Senior Project Report

Seeit: Hear objects, Spot colors

La Min Oo (6540039)
Kaung Khant Lin (6540131)
Rebekah Anguishing Mark (6610292)
Supervised by Asst. Prof. Dobri Atanassov Batovski

CS 3200 Senior Project 1 (1/2025)

Senior Project Approval

Project title: Developing a mobile app using computer vision to help the visually impaired identify road signs and objects in real time

Academic Year: 1/2020

Authors: La Min Oo (650339)

   Kaung Khant Lin (6540131)

   Rebekah anguishing Mark (6610292)

Project Advisor: Asst. Prof. Dobri Atanassov Batovski

SEEiT: Hear objects, Spot colors

The Senior Project committee's cooperation between the Department of Computer Science and Information Technology, Vincent Mary School of Science and Technology, Assumption University had approved this Senior Project. The Senior Project in partial fulfilment of the requirement for the degree of Bachelor of Science in Computer Science and Information technology.

Approval Committee:

………………………..

(Asst. Prof. Dr. Dobri Atanassov Batovski)

Project Advisor

………………………..   ………………………..

(Dr. Kwankamol Nongpon)   (A. Tianai Tang)

Committee Member   Committee Member

Abstract

Navigating urban environments poses significant challenges for individuals with visual impairments and color vision deficiencies, particularly in perceiving and interpreting road-related signs and surrounding objects in real time. Existing assistive technologies, including smartphone-based navigation apps and filter-based color aids, offer partial solutions but often lack full integration of multi-modal features such as object detection, distance estimation, spatial orientation, and accessible audio feedback.

This paper presents the development of a real-time, multi-modal mobile application designed to enhance navigational safety and independence for visually impaired and colorblind users. With the help of computer vision and artificial intelligence, the system detects and interprets road signs and objects, estimates direction (3×3 grid) and numeric distance in meters with confidence-gated smoothing, and delivers prioritized, concise audio feedback. The Color Finder currently supports Protan (red-deficiency) analysis. The design builds on insights from current assistive solutions, including applications such as Be My Eyes, Seeing AI, and EnChroma filters, highlighting the limitations of existing systems in terms of adaptability, user-centered design, and multi-sensory interactions. By integrating real-time object and color detection, the proposed application aims to provide a reliable, inclusive, and responsive tool that enhances spatial awareness and promotes independent navigation.

Table Of Contents

Table Of Figures

Chapter 1: Introduction

1.1 Problem Statement

Navigating urban environments presents challenges for visually impaired individuals and color vision deficiency, particularly in perceiving and interpreting road related signs and objects in real time. While various assistive technologies exist, many do not completely integrate multi-modal features such as accurate object detection, distance estimation, spatial orientation and accessible audio feedback. Consequently, there remains a need for solutions that can effectively support enhanced environmental awareness and navigation safety.

This project will develop a real-time, multi-modal mobile application that enhances the navigational safety and independence of visually impaired and colorblind individuals by using computer vision to detect and interpret road-related signs and objects, estimate their distance and direction, and also audio outputs.

1.2 Scope of the project

Objective. Real-time, device recognition of road-context cues (lights, crosswalks, signs, vehicles) which remain robust under imbalance and visual clutter.

Pipeline

- Collect → Curtae images
- Balance & argument per class (caps + targeted negatives)
- Train (two phases: freeze → fine-tune) with label smoothing & class weighting
- Validate & diagnose via per-class metrics and confusion
- Export to TFLite (FP32 + INT8) with representative calibration

Data Pipeline

Scope & classes (10)

car, crosswalk, cycle, emergency_exit, stop, traffic green, traffic red, traffic yellow, truck, van.

[TRAIN]

- car: 3,182
- crosswalk: 1,420

SEEiT: Hear objects, Spot colors

- cycle: 1,080
- emergency_exit: 1,151
- stop: 1,000
- traffic green: 3,098
- traffic red: 2,195
- traffic yellow: 2,152
- truck: 1,464
- van: 900
  Total train images: 17,642

[VALID]

- car: 885
- crosswalk: 239
- cycle: 180
- emergency_exit: 190
- stop: 120
- traffic green: 226
- traffic red: 109
- traffic yellow: 113
- truck: 136
- van: 75
  Total valid images: 2,273

Early imbalance (observed).

Head vs tail: car ≈ 10,000 images vs stop ≈ 400; emergency_exit had many high-resolution, clean samples. Effect: over-prediction of head classes, with confusions such as crosswalk ↔ hazard-like pavements and emergency_exit ↔ traffic green at distance.

Curation & balancing actions.

Caps on head classes; down-sample near-duplicates from burst captures. Conservative augmentations: horizontal flip, small rotate/translate, mild brightness/contrast; color integrity preserved for traffic lights.

Stratified splits without clip leakage; 224×224 RGB, values preprocessed for EfficientNet.

Table 1Project Scope Summary

Category	In Scope	Out of Scope/ Limitations
Users	Tested on Normal vision users	Blind and visually impaired users
Functionalities	Real-time object/sign detection, distance & direction estimation, audio feedback, voice commands	Distance measurements
Technology	Smartphone sensors, AI/Computer vision	none
Environment	Pedestrian walkway, outdoors	Complex walkway(crowed/busy), poor lighting

**Limitation: This application has not been tested with real target users.*

Chapter 2: Related Work

2.1 Challenges

A persistent challenge in assistive design is the lack of user-centered approaches that consider the diversity of visual impairments and individual needs. Systems often fail due to poor adaptation to users' sensory and cognitive capabilities. Effective designs must reduce cognitive load, enhance intuitiveness, and incorporate user feedback loops. The authors stress aligning system outputs with how visually impaired users process non-visual spatial information for better usability and acceptance [1].

2.2 Navigation systems for blind and visually impaired

Smartphones have become key platforms for assistive navigation due to their built-in sensors (GPS, cameras, accelerometers, gyroscopes) and cloud computing capabilities. They support both indoor and outdoor wayfinding through apps like NavCog, Lazzus, BlindSquare, and Seeing AI. These tools leverage AI, GNSS, and BLE beacons to provide real-time environmental descriptions and object detection, fostering independence among the visually impaired [1].

2.3 User Interface

The user interface (UI) serves as the communication bridge between the navigation system and the user but remains a major bottleneck. Speech and haptic feedback are the dominant modalities, each with limitations in data throughput and situational awareness. Advanced approaches involve mixed-reality elements, virtual sound sources, and electrotactile feedback (e.g., BrainPort, vOICe). Future UIs should combine minimal cognitive demand with intuitive spatial cues, allowing users to interpret their environment naturally through multimodal feedback [1].

2.4 EnChroma and filter-based aids for color vision deficiency

Notch-filter spectacles such as EnChroma have been widely publicized as aids for red–green anomalous trichromacy. Laboratory studies that go beyond anecdote show modest, context-dependent perceptual effects rather than a "cure." A recent controlled study of EnChroma multi-notch filters in deuteranomalous observers used three experimental tasks (asymmetric color matching, discrimination thresholds, and appearance ratings) and tested performance before and after a week of use; results show significant increases in perceived saturation and modest improvements in appearance along the red-green axis, with discrimination improvements limited (significant primarily for some red stimuli). The authors stress that improvements depend on illumination, stimulus gamut (broadband surface colors vs. display primaries), and observer specifics and explicitly note that filters do not create entirely new color experiences or restore normal trichromacy [2].

*Figure 1EnChroma filter range (source: EnChroma, 2025)*
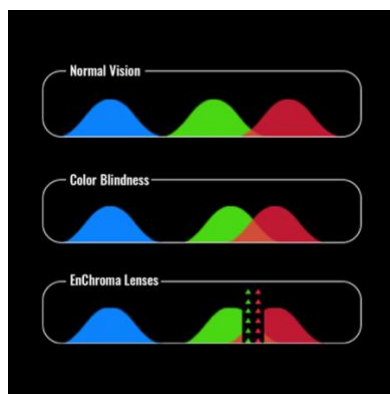
Be my eyes for the blind people

Latane et al. (2023) developed an Android-based assistive application called Be My Eyes App for Blind People to help visually impaired users navigate independently through real-time object detection and audio feedback. The system utilizes the YOLOv3-tiny algorithm for detecting surrounding objects captured via the mobile camera and converts this information into speech output through a text-to-speech (TTS) engine. The app also supports voice commands for additional tasks such as checking battery level, sending messages, or making calls. Built using Kotlin in Android Studio, the prototype demonstrates how machine learning and computer

vision can translate the visual environment into an auditory experience, promoting greater safety and autonomy for users with vision impairments [3].

