Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

WVAB: A Low-Latency Wireless Assistive Vision Framework for Risk-Aware Navigation and Multilingual Audio Guidance

Md Shahanur Islam Shagor

smshagor.ru@gmail.com

Voronezh State University of Forestry and Technologies https://orcid.org/0009-0003-7730-3202

Research Article

Keywords: assistive vision, visual impairment, object detection, wireless multimedia, navigation assistance, multilingual audio feedback

Posted Date: August 31st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10854126/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Abstract

Affordable assistive vision requires more than accurate object detection: camera delay, wireless transport, decision logic, and spoken feedback must remain temporally aligned if scene information is to be useful. This article presents WVAB, a host-assisted wireless assistive-vision framework that combines an ESP32-CAM sensing node, freshness-oriented UDP image transport, COCO-pretrained YOLOv8n inference, a deterministic C + + lane-risk NavigationPlanner, and offline multilingual speech generation. The study evaluates three questions: whether the processing pipeline satisfies a 200 ms prototype timing target, how camera source affects latency and processed throughput, and how closely planner outputs agree with manually annotated navigation commands. Timing was measured over 500 consecutive frames on an Intel Core i5 laptop with an NVIDIA GTX 1650. GPU processing-to-TTS-queue latency was 78.6 ± 14.1 ms at 22.3 FPS, compared with 161.3 ± 28.4 ms and 6.8 FPS in CPU-only execution. End-to-queue latency was 62.6 ± 13.1 ms for USB webcam input, 95.4 ± 16.2 ms for ESP32-CAM UDP, and 114.6 ± 19.7 ms for smartphone MJPEG. In 200 annotated video scenarios, planner–annotation agreement was 91.5% indoors and 87.3% outdoors. These findings establish prototype-level technical feasibility while also exposing important limits: the evaluation does not measure audible speech onset, calibrated depth, clinical safety, or mobility outcomes with blind or low-vision participants. The contribution is therefore a reproducible systems architecture and timing-aware evaluation, not a new detector or a validated replacement for established mobility aids.

1 Introduction

Vision impairment affects mobility, access to information, education, employment, and participation in everyday activities. The World Health Organization estimates that at least 2.2 billion people live with near or distance vision impairment, with at least 1 billion cases either preventable or still unaddressed [1]. Independent travel remains a particularly demanding task because useful environmental information changes quickly: a pedestrian enters a walking corridor, a vehicle approaches from the side, furniture constrains an indoor passage, or a temporary obstruction appears where the user expects free space.

The white cane, guide dog, orientation and mobility training, and human assistance remain foundational mobility resources. Electronic travel aids are best understood as complementary systems rather than replacements. Reviews spanning ultrasonic, infrared, wearable, camera-based, and multimodal devices repeatedly identify the same practical tension: richer sensing can provide more semantic information, but it also increases computational load, device complexity, power demand, cognitive burden, and the number of failure modes that must be handled [2–5]. These trade-offs are especially important in camera-based systems because recognizing an object is only one stage in a longer perception-to-feedback chain.

Computer vision has expanded the information that assistive systems can derive from ordinary scenes. Scene-understanding research for blind and visually impaired users now includes object recognition, obstacle detection, navigation, text access, and scene description [6]. Recent systems have coupled

RGB-D sensing, inertial measurements, time-of-flight sensors, haptics, smartphones, and deep neural detectors in different combinations [8–13]. These studies demonstrate substantial progress, but they also show that the design question is not simply which detector has the highest benchmark accuracy. A deployable assistive pipeline must decide which detections matter, keep frames current, produce stable spatial guidance, avoid filling an audio queue with stale announcements, and make its operating limits visible.

WVAB (Wireless Vision-Aid for the Blind) was developed around this systems problem. The evaluated configuration uses a low-cost ESP32-CAM as a wireless imaging node while a host computer performs object detection, deterministic lane-risk reasoning, and speech preparation. Smartphone MJPEG and direct USB video are retained as comparison inputs. The architecture intentionally separates camera acquisition, transport, detection, planning, and audio generation so that each stage can be timed and replaced independently. The principal contribution is not a new neural-network architecture. Rather, it is a timing-aware multimedia pipeline that integrates semantic perception with frame-freshness control, bounded navigation reasoning, and multilingual feedback.

1.1 Research questions

The study is organized around three research questions rather than a broad product demonstration:

RQ1. Can the evaluated host-assisted WVAB pipeline satisfy a predefined processing-to-TTS-queue target below 200 ms while sustaining at least 10 processed frames per second on the primary GPU configuration?

RQ2. How does the input path—direct USB, ESP32-CAM over UDP, or smartphone MJPEG—affect camera-to-TTS-queue latency and processed throughput under the reported local test conditions?

RQ3. How closely do deterministic NavigationPlanner outputs agree with manually annotated scene-level navigation commands in indoor and outdoor video scenarios?

1.2 Contributions and scope

The article makes five specific contributions. First, it defines a modular perception-to-feedback architecture in which sensing, transport, inference, navigation reasoning, and speech preparation have explicit interfaces. Second, it describes a freshness-oriented wireless video path that discards incomplete or superseded frames instead of allowing retransmission delay to accumulate. Third, it formalizes the C + + NavigationPlanner as a bounded heuristic risk model over left, centre, and right image regions. Fourth, it reports stage-level latency, camera-source timing, and throughput measurements rather than treating detector inference as a proxy for end-to-end responsiveness. Fifth, it reports a preliminary planner–annotation agreement study while clearly separating technical scene evaluation from clinical or human-subject mobility validation.

The reported experiments correspond to the host-assisted configuration used during the measurement campaign. The public WVAB repository has since evolved to a broader v2.0.0 platform with a standalone mobile application, authenticated and encrypted camera transport, and additional navigation research modules. Those later features are deliberately not mixed into the measurements in this paper. This boundary is important for reproducibility: a feature present in the current codebase is not presented as experimentally validated here unless it was part of the evaluated configuration.

2 Related work

2.1 Electronic travel aids and wearable design

Electronic travel aids have been studied for decades, initially with strong emphasis on proximity sensing and obstacle avoidance. Dakopoulos and Bourbakis [2] reviewed wearable obstacle-avoidance systems and highlighted real-time response, reliability, usability, portability, and cost as recurring evaluation dimensions. Elmannai and Elleithy [3] later surveyed sensor-based assistive devices and similarly identified limitations in capability, accuracy, and practical deployment. More recent reviews broaden the picture to navigation, localization, feedback modalities, mobile platforms, and deep learning [4, 7].

Hersh [5] argues that technical performance alone is insufficient when a device is intended to be worn or carried during travel. Weight, comfort, attachment, battery life, robustness, learnability, cultural and language factors, and compatibility with a long cane or guide dog all influence whether an assistive device is useful in practice. That observation motivates a conservative position in this article: the present study evaluates a multimedia processing pipeline, not the full human factors of a wearable product. The ESP32-CAM node is small enough to support wearable experiments, but wearability and acceptance are not inferred from its physical size.

