Engineering and Technology Journal e-ISSN: 2456-3358

Volume 11 Issue 08 August-2026, Page No.-11058-11065

DOI: 10.47191/etj/v11i08.02, I.F. – 8.595

Walk Assist: An Integrated Arcore and Vision-Language Based Mobile Walking Assistance System For Blind And Low-Vision Pedestrians

Daehyun Kwon[1], Seonu Jeon[2], Seonkyu Lee[3], Yeo Junho[4], Seungjae Lee[5]*

[1,2,3,4,5]Computer Engineering Department, Sunmoon University, Korea

ABSTRACT: This paper presents WalkAssist, an Android-based walking assistance system designed for blind and low-vision pedestrians. The system integrates ARCore spatial sensing, YOLO-based object segmentation, one-shot OCR, Gemini 2.5 Flash-Lite API scene description, map-based pedestrian route guidance, and multimodal feedback into a single mobile prototype. ARCore camera pose, hit tests, raw depth, and depth confidence are utilized to estimate the walking zone and nearby obstacle distance. YOLO26n-seg detects objects and provides segmentation masks, which are combined with depth samples to improve object-level distance awareness. OCR reads visible text when requested, while the Gemini 2.5 Flash-Lite API provides natural-language scene descriptions for additional context. The feedback layer delivers information through an on-screen overlay, text-to-speech, vibration, tones, and accessibility announcements. The implemented prototype demonstrates the feasibility of combining multiple perception and feedback modules on a smartphone to support pedestrian awareness. Development also revealed practical challenges, including ARCore depth instability, low-light sensitivity, and VLM latency, which were mitigated through confidence filtering, fallback depth sampling, output prioritization, and local walking-zone feedback prior to remote VLM responses.

KEYWORDS: Walking assistance, ARCore, vision-language model, YOLO segmentation, accessibility feedback

I. INTRODUCTION

Independent walking requires continuous awareness of nearby obstacles, readable signs, route direction, and the open area in front of the body. For blind and low-vision pedestrians, obtaining this critical information without personal assistance remains a significant challenge [1]. Traditional aids such as white canes and audio navigation services are highly useful, but they often lack real-time object-level visual context and short-range spatial estimation capabilities [10]. With recent advancements in smartphone cameras, augmented reality (AR) APIs, and multimodal artificial intelligence models, modern mobile devices can serve as integrated platforms for experimental walking assistance.

This paper proposes WalkAssist, an Android walking assistance prototype that combines ARCore, YOLO object segmentation, OCR, the Gemini 2.5 Flash-Lite API, pedestrian route guidance, and multimodal feedback. The primary goal of the system is not to replace certified mobility aids, but to explore how heterogeneous perception modules can be cohesively integrated into a single accessible mobile interface. In this architecture, ARCore estimates spatial structure and walking-zone distance [2], YOLO26n-seg detects objects with segmentation masks [9], OCR reads visible text on demand [4], and the Gemini 2.5 Flash-Lite API describes the current camera scene when the user requests additional semantic context [6]. Subsequently, the feedback layer translates these analytical results into comprehensive overlay text, speech, vibration, and accessibility announcements [14].

The main contribution of this work is the design and implementation of an integrated walking-assistance pipeline on the Android platform. Unlike conventional systems that rely exclusively on isolated object detection or standalone scene-description models, WalkAssist cleanly separates short-range spatial awareness, object analysis, text recognition, semantic scene description, and user feedback into cooperating modules. Furthermore, practical development hurdles—such as vision-language model (VLM) latency, ARCore depth instability, and low-light sensitivity—are analyzed and addressed as core implementation lessons to ensure robust runtime performance.

II. RELATED WORK

A. Assistive Navigation and Depth Awareness

Assistive navigation research has long explored the utilization of depth sensors, wearable hardware, haptic feedback interfaces, and smartphone-based localization techniques to assist visually impaired pedestrians in navigating complex urban environments [10]-[12]. Among these methodologies, depth-based systems are particularly advantageous as they enable direct estimation of nearby obstacle distances rather than merely recognizing object categories.

