# Employing Computer Vision on a Smartphone to Help the Visually Impaired Cross the Road

**Muhammad Imaad M. Ismail, Mahmoud A. A. Mousa**

Heriot-Watt University
School of Mathematical and Computer Sciences
mm2027@hw.ac.uk, m.mousa@hw.ac.uk

## Abstract

This paper presents a smartphone-based hybrid computer vision system designed to assist visually impaired (VI) individuals in safely navigating pedestrian crosswalks. Existing assistive technologies often depend on controlled crossings and require external hardware, limiting their usability in diverse real-world scenarios. In contrast, this system leverages a standard smartphone camera to detect vehicles and recognize pedestrian traffic lights in real time. The proposed framework integrates two lightweight YOLOv11 models—one for vehicle detection and another for pedestrian traffic light classification—alongside MiDaS v2.1 for monocular depth estimation. These models were trained on public datasets (KITTI and blind-assist1), optimized using TensorFlow Lite, and deployed as two Android applications providing auditory feedback for real-time guidance. Performance evaluations demonstrate high accuracy in object detection and reliable depth estimation under various conditions. Usability testing further confirms the practicality of the system in live environments. By combining accessibility, mobility, and context-aware scene understanding, this work offers a low-cost, deployable alternative for improving independent mobility in the VI community.

## Introduction

Computer Vision (CV) techniques have enabled smartphones to understand dynamic environments in real time, supporting applications such as autonomous driving, robotics, and assistive navigation. The proliferation of lightweight deep learning (DL) models, combined with optimized mobile inference engines like TensorFlow Lite, has made real-time vision feasible on low-power devices (Ignatov et al. 2019; Wang, Cui, and Lai 2019).

This capability is particularly transformative in assistive technologies for visually impaired (VI) individuals, where scene awareness, traffic understanding, and obstacle avoidance must be performed in real-time under hardware constraints (Vijetha and Geetha 2024; Vrysis et al. 2024). Traditional assistive systems often rely on cloud processing, wearable sensors, or infrastructure-based crossings, which can be costly, less portable, or unreliable in uncontrolled environments (Khan et al. 2020; Li et al. 2020).

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In contrast, we propose a smartphone-only system for safe road crossing that integrates two real-time DL pipelines: a vehicle detection and depth estimation app for uncontrolled crossings, and a pedestrian light recognition app for controlled crossings. Our approach uses YOLOv11 for object detection (Khanam and Hussain 2024), MiDaS v2.1 for monocular depth estimation (Sarızeybek and Isık 2022), and is fully optimized for on-device inference. Both apps include audio feedback using Android's TTS engine, issuing verbal cues like "safe to cross" or "car approaching".

Unlike other systems that require extra hardware (e.g., LiDAR, GPS, stereo cameras), our method leverages only the monocular rear camera of a smartphone, improving portability and accessibility. Evaluations across benchmarks such as KITTI (Geiger et al. 2013) and blind-assist1 (X 2024) show our system achieves comparable precision to recent state-of-the-art works, while maintaining real-time performance on mid-tier Android devices.

The contributions of this paper are as follows:

- A hybrid Android-based vision system to support both controlled and uncontrolled pedestrian crossings.
- Integration of YOLOv11 and MiDaS models for combined object detection and depth estimation in real-time.
- Modular dual-app structure optimized for low inference latency using TensorFlow Lite.
- Comprehensive evaluation on datasets and real-world testing for usability and responsiveness.

## Related Work

Recent advances in DL compression, on-device inference, and monocular perception have enabled complex CV systems to operate on smartphones. This section discusses the core components relevant to the proposed system: real-time DL on mobile devices, monocular depth estimation, object detection and tracking, and hybrid assistive tools for the VI.

### Deep Learning on Mobile Devices

Running DL models on smartphones is constrained by thermal limits, power budgets, and memory capacity. Frameworks such as TensorFlow Lite have enabled real-time inference on Android by supporting quantization and operator fusion (Tang 2018; Ignatov et al. 2019). Several studies benchmark DL performance across mobile CPUs and

GPUs, showing that lightweight models like MobileNet and YOLOv5 can run efficiently even on mid-range devices (Ignatov et al. 2019).

Cai et al. developed YOLObile, a compression-optimized YOLOv4 variant for smartphones, achieving a 14× reduction in size with minor accuracy tradeoffs. Furthermore, Wang, Cui, and Lai highlighted the challenges and best practices for integrating DL in mobile edge environments.

