BlindNav: YOLO + LLM for Real-Time Navigation Assistance for Blind Users

Kaizhen Tan, Yufan Wang, Yixiao Li, Hanzhe Hong, Nicole Lyu

Introduction to Artificial Intelligence 95-891, Carnegie Mellon University

Abstract

This project investigates whether a camera-based AI assistant can meaningfully support blind and low-vision pedestrians during everyday walking in urban streets. We design a three-mode system, Walking Mode, Crossroad Mode, and Chatting Mode, that runs on egocentric video from the user's camera. A YOLOv8-based segmentation model is fine-tuned on a blind-path dataset to follow tactile paving and detect obstacles in Walking Mode, while a YOLOv8n detector trained on a merged 20-class street dataset drives Crossroad Mode by monitoring vehicles, pedestrians, traffic lights, and pedestrian signals. Both modes use a rule-based event layer to convert frame-level detections into simple navigation states, which are then turned into short voice messages. Chatting Mode passes a structured scene summary to a local Llama 3.2 language model running via the Ollama API to answer user questions via voice. Experiments on Bilibili videos and on-site footage in Pittsburgh show that the detector reaches an mAP50 of about 0.81, and the tactile-path segmenter reaches an mAP50 of 0.899 with high precision and recall, supporting a near–real-time prototype on CPU.

1. Introduction, Problem Definition, and Motivation

1.1 Problem Background

Vision impairment is a massive and worldwide concern. According to the research, it is estimated that there were 43 million blind people worldwide in 2020, and several hundred million people are moderately and severely vision-impaired. The research also forecasted that the total population of vision-impaired individuals may exceed 1.7 billion by 2050 due to population growth (Bourne et al.), making vision disability a significant health issue.

For people who are blind or have very low vision, independent mobility is one of the most important parts of their lives. Therefore, being able to go to work, school, grocery stores, and social events without relying on another person is essential. In the city, the streets are full of people, cars, and bikes that are all moving at various speeds and dynamically. Blind individuals are mainly dependent on sound, touch, and memory, which is not detailed and is not accurate, and this puts them in a severely dangerous condition while walking in such environments.

1.2 Limitations of Existing Solutions

Conventional assistive devices are helpful, but not definitive. The long cane and guide dog are able to identify most obstructions on the ground and near the person. Nevertheless, they fail to provide a complete representation of what is happening ahead of the person. They are not able to determine if a traffic light is red, if the tactile path continues forward, or if a car is blocking the crossing path. GPS-assisted navigation software is able to tell a person which road to take and which corner to turn into, but it is not "conscious" of traffic and sidewalk conditions. They also fail to identify minor details that are essential to determining safe passage, such as a bike on the path. Consequently, most critical decisions are left up to the person's assessment of what is happening, often leading to possible accidents and the avoidance of unknown paths.

This situation shows a clear gap. The current tool is either localized obstacle detection (cane, guide dog) or route information (GPS), but not a rich, detailed understanding of what is happening on the scene. The essential problem, the motivation behind the project, is therefore: can we apply the latest computer vision and large language model to understand what happens on urban streets when viewed from first person, and then turn it into very short and safety-oriented messages to enable vision-impaired pedestrians to navigate their streets on a daily basis?

1.3 Project Goals and System Overview

The goal of this project work is to develop a multi-mode AI assistant that supports visually impaired and blind users while they walk and interact in everyday environments. The AI assistant is dependent on the camera video input from the user, and the user has the option to select between three modes, which include walking mode, crossroad mode, and chatting mode. All the modes are dependent on the vision models connected to a language model, which produces short voice messages that guide the user.

In walking mode, the system focuses on blind path guidance. A semantic segmentation YOLOv8 model is fine-tuned on blind path datasets and used to trace the tactile path. With the segmentation, the system is able to determine if the user is on the path. The system will then issue simple commands to the user to walk straight, shift left, and shift right. The same semantic segmentation YOLOv8 model is also used to identify any obstacles along the path during the walk. As soon as the model detects that the obstacles are too close to the user or block the tactile path, the system produces events to the language model, which notifies the user on how to walk past the obstacles.