Recently, ARCore-based frameworks have emerged as an attractive solution for mobile prototyping because they natively provide robust pose tracking, hit tests, plane detection, and raw depth data through standard commercial Android devices [2], [3]. Despite these advantages, ARCore

depth remains an API-level estimation derived from visual inertial odometry, meaning its accuracy depends heavily on hardware support, camera motion dynamics, ambient lighting conditions, and scene texture richness [5]. Consequently, robust mobile navigation systems must account for these inherent hardware limitations through algorithmic filtering and fallback mechanisms.

B. Object Recognition, OCR, and Vision-Language Models (VLMs)

Accurate object detection and semantic segmentation are critical for identifying dynamic hazards such as pedestrians, vehicles, street furniture, and construction barriers. In the WalkAssist architecture, YOLO26n-seg is adopted because it efficiently delivers both bounding boxes and fine-grained segmentation masks optimized for real-time mobile inference via TensorFlow Lite [9]. Beyond spatial obstacles, public navigation heavily relies on text comprehension; thus, optical character recognition (OCR) plays a vital role in reading street signs, store labels, and public notices using mobile-optimized text-recognition APIs [4].

In parallel, Vision-Language Models (VLMs) have introduced advanced capabilities by summarizing complex visual scenes into natural-language descriptions. Architectures such as Florence-2 and modern cloud-based foundation models can seamlessly bridge visual inputs with textual reasoning [6]-[8]. However, deploying VLMs for real-time pedestrian assistance introduces significant latency and safety challenges. While dense semantic descriptions offer rich contextual awareness, they are often too slow or unpredictable for immediate collision avoidance. Therefore, WalkAssist establishes a hybrid paradigm: immediate, safety-critical distance cues are handled via local ARCore and YOLO processing, whereas VLM outputs are strategically integrated as a supplementary, asynchronous description layer.

III. SYSTEM DESIGN

A. Overall Architecture

WalkAssist is organized as a live AR pipeline connected to an accessibility-oriented feedback layer. Each camera frame supplies RGB image data, ARCore pose, hit-test results, raw depth, and depth confidence. The spatial-awareness module estimates the open walking zone in front of the user and tracks obstacle distance across left, center, and right lanes. The vision module performs YOLO segmentation and combines detected objects with available depth samples. Optional modules, including OCR, Gemini scene description, and route guidance, are triggered by user interaction or navigation state.