## Monocular Depth Estimation (MDE)

MDE aims to infer spatial depth using a single RGB image. Traditional methods like Inverse Perspective Mapping (IPM) required prior knowledge of scene geometry and calibration (Bao and Wang 2016), which limits robustness. DL-based approaches generalize better by learning pixel-to-depth mappings from large-scale datasets.

MiDaS (Ranftl et al. 2022) is one of the most generalizable models for MDE, trained across diverse datasets like KITTI, NYU Depth, and MegaDepth. It captures multi-scale spatial features through a Transformer-based encoder-decoder architecture. Researchers successfully deployed MiDaS v2.1 on Android devices, reporting acceptable inference times and strong structural consistency—further validating its suitability for mobile assistive systems (Sarızeybek and Isık 2022). External benchmarking studies demonstrate that MiDaS achieves superior structural consistency and edge sharpness compared to DenseDepth in edge-device environments (Padkan et al. 2023). Object-specific distance estimation has been explored in monocular setups (Zhu and Fang 2019), but often requires depth post-processing unsuitable for mobile inference.

## Monocular Object Detection and Tracking

Real-time object detection on mobile platforms typically uses one-stage detectors like YOLO due to their low latency. YOLOv11 (Khanam and Hussain 2024) introduces architectural improvements such as lightweight spatial attention and enhanced residual connections to improve detection accuracy while reducing computational load.

Object tracking is frequently achieved using DeepSORT, which combines Kalman filtering and CNN-based appearance features. YOLOv4 combined with DeepSORT has been used for real-time vehicle tracking (Zuraimi and Zaman 2021). Similarly, YOLOX with DeepSORT has been employed to estimate vehicle speed and position with high accuracy on monocular streams (Zhang et al. 2024).

## Hybrid Assistive Systems

Hybrid systems that combine object detection with depth estimation are emerging as powerful tools for navigation assistance. Obs-tackle (Vijetha and Geetha 2024) uses MiDaS and semantic segmentation to evaluate obstacle proximity. It also integrates TopFormer for lane boundary recognition and provides audio feedback. However, it is implemented on high-end smartphones with 8GB RAM and lacks modular design, which can lead to overheating when multiple models run simultaneously.

A wearable navigation system combining a monocular camera, GPS, and ultrasonic sensors has been implemented for outdoor localization (Khan et al. 2020). Though it is effective in that general context, it does not support traffic signal recognition or vehicle motion estimation.

CrossSafe (Li et al. 2020) targets pedestrian signal recognition for the visually impaired using a CNN, but it lacks support for uncontrolled crossings. Meanwhile, LytNet, a pedestrian signal classifier with over 96% accuracy, was proposed and optimized for iOS, though it depends on a pre-trained MobileNet, and its restriction to iOS devices restricts its use to a specific demographic (Yu, Lee, and Kim 2019). A CNN-based system for blind navigation was presented, achieving strong accuracy using obstacle-aware segmentation, though their model was not designed for mobile deployment (Atitallah et al. 2024). Another vision-based system was implemented that detects multiple scene types for the visually impaired, though it does not perform fine-grained traffic context recognition (Divina, Richard, and Raimond 2023). These solutions address important aspects of visual navigation but are either specialized, platform-dependent, or insufficiently generalized to handle complex crossing scenarios.

## Assistive Vision Systems for the Visually Impaired

Obs-tackle (Vijetha and Geetha 2024) is a hybrid system combining segmentation and depth estimation for obstacle detection using MiDaS and TopFormer. It provides safe path suggestions to VI users but is computationally intensive for real-time use on phones. Another Android-based navigation aid was proposed that used SSD-MobileNet and GPS but lacked support for dynamic vehicle tracking (Khan et al. 2020). CrossSafe (Li et al. 2020) employed a CNN to detect pedestrian lights and provide haptic feedback using Jetson TX2 hardware. LytNet (Yu, Lee, and Kim 2019), a lightweight pedestrian signal detector that achieves 96% accuracy on iOS. Unlike our system, these methods are either dependent on external hardware or lack vehicle motion estimation. Another notable system was proposed which integrates YOLOv4-based object detection, pedestrian traffic light recognition, and distance estimation into a head-mounted mobile assistive device (Tian et al. 2021). While their approach achieved high accuracy across scene components, including crosswalk boundaries, pedestrians, and traffic signals, it was primarily tailored to controlled crossings and required dedicated hardware such as the Intel RealSense camera, making it less accessible than a smartphone-only solution. Our proposed system, by contrast, supports both controlled and uncontrolled scenarios using only a smartphone camera, offering a more accessible solution for VI users.