2.2 Vision-based perception and navigation

Camera-based systems can provide semantic information that proximity-only sensors cannot. Bai et al. [8], for example, combined an RGB-D camera and inertial sensing with a smartphone to support both environmental perception and navigation. Katzschmann et al. [9] pursued a different design point using time-of-flight sensing and haptic feedback for local obstacle avoidance. Neugebauer et al. [10] evaluated visual-to-auditory sensory substitution in a pilot study with blind participants, illustrating both the promise and the difficulty of translating visual information into actionable audio. These approaches differ in sensing and feedback, but all must manage a conversion from dense environmental data to a limited human output channel.

Deep object detection has made semantic perception more practical on commodity hardware. The Microsoft COCO dataset provides a widely used benchmark and set of everyday object categories [14]. WVAB uses the Ultralytics YOLOv8n software implementation [15] with COCO-pretrained weights because the small model offers a practical accuracy–latency trade-off for the tested host. Importantly, this paper does not report COCO benchmark performance as an original WVAB result. The detector is an

upstream component; the research questions concern integration, timing, transport, and navigation reasoning.

Detection-focused research continues to improve models specifically for assistive scenes. Wang et al. [13] proposed YOLO-OD for obstacle detection in visually impaired navigation assistance, emphasizing architectural changes to improve detection of cluttered, occluded, and small objects. Such work is complementary to WVAB. A stronger detector could be substituted at the detection interface, but it would still need to satisfy the timing, prioritization, and feedback constraints of the end-to-end system.

2.3 Audio feedback and cognitive burden

Audio is attractive because it can communicate object identity and spatial instructions without occupying the hands, but it is also serial: a message spoken now may describe a scene that has already changed by the time it finishes. Human-centered studies show that assistive feedback must be evaluated not only for information content but also for cognitive load and task performance [10, 11]. Mobile accessibility research likewise shows that speech and camera-based functions are useful, while their usability depends on interaction design and user context [12].

WVAB addresses this constraint at the software level through filtering, cooldowns, bounded queues, and a concise command vocabulary. These mechanisms are engineering controls, not evidence that cognitive load has been reduced for blind users. Establishing that claim would require a controlled participant study. The distinction matters because technically reducing the number of queued messages is not equivalent to demonstrating lower cognitive workload.

2.4 System-level gap addressed by WVAB

Across the literature, there is no single dominant architecture. Some systems invest in richer depth sensing; others prioritize wearable form factor, haptics, mobile processing, or detector innovation. WVAB focuses on a narrower systems question: whether low-cost wireless image capture, host-side semantic detection, deterministic navigation reasoning, and multilingual speech preparation can be combined while keeping the pipeline temporally bounded. This makes the work particularly relevant to multimedia systems research, where acquisition, transmission, analysis, and user-facing media output must be considered together rather than as isolated modules.

Table 1
Representative assistive-navigation approaches and the specific gap addressed by WVAB

Study	Sensing / processing	Feedback / objective	Evaluation emphasis
Katzschmann et al. [9]	Time-of-flight wearable sensing	Haptic local navigation	Safety-oriented local obstacle avoidance with users
Bai et al. [8]	RGB-D + IMU + smartphone	Audio; navigation + object recognition	Integrated indoor/outdoor assistive device
Neugebauer et al. [10]	Mobile AR / visual-to-auditory substitution	Audio sensory substitution	Pilot navigation and recognition study with blind participants
Wang et al. [13]	Camera + specialized YOLO detector	Obstacle detection for navigation assistance	Detector architecture and public-dataset performance
WVAB (this study)	ESP32-CAM / smartphone / USB + host YOLOv8n	Lane-risk command + multilingual TTS queue	Stage timing, camera-source latency, throughput, preliminary planner agreement

The table is qualitative. Reported metrics across studies are not directly compared because sensing hardware, datasets, participants, and endpoints differ.

3 Problem formulation and design requirements

At time t, the visual scene is represented after detection as a set of objects E(t) = {o1, o2, ..., on}. Each object oi contains a semantic class ci, bounding box bi, and detector confidence confi. For a frame of width W and height H, the planner uses normalized horizontal position and apparent bounding-box area rather than raw pixel coordinates. The system must transform this changing set into a small command vocabulary I(t) ∈ {GO STRAIGHT, GO LEFT, GO RIGHT, SLOW, STOP} and a speech message suitable for asynchronous delivery.

The design is governed by two types of constraint. The first is computational: the system should keep recent frames moving through the pipeline without allowing network buffering or speech preparation to create unbounded lag. The second is epistemic: image occupancy is not equivalent to walkable ground, a large bounding box is not calibrated depth, and a detector confidence score is not collision probability. The planner therefore operates on heuristic risk values and must be described as decision support rather than autonomous navigation.

Table 2
Design requirements and study boundaries

Requirement	Operational interpretation
Timeliness	Prototype target below 200 ms from the relevant processing/camera timing start point to TTS queue insertion.
Processing rate	At least 10 fully processed frames per second for the primary wireless GPU path.
Semantic filtering	Only navigation-relevant detections are forwarded to the reported planner configuration.
Spatial reasoning	Apparent risk is aggregated across left, centre, and right image regions.
Speech control	Routine repetition is suppressed and the queue is bounded to reduce stale announcements.
Distance claim	Bounding-box area is used as an apparent-size heuristic, not as calibrated metric depth.
Cost claim	Approximately USD 15 refers to the ESP32-CAM sensing node, not the complete host-assisted system.
Validation boundary	Technical video and bench evaluation only; no blind or low-vision mobility trial is reported.

4 WVAB system architecture and methods

The evaluated pipeline is decomposed into Camera Acquisition (C), Transport and Frame Handling (T), Detection (D), Navigation Planning (N), and Audio Preparation (A). If F(t) denotes an input frame, the functional chain can be written as:

$$I(t) = A \circ N \circ D \circ T \circ C(F(t)) \quad (1)$$

The composition emphasizes interface separation rather than mathematical novelty. Each stage consumes a bounded representation from the previous stage, and the timing study records the contribution of major stages to the final queue-insertion endpoint.

4.1 Camera acquisition

The primary wireless source is an AI-Thinker ESP32-CAM with an OV2640 sensor. In the evaluated firmware, JPEG frames were captured at SVGA resolution (800 × 600) using the ESP32 camera quality parameter 10. Typical compressed frame sizes in the reported configuration were approximately 30–90 KB. Frames were decoded and resized to 640 × 480 on the host before detector inference. The approximately USD 15 hardware figure used elsewhere in the project applies only to this sensing node

and its basic supporting components; it excludes the host computer, audio device, power system, and mounting accessories.

Two alternative inputs were used to contextualize the wireless path. A Samsung Android smartphone supplied an HTTP MJPEG stream, representing a common network-camera workflow with a higher-quality consumer camera. A Logitech C920 USB webcam supplied frames directly through OpenCV, removing wireless transport from the timing path. Because the camera devices and protocols differ, the comparison is a systems comparison, not a protocol-only controlled experiment.