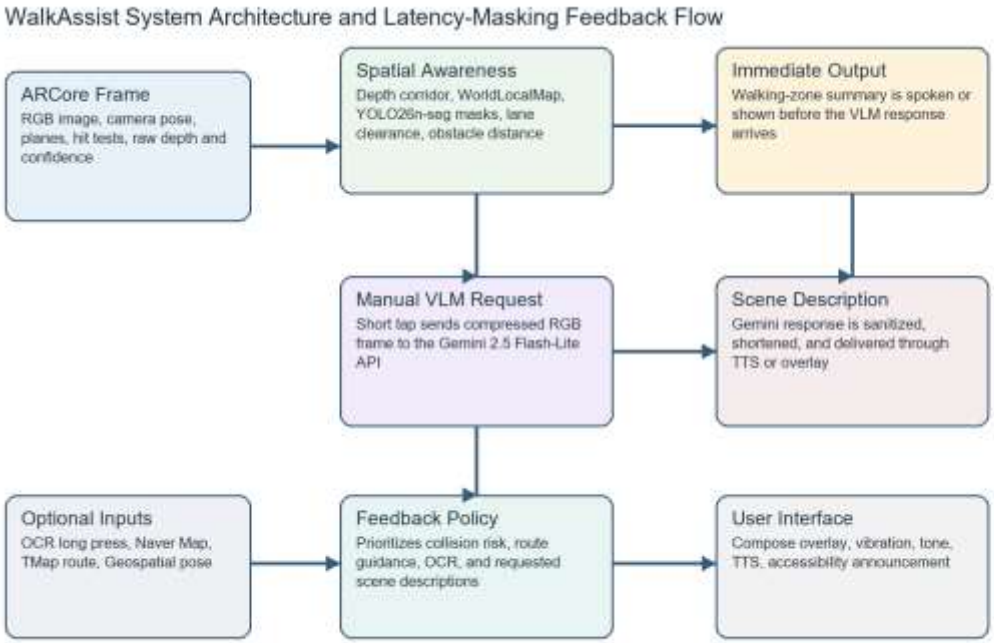

Fig. 1. Integrated WalkAssist architecture for spatial sensing, vision analysis, and feedback.

TABLE I: FUNCTIONAL MODULES OF WALKASSIST

Module	Main Technology	Function in the System
Spatial sensing	ARCore pose, hit tests, raw depth, confidence	Estimates walking-zone openness, obstacle distance, lane state, and collision risk.

Object perception	YOLO26n-seg with TensorFlow Lite	Detects objects and uses segmentation masks for object-depth fusion.
Text recognition	ML Kit OCR	Reads visible text when the user performs a long-press request.
Scene description	Gemini 2.5 Flash-Lite API	Generates a short natural-language description when the user taps the AR screen.
Route guidance	Naver Map, Geocoder, TMap pedestrian route	Searches destinations, displays pedestrian routes, provides TTS route instructions, warns when the user leaves the route, and gives direction-alignment guidance.
Feedback output	Compose overlay, TTS, vibration, tone, accessibility API	Converts analysis results into screen, speech, haptic, and accessibility feedback.

B. ARCore-Based Walking-Zone Estimation

The ARCore module estimates the user's surrounding space by combining camera pose, screen corridor hit tests, and raw-depth samples. The system interprets the camera-forward direction as the primary walking direction and divides the visible area into left, center, and right lanes. For each lane, depth and hit-test observations are accumulated into a local occupancy map. The resulting state includes approximate obstacle distance, available walking-zone distance, and risk level. This local spatial feedback is the fastest and most frequently updated part of the system.

C. YOLO Object Segmentation and Depth Fusion

YOLO26n-seg detects objects from the current camera image and returns bounding boxes, class labels, confidence values, and segmentation polygons. WalkAssist does not simply announce every detected object. Instead, it combines object masks with ARCore raw-depth samples to estimate whether the object is close enough to affect walking. If enough depth samples are available inside the segmentation mask, the object distance is estimated from those samples. If the mask is unreliable or too sparse, the system falls back to a bounding-box sampling strategy. This fusion process makes object feedback more spatially meaningful.

D. OCR and Gemini Scene Description

Two user-requested recognition functions are provided. A long press invokes OCR to read visible text, which is useful for signs, labels, and notices. A short tap invokes the Gemini 2.5 Flash-Lite API to describe the camera scene in natural language. The Gemini prompt asks for concise pedestrian-oriented descriptions and the response is sanitized before speech output. These functions are intentionally separated from the continuous spatial loop because they may take longer than frame-by-frame ARCore processing.

IV. IMPLEMENTATION

The WalkAssist prototype is implemented natively in Kotlin for the Android platform, structuring the runtime environment into modular perception, spatial processing, and accessibility-oriented feedback layers. The core runtime is managed by WalkAssistArFragment, which coordinates ARCore session updates, camera frame acquisition, raw-depth data access, hit-test execution, YOLO frame analysis, OCR processing, and manual VLM requests.

A. Spatial Sensing and Perception Pipeline

The system continuously ingests live camera frames containing RGB data, camera poses, hit-test outcomes, raw depth matrices, and associated confidence values. ArMeasurementBridge serves as the core communication bridge, publishing real-time spatial metrics to the user interface and feedback layers.

Walking-Zone Estimation: The spatial-awareness module projects the camera-forward vector into a structured corridor, dividing the immediate field of view into distinct left, center, and right lanes. Depth and hit-test observations are accumulated into a local occupancy map to track lane clearance and collision risks.