In crossroad mode, the system supports safe crossing of the road. The user is able to switch from walk mode to crossroad mode with a simple interaction, such as saying the command "I am ready to cross." After entering crossroad mode, a YOLOv8 detection model is applied to discover traffic lights, pedestrian lights, cars, and objects. The system checks whether the pedestrian signal shows it is safe to cross, and if there are vehicles and pedestrians who may still present a risk. These detections are then merged into crossing events such as "safe to cross" or "wait, car approaching," and then sent to the language model. The language model generates voice commands to assist the user.

In chatting mode, the system is more like a scene describer. The detection pipeline is the same, but it is more geared towards general understanding. For example, in a supermarket, the user can use chatting mode to detect items, and the user can ask what fruits are in front of them. The detections are converted into a structured description and sent to the language model, which answers in simple spoken language.

2. Preliminary Literature Review

2.1 Egocentric Navigation Assistance on Smartphones

Jadhav et al. introduce AI Guide Dog (AIGD), a lightweight egocentric navigation system for visually impaired people that runs on a smartphone camera (Jadhav et al.). The system takes video from a phone held by the user and uses a deep neural network to predict high-level walking commands such as "go straight," "turn left," or "turn right." Instead of building a full map, AIGD treats navigation as a multi-label classification problem: for each frame, the model predicts which actions are safe and appropriate. The authors also design goal-based outdoor navigation by combining these predictions with a routing engine, and they report real-time performance on consumer devices.

In the modeling section, Jadhav et al. discuss the results of the different neural network designs that they attempted. They start with a CNN for direction prediction given a single image. They extend the CNN to ConvLSTM and PredRNN networks for very short video sequences. Afterward, they introduce a CNN+LSTM hybrid approach for direction prediction. They also test the CNN+LSTM approach conditioning on high-level intent features from a routing engine. Although the accuracy of PredRNN was the highest, it is too heavy for real-time phone use, so the final system chooses CNN+LSTM over PredRNN since the accuracy-latency tradeoff was clearly better.

AIGD is important for our work because it proves that an egocentric, camera-only model can run in real time on a smartphone and provide useful guidance. However, it outputs only focus on direction labels and does not explicitly detect objects such as tactile paths, crosswalks, or detailed traffic-light states. Its interaction model is also limited to navigation commands. Our system builds on the same egocentric idea but adds mode-specific guidance: blind-path following in walking mode, safe crossing in crossroad mode, and scene description in chatting mode.

2.2 Tactile Paving Detection and Tracking

Takano et al. propose Tenji10K, a large dataset for tactile paving detection and tracking, and use it to evaluate computer-vision methods. Tenji10K includes 20 video sequences and 10,000 images captured in real outdoor environments in Japan. Each sequence contains tactile paving from the user's viewpoint and is labeled with mask and contour information. The dataset includes multiple scenarios such as different paving types, lighting changes, occlusions, and complex background conditions. This makes Tenji10K a strong benchmark for blind-path detection research.

The authors then analyzed three techniques for detecting tactile paving: HSV color thresholding, U-Net semantic segmentation, and multi-objective evolutionary algorithm (MOEA). HSV techniques for detection are fast, but their accuracy varies in different lighting conditions and colors. In contrast, the U-Net approach provides more precise segmentation values. MOEA further improves the accuracy of ground truthing values for a smooth detection.

This work is highly relevant to our walking mode, where we also focus on following tactile paths. Tenji10K shows that tactile paving can be reliably detected, but also reveals common failure cases such as shadows, glare, and partial occlusion. Our project follows a similar direction but uses a YOLOv8-based segmentation model instead of U-Net or color thresholding, aiming for real-time performance on general street scenes. We then connect the segmentation output to a rule-based event layer and a language model, so the system can not only detect the tactile path but also warn about obstacles on it and give explicit voice commands.

2.3 Summary and Gap

AI Guide Dog and Tenji10K represent two key lines of work. AI Guide Dog shows that egocentric navigation can be deployed on smartphones using a CNN+LSTM model to predict safe walking directions (Jadhav et al.). Tenji10K provides a large, realistic dataset for tactile paving detection and tracking and compares HSV thresholding, U-Net segmentation, and MOEA-based methods for path detection (Takano et al.). However, each research focuses on a single core function. Our project aims to combine and extend these ideas into a unified multi-mode system. We use YOLOv8-based detection and segmentation to support blind-path following, safe street crossing, and general scene description, and we connect the outputs to a local language model that produces short voice guidance. In this way, our system moves from single-task perception toward a multi-mode assistant that can help blind pedestrians in several common street and everyday scenarios.