4.2 Freshness-oriented UDP transport

The ESP32-CAM path used an application-level UDP chunking scheme. UDP itself does not guarantee delivery, ordering, duplicate suppression, or congestion control [16, 17]. The evaluated WVAB prototype therefore optimized for frame freshness rather than reliable byte-stream completion: JPEG frames were split into chunks, identified by frame and chunk metadata, and reassembled only when complete. Incomplete or stale frames were discarded instead of delaying newer frames while waiting for retransmission.

The reported prototype permitted large application payloads that could approach IP-fragmentation territory on ordinary Wi-Fi networks. That design was acceptable for a controlled local experiment but is not presented as a production network recommendation. RFC guidance makes clear that UDP applications must account for path MTU, congestion, loss, and fairness [17]. The current public WVAB v2.0.0 codebase has subsequently hardened the remote-camera path with authenticated encryption and replay protection; those later changes are outside the timing data reported in this article.

4.3 Detection and navigation-relevant filtering

The host uses COCO-pretrained YOLOv8n [14, 15]. The detector returns bounding boxes, class labels, and confidence scores. The reported planner configuration applies a confidence threshold and an 11-class navigation whitelist before reasoning. This filtering has two purposes: it limits downstream computational and speech load, and it prevents the planner from treating every visually detectable object as equally relevant to movement.

For a retained bounding box with horizontal corners x1 and x2, width w, and height h, normalized horizontal centre xi and apparent area Ai are defined as:

x_i = (x_1 + x_2) / (2W)	**(2)**
A_i = (w_i h_i) / (WH)	(3)

Ai is a scale-normalized image-space quantity. It increases as an object occupies more of the frame, but it is affected by physical object size, camera focal length, viewpoint, truncation, and detector box quality. For that reason it is used only as a coarse proximity cue in the planner.

4.4 Risk-aware lane planner

The NavigationPlanner is implemented in C + + 17. It assigns each retained detection a class-dependent hazard weight W(ci), multiplies that weight by detector confidence, and amplifies the result according to apparent area. The bounded proximity term used in the evaluated planner is:

$$P(A_i) = 1 + \min(4A_i, 2) \quad (4)$$

The object-level heuristic risk score is then:

$$R_i = W(c_i) \cdot conf_i \cdot P(A_i) \quad (5)$$

The image plane is partitioned into three lateral regions. A detection belongs to the left lane when xi < 0.33, to the centre when 0.33 ≤ xi ≤ 0.67, and to the right when xi > 0.67. Regional scores are simple sums of object scores:

$$R_L = \Sigma R_i (o_i \in L), R_C = \Sigma R_i (o_i \in C), R_R = \Sigma R_i (o_i \in R) \quad (6)$$

When movement is permitted, the planner favors the region with the lowest aggregate risk. A configurable warning threshold can trigger SLOW when centre-lane risk becomes elevated, while a higher safety condition can trigger STOP when all available directions are judged highly occupied. A short temporal mode filter is applied to reduce rapid left–right command oscillation. The values are deliberately described as heuristic risks. No probability calibration has been performed, and the planner does not infer traversability of unseen ground or detect all drop-offs.

4.5 Multilingual speech preparation

Audio generation is decoupled from perception through an asynchronous producer–consumer queue. The reported host implementation uses offline pyttsx3 synthesis with optional Microsoft SAPI integration on Windows. Localized labels are resolved from a JSON configuration at runtime, allowing language changes without retraining the object detector. This is a software localization mechanism rather than a language-comprehension study.

Three controls limit stale output. First, the queue is bounded to 20 messages. Second, a per-object cooldown of 2.5 s suppresses routine repetition. Third, routine announcements are separated by at least 0.5 s unless a higher-priority hazard is produced. These choices reduce software-level repetition, but the present experiment timestamps queue insertion rather than the start or completion of audible speech. Consequently, the reported latency must not be described as full acoustic end-to-end latency.

4.6 Frame freshness and load control

The receiver restricts buffering to recent frames and discards outdated content so that transient network or compute pressure does not create an ever-growing lag. Under high processor load, adaptive frame skipping reduces the inference rate. This distinction is important when reading the results: camera capture rate, network receive rate, detector invocation rate, and fully processed navigation throughput are not interchangeable quantities. The reported FPS values are fully processed throughput at the queue endpoint.

4.7 Manuscript preparation and reporting integrity

A generative AI language model was used during manuscript preparation to assist with language organization and drafting. The author reviewed the technical content against project records, separated measured results from later codebase features, checked cited bibliographic information, and retains full responsibility for the manuscript. No AI-generated experimental observations, detector benchmark values, participant outcomes, or missing raw measurements were introduced to fill gaps in the project record.

5 Experimental methodology

5.1 Host configuration

The timing campaign used a Windows 10 laptop equipped with an Intel Core i5 10th-generation processor, 4 GB system RAM, and an NVIDIA GTX 1650 GPU with 4 GB VRAM. Python 3.11 coordinated acquisition, decoding, YOLOv8n inference, and speech preparation. The C + + 17 NavigationPlanner consumed normalized detections. Table 3 records the configuration used to interpret the results.

Table 3
Host-assisted experimental configuration

Component	Configuration
Host CPU	Intel Core i5, 10th generation
System memory	4 GB RAM
GPU	NVIDIA GTX 1650, 4 GB VRAM
Operating system	Windows 10
Primary camera	AI-Thinker ESP32-CAM with OV2640
Alternative sources	Samsung Android smartphone MJPEG; Logitech C920 USB webcam
Detector	COCO-pretrained YOLOv8n
Planner	C + + 17 NavigationPlanner
Speech layer	pyttsx3; optional Microsoft SAPI
Timing endpoint	Generated message inserted into the TTS queue

5.2 Latency and throughput measurement

Stage timing was summarized over 500 consecutive frames. The first timing analysis excludes camera acquisition and reports the major processing stages up to TTS queue insertion: wireless transport, JPEG decoding, YOLO inference, navigation planning, and TTS preparation/queue work. GPU and CPU-only configurations use the same downstream planner and queue logic, allowing the effect of inference acceleration to be seen in context.

The study reports arithmetic mean and standard deviation because the archived project package contains those summary values rather than raw per-frame timestamp traces. Median, p95, p99, and distributional tests therefore cannot be reconstructed without inventing data. The 200 ms requirement is treated as a project-specific prototype target, not a universal human-factors threshold.

5.3 Camera-source comparison

A second timing analysis includes acquisition and compares the three input paths under the reported local test conditions with GPU inference enabled. The USB webcam provides a direct baseline. The ESP32-CAM path includes acquisition, UDP transmission, reassembly, and JPEG decode. The smartphone path includes acquisition, TCP/MJPEG transport, and JPEG decode. Total camera-to-queue time is the sum of the measured stages for each path. Because both source hardware and protocol change across conditions, causal statements about UDP versus TCP alone are avoided.