## Critical Analysis and Research Gaps

Most current systems focus exclusively on controlled crossings, using traffic light cues without supporting unregulated environments. Others depend on additional hardware such as compute modules or headsets, limiting accessibility. Moreover, few combine traffic light recognition, vehicle detection, and depth estimation into a unified mobile pipeline.

Moreover, current solutions often lack modularity or require persistent network access, limiting their deployment in real-world urban settings. Few systems attempt to fuse

traffic light classification with depth-aware object detection on mobile devices in a fully offline pipeline.

This work addresses these gaps by proposing a lightweight, dual-app smartphone system integrating all three functionalities. It operates entirely on-device and offers audio-based scene feedback tailored for VI pedestrians in both controlled and uncontrolled settings.

## Proposed Framework

The proposed system consists of two Android applications that operate independently to assist VI users in crossing roads safely. One application handles vehicle detection and depth estimation, while the other is responsible for pedestrian traffic light classification. This dual-app architecture was chosen due to mobile hardware limitations and to prevent overloading from running multiple heavy models simultaneously.

### Application Structure and Architecture

The applications were implemented using Android Studio and Kotlin, utilizing the CameraX API for real-time frame capture. Frame analysis and overlay rendering were handled asynchronously to maintain throughput, and bounding boxes were drawn using a custom `OverlayView`. Safety logic was encapsulated in modular `Activity.kt` classes for maintainability. The full system was initially designed as a single app; however, due to issues such as segmentation faults, overheating, and reduced responsiveness, it was split into two separate applications optimized for performance. The user can seamlessly switch between these via Google Assistant commands.

**Vehicle Detection and Depth Estimation App.** As illustrated in Figure 1, this application includes three main modules:

- A YOLOv11-based object detector trained to identify vehicles such as cars, vans, trucks, and trams.
- A MiDaS v2.1 Small model for monocular depth estimation, providing pixel-wise depth information from a single 256x256 image. MiDaS was selected not only for its competitive accuracy but also for its superior performance in maintaining edge sharpness in Android deployment, which is critical to distinguish vehicles from urban backgrounds where clear object boundaries aid in safer decision-making.
- A Text-to-Speech (TTS) feedback system that generates safety alerts based on proximity and motion analysis.

Real-time camera frames are analyzed via `FrameAnalyser.kt`, and asynchronous coroutines are used to maximize throughput. Detected objects are tracked across frames using a lightweight tracker (`Tracker.kt`) which helps smooth out noise in depth data. A rule-based engine embedded in `DepthEstimationActivity.kt` determines crossing safety based on object position and movement trends, issuing verbal commands such as "Don't cross, car is approaching fast" or "Safe to cross, car is far and stable". This module builds on the open-source Android implementation by haruncetin, which served as the base for integrating MiDaS depth inference into a real-time Android environment. The original framework was extended and restructured to support dynamic bounding box evaluation, audio feedback, and frame analysis tailored for pedestrian crossing safety.

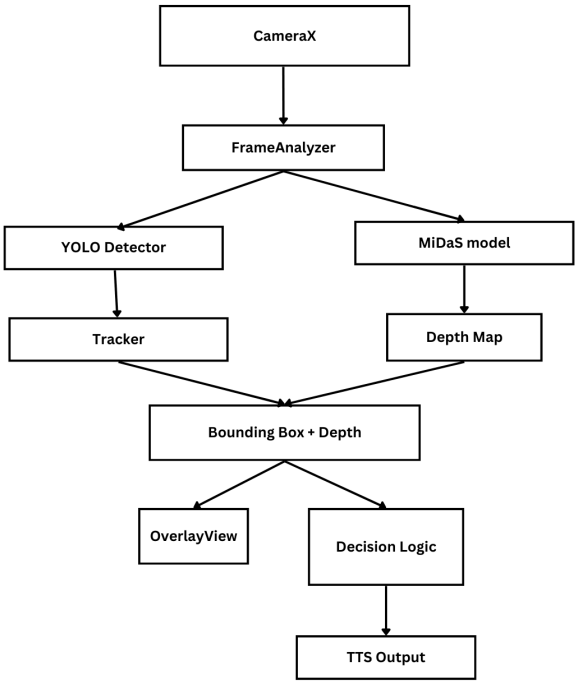


Figure 1: Structure of the vehicle detection with depth estimation application.