3. Data Preparation

3.1 Source

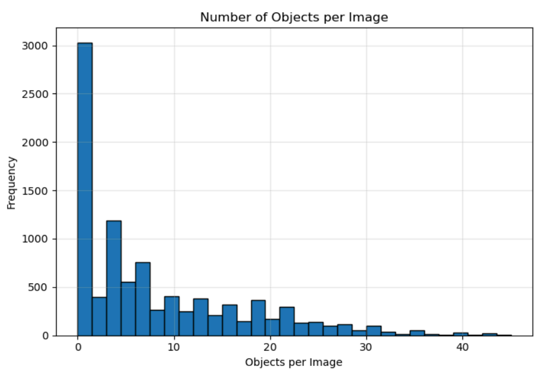
In the object detection part of the project, the goal is to support multimodal scene understanding and richer language feedback. This required a dataset with enough semantic variety while still fitting the needs of a detection model. To build such a dataset, we selected five public datasets from Roboflow, covering street objects, traffic-light states, pedestrian crossing signals, and tactile guide paths. These datasets were created independently, with different label names and class structures, so they could not be used directly as one unified dataset.

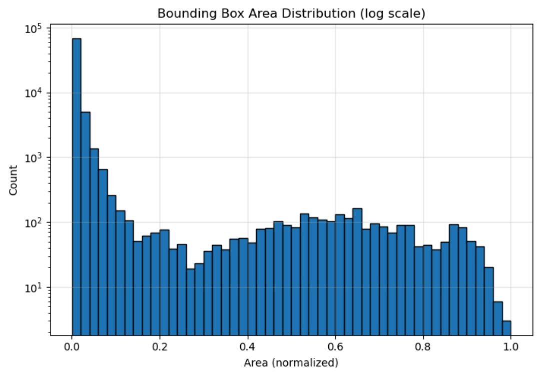
To address this, we wrote an automated script to clean the data, standardize class definitions, and merge all sources into a single YOLO-ready dataset. The final merged dataset contains 9,537 labeled images and 79,069 bounding boxes across 20 classes, including vehicles, pedestrians, street objects, tactile paths, traffic-light states, and pedestrian signals. This unified dataset provides the foundation for training a model that can later generate more detailed and diverse language feedback.

The semantic segmentation model was trained on a specialized dataset sourced from the Roboflow Universe platform. The dataset, titled "Blind road" (version 5), was provided by a Roboflow user under a CC BY 4.0 license. It consists of 120 images annotated for blind‑road (tactile paving) segmentation in YOLOv8 format. The images were split into training (100 images), validation (12 images), and test (8 images) sets, following the common practice in computer vision to ensure separate data for model development and evaluation. The annotations provide pixel‑level masks for the single class "blind‑road," enabling the model to learn precise boundaries of the tactile path.

3.2 EDA & Visualization

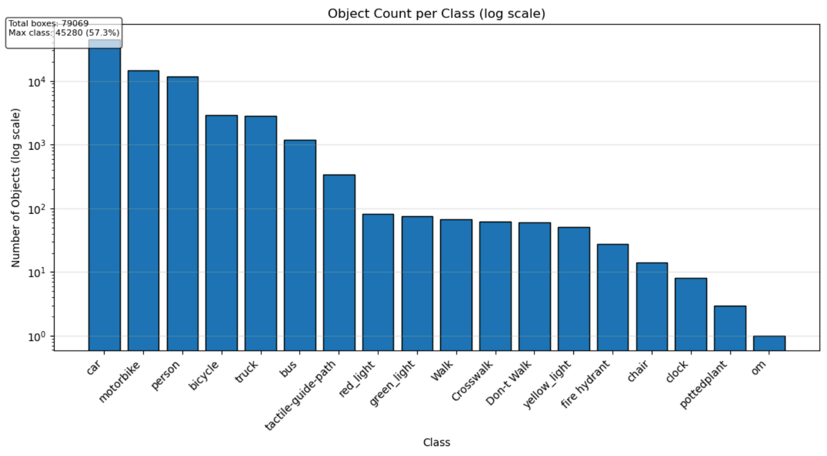
For the object detection dataset, we conducted descriptive analyses to better understand its structure and potential sources of bias.