Object Segmentation & Fusion: YOLO26n-seg processes camera inputs to extract bounding boxes and segmentation polygons. To avoid alerting users to distant or irrelevant objects, the system fuses these segmentation masks with ARCore raw-depth samples. If sufficient valid depth points exist within a mask, object distance is derived locally; otherwise, the system seamlessly falls back to a bounding-box sampling strategy.

B. On-Demand Recognition and Route Guidance

To balance real-time processing performance with rich contextual awareness, compute-intensive or user-triggered tasks are isolated from the continuous spatial loop:

OCR and VLM Integration: A long-press user interaction triggers ML Kit OCR to extract visible text from signs and labels. A short tap on the AR screen captures the current frame, resizes it with an upper bound defined by the VLM_ANALYSIS_MAX_LONG_SIDE constant (768 pixels), and transmits the compressed bitmap to the Gemini 2.5 Flash-Lite API for natural-language scene description.

Pedestrian Route Guidance: The map interface supports destination searching, dynamic pedestrian route rendering, Turn-by-Turn TTS guidance, off-route warnings, and heading-alignment assistance calculated from route bearings and device orientation. Optional AR Geospatial integration can visually project directional hints onto the camera overlay when active.

C. Policy-Based Multimodal Feedback Layer

Because simultaneous perception modules can generate conflicting or overwhelming notifications, output delivery is governed by a policy-based runtime architecture. ArFeedbackMapper transforms raw spatial measurements into standardized feedback requests, while FeedbackPolicy evaluates and prioritizes outputs based on collision severity, route instructions, OCR text, Gemini scene descriptions, and hardware sensor status. Finally, FeedbackManager executes the prioritized notifications through a combination of Jetpack Compose overlays, text-to-speech (TTS) announcements, distinct haptic vibrations, auditory tones, and Android accessibility framework events [14]. This strict separation of concerns ensures that a walking pedestrian receives only the most critical, actionable guidance without cognitive overload.

V. PROTOTYPE RESULTS AND EVALUATION

The implemented WalkAssist prototype successfully unifies live AR visualization, real-time spatial distance tracking, object perception, and policy-driven multimodal feedback into a cohesive mobile interface. During standard execution, users receive continuous, immediate updates regarding the openness of the center walking corridor, the proximity of surrounding obstacles, and the stability status of the device's spatial sensors. Furthermore, users can seamlessly trigger on-demand OCR or Gemini-powered scene descriptions directly from the active AR screen without disrupting the core navigation workflow.

A. User Interface and Spatial Feedback Verification

Preliminary functional evaluations were conducted by non-visually-impaired researchers across controlled indoor spaces and real-world outdoor pedestrian environments. All tests were executed under sufficient ambient lighting to ensure stable camera tracking and raw-depth estimation. These evaluations were primarily designed to validate end-to-end prototype integration, responsiveness, and functional reliability rather than to establish formal clinical safety or broad statistical usability for blind and low-vision populations.