### Pedestrian Light Detection App

The second application, shown in Figure 2, is a streamlined tool focused on detecting and classifying pedestrian traffic lights using YOLOv11 Nano. It includes:

- Camera feed capture and preprocessing using the CameraX API.
- Inference using a TFLite model trained on the `blind-assist1` dataset with classes: `red-pedestrian-light`, `green-pedestrian-light`, and `traffic-light`.
- Confidence filtering, non-max suppression, and OverlayView rendering for bounding boxes and class labels.
- Audio feedback using Android's TTS API, updated every few seconds to avoid information overload.

Both models were trained with data augmentation techniques (e.g., mosaic, flips, color jittering) and optimized using AMP for faster training. This model structure achieved high classification accuracy, particularly for red and green pedestrian light classes. In the pedestrian light application,

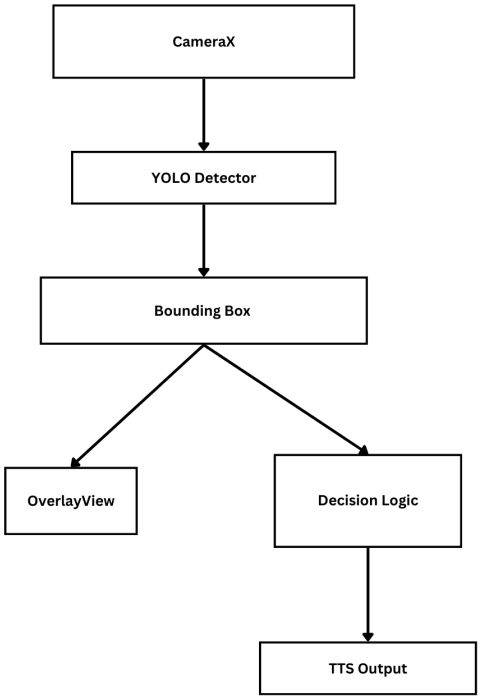


Figure 2: Structure of the pedestrian traffic light detection application.

the detection pipeline follows a standard object classification flow: the TFLite interpreter processes each camera frame to identify and classify bounding boxes. The output is filtered using confidence thresholds and non-maximum suppression to isolate the most prominent signal. Only detections labeled as `green-pedestrian-light` or `red-pedestrian-light` are used to drive feedback. Based on the active class and detection confidence, the system issues a verbal prompt advising the user to either cross or wait. This logic was implemented natively in Kotlin using Android's Text-to-Speech API and timer constraints to avoid repeated prompts.

### Decision Logic and Feedback Flow

Each application independently determines whether it is safe to cross based on different input cues. The vehicle detection app uses object depth category (Near, Close, Far) and movement patterns, while the pedestrian light app uses the classification result. Together, they offer a full understanding of both controlled and uncontrolled crossings.

All outputs are relayed to the user via voice feedback. Audio is only triggered if the safety status has changed, to avoid cognitive overload. Constants such as thresholds, model paths, and voice update intervals are defined centrally in `Constants.kt`. The core logic that governs real-time safety feedback in the vehicle detection app is outlined in Figure 3. It illustrates how YOLOv11 detections and MiDaS-generated depth values are processed together to trigger appropriate audio warnings based on object distance and motion trends.

```
# Pseudocode: Depth-Aware Vehicle Detection and Safety Assessment

load YOLOv11 model
load MiDaS depth estimation model
initialize camera feed

while camera_is_active:
    frame = capture_camera_frame()
    depth_map = depth_model.getDepthMap(frame)
    boxes = detector.detect(frame)

    for box in boxes:
        box.depth = getDepthAtBox(box, depth_map)
        if box.depth < 10:
            category = "Near"
        elif box.depth < 30:
            category = "Close"
        else:
            category = "Far"

        if category == "Near" and object_moving_towards(box):
            speak("Don't cross, fast-approaching car")
        elif category == "Far" and has_been_far_for_long(box):
            speak("Safe to cross")
```

Figure 3: Simplified pseudocode for YOLOv11 and MiDaS-based vehicle detection with real-time audio feedback.

## Experiments

This section presents the evaluation of the proposed system through both model performance metrics and user testing. We report results for the YOLOv11-based vehicle and traffic light detection models, and summarize feedback from VI participants who tested the system.

### Dataset Splits

Two datasets were used to train and evaluate the proposed system: the KITTI dataset for vehicle detection and the `blind_assist1` dataset for pedestrian traffic light detection.