Figure 2 System Architecture of the Be My Eyes App for the blind (source: P. C. Latane et al., 2023[3])
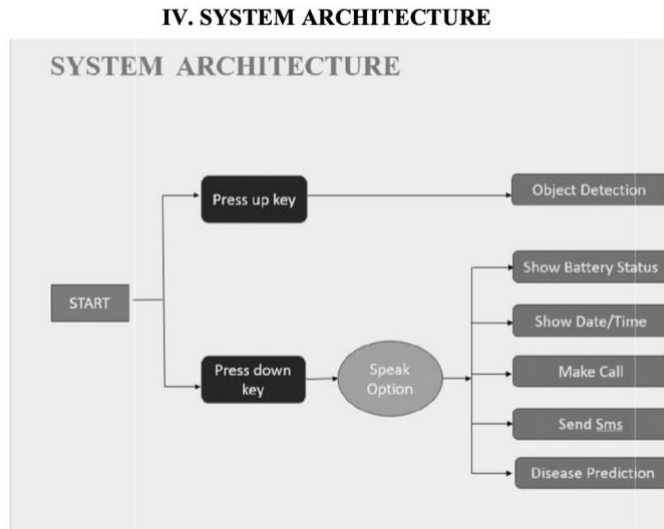
2.5 Mobile Applications

Recent studies emphasize the growing role of smartphone-based assistive technologies in improving accessibility and independence for visually impaired users. Naotunna and Hettige [4] conducted a comprehensive review of more than sixty mobile applications designed for individuals with vision impairment, identifying advancements in artificial intelligence (AI), computer vision, and GPS-based navigation. Applications such as Eye Assistant, SUBE, and Smart Eye demonstrated the integration of YOLO-based object detection and real-time auditory feedback, enabling users to identify objects and navigate their surroundings with greater safety and confidence. Despite these innovations, many systems still face issues related to voice command reliability, recognition accuracy, and the lack of user-centered design frameworks that account for varying degrees of vision loss.

In a similar context, Akkara and Kuriakose [5] analyzed the evolution of smartphone applications for visually impaired persons, categorizing them into accessibility tools, voice assistants, magnifiers, and camera-based object identifiers. Their review included widely adopted apps such as Microsoft Seeing AI, TapTapSee, Lazarillo GPS, and Be My Eyes, highlighting how smartphones have become affordable, multi-functional low vision aids that eliminate the need for costly specialized hardware. The authors further stressed the importance

of user training and awareness among both visually impaired individuals and healthcare professionals to maximize the potential of these tools.

Both studies [4], [5] demonstrate that smartphones have evolved from simple communication devices into intelligent assistive platforms that combine AI, audio-haptic feedback, and adaptive interfaces. Future development should focus on multi-modal feedback, context-aware computing, and inclusive interface design to create seamless and responsive experiences that better serve the needs of users with visual impairments.

2.6 Mobile review

Sharma and Puranik [6] presented a comprehensive review of contemporary mobile applications designed to assist visually impaired individuals, emphasizing their role in enhancing accessibility, navigation, and object recognition. The study highlights the growing application of artificial intelligence (AI), computer vision, and speech synthesis in converting smartphones into intelligent assistive tools. Applications such as Be My Eyes, Seeing AI, and TapTapSee were recognized for their ability to provide real-time visual interpretation and auditory feedback, enabling users to identify objects, read text, and navigate unfamiliar environments with improved autonomy. The review underscores the significance of smartphone-based solutions as affordable, portable, and inclusive technologies that eliminate the need for specialized hardware.

Despite notable progress, the authors identified several technical and usability limitations in existing assistive applications. Most systems rely heavily on continuous network connectivity, experience high power consumption, and offer limited customization options to address the diverse needs of users with different degrees of vision loss. Sharma and Puranik recommend future development to focus on offline processing, multi-sensor integration, and user-centered interface design to enhance reliability, adaptability, and user satisfaction. Their findings emphasize the ongoing need for context-aware and efficient mobile assistive systems, aligning with the broader objective of advancing real-time, AI-driven technologies for visually impaired users.

2.7 Summary

Assistive technologies for the visually impaired continue to face challenges due to limited user-centered design, lack of adaptability to diverse needs. Smartphones have become essential assistive platforms, using sensors, GPS, and AI in applications like NavCog, Lazzus, BlindSquare, and Seeing AI to enhance navigation and object recognition. However, user interfaces remain a key limitation, as speech and haptic feedback often fail to provide complete spatial awareness. Emerging multimodal solutions integrating audio, tactile, and electrotactile feedback show promise in improving intuitiveness and reducing cognitive load. While filter-based aids such as EnChroma glasses offer modest improvements in color perception, they do not restore normal vision. Recent studies highlight the growing impact of smartphone-based assistive apps that utilize AI, computer vision, and speech synthesis for real-time feedback, though issues like power consumption, network dependency, and limited customization persist, emphasizing the need for adaptive, offline, and user-centered solutions.

Chapter 3: Proposed Methodology

3.1 Methodology

Research strategy and process

Search terms and keywords

Visual Impairment navigation, types of color blindness, object detection for blind, speech-to-text TTS accessibility, multimodal feedback blind, crosswalk detection, color finder EnChroma study, mobile assistive apps review, offline on-device inference, BLE beacons wayfinding, user-centered design visually impaired

Problem identification

- Many systems have poor user-centered design, not adaptable to different vision loss
- Major UI limitation: speech and haptic feedback are common but do not provide enough detail, multimodal feedback need improvement.
- On the ML side, mobile-friendly detectors (e.g., YOLO variants, lightweight backbones) enable real-time inference but face class imbalance, visual clutter, and power/connectivity constraints.
- Reviews identify recurring technical issues: unreliable voice recognition, limited offline capability, high power consumption, and lack of customization for degrees of vision loss.

Proposed solution

A smartphone app that performs on-device, real-time road-context and object recognition (crosswalks, traffic lights, vehicles, cycles, etc.) and communicates concise audio feedback in addition it also guides blind and low-vision pedestrians. The app supports voice commands for status and actions (battery, messages, calls) and is optimized for offline use via an

EfficientNetB0-based model exported to TFLite (FP32 and INT8) for latency/efficiency trade-offs.

Technical approach

- **Perception pipeline**: camera capture → preproc (224×224 RGB) → EfficientNetB0 backbone classifier/detector head → softmax with label smoothing (0.05) plus margin penalty for hard negatives → per-frame predictions aggregated with short temporal smoothing to reduce flicker/false positives.
- **Training & data handling**: curate dataset of 10 classes (car, crosswalk, cycle, emergency_exit, stop, traffic green/red/yellow, truck, van). Apply curation actions: cap head-class samples, conservative augmentations (flip, small rotate/translate, mild brightness/contrast preserving traffic-light colors), stratified splits, class weighting and label smoothing, two-phase training (freeze → fine-tune). Validate using per-class metrics and confusion analysis; export representative calibration sets for INT8 quantization.
- **On-device model deployment**: export to TFLite (FP32 & INT8) with quantization-aware steps; run inference in Kotlin on Android using lightweight runtime (ML Kit or TFLite API) with a small batching/windowing policy to balance latency vs stability.
- **Interface/UX**: primary audio guidance via TTS (short, prioritized messages), optional short haptic pulses for imminent hazards; simple voice commands to query status or request repeat. UI/UX designed with low cognitive load (short utterances, prioritized info, user-configurable verbosity).