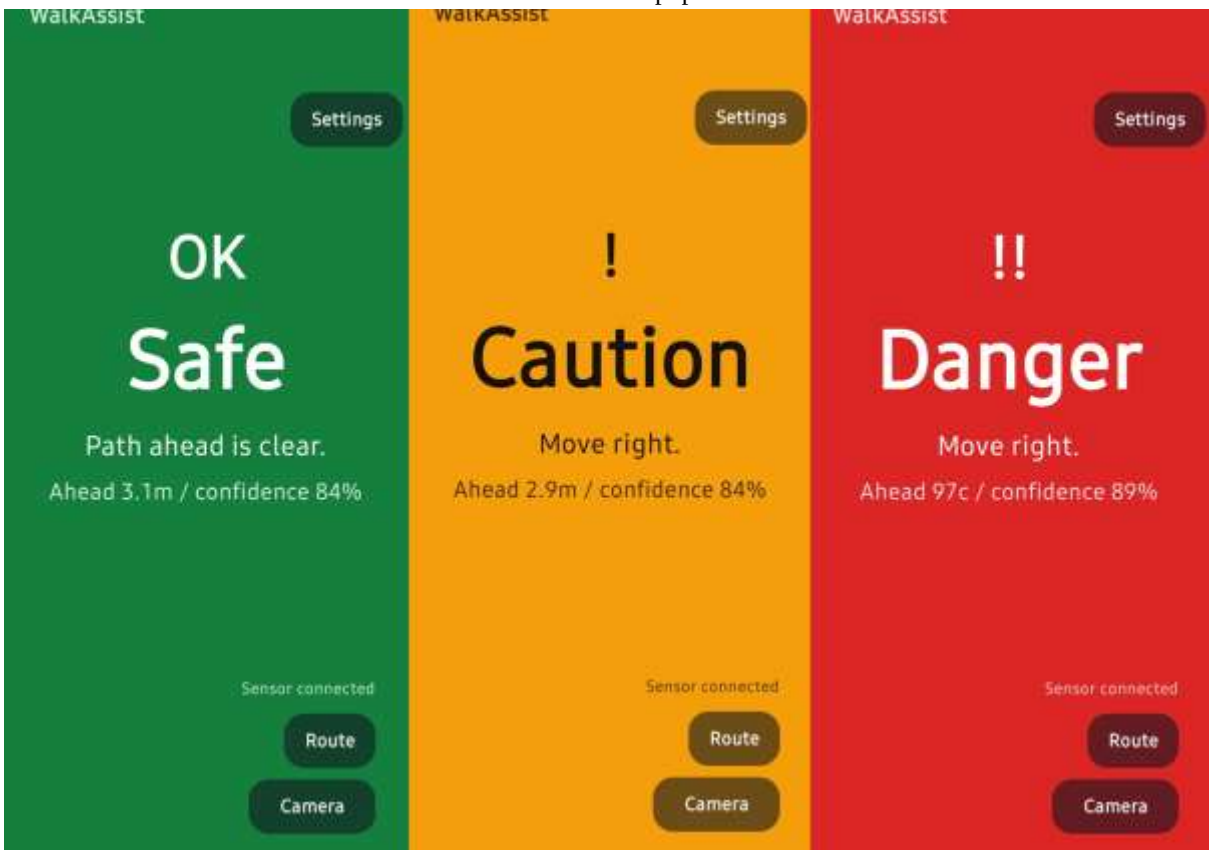

Fig. 2. Main WalkAssist execution screen.

As illustrated in Fig. 2, the Jetpack Compose-based runtime overlay dynamically adapts its visual representation based on collision risk levels:

- Safe State (Green): Indicates a clear path ahead, displaying real-time distance and sensor confidence metrics.
- Caution State (Yellow): Alerts the user to minor obstacles or restricted corridor clearance, suggesting lateral adjustments (e.g., moving right).
- Danger State (Red): Emits high-priority warnings for immediate collision hazards at close range.

B. Vision-Language Model Latency Performance

Because the system architecture prioritizes low-latency safety responses, VLM execution times were rigorously measured during preliminary functional testing under stable lighting conditions. As summarized in Fig. 3, comparing local on-device inference against cloud-based APIs revealed substantial performance variations:

- On-Device Attempt: An initial implementation utilizing the Florence-2 0.23B model locally on mobile hardware required an average latency of 41.5 seconds, rendering it entirely unfeasible for real-time pedestrian assistance.
- Adopted API Solution: Integrating the remote Gemini 2.5 Flash-Lite API drastically reduced the response time to 2.63 seconds.