5.4 Navigation-decision evaluation

The planner was evaluated on 200 manually annotated video clips: 100 indoor and 100 outdoor scenarios. Annotators reviewed the visible scene and selected an expected command from GO STRAIGHT, GO LEFT, GO RIGHT, SLOW, or STOP. Agreement was counted when the planner output matched the annotated command. The reported project package does not contain a full annotation manual, clip-duration distribution, inter-annotator agreement statistics, or command-level confusion matrix. The result is therefore reported as preliminary planner–annotation agreement rather than navigation accuracy or safety effectiveness.

No blind or low-vision participant was asked to navigate with the system in this study. The clips are used as technical scene inputs. No claim is made about collision reduction, route completion, trust, cognitive load, or user preference.

5.5 Analysis strategy

The analysis is descriptive. For RQ1, the GPU and CPU totals are compared with the predefined 200 ms and 10 FPS project targets. For RQ2, camera-to-queue latency and processed FPS are compared across the three source paths, and percentage differences are derived from the reported means. For RQ3, agreement is reported separately for indoor and outdoor clips. Inferential statistics are not applied because the available archive does not include the raw timing traces or per-clip command labels needed for defensible hypothesis testing.

6 Results

6.1 Processing latency and throughput

Table 4
Pipeline timing excluding camera acquisition, summarized over 500 consecutive frames

Stage / metric	GPU	CPU only
YOLO inference (ms)	11.6 ± 2.1	94.3 ± 18.7
UDP transport (ms)	18.4 ± 4.2	18.4 ± 4.2
JPEG decode (ms)	3.1 ± 0.8	3.1 ± 0.8
NavigationPlanner (ms)	0.3 ± 0.1	0.3 ± 0.1
TTS preparation / queue (ms)	45.2 ± 12.3	45.2 ± 12.3
Processing-to-queue total (ms)	78.6 ± 14.1	161.3 ± 28.4
Processed throughput (FPS)	22.3	6.8

The endpoint is queue insertion, not audible speech. The total includes the listed transport, decode, inference, planner, and TTS-preparation stages but excludes camera acquisition.

The GPU configuration satisfied both predefined project targets. Its mean processing-to-queue latency was 78.6 ms, leaving substantial margin below 200 ms, and it sustained 22.3 fully processed FPS. The CPU-only configuration also remained below the 200 ms mean latency target at 161.3 ms, but its throughput fell to 6.8 FPS and therefore did not satisfy the 10 FPS design target.

Using the reported means, GPU acceleration reduced total processing-to-queue latency by approximately 51.3% relative to CPU-only execution and increased processed throughput by a factor of about 3.28. The difference is dominated by detector inference: YOLOv8n fell from 94.3 ms on CPU to 11.6 ms on GPU, whereas planner time remained 0.3 ms in both modes. This result answers RQ1: the measured pipeline target is supported for the tested GPU host, but not fully for CPU-only operation because throughput is below the project requirement.

6.2 Camera-source latency

Table 5
Camera-to-TTS-queue timing by input source

Metric	ESP32-CAM UDP	Smartphone MJPEG	USB webcam
Frame acquisition (ms)	16.8 ± 3.7	22.4 ± 5.9	5.2 ± 1.4
Transport (ms)	18.4 ± 4.2	31.7 ± 6.8	0
JPEG decode (ms)	3.1 ± 0.8	3.4 ± 0.9	0
YOLO inference (ms)	11.6 ± 2.1	11.6 ± 2.1	11.6 ± 2.1
NavigationPlanner (ms)	0.3 ± 0.1	0.3 ± 0.1	0.3 ± 0.1
TTS preparation / queue (ms)	45.2 ± 12.3	45.2 ± 12.3	45.2 ± 12.3
Camera-to-queue total (ms)	95.4 ± 16.2	114.6 ± 19.7	62.6 ± 13.1
Processed throughput (FPS)	20.8	18.2	24.5

GPU inference was enabled for this comparison. The devices and transport protocols differ simultaneously; differences should be interpreted as input-path differences rather than isolated protocol effects.

The direct USB webcam path produced the lowest measured camera-to-queue latency, 62.6 ms, and the highest throughput, 24.5 FPS. This is expected because it avoids wireless transport and network JPEG reassembly. The ESP32-CAM wireless path reached 95.4 ms at 20.8 FPS, while the smartphone MJPEG path reached 114.6 ms at 18.2 FPS. All three means were below the 200 ms project timing target and all three exceeded 10 processed FPS in the GPU configuration.

Relative to the smartphone path, the ESP32-CAM path was approximately 16.8% lower in total mean camera-to-queue latency. The measured transport-stage mean was 18.4 ms for ESP32-CAM UDP and 31.7 ms for smartphone MJPEG, a difference of about 42.0%. It would be incorrect, however, to attribute the full difference to UDP alone because the source device, encoder behavior, stream format, and

protocol stack also change. RQ2 is therefore answered at the systems level: input-path choice materially affects latency, and in this local setup the direct USB path was fastest, followed by ESP32-CAM UDP and smartphone MJPEG.

6.3 Preliminary navigation-decision agreement

Table 6
Preliminary NavigationPlanner agreement with manually annotated commands

Scenario group	Clips	Agreement
Indoor	100	91.5%
Outdoor	100	87.3%

Agreement is a technical scene-level measure. It is not a clinical navigation-success or safety metric.

Planner–annotation agreement was 91.5% for the 100 indoor clips and 87.3% for the 100 outdoor clips. Because the groups are equal in size, the descriptive overall agreement is 89.4%. The lower outdoor result is consistent with the project notes, which identify borderline centre-lane scores and lateral left/right decisions as a recurring source of disagreement. Without raw command labels and a confusion matrix, that explanation cannot be quantified further here.

RQ3 is therefore only partially resolved. The result shows that the deterministic planner often produced the same command as the manual scene annotation under the reported test conditions, with somewhat higher agreement indoors. It does not establish that the command is safe, that annotators agree with one another, or that a blind traveler would follow the instruction successfully.

7 Discussion

7.1 What the results demonstrate

The strongest result is the system-level timing evidence. On the tested GPU host, WVAB kept the full measured chain from network transport through queue insertion below 100 ms on average for the primary ESP32-CAM path while sustaining more than 20 processed FPS. This is materially different from reporting detector inference alone. Inference accounted for 11.6 ms of the GPU pipeline, but speech preparation and transport together accounted for a larger share. Optimizing only the neural network would therefore miss important sources of delay.

The CPU result is equally informative. A mean of 161.3 ms might appear acceptable if latency is the only criterion, yet processed throughput was 6.8 FPS. This illustrates why latency and throughput should be reported together. A system can process an individual frame within a nominal deadline while still failing to refresh its decisions frequently enough for the design target. For the tested YOLOv8n configuration,

GPU acceleration is not merely a convenience; it is what allows the measured pipeline to satisfy both timing and rate requirements simultaneously.