**Vehicle Detection (KITTI)**: From the full 7,481 training images provided by the KITTI object detection benchmark (Geiger et al. 2013), we used 7,001 images for training and 480 for validation. A separate set of 7,518 images was used for testing, consistent with standard practices in the literature.

**Pedestrian Light Classification (blind_assist1)**: The dataset originally included 622 red, 561 green, and 568 general traffic light instances, but only the first two were retained due to low classification performance of the third class during validation. The dataset was split into 79% for

training, 16% for validation, and 5% for testing. These splits ensured balanced representation of red and green pedestrian lights, which are critical to safe crossing decisions (X 2024).

## Training Configuration

**Pedestrian Light Model (YOLOv11 Nano)**:

- Optimizer: AdamW, Learning Rate: 0.001429, Momentum: 0.9, Weight Decay: 0.0005
- Epochs: 80 with early stopping after 10 non-improving epochs (best at epoch 64)
- Input Resolution: 640×640
- Augmentations: Mosaic, horizontal flip, and color jittering
- AMP (Automatic Mixed Precision): Enabled for faster training

**Vehicle Detection Model (YOLOv11 Nano)**:

- Optimizer: AdamW, Learning Rate: 0.00125
- Epochs: 50
- Image Resizing: 1242×375 → 640×640
- Augmentations: 50% horizontal flip, full mosaic augmentation
- AMP: Enabled

Training was performed using the Ultralytics YOLOv11 training pipeline (Jocher, Qiu, and Chaurasia 2023), monitored using mAP and loss metrics across training and validation splits.

## Evaluation Metrics

System performance was evaluated using precision, recall, F1-score, and mean Average Precision (mAP) at IoU thresholds of 0.5 and 0.5:0.95.

**Precision**:

$$Precision = \frac{TP}{TP + FP}$$

**Recall**:

$$Recall = \frac{TP}{TP + FN}$$

**F1-Score**:

$$F1Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

**Mean Average Precision (mAP)**: Area under the precision-recall curve averaged across all classes and IoU thresholds.

Performance was also monitored using classification loss, objectness loss, and Distribution Focal Loss (DFL) across training epochs. No overfitting was observed, and all validation metrics improved steadily.

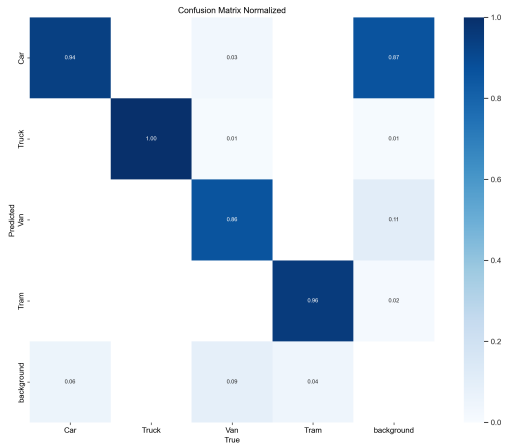


Figure 4: Confusion matrix for YOLOv11 vehicle detection.

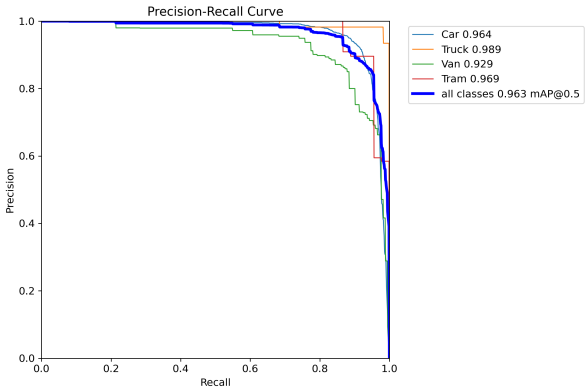


Figure 5: Precision-recall curve for YOLOv11 vehicle detection.

## Vehicle Detection Performance

The YOLOv11 model for vehicle detection was evaluated using the KITTI dataset and a custom annotated set. The normalized confusion matrix in Figure 4 indicates high precision across all classes, with the *Truck* and *Tram* classes achieving perfect classification rates.

Figure 5 shows the precision-recall (PR) curve. The model achieves a mean average precision (mAP@0.5) of 0.963, demonstrating strong generalization performance under real-world conditions.

On an Android 11 device equipped with a Qualcomm SDM765G 5G OCTA-core processor, the vehicle detection model achieved an average inference time of approximately 315 milliseconds per frame. Although not strictly real-time, this was sufficient for pedestrian-level response time in urban navigation scenarios.