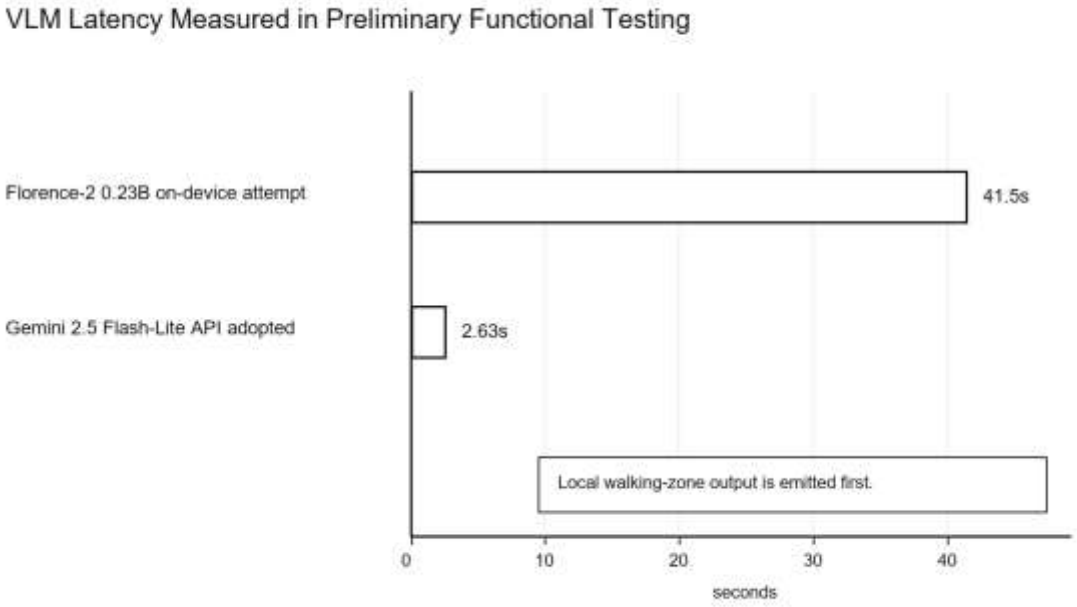

Fig. 3. VLM latency measured during preliminary functional testing under camera-stable lighting conditions.

Despite the significant speedup achieved by the Gemini API, 2.63 seconds remains too slow for uninterrupted, continuous walking guidance. To mitigate this perceptual lag, WalkAssist implements a latency-masking feedback strategy: local ARCore spatial-awareness outputs are emitted instantaneously to handle immediate collision risks, while Gemini scene descriptions are delivered asynchronously as supplementary context once the remote response successfully arrives. For manual VLM requests, the application preprocesses the current frame by converting it into an upright bitmap constrained by the VLM_ANALYSIS_MAX_LONG_SIDE constant (768 pixels), ensuring optimal payload size and transmission efficiency over mobile networks.

VI. Technical Challenges and Mitigation Strategies

TABLE II: Development Issues Encountered in WalkAssist

Issue	Impact on the Prototype	Implemented Response
ARCore depth noise	Distance values sometimes changed sharply or disappeared.	Applied confidence checks, median sampling, smoothing, and fallback hit-test logic.
Object-depth alignment	YOLO detections alone did not indicate whether an object blocked the path.	Combined segmentation masks with raw-depth samples and used bounding-box fallback sampling.

Multiple feedback sources	Obstacle, route, OCR, and VLM messages could compete for output.	Separated mapping, policy, and runtime layers to prioritize feedback.
VLM response delay	Remote scene descriptions were slower than local spatial updates.	Announced walking-zone information first and used Gemini output as delayed context.
Unsafe or internal text	AI output could include unsuitable phrases or detector terms.	Sanitized scene-description text and removed internal evidence keys before TTS.
Low-light operation	Tracking and depth quality degraded under poor lighting.	Displayed sensor state and treated ARCore distance as advisory rather than exact.

Developing a real-time mobile walking-assistance application introduces unique software and hardware challenges, particularly when fusing heterogeneous vision models, spatial depth sensors, and multimodal feedback layers on resource-constrained smartphones. Table II summarizes the core technical hurdles encountered during the development of WalkAssist and the corresponding algorithmic and architectural solutions implemented to maintain system reliability and safety.