7.2 Camera and transport implications

The camera comparison quantifies a familiar engineering trade-off. Direct USB acquisition is fastest but constrains placement to a wired host. Smartphone MJPEG is convenient and benefits from a capable camera, but its measured network path added more delay in this setup. ESP32-CAM offers a compact wireless sensor with lower measured total latency than the smartphone path, at the cost of a more constrained image sensor and a custom transport stack. The right choice therefore depends on deployment goals rather than a single latency number.

The UDP result must also be interpreted responsibly. UDP can reduce transport overhead and avoid head-of-line blocking caused by retransmission, but it transfers reliability responsibilities to the application and can behave poorly if datagram sizing, congestion, authentication, or reassembly are ignored [16, 17]. The evaluated prototype prioritized freshness and worked on a controlled local network. A production wearable should use MTU-aware packetization, authenticated encryption, bounded state, explicit frame age, and a fail-safe response to prolonged loss. The current v2.0.0 repository has moved in that direction, but new security behavior requires its own performance measurements before publication-level claims are made.

7.3 Planner contribution and interpretability

The NavigationPlanner contributes negligible measured compute time—0.3 ms per processed frame—while exposing its decision variables explicitly. This interpretability is useful in an assistive prototype. A developer can inspect the class weight, confidence, apparent area, lane assignment, and threshold that led to a command. The design also makes the planner replaceable: a future probabilistic or learned policy could consume the same normalized detections without changing camera or speech modules.

At the same time, the planner's simplicity defines its limits. Apparent bounding-box area is not depth; lane occupancy is not traversability; and class-weighted sums do not model pedestrian intent, vehicle velocity, stairs, curbs, transparent objects, or free-ground geometry. The 89.4% aggregate scene agreement should therefore be read as a software consistency signal, not as evidence of safe independent navigation. The next planner study should use a documented annotation protocol, multiple annotators, inter-rater agreement, per-command precision and recall, and explicit analysis of STOP false negatives.

7.4 Relation to assistive-vision research

Compared with research that introduces new sensors or detector architectures [8, 9, 13], WVAB's contribution is architectural integration. It demonstrates how ordinary multimedia components can be connected with bounded timing and transparent decision logic. This aligns with broader reviews that call for attention to portability, feedback design, robustness, and user burden rather than detector performance in isolation [3–7].

The absence of a user trial is a meaningful limitation, not a minor omission. Neugebauer et al. [10] and Lupu et al. [11] show why human evaluation can change the interpretation of technically promising feedback: task performance, cognitive load, training, and modality preference cannot be predicted from software latency alone. Vrysis et al. [12] similarly demonstrate that accessibility features must be assessed in the context of actual low-vision use. WVAB should therefore be considered ready for more rigorous technical replication and carefully supervised user-centered research, not for unsupervised mobility deployment.

7.5 Threats to validity

Several threats limit generalization. First, the timing data come from one host platform and local network environment. Router load, radio interference, camera firmware, thermal state, operating-system scheduling, and model version can change the result. Second, the archive contains mean and standard deviation summaries but not the full timing traces, preventing tail-latency analysis. Third, the camera comparison changes both hardware and transport protocol, so it is not a clean network protocol experiment. Fourth, the navigation study lacks a published clip set, detailed annotation guide, inter-annotator reliability, and command confusion matrix. Fifth, no calibrated distance ground truth or perception benchmark was collected for this manuscript. Sixth, no blind or low-vision participants evaluated the guidance.

These constraints also explain what is intentionally absent from the article. Earlier project drafts contained detector accuracy numbers that could not be tied to a reproducible WVAB dataset split and evaluation protocol. Those values are not repeated here. Similarly, the article does not compare commercial device prices or claim that the complete system costs USD 15. Removing unsupported precision strengthens the reproducibility of the remaining findings.

8 Safety, ethics, and privacy considerations

WVAB is an experimental secondary aid. It has not been clinically validated, safety certified, or evaluated as a replacement for a long cane, guide dog, orientation and mobility training, or other established mobility support. Object detection can fail under motion blur, low light, glare, occlusion, unusual object appearance, transparent obstacles, or camera obstruction. Image-space occupancy cannot reliably detect every drop-off or confirm that apparently clear ground is traversable.

The study reported here does not enroll blind or low-vision participants and does not report human mobility outcomes. Any future participant study should be reviewed under the applicable institutional ethics process, use informed consent, employ supervised routes and independent safety measures, and involve orientation-and-mobility expertise. Outcomes should include not only completion time but also collisions or near misses, STOP false negatives, workload, message comprehension, trust, and learning effects.

Privacy and network security are also part of assistive-system safety. A wearable camera can capture bystanders, private interiors, or sensitive contextual information. The historical evaluated path assumed a trusted local network and is not suitable as a production security baseline. Deployment-oriented versions should minimize retention, process locally where possible, encrypt and authenticate remote transport, provide visible failure states, and avoid silently continuing when camera or inference integrity is uncertain.

9 Conclusion and future work

This article evaluated WVAB as a low-latency multimedia pipeline for assistive visual perception and navigation guidance. The system combines an ESP32-CAM wireless sensing node, host-side YOLOv8n object detection, a deterministic C + + lane-risk planner, and multilingual offline speech preparation. On the tested GPU host, processing-to-TTS-queue latency averaged 78.6 ± 14.1 ms at 22.3 FPS. When camera acquisition was included, the ESP32-CAM path averaged 95.4 ± 16.2 ms at 20.8 FPS, compared with 114.6 ± 19.7 ms for smartphone MJPEG and 62.6 ± 13.1 ms for direct USB. Preliminary planner–annotation agreement reached 91.5% indoors and 87.3% outdoors across 200 annotated clips.

The results support prototype-level feasibility of the host-assisted architecture, particularly when GPU acceleration is available. They do not establish clinical effectiveness, independent-navigation safety, calibrated distance accuracy, or user acceptance. The next technical study should archive raw timing traces and report median, p95, and p99 latency; evaluate packet loss and MTU-aware secure transport; benchmark quantized edge inference on Raspberry Pi-class hardware; and publish a reproducible navigation annotation protocol with command-level errors. Only after those engineering stages should supervised human-centered evaluation be used to test whether focused audio guidance improves mobility without increasing cognitive burden.

Statements and Declarations

Funding. No funding was received for conducting this study.

Competing interests. The author has no relevant financial or non-financial interests to disclose.

Author contributions. Md Shahanur Islam Shagor: conceptualization, system architecture, software, methodology, investigation, validation, formal analysis, visualization, and manuscript preparation.

Ethics and consent. No blind or low-vision participants were enrolled in the reported technical evaluation, and no clinical or mobility trial is reported. Future human-subject evaluation will require the applicable institutional ethics review and informed consent.

Data availability. The article reports the archived summary measurements available for the evaluated prototype. Raw per-frame timing traces, the complete navigation annotation protocol, and the original

command-level labels are not publicly archived; consequently, distributional or command-level statistics are not reconstructed in this manuscript.

