ratio band of 1 ± 0.25 is wide enough to allow for hand-pose variation but narrow enough to prevent the user from overshooting the object by a large margin. These tolerances are all software-adjustable and should be tuned further after closed-loop user testing with the final wearable form factor.

3. Cost

Table 2 lists an estimated bill of materials for the wearable portion of the system. The host laptop or GPU workstation is treated as existing lab infrastructure and is therefore excluded from the hardware subtotal. Cloud API usage is included separately because it directly affects the feasibility of the senior design demonstration.

Item	Qty.	Unit Cost	Extended Cost
Seeed XIAO ESP32S3 Sense wearable controller/camera module	1	$18	$18
ICM42688 IMU breakout or equivalent inertial sensor	1	$10	$10
MAX98357A I2S audio amplifier	1	$4	$4
Bone-conduction earphone or compact speaker/headphone assembly	1	$24	$24
Li-ion battery pack, charger, and protection hardware	1	$12	$12
Mounting hardware, wiring, connectors, and prototype PCB	1	$12	$12
3D-printed frame or wearable fixture materials	1	$15	$15
Prototype wearable subtotal			**$95**
DashScope API credits and network/service contingency	1	$20	$20
Estimated system total			**$115**

Table 2: Estimated cost for the senior design navigation glasses system.

This cost is reasonable for a senior design build because the expensive computation is offloaded to an existing host machine. If the system were migrated fully on-device, the cost would rise substantially due to the need for a more capable embedded compute platform and a more integrated industrial design. For the current graduation-project scope and demonstration goals, the split architecture is the more pragmatic choice.

4. Schedule

The development sequence suggests a semester-long integration path in which data transport and device bring-up came first, followed by navigation logic, then multimodal interaction and item search. Table 3 captures that schedule in a form suitable for the design document.

Time	Primary Goal	Expected Deliverable	Status
Weeks 1–2	Requirements definition and architectural decomposition	Project scope, mode list, endpoint plan, and initial hardware selection	Completed
Weeks 3–5	Wearable bring-up and transport backbone	Camera/audio streaming, browser viewer, synchronized recorder	Completed
Weeks 6–8	Blind-path navigation	Segmentation pipeline, centerline extraction, obstacle filtering, spoken direction cues	Completed
Weeks 9–11	Crosswalk approach and crossing logic	Crosswalk geometry estimation, traffic-light gating, transition logic in the central controller	Completed
Weeks 12–13	Multimodal interaction and item search	ASR hotword reset, Qwen-Omni streaming, open-vocabulary search, hand guidance	Completed
Weeks 14–16	Integration hardening and demo preparation	Threshold tuning, safety messaging, UI cleanup, end-to-end demonstration recording	In progress

Table 3: Development schedule for the senior design project.

The remaining work after this design stage is not a major architectural rewrite. It is primarily validation work: reducing false prompts in cluttered scenes, tuning thresholds with more walking footage, and documenting operating limits clearly enough that the system is not mistaken for a fully validated mobility aid.

5. Ethics and Safety

5.1 Ethics

This senior design project deals with sensitive sensory data and a user population for whom system errors carry a higher consequence than in a typical consumer demo. The system should therefore be described as an assistive graduation-design prototype for engineering validation, not as a finished mobility device. The system streams first-person video, environmental audio, and inferred user intent to a host machine and to cloud APIs, so privacy is a primary concern. Any future deployment should clearly disclose what data leaves the device, what is stored for debugging, and how long those recordings persist.

There is also a fairness and robustness issue. The perception subsystems are trained or tuned on limited datasets and hand-coded thresholds, which means performance will vary across lighting conditions, weather, sidewalk materials, intersection layouts, and object appearance. Overstating the system as a general mobility replacement would therefore be misleading. The ethical position for this senior design project is to describe it as an assistive prototype, not an autonomous mobility device.

5.2 Safety

The main safety hazards are false crossing permission, contradictory directional prompts, and over-reliance on cloud services during a time-critical task. The present architecture includes several mitigations. First, state transitions use counters and cooldowns so that one unstable frame does not switch the user from sidewalk following into crossing. Second, green-light permission is gated by repeated detection rather than a single classification. Third, the ASR layer supports explicit stop hotwords so the user can silence the system quickly. Fourth, the item-search mode is isolated from navigation mode, reducing the chance that near-field grasp guidance will compete with road-crossing prompts.

Even with these mitigations, the current senior design prototype should be treated as advisory only. A safe operating procedure would require the user to confirm intersections independently, require an assistant during testing, and avoid using the system in dense traffic or unfamiliar environments. Before any real deployment, the system would need human-subject testing, reliability metrics under varied scenes, and a formal failure-mode review. Until then, the safest framing is that this system assists orientation and experimentation but does not replace a cane, guide dog, or established mobility practice.

References

[1] S. Ramrez, “FastAPI,” online documentation. [Online]. Available: https://fastapi.tiangolo.com/

[2] Ultralytics, “Ultralytics YOLO Docs,” online documentation. [Online]. Available: https://docs.ultralytics.com/

[3] Google, “MediaPipe Hand Landmarker,” online documentation. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

[4] Alibaba Cloud, “DashScope Developer Guide,” online documentation. [Online]. Available: https://help.aliyun.com/zh/model-studio/

[5] Qwen Team, “Qwen-Omni Technical Overview,” online documentation. [Online]. Available: https://qwenlm.github.io/

[6] 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 *Proc. ECCV*, 2022.

[7] B. D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision,” in *Proc. Imaging Understanding Workshop*, 1981.

[8] Seeed Studio, “XIAO ESP32S3 Sense,” hardware documentation. [Online]. Available: https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/

[9] TDK InvenSense, “ICM-42688-P Datasheet,” hardware documentation. [Online]. Available: https://invensense.tdk.com/