A. Sensor Instability and Environmental Sensitivity

ARCore Depth Noise: Raw depth measurements obtained from consumer mobile devices often exhibit high-frequency jitter, sudden dropouts, or spatial inaccuracies due to textureless surfaces or rapid camera motion. To stabilize distance estimations, the system integrates multi-frame median sampling, temporal smoothing filters, and a confidence-check mechanism backed by fallback hit-test logic.

Low-Light Degradation: Visual-inertial odometry and depth sensing heavily rely on adequate ambient illumination. Under poor lighting conditions, tracking reliability degrades significantly. Rather than failing silently, WalkAssist explicitly exposes real-time sensor status via the UI and treats ARCore distance metrics as advisory feedback rather than exact spatial measurements.

B. Real-Time Perception and Model Latency

Object-Depth Alignment: Standard object detection bounding boxes fail to specify whether an identified obstacle actually obstructs the user's immediate walking path. WalkAssist resolves this by fusing YOLO26n-seg segmentation masks with ARCore raw-depth samples. If sufficient depth points are captured within a mask, precise object-level distance is derived; otherwise, the system gracefully resorts to bounding-box fallback sampling.

VLM Response Latency: Remote cloud-based vision-language models, despite being significantly faster than on-device alternatives, still introduce noticeable network and inference delays (e.g., 2.63 seconds for the Gemini 2.5 Flash-Lite API). To prevent dangerous interaction lags, WalkAssist decouples local spatial safety tracking from semantic scene description. Local spatial warnings are emitted instantaneously, while VLM outputs are delivered asynchronously as delayed contextual enhancements. Furthermore, raw AI responses are thoroughly sanitized to strip out internal confidence keys or inappropriate descriptive terms before text-to-speech rendering.

C. Multimodal Feedback Management

Information Overload and Conflict: Running concurrent perception modules (spatial tracking, object detection, OCR, and VLM descriptions) frequently results in competing or redundant output signals. A visually impaired pedestrian walking in a dynamic environment can easily be overwhelmed by excessive audio-haptic cues. To address this, WalkAssist enforces a strict separation between measurement mapping, policy evaluation, and runtime execution. The FeedbackPolicy layer dynamically filters and prioritizes messages based on collision risk severity, ensuring that critical safety warnings take absolute precedence over auxiliary route guidance or text readings.

VII. CURRENT LIMITATIONS AND FUTURE RESEARCH DIRECTIONS

While WalkAssist successfully demonstrates the feasibility of integrating heterogeneous perception models, AR spatial sensing, and policy-driven feedback into a unified mobile application, several inherent limitations must be addressed in future work to transition from an experimental prototype to a reliable assistive tool.

A. Sensor Limitations and Environmental Constraints

- Approximate Depth Estimation: The current prototype relies on software-level ARCore depth estimation rather than dedicated, hardware-calibrated safety-grade depth sensors. Consequently, distance measurements remain approximate and should not be interpreted as

exact metric distances suitable for critical safety decisions.

- Environmental Vulnerability: Sensor performance heavily degrades under adverse real-world conditions, including low-light environments, low-texture flooring, highly reflective surfaces, rapid camera movements, and partial object occlusions. Under such circumstances, while the application continues to provide adaptive UI feedback, the underlying spatial data lacks the reliability required for unassisted real-world navigation.

B. Vision-Language Model Latency and Real-Time Constraints

- Asynchronous Semantic Lag: Although adopting the Gemini 2.5 Flash-Lite API drastically improved response times compared to on-device alternatives (reducing latency from 41.5 seconds down to 2.63 seconds), a delay of over two seconds remains a critical bottleneck for continuous, real-time walking guidance. Seamless semantic integration requires near-instantaneous inference to keep pace with dynamic pedestrian environments.

C. Future Research Directions

- Hardware-Enhanced Sensing: Future iterations of WalkAssist should incorporate dedicated hardware sensors, such as Time-of-Flight (ToF) or RGB-D modules, to substantially enhance spatial measurement stability, accuracy, and overall user safety.
- Advanced Real-Time VLMs: Research efforts will also focus on adopting emerging ultra-low-latency edge-cloud hybrid VLM architectures. Such improvements are essential to bridge the gap between rich semantic scene understanding and the immediate responsiveness demanded by visually impaired pedestrians.