## Pedestrian Traffic Light Detection Performance

The pedestrian light model, also based on YOLOv11, was trained on the `blind-assist1` dataset. Figure 6 illustrates the confusion matrix, with 98% accuracy for red lights

and 96% for green lights. The `traffic-light` class showed lower performance and was excluded from final inference logic.

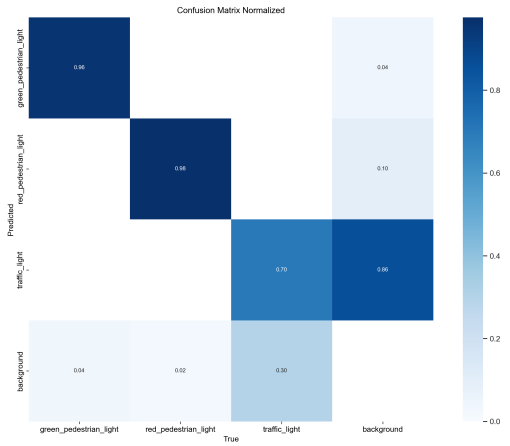


Figure 6: Confusion matrix for YOLOv11 pedestrian traffic light detection.

As shown in Figure 7, the model achieves an mAP@0.5 of 0.873 and strong recall at lower confidence thresholds, supporting robust operation under varied lighting conditions.

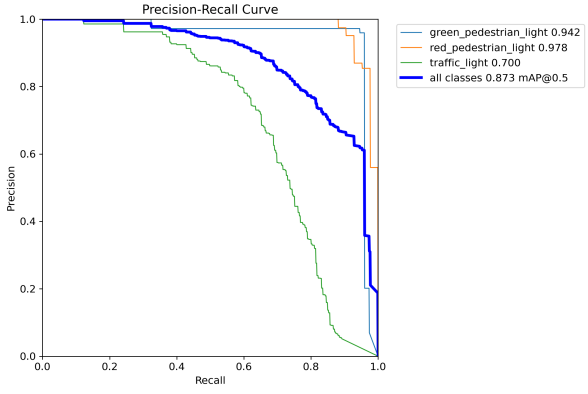


Figure 7: Precision-recall curve for pedestrian traffic light detection.

The pedestrian light detection model demonstrated a similar average inference time of 305 milliseconds per frame on the same hardware. Despite the heavier load, the application maintained responsiveness suitable for assistive feedback, given the slower pace and timing of pedestrian light changes. Although the average inference time of 305–315ms per frame is below the typical 30 FPS benchmark for real-time video, this latency is acceptable for pedestrian navigation tasks where user movement is relatively slow, and feedback intervals of under one second remain effective and non-disruptive. Similar latency tolerance has been reported in other VI navigation systems that prioritize safe and clear verbal feedback over frame rate (Vijetha and Geetha 2024).

## Usability Testing

To evaluate real-world utility, a small-scale usability study was conducted with VI participants. They rated both applications across clarity, responsiveness, and perceived usefulness. Results in Figure 8 show that audio guidance was rated highly, with an average score of 4.4/5.

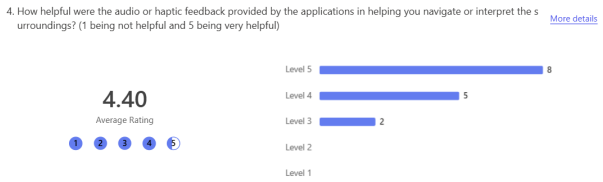


Figure 8: Helpfulness of audio feedback as rated by users.

When asked about the system's potential in real-world scenarios, 87% of participants responded positively (Figure 9), citing increased confidence and perceived safety as key benefits. A small portion indicated that the applications could be helpful with significant improvements.

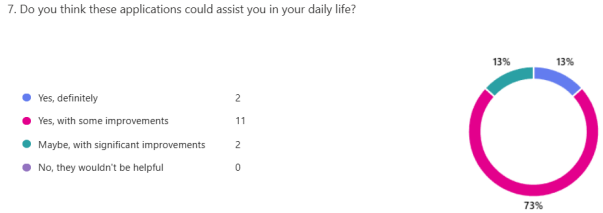


Figure 9: Perceived real-world usefulness of the system.

Participants were also asked about the overall ease of use for each application and the responsiveness of the system. As shown in Figure 10, the majority of users found both apps intuitive and easy to navigate.