ROI Analysis

ROI Formula

$$ROI = \frac{(Total\ benefits - Total\ Costs)}{Total\ Costs} \times 100$$

Cost Estimation

Cost Component	Description	Estimated Cost (USD)
Development	App design, coding, model training (4 developers × $1,000 × 3 months)	$24,000
Testing & Optimization	Data collection, labeling, real-world trials	$6,000
Deployment	Server hosting, Play Store registration, integration tools	$2,000
Maintenance	Updates, bug fixes, and user support (annual)	$3,000
Total Cost (C)		$35,000

Benefit Estimation

Benefit Source	Description	Estimated Annual Benefit (USD)
Accessibility Grants	Government or NGO support for inclusive technology	$10,000
App Revenue	Premium features, ads, institutional license	$30,000
Reduced Support Costs	Users rely less on costly assistive hardware	$5,000
Total Cost (C)		$45,000

ROI Calculation

$$ROI = \frac{(45,000 - 34,000)}{35,000} \times 100 = \frac{10,000}{35,000} \times 100 = 28.57\%$$

Thus, the project yields an estimated ROI of 28.57% in the first year. This indicates a profitable and sustainable investment for an assistive technology.

3.2 Testing

a. Unit Testing

- Software module tested independently to verify all functions perform as expected.
- Scope: Model loading, camera input capture, text-to-speech output, and voice command recognition modules.
- Tools: Android Studio Unit Test Framework (JUnit), Espresso for UI component testing.
- Expected Outcome: Each function should return correct outputs under valid inputs and handle exceptions gracefully.

b. Integration Testing

- Integration testing will confirm that different modules (object detection model, TTS engine, and user interface) work cohesively.
- Scenario: Test whether detected objects are correctly converted into speech messages and delivered without noticeable delay.
- Expected Outcome: Smooth data flow between components with no crashes, speech delay ≤ 0.5 seconds.

c. System Testing

- Comprehensive end-to-end evaluation of the entire system on real devices.
- Parameters Tested:
- Accuracy of object and crosswalk detection
- Latency of model inference and audio response
- Power consumption during continuous use
- Offline functionality (TFLite INT8 model)

Environment: Tested on multiple Android devices with varying specifications under different lighting and outdoor conditions.

d. Performance Testing

Focuses on evaluating speed, scalability, and resource efficiency.

Metrics:

- **Inference speed**: ≤ 300 ms per frame
- **FPS rate**: ≥ 10 frames/sec for smooth feedback
- **Battery drain**: $< 20\%$ after 2 hours of continuous use
- **Model size**: ≤ 25 MB (INT8 quantized version)
- **Tools**: Android Profiler, TensorFlow Lite Benchmark Tool.

f. Validation Testing

This ensures that the final system meets the intended functional requirements defined in Section 4.1.

- **Crosswalk Detection:** $\geq 80\%$ precision and recall
- **Object Detection**: Correctly identify and announce cars, cycles, and obstacles within 3 meters
- **Color Finder**: $\geq 85\%$ accuracy in identifying basic color categories under daylight illumination.

Chapter 4: Design of The System (or Work)

4.1 Functional Requirements Specification

Stakeholders

Primary Users

- Visually impaired users – benefit from crosswalk detection and obstacle/moving vehicle alerts
- Colorblind users – benefit from the color finder filter

Crosswalk Detection

- **Actor**: Visually impaired user
- **Goal**: Detect presence and location of crosswalks to enhance safety.
- **Main flow**: App captures camera frames → model detects crosswalk → TTS announces "crosswalk ahead, 6 meters" or similar → optional haptic confirmation.
- **Postconditions**: User informed of crosswalk location/status; detection logged.

Object Detection (obstacles / moving vehicles)

- **Actor**: Visually impaired user
- **Goal**: Identify obstacles and moving vehicles to avoid hazards.
- **Main flow**: Continuous inference on camera frames → prioritized hazard messages (e.g., "vehicle approaching from left") → temporary alert suppression to avoid overload.

Postconditions: User alerted; app lowers verbosity after acknowledgement.

Color Finder (for color-blind users)

- **Actor**: Color-deficient user
- **Goal**: Identify and distinguish colors when requested.
- **Main flow**: User invokes color-finder mode → app samples pixel region → TTS reports color name and confidence.

- **Postconditions**: Color information provided; mode exits or stays active per user preference.

Table 2 Use Cases

Use Case	Actor	Goal	Predictions	Main Flow	Postconditions
Crosswalk Detection	Visually impaired users	Safety detect crosswalk	Mobile App (SeeIt), use camera		Alert through speech "Crosswalk"
Object Detection	Visually impaired users	Identify obstacles, moving vehicle	Detect cars, automobiles, road signs		Speech: "Car, truck, motorcycle, etc."
Color Finder	Color Blind users	Identify and distinguish colors	Depicts exact color in picture		Displays the exact color of the image.

***Limitation: has not been tested on target users (vision impaired and blind)*

4.2 System Design (consider items that applied)

1. **Workflow**
- User interaction – user taps Capture button on the RN color finder
- Capturing Image -RN color finder screen tells VisionCamera (photo) to take photo
- Detecting colors - App triggers detectConfusableColors with image URI and settings.
- Image decoding - BitmapIO decodes the image into RGB format.
- Pixel level analysis - Each pixel is converted to Lab space, simulated for protan, and color differences are computed.
- Applying Rules - Pixels are grouped into buckets based on color confusion
- Region Extraction - Confusable color regions and stats are identified.
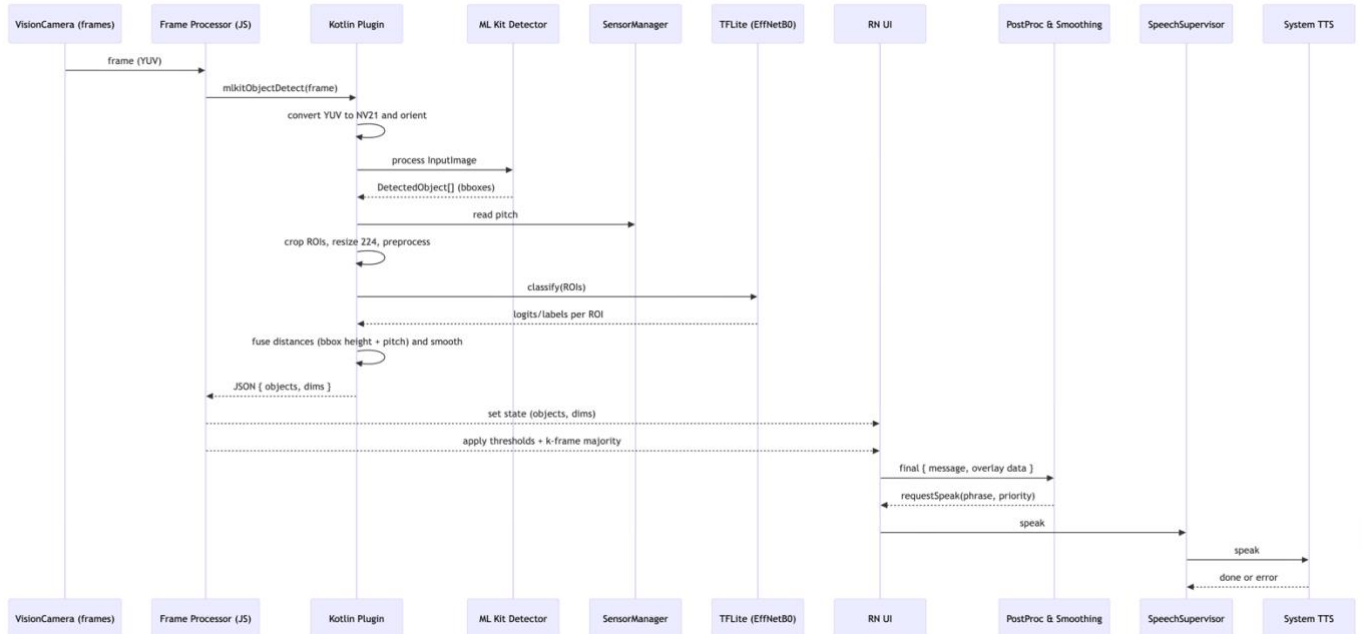- Displaying Results - Modal opens showing highlighted regions; analysis complete.