Code availability. The WVAB source code is publicly available at https://github.com/smshagor-dev/wireless-Vision-Aid-for-the-blind. The current v2.0.0 repository contains later mobile, security, packaging, and navigation features beyond the host-assisted configuration measured in this article; readers should use the paper's stated experimental boundary when reproducing the reported numbers.

Use of generative AI. A generative AI language model was used to assist with language organization and drafting. The author reviewed and verified the technical claims, measurements, references, interpretations, and final text and accepts full responsibility for the manuscript.

References

1. World Health Organization (2026) Blindness and vision impairment. WHO Fact Sheet, 10 February 2026. Accessed 26 August 2026 https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment
2. Dakopoulos D, Bourbakis NG (2010) Wearable obstacle avoidance electronic travel aids for blind: a survey. IEEE Trans Syst Man Cybern C Appl Rev 40(1):25–35. https://doi.org/10.1109/TSMCC.2009.2021255
3. Elmannai W, Elleithy K (2017) Sensor-based assistive devices for visually-impaired people: current status, challenges, and future directions. Sensors 17(3):565. https://doi.org/10.3390/s17030565
4. Messaoudi MD, Menelas B-AJ, Mcheick H (2022) Review of navigation assistive tools and technologies for the visually impaired. Sensors 22(20):7888. https://doi.org/10.3390/s22207888
5. Hersh M (2022) Wearable travel aids for blind and partially sighted people: a review with a focus on design issues. Sensors 22(14):5454. https://doi.org/10.3390/s22145454
6. Valipoor MM, de Antonio A (2023) Recent trends in computer vision-driven scene understanding for VI/blind users: a systematic mapping. Univers Access Inf Soc 22:983–1005. https://doi.org/10.1007/s10209-022-00868-w
7. Okolo GI, Althobaiti T, Ramzan N (2024) Assistive systems for visually impaired persons: challenges and opportunities for navigation assistance. Sensors 24(11):3572. https://doi.org/10.3390/s24113572
8. Bai J, Liu Z, Lin Y, Li Y, Lian S, Liu D (2019) Wearable travel aid for environment perception and navigation of visually impaired people. Electronics 8(6):697. https://doi.org/10.3390/electronics8060697
9. Katzschmann RK, Araki B, Rus D (2018) Safe local navigation for visually impaired users with a time-of-flight and haptic feedback device. IEEE Trans Neural Syst Rehabil Eng 26(3):583–593. https://doi.org/10.1109/TNSRE.2018.2800665

10. Neugebauer A, Rifai K, Getzlaff M, Wahl S (2020) Navigation aid for blind persons by visual-to-auditory sensory substitution: a pilot study. PLoS ONE 15(8):e0237344. https://doi.org/10.1371/journal.pone.0237344
11. Lupu R-G, Mitruț O, Stan A, Ungureanu F, Kalimeri K, Moldoveanu A (2020) Cognitive and affective assessment of navigation and mobility tasks for the visually impaired via electroencephalography and behavioral signals. Sensors 20(20):5821. https://doi.org/10.3390/s20205821
12. Vrysis L, Almaliotis D, Almpanidou S et al (2024) Mobile software aids for people with low vision. Multimed Tools Appl 83:30919–30936. https://doi.org/10.1007/s11042-023-16639-5
13. Wang W, Jing B, Yu X, Sun Y, Yang L, Wang C (2024) YOLO-OD: obstacle detection for visually impaired navigation assistance. Sensors 24(23):7621. https://doi.org/10.3390/s24237621
14. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer Vision – ECCV 2014. LNCS 8693. Springer, Cham, pp 740–755. https://doi.org/10.1007/978-3-319-10602-1_48
15. Jocher G, Chaurasia A, Qiu J (2023) Ultralytics YOLOv8, version 8.0.0 [computer software]. Ultralytics. Accessed 26 August 2026. https://github.com/ultralytics/ultralytics
16. Postel J (1980) User Datagram Protocol. RFC 768. Internet Engineering Task Force. https://doi.org/10.17487/RFC0768
17. Eggert L, Fairhurst G, Shepherd G (2017) UDP usage guidelines. RFC 8085. Internet Engineering Task Force. https://doi.org/10.17487/RFC8085

Figures

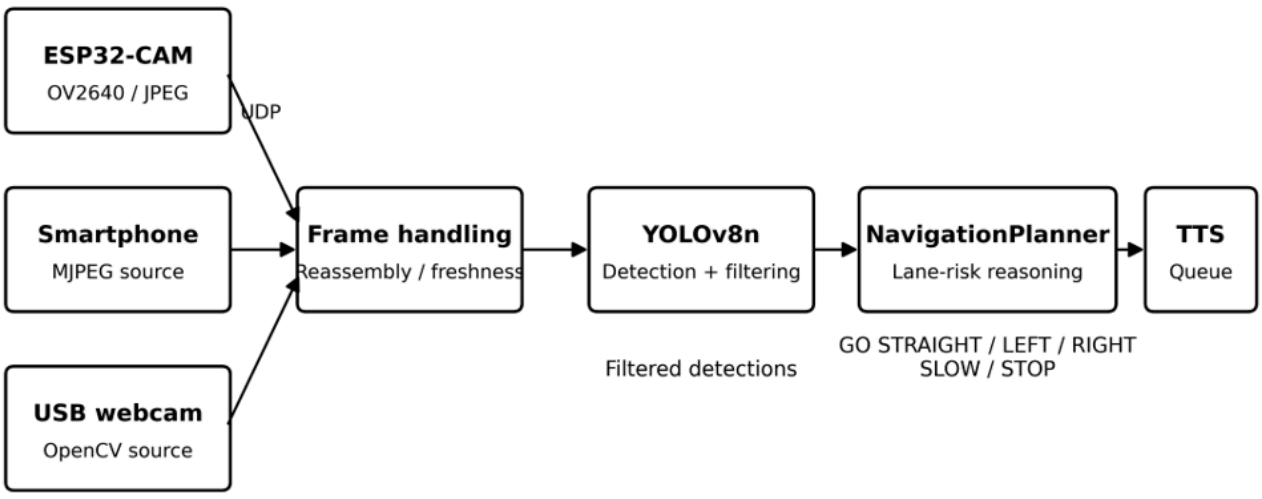
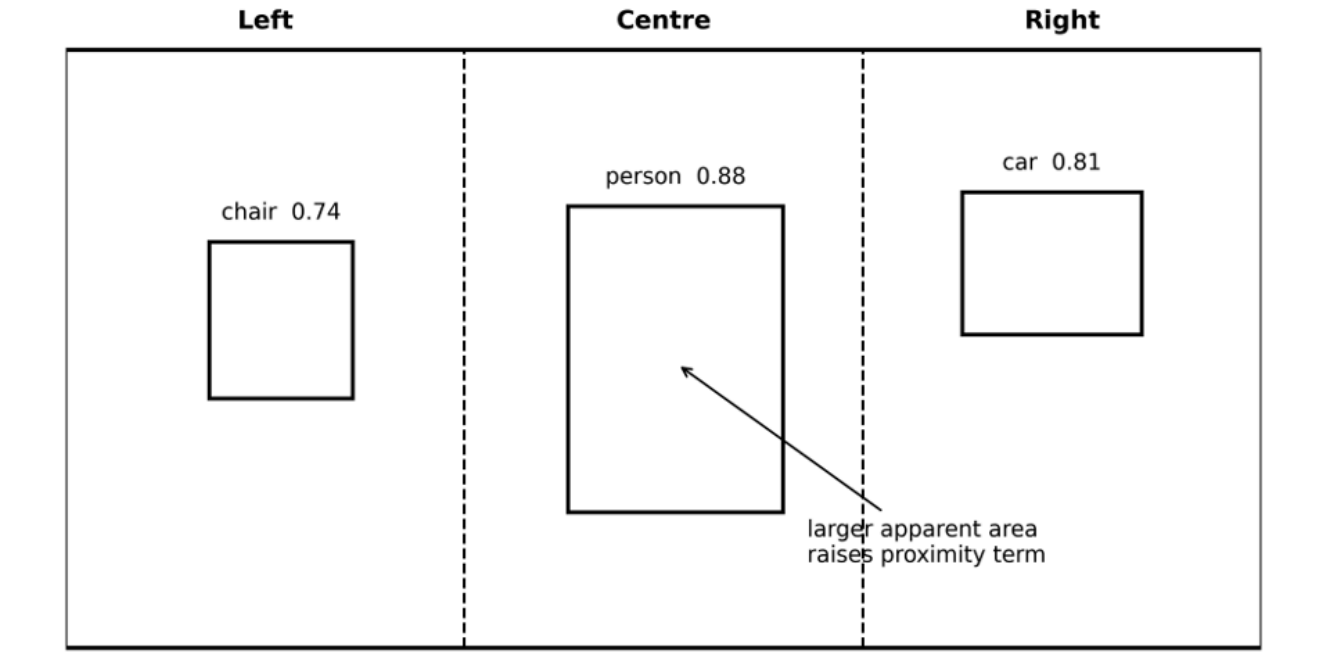