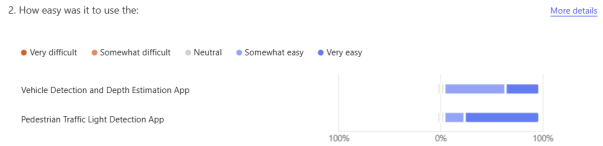


Figure 10: User rating of application ease of use.

Additionally, Figure 11 illustrates that response times were perceived as adequate for real-time crossing assistance, with most participants rating them as satisfactory or better.


Figure 11: User perception of application responsiveness.

## Limitations

Despite promising results, the system has several limitations. Firstly, the average inference speed of approximately 300 milliseconds could be enhanced to perform faster for rapidly changing environments, potentially delaying critical safety cues. Secondly, the depth estimation module relies on relative depth rather than metric absolute values, which could introduce uncertainty in distance estimation in complex scenes with multiple occlusions or varied elevation. Thirdly, while the use of two separate Android applications improves performance and reduces memory issues on mobile devices, it introduces complexity. Users must manually switch between apps depending on the crossing type (controlled vs. uncontrolled), which could be cognitively demanding in real-world scenarios. In high-stress or time-sensitive situations, this manual context-switching could lead to incorrect app usage or decision delays, increasing the likelihood of unsafe crossing attempts. Automating this process is vital for the future development of the system. Additionally, although initial user feedback was positive, the usability evaluation was limited to a small sample of participants of 15 users. Testing was conducted under relatively stable and clear weather conditions. Environmental factors such as poor lighting, rain, occlusion or extremely crowded roads may degrade model accuracy and lead to confusion. These challenging conditions may not only reduce detection accuracy but also delay critical safety alerts, posing a risk in fast-moving or unpredictable traffic scenarios. Therefore, broader and more diverse testing strategies including low-light, rainy, or high-traffic scenarios is required to reconfirm the robustness and generalization of the system. Lastly, while the system integrates several state-of-the-art models effectively for real-time pedestrian assistance, it does not introduce new architectures or novel learning paradigms. The primary contribution lies in thoughtful integration, deployment, and real-world evaluation of existing methods within a smartphone framework. Future work will focus on merging both applications into a unified system with adaptive mode switching, improving inference speed, and expanding usability trials to include diverse environmental conditions and user populations.

## Conclusion

This paper presented a smartphone-based hybrid computer vision system designed to assist VI individuals in crossing roads safely in both controlled and uncontrolled environments. The system integrates two lightweight YOLOv11 models for vehicle and pedestrian traffic light detection, along with the MiDaS v2.1 depth estimation network, all optimized for on-device inference using TensorFlow Lite.

By deploying the models across two Android applications, the framework addresses key challenges of mobile computation, including thermal throttling and memory constraints. Usability testing with VI users confirmed the system's accessibility and real-world viability, while experimental evaluation demonstrated high classification accuracy, effective depth-based reasoning, and reliable audio feedback delivery. This work serves as a successful proof of concept, demonstrating that a fully functional pedestrian assistance tool can be deployed using only a smartphone's onboard camera and processing capabilities.

Inference performance on mid-range hardware (Qualcomm SDM765G, Android 11) yielded 305–315ms per frame per model. Although not real-time by traditional benchmarks, the response time proved sufficient for pedestrian navigation scenarios. While MiDaS provides highly structured depth maps, its output is in relative units, which can limit metric accuracy without calibration. While this is an acceptable trade-off for VI feedback, it is important to improve on this aspect in future versions.

The work also demonstrates that existing deep learning architectures, when adapted carefully, can support real-time assistive technologies on widely available smartphones. Future work will focus on performance optimization through model quantization and multi-threaded execution, as well as incorporating additional context such as auditory scene analysis or semantic segmentation for more comprehensive safety cues.

## References

Atitallah, A. B.; Said, Y.; Atitallah, M. A. B.; Albekairi, M.; Kaaniche, K.; and Boubaker, S. 2024. An effective obstacle detection system using deep learning advantages to aid blind and visually impaired navigation. *Ain Shams Engineering Journal*, 15(2): 102387.

Bao, D.; and Wang, P. 2016. Vehicle distance detection based on monocular vision. In *2016 International Conference on Progress in Informatics and Computing (PIC)*, 187–191. IEEE.

Cai, Y.; Li, H.; Yuan, G.; Niu, W.; Li, Y.; Tang, X.; Ren, B.; and Wang, Y. 2021. Yolobile: Real-time object detection on mobile devices via compression-compilation co-design. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 955–963.