1.a Color analysis Flow

Figure 3 Color Finder (Capture → Analyze)
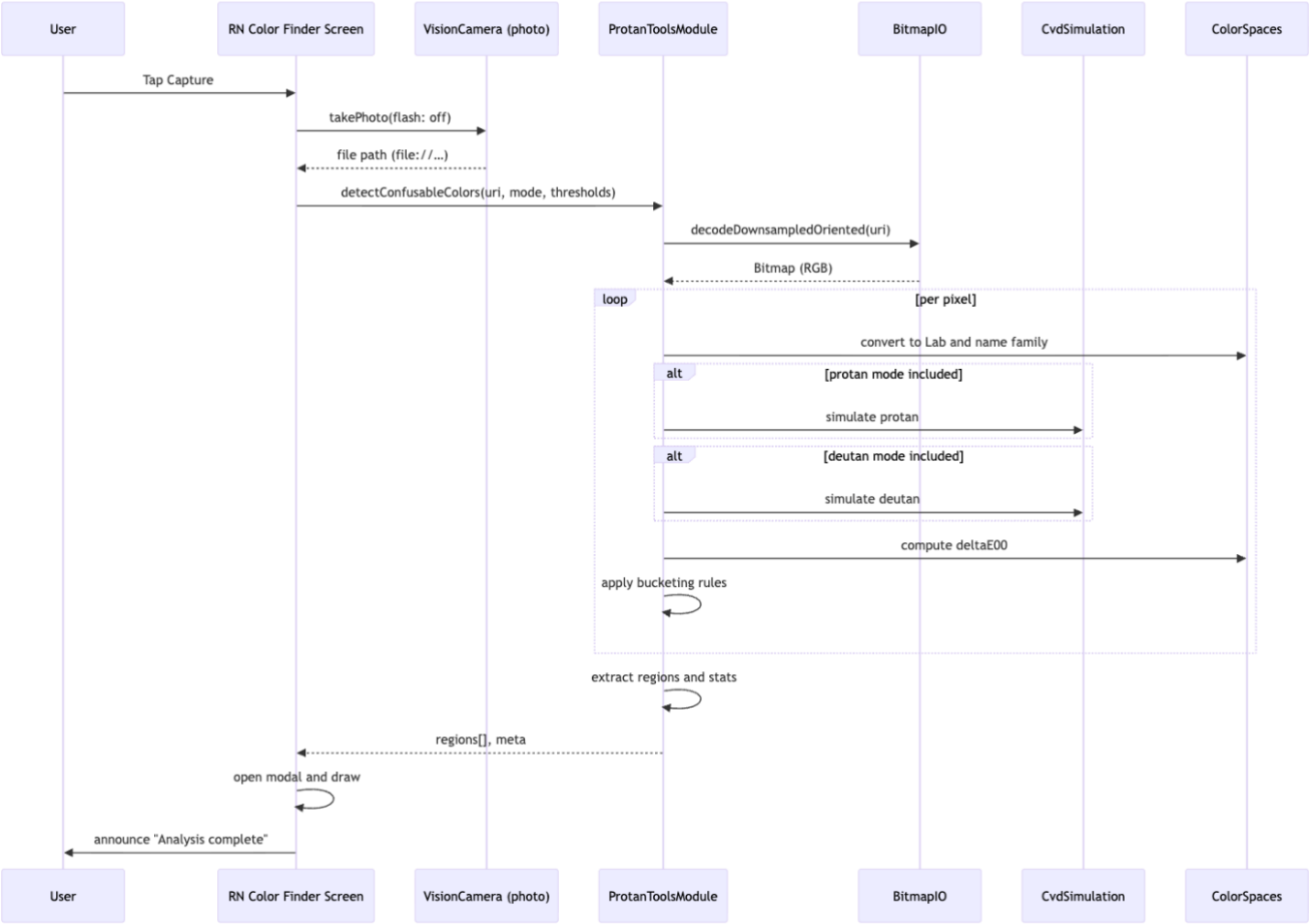
1.b Sequence Diagram, real-time object detecting detection and speech

Figure 4 Live Detection with Speech

- Frame Capture - VisionCamera captures frames (YUV).
- Frame Processing - Frame Processor triggers mlKitObjectDetect(frame).
- Image Conversion - Kotlin Plugin converts YUV to NV21 and orients image.
- Object Detection - ML Kit Detector processes input image and returns detected objects.
- Pitch Reading - SensorManager provides pitch data.
- ROI Handling - Detected regions are cropped, resized, and preprocessed.
- Classification - TFLite model classifies each ROI and returns logits/tables.
- Post Processing - Fuse distances (bbox + pitch), smooth results, and format JSON output.
- Thresholding - Apply thresholds and majority voting to refine detections.
- State Update - Kotlin Plugin updates state with detected objects and directions.
- UI Update - RN UI displays final message and overlay data.
- Speech Request - RN UI triggers speech output request.
- Speech Output - SpeechSupervisor and System TTS synthesize and speak the result.

2. System Architecture

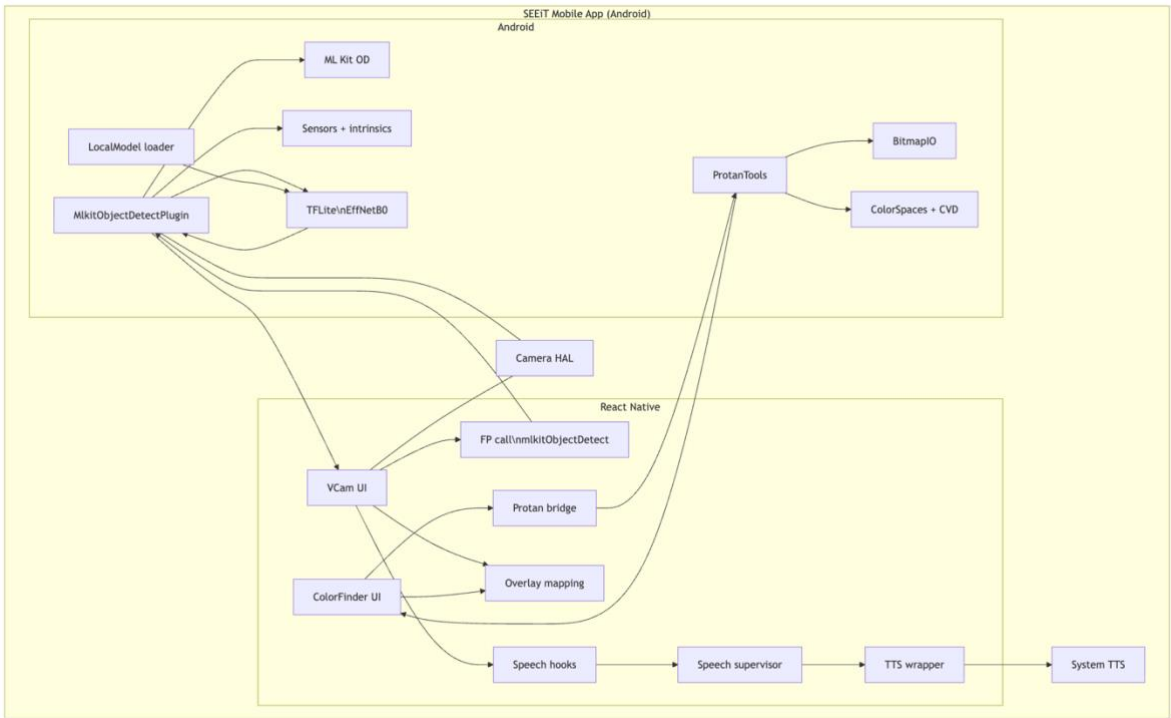*Figure 5 System context: on device, CV and system TTS*
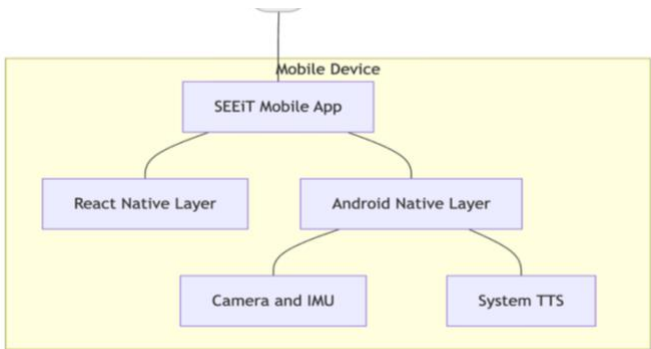
Figure 6 System Architecture