Figure 1

Evaluated host-assisted WVAB perception-to-feedback pipeline. Smartphone and USB inputs are comparison paths; ESP32-CAM over UDP is the primary wireless path. The study endpoint is TTS queue insertion, not audible speech onset



Detections are assigned to image lanes; class weight, confidence, and apparent area contribute to heuristic lane risk.

Figure 2

Simplified lane-risk formulation. Normalized horizontal centre assigns detections to left, centre, or right; class weight, confidence, and apparent area contribute to a bounded heuristic risk score

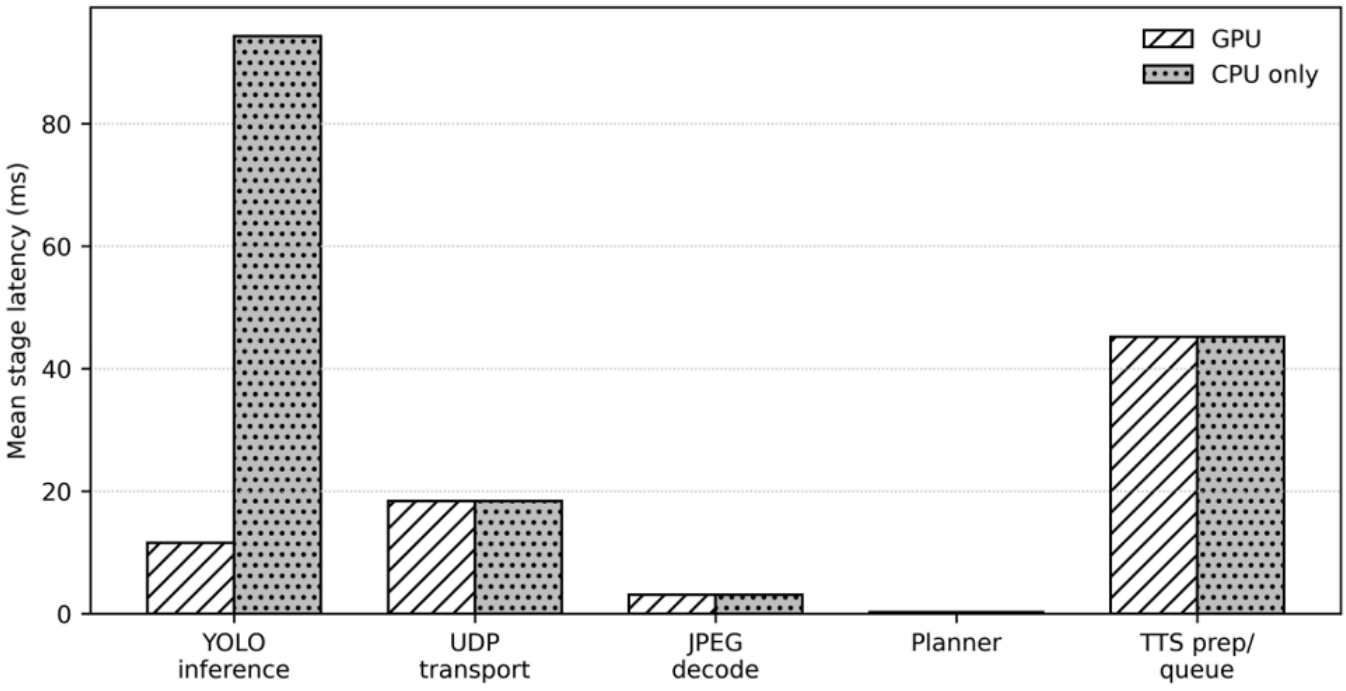


Figure 3

Mean stage latency for GPU and CPU-only execution. Error bars are not shown in the plot; numerical mean ± standard deviation values are given in Table 4