Divina, T.; Richard, R. P.; and Raimond, K. 2023. Assistance for Visually Impaired People in Identifying Multiple Scenes Using Deep Learning. In *International Conference on Data Intelligence and Cognitive Informatics*, 547–556. Springer.

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237.

haruncetin. 2022. GitHub - haruncetin/MonocularDepthEstimation: In this repository, monocular depth estimation is implemented using MiDaS v2.1 small model for Android mobile devices.

Ignatov, A.; Timofte, R.; Kulik, A.; Yang, S.; Wang, K.; Baum, F.; Wu, M.; Xu, L.; and Van Gool, L. 2019. Ai benchmark: All about deep learning on smartphones in 2019. In 2019 IEEE. In *CVF International Conference on Computer Vision Workshop (ICCVW)*, 3617–3635.

Jocher, G.; Qiu, J.; and Chaurasia, A. 2023. Ultralytics YOLO.

Khan, M. A.; Paul, P.; Rashid, M.; Hossain, M.; and Ahad, M. A. R. 2020. An AI-based visual aid with integrated reading assistant for the completely blind. *IEEE Transactions on Human-Machine Systems*, 50(6): 507–517.

Khanam, R.; and Hussain, M. 2024. YOLOv11: An Overview of the Key Architectural Enhancements. *arXiv preprint arXiv:2410.17725*.

Li, X.; Cui, H.; Rizzo, J.-R.; Wong, E.; and Fang, Y. 2020. Cross-Safe: A computer vision-based approach to make all intersection-related pedestrian signals accessible for the visually impaired. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 2 1*, 132–146. Springer.

Padkan, N.; Trybala, P.; Battisti, R.; Remondino, F.; and Bergeret, C. 2023. EVALUATING MONOCULAR DEPTH ESTIMATION METHODS. *The international archives of the photogrammetry, remote sensing and spatial information sciences/International archives of the photogrammetry, remote sensing and spatial information sciences*, XLVIII-1/W3-2023: 137–144.

Ranftl, R.; Lasinger, K.; Hafner, D.; Schindler, K.; and Koltun, V. 2022. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3): 1623–1637.

Sarızeybek, A. T.; and Isık, A. H. 2022. Monocular Depth Estimation and Detection of Near Objects. *Uluslararası Teknolojik Bilimler Dergisi*, 14(3): 124–131.

Tang, J. 2018. *Intelligent Mobile Projects with TensorFlow: Build 10+ Artificial Intelligence Apps Using TensorFlow Mobile and Lite for IOS, Android, and Raspberry Pi*. Packt Publishing Ltd.

Tian, Y.; Chen, X.; Shen, Y.; Huang, Z.; Yuan, W.; and Wang, H. 2021. Scene Understanding for Pedestrian Crossing Assistance Using YOLOv4 and Depth Sensing. *Sensors*, 21(11): 3846.

Vijetha, U.; and Geetha, V. 2024. Obs-tackle: an obstacle detection system to assist navigation of visually impaired using smartphones. *Machine Vision and Applications*, 35(2): 20.

Vrysis, L.; Almaliotis, D.; Almpanidou, S.; Papadopoulou, E. P.; Oikonomides, K.; Chatzisavvas, K. C.; and Karampatakis, V. 2024. Mobile software aids for people with low vision. *Multimedia Tools and Applications*, 83(10): 30919–30936.

Wang, Z.; Cui, Y.; and Lai, Z. 2019. A first look at mobile intelligence: Architecture, experimentation and challenges. *IEEE Network*, 33(4): 120–125.

X, M. 2024. blind_assist1 Dataset. https://universe.roboflow.com/markus-x-cgbeg/blind_assist1. Visited on 2025-03-19.

Yu, S.; Lee, H.; and Kim, J. 2019. Lytnet: A convolutional neural network for real-time pedestrian traffic lights and zebra crossing recognition for the visually impaired. In *International Conference on Computer Analysis of Images and Patterns*, 259–270. Springer.

Zhang, K.; Wu, F.; Sun, H.; and Cai, M. 2024. Monocular vehicle speed detection based on improved YOLOX and DeepSORT. *Neural Computing and Applications*, 36(17): 9643–9660.

Zhu, J.; and Fang, Y. 2019. Learning object-specific distance from a monocular image. In *Proceedings of the IEEE/CVF International Conference on computer vision*, 3839–3848.

Zuraimi, M. A. B.; and Zaman, F. H. K. 2021. Vehicle detection and tracking using YOLO and DeepSORT. In *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 23–29. IEEE.