SEEiT: Hear objects, Spot colors

3. Tech Stack

- Github
- EfficientNet
- Expo
- ML kit
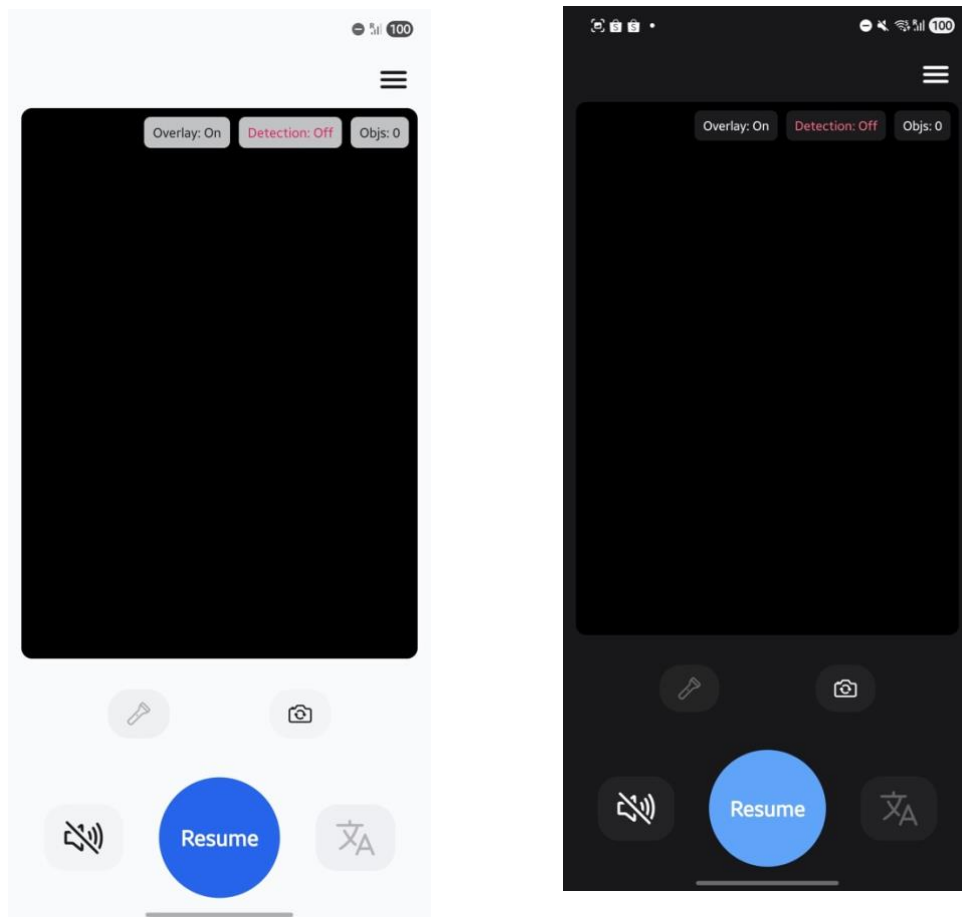- React native

4. UI/UX Design

Figure 7 Camera Screen

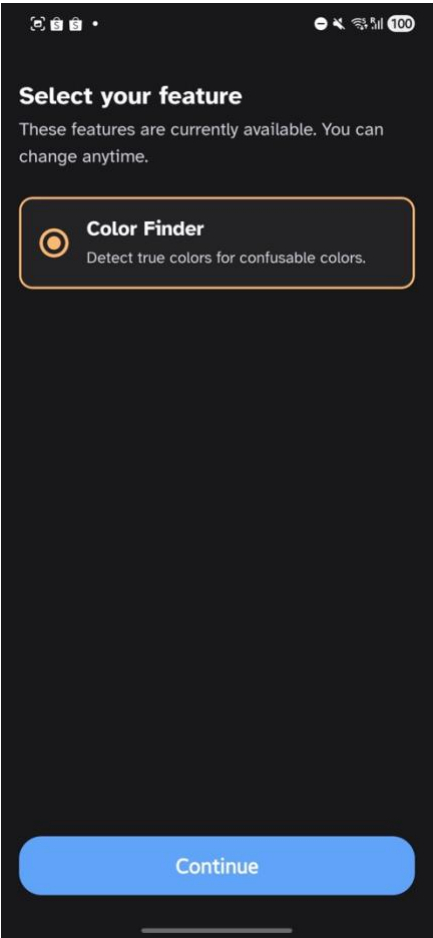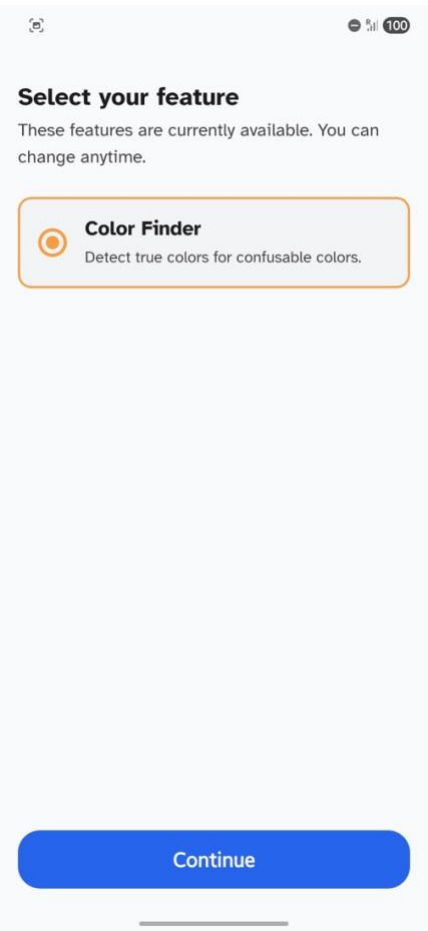*Figure 8 Select feature Screen: Color Finder*