VIII. CONCLUSION

This paper presented WalkAssist, an integrated Android-based walking-assistance prototype designed to support blind and low-vision pedestrians through a unified mobile interface. By combining ARCore spatial sensing, YOLO-based object segmentation, ML Kit OCR, the Gemini 2.5 Flash-Lite API for semantic scene description, map-based route guidance, and a policy-driven multimodal feedback layer, the system demonstrates the practical feasibility of orchestrating heterogeneous perception modules on commodity smartphones. Rather than claiming formal clinical effectiveness or certified safety validation, this work highlights how disparate vision and spatial technologies can be cohesively structured into a single assistive workflow.

Furthermore, empirical development experience revealed critical real-world constraints: ARCore-based distance estimation remains approximate, environmental factors such as low-light conditions severely degrade tracking reliability, and remote VLM latency continues to pose a barrier to instantaneous semantic navigation. To mitigate these challenges, WalkAssist implements robust architectural strategies, including confidence filtering, local-first spatial prioritization, latency-masked asynchronous VLM delivery, and strict feedback policy management. Future work will focus on overcoming hardware limitations by integrating dedicated ToF or RGB-D sensors for enhanced spatial accuracy, as well as adopting advanced, ultra-low-latency vision-language models to achieve true real-time semantic scene understanding for visually impaired users.

VIIII. ACKNOWLEDGMENTS

"This research was supported by the MSIT(Ministry of Science ICT), Korea, under the National Program for Excellence in SW, supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation) in 2026"(No. 2024-0-00023).

REFERENCES

1. World Health Organization, "Blindness and vision impairment," Fact sheet, Feb. 10, 2026. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment
2. Google, "Use Depth in your Android app," ARCore Developer Guide. [Online]. Available: https://developers.google.com/ar/develop/java/depth/developer-guide
3. Google Codelabs, "ARCore Raw Depth." [Online]. Available: https://codelabs.developers.google.com/codelabs/arcore-rawdepthapi
4. Google, "Recognize text in images with ML Kit on Android." [Online]. Available: https://developers.google.com/ml-kit/vision/text-recognition/v2/android
5. Google, "ARCore supported devices." [Online]. Available: https://developers.google.com/ar/discover/supported-devices
6. Google Cloud, "Gemini 2.5 Flash-Lite." [Online]. Available: https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash-lite
7. B. Xiao et al., "Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks," arXiv:2311.06242, 2023.
8. Microsoft, "microsoft/Florence-2-base," Hugging Face model card. [Online]. Available: https://huggingface.co/microsoft/Florence-2-base

9. TensorFlow, "TensorFlow Lite for Android." [Online]. Available: https://android.googlesource.com/platform/external/tensorflow/+/main/tensorflow/lite/g3doc/android/index.md
10. Y. H. Lee and G. Medioni, "A RGB-D camera based navigation for the visually impaired," in Proc. RSS Workshop RGB-D, 2011.
11. F. Katzschmann, B. Araki, and D. Rus, "Safe Local Navigation for Visually Impaired Users With a Time-of-Flight and Haptic Feedback Device," IEEE Trans. Neural Systems and Rehabilitation Engineering, vol. 26, no. 3, pp. 583-593, 2018.
12. S. K. Agrawal et al., "An ARCore Based User Centric Assistive Navigation System for Visually Impaired People," Applied Sciences, vol. 9, no. 5, 989, 2019.
13. ONNX Runtime, "NNAPI Execution Provider." [Online]. Available: https://onnxruntime.ai/docs/execution-providers/NNAPI-ExecutionProvider.html
14. Android Developers, "TextToSpeech," Android API reference. [Online]. Available: https://developer.android.com/reference/android/speech/tts/TextToSpeech