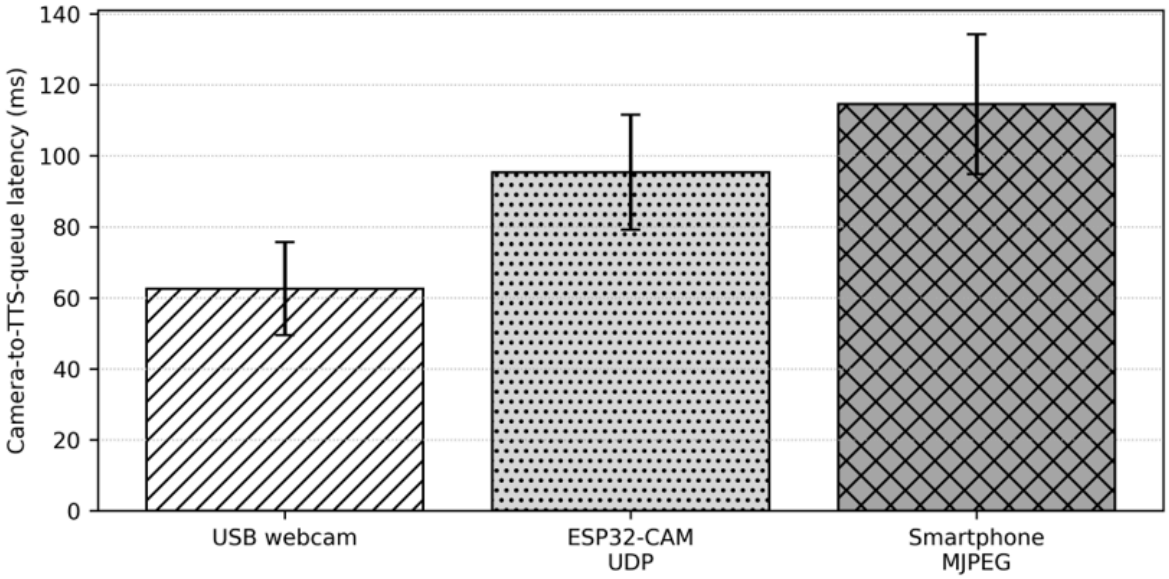


Figure 4

Mean camera-to-TTS-queue latency for the three evaluated input paths. Error bars show the reported standard deviation

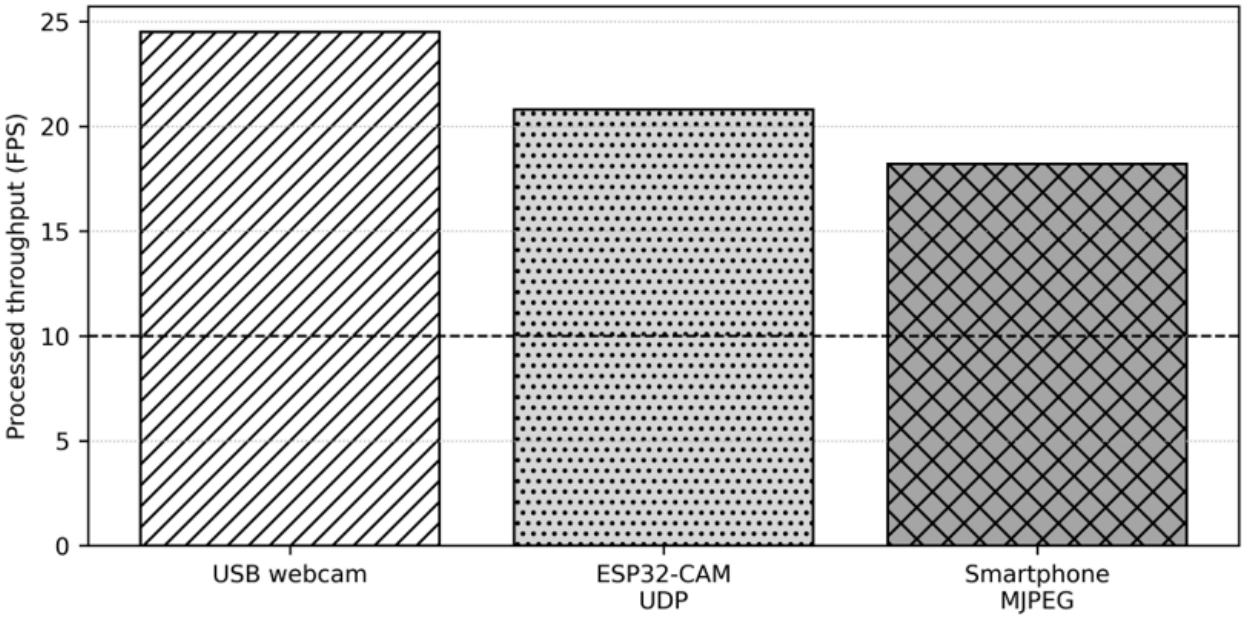


Figure 5

Fully processed throughput for the three evaluated input paths. The dashed line marks the 10 FPS project design target

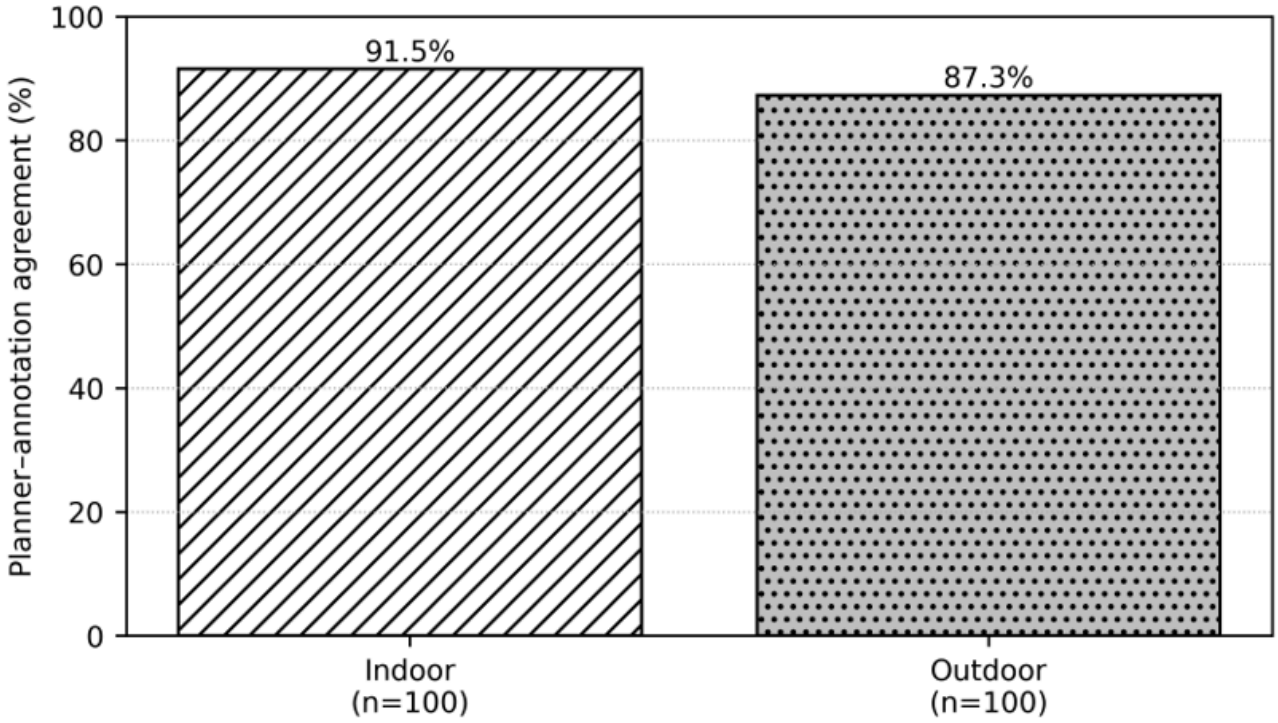


Figure 6

Preliminary planner–annotation agreement for indoor and outdoor video scenarios