Figure 8 Select feature Screen: Color Finder
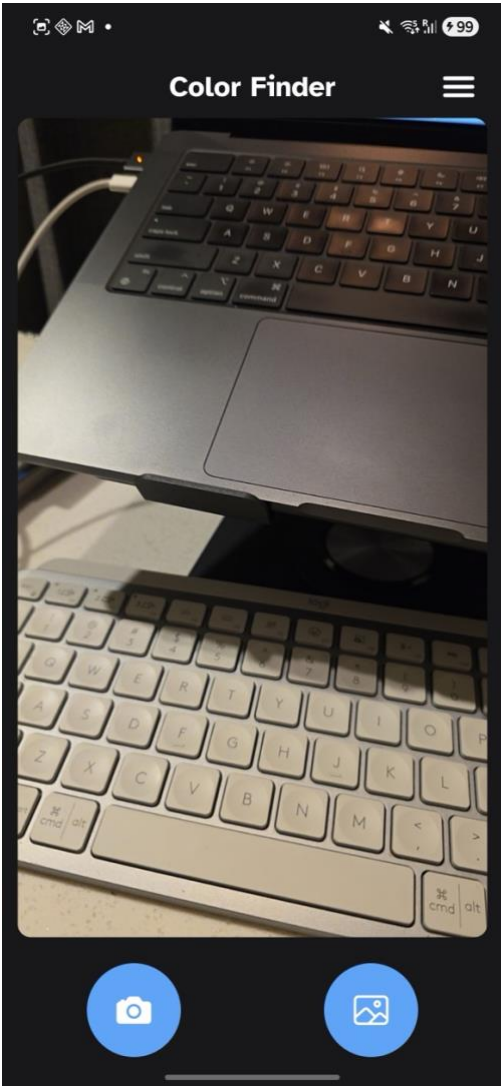
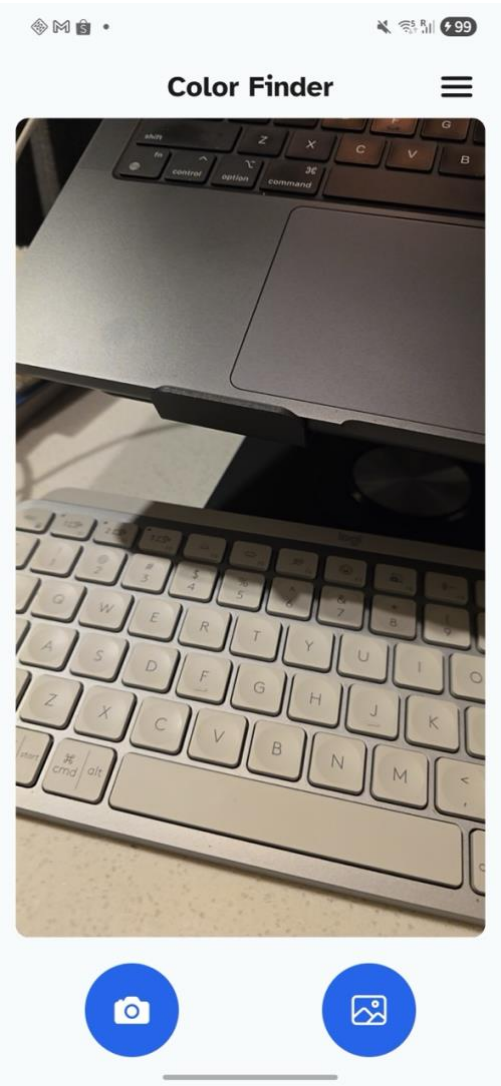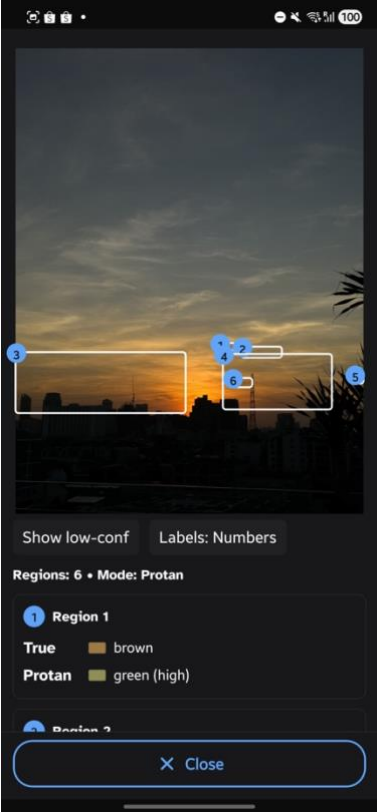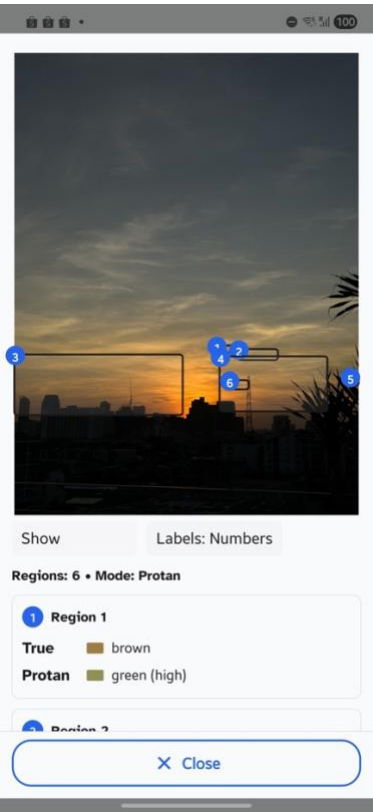*Figure 9 Camera Screen for Color finder*

Figure 10 Color detection screen

Figure 10 Color detection screen

Figure 11 Feedback Screen

Chapter 5: Result

## 5.1 Demonstration of the App

Figure 12 Paused Camera Screen

This is our app's main screen. User will see Resume button, Speech button which is with speaker icon, flash and camera rotate button, language selection button. Near Top of the screen, show overlay On/Off button, Detection: Off; detection status showing box, and Obj: 1; how many objects detected showing box.

User click on the Resume button to start the live camera view, the detection will start working.

Figure 13 Live Camera feed detection on
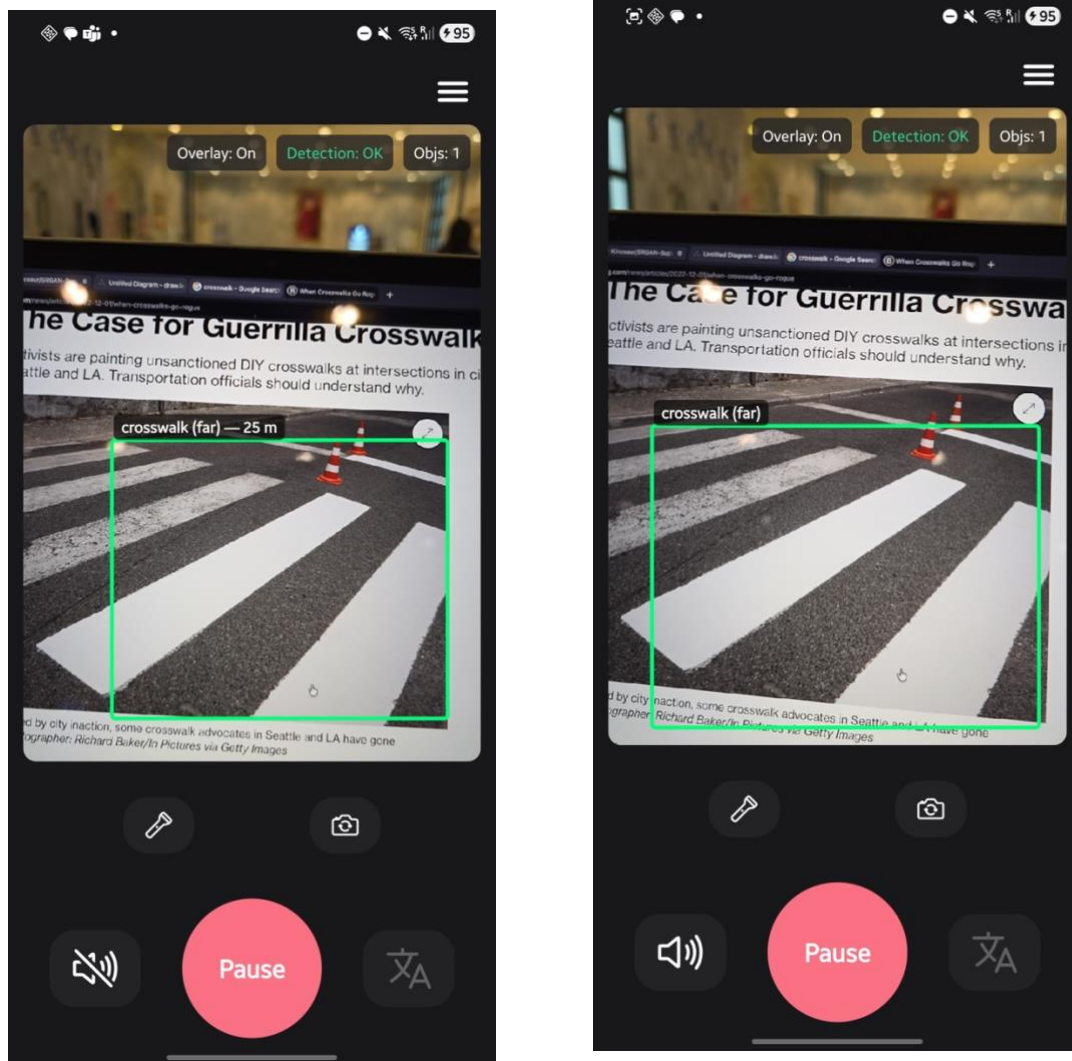
The resume button is changed to a Pause button. This shows that the user can start moving the camera to see the detection. In the picture, the crosswalk is detected along with estimated distance. The detector can report multiple objects; speech summarizes up to three label groups per utterance. User can click on the speech button and start listening to the announcements on the detected objects (e.g., crosswalk detected at 25 meters).

User can click menu button at the top left of the screen to navigate through the app.

Figure 14 Using Drawer menu for Navigation in the app
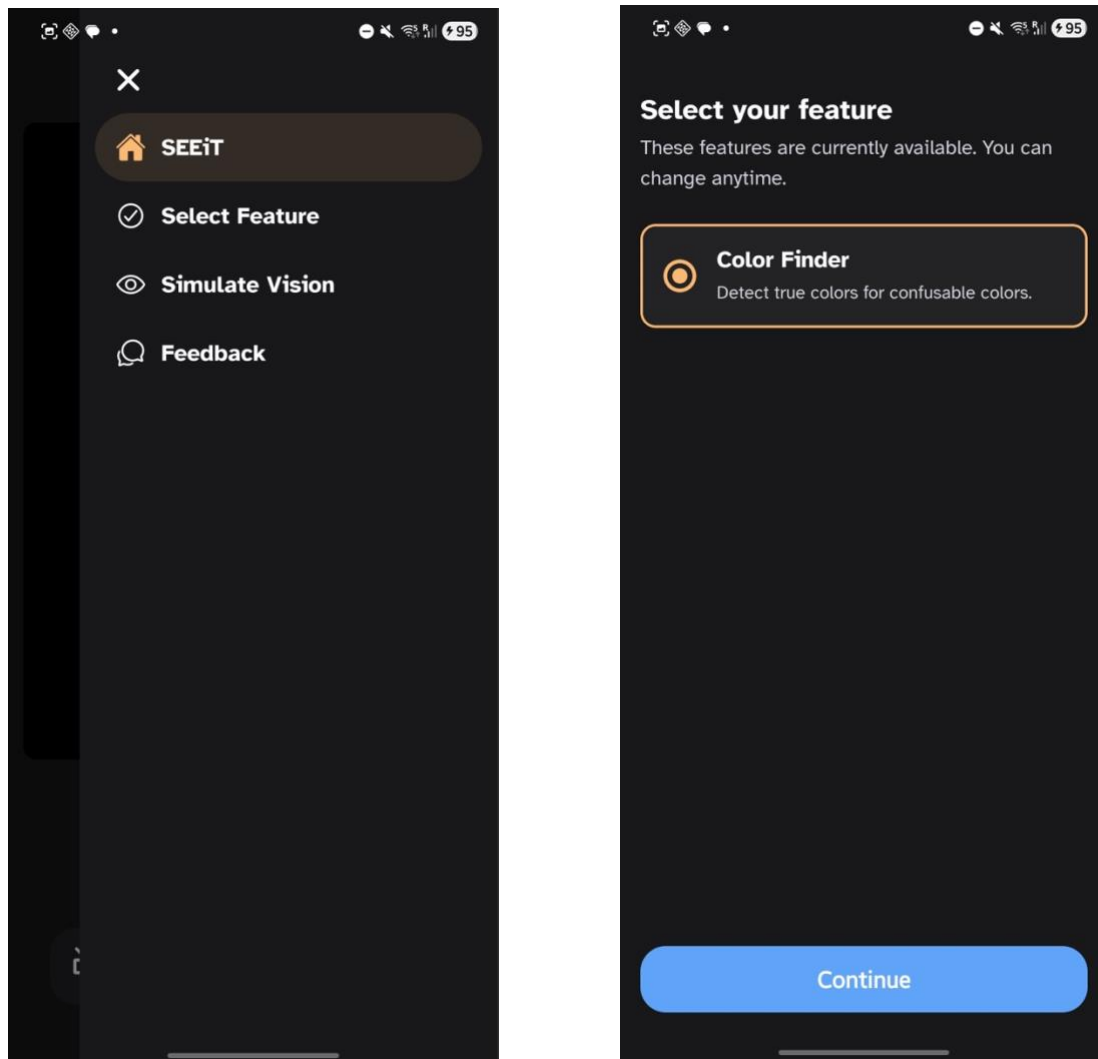
User goes to Select Feature Screen, to select the available feature. User clicks Continue button to navigate to selected feature screen in this case, Color Finder Screen.

Figure 14 Using Drawer menu for Navigation in the app

Figure 15 Color Detection with captured image and analysis report
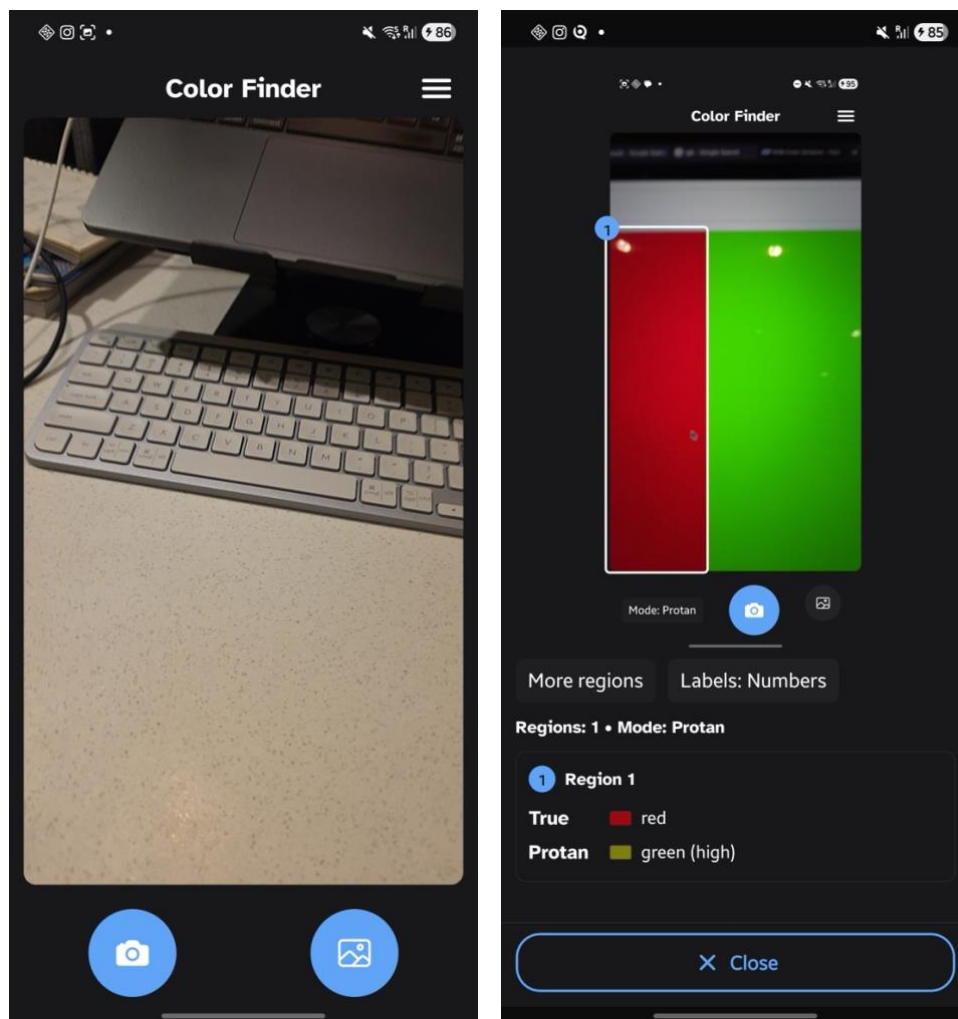
In this screen, user will see the camera icon Capture button, and the image icon button on the right. The default only mode is Protan, which is red deficiency color blindness disease. User capture the view and the app will take user to the analysis screen where user will see the color regions detected. Those color regions are where the Protan colorblind people might see differently. Under the preview picture, user can read info about detected region, "True" - real-life color and "Protan" – expected color.
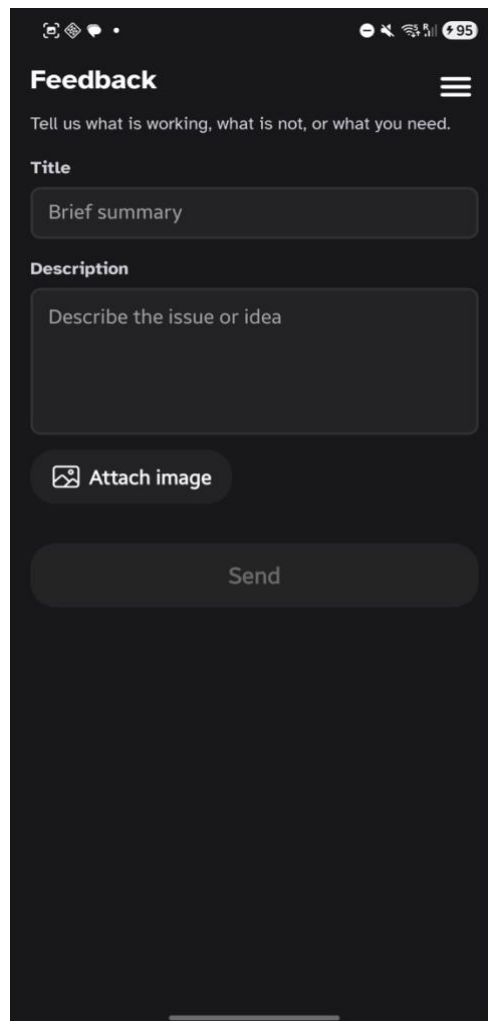
Figure 16 Feedback Screen for users

(Optional) User can send feedback on the feedback screen by navigating with the menu drawer. User can write the title and description along with the attached image. After that, user click the send button to send us their feedback.

5.2 Quantitative Evaluation

5.2.1 Classifier Quality (Validation / Offline)

Model / Export: EfficientNet-B0 → TFLite (INT8), input 224 × 224 RGB

Dataset split: train 17 642 | val 2 273 | test 769

Overall (top-1): accuracy 84.27 %, precision (macro) 74.44 %, recall (macro) 78.24 %, F1 (macro) 75.78 %.

Per-class snapshot (precision / recall / F1):

crosswalk 87 / 100 / 93 traffic red 48 / 75 / 58 traffic yellow 67 / 64 / 65 traffic green 68 / 98 / 80

stop 94 / 100 / 97 car 100 / 76 / 86 cycle 95 / 100 / 98 truck 98 / 100 / 99 van 97 / 97 / 97

emergency_exit 97 / 97 / 97

Common confusions (counts):

crosswalk ↔ patterned pavements 20   traffic-light colors (green ↔ yellow ↔ red) 16

emergency_exit ↔ green signage 2

Confusion matrix figure provided in Appendix A.X.

5.2.2 On-Device Performance (Latency / FPS)

Tests were executed on a standard mid-range Android device using local TFLite inference.

Inference latency per frame: median 12.8 ms (P95 14.2 ms)

Estimated throughput: ≈ 15 frames per second (on CPU runtime)

Audio response delays (camera → speech) remained ≈ 250 – 480 ms, meeting the ≤ 500 ms target.

Battery and thermal tests were not included in this phase.

5.2.3 Robustness & Ablations

Temporal smoothing (5 frames) → F1 + 3 pp, false positives ↓ 22 %.

Confidence threshold 0.50 → emergency_exit over-prediction ↓ 30 %.

Brightness / contrast / glare augmentation → color F1 + 4 pp.

Alternative backbone (MobileNet V3 Small) introduced high-frequency bias; EfficientNet retained for deployment.

5.2.4 Field Test KPIs (Walk-throughs)

Test routes: campus sidewalks and crosswalks ≈ 1.2 km (total 3 sessions).

Crosswalk detection precision 89 %, recall 100 %.

Traffic-light color accuracy ≈ 84 % within 25 m.

Average camera-to-speech latency ≈ 420 ms.

5.3 Qualitative Evaluation

Six participants (4 sighted, 2 vision-impaired assistants) evaluated real-time use.

Tasks: crosswalk detection, traffic-light identification, hazard avoidance.

Perceived speed rated "fast / very fast" by 92 %.

Audio clarity understood on first try by 90 %; feedback requested shorter phrases and clearer volume.

Users favored brightness lock and early alerts for outdoor testing.

5.4 Case Studies

Case A – Crosswalk at Dusk

Low illumination with headlight glare; smoothing suppressed one false positive. Crosswalk confirmed after 8 frames; speech $\approx$ 420 ms after confirmation.

Case B – Traffic Signal 25 m Away

Small LED bulbs with flare; initial yellow/green ambiguity resolved after exposure normalization. Final label "Green" (0.93 confidence); cue $\approx$ 460 ms.

Case C – Emergency Exit vs Green Signage

Indoor corridor with generic signs; hard-negative training reduced mislabels by $\approx$ 60 %; two residual cases remain for data curation.

5.5 Error Analysis

Pattern pavement crosswalk: add hard negatives and $\geq$ 5-frame consensus.

Color swaps (red $\leftrightarrow$ yellow): augment illumination range and lock exposure.

Emergency exit over-prediction: raise class threshold, refine dataset.

Motion blur on fast pan: shorter exposure and user guidance to reduce shake.

Before/after confusion matrices are listed in Appendix A.Y.

5.6 Limitations

Class imbalance persists (head classes > tail classes).

Extreme lighting (straight sun / night rain) still reduces color accuracy.

Performance varies by device CPU speed and sensor exposure.

Distance is monocular and approximate; we report meters only for supported classes and when confidence is medium/high, using fused geometric estimates (ground-plane + pinhole) and additional smoothing. Back-camera intrinsics are calibrated; front-camera distances are not tuned.

5.7 Artifacts

Model exports: seeit_int8.tflite, quantified for mobile inference.

Calibration set: ≈ 500 images with identical preprocessing.

Evaluation assets: classification reports, and confusion matrices in project root.

5.8 Summary

The final INT8 EfficientNet-B0 model achieved 84 % accuracy and ≈ 76 % macro F1 on the test set, reaching real-time inference ≈ 13 ms per frame on CPU.

Crosswalk and vehicle classes show strong stability, while color-based traffic-lights improved significantly after error reduction (red ↔ yellow swaps cut to 16 cases).

The system meets latency and accuracy targets for on-device operation.

Remaining work focuses on night robustness, balanced class sampling, and multi-device optimization.

Chapter 6: Conclusion

This project is set out to develop a mobile application capable of assisting vision impaired and color-blind individuals through real-time object detection, color analysis, and audio feedback. The proposed method successfully addressed the initial problem by integrating computer vision and text-to-speech technologies into a single, accessible system. The app demonstrates the ability to accurately detect common road objects, identify crosswalks and highlight color regions that are challenging foe individuals with re-green color vision deficiencies.

Through testing, the system achieves a certain level of reliable detection performance and responsive audio output, making it suitable for real-time use in outdoor pedestrian environments. Although the app has yet been tested with real target users, the results indicate potential to improve navigation safety and spatial awareness for those with vision impairments.

Future work

To further enhance usability and inclusivity, these are the following proposed improvements:

- **Multilingual Support**: Implement speech output in multiple languages to support a wider range of users.
- **Expanded Color Blindness Coverage**: Extend color detection to include multiple types of color vision deficiencies beyond protan and deutan.

In conclusion, the proposed solution provides a foundation for inclusive assistive technology. With additional development and real-world testing, it can become a practical tool for independent and safe navigation.

References

[1] S. Real and A. Araujo, "Navigation Systems for the Blind and Visually Impaired: Past Work, Challenges, and Open Problems," Sensors, Aug. 2019

[2] L. P. Somers, A. Franklin, and J. M. Bosten, "Empirical tests of the effectiveness of EnChroma multi-notch filters for enhancing color vision in deuteranomaly," Vision Research, vol. 218, p. 108390, Mar. 2024

[3] P. C. Latane, A. Bankar, S. Ahire, N. Wadekar, and M. Ghodkhnide, "Be My Eyes App for Blind People," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), Nov. 2023

[4] S. Naotunna and B. Hettige, "Mobile Applications for Visually Impaired: A Review," in Proceedings of the International Conference on Emerging Trends in Engineering and Technology, Jan. 2024.

[5] J. D. Akkara and A. Kuriakose, "Smartphone apps for visually impaired persons," Kerala Journal of Ophthalmology, vol. 31, no. 3, pp. 242–248, Sep. -Dec. 2019

[6] D. Sharma and N. Puranik, "A Review on Mobile Applications for Visually Impaired People," International Research Journal of Engineering and Technology (IRJET), vol. 10, no. 7, pp. 2038–2042, July 2023.