The histogram of objects per image shows that most images contain only a few objects (1–5), but some images include more than 20. This indicates that the dataset covers both simple scenes and dense urban environments, which helps the model learn across different levels of complexity.

The bounding box area distribution, plotted on a log scale, shows that small objects make up a large portion of the dataset, especially items like traffic lights and pedestrian signals. This places higher demands on the model's ability to detect small targets. Medium and large boxes appear in the dataset as well, but they are far less common.



The class distribution displays a clear long-tail pattern. Common categories such as car, motorbike, and person have far more samples. Accessibility-related classes like tactile-guide-path and pedestrian signals (Walk, Don't Walk, Crosswalk) are much less frequent. Rare street objects such as chair, clock, potted plant, and other markers (om) appear even less often.

This imbalance means the model will naturally lean toward high-frequency classes during training, while performing worse on rare but important categories such as U.S. pedestrian signals. These issues need to be considered explicitly in the training strategy.

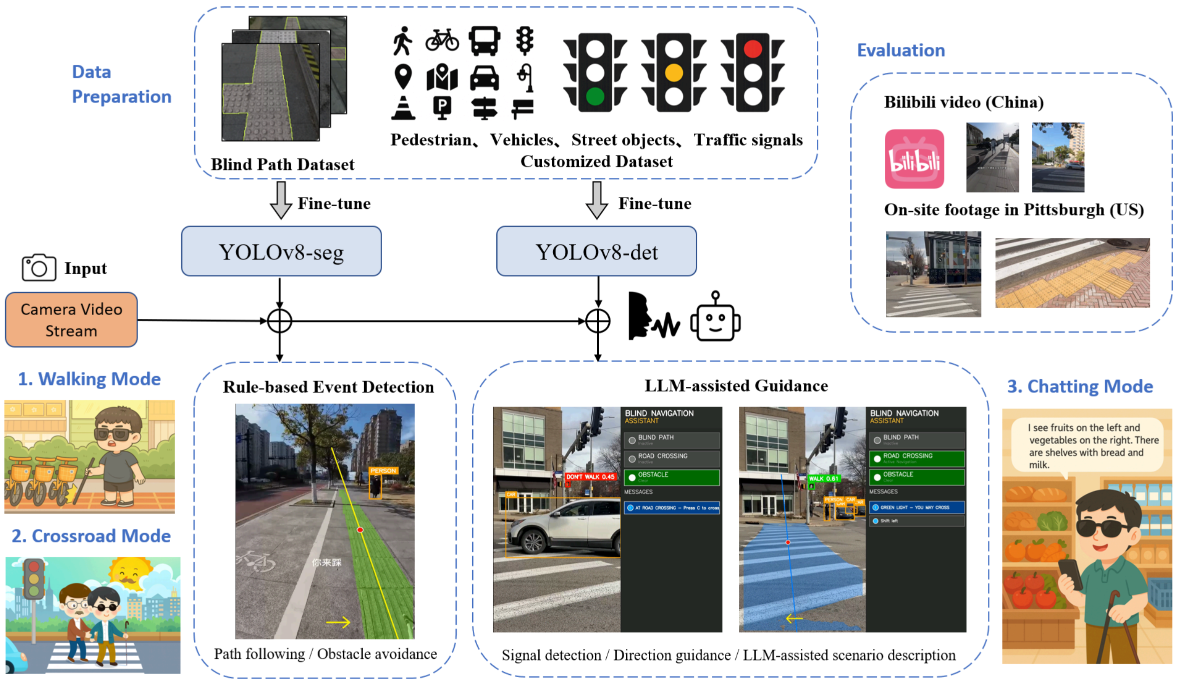
For the semantic model, exploratory data analysis was conducted to understand the characteristics of the segmentation dataset. The dataset contains 120 images with 126 annotated instances of blind roads, resulting in an average instance density of 1.05 per image. Most images (92.5%) contain exactly one instance, indicating that the dataset focuses on segmenting primary, continuous tactile paths rather than multiple scattered elements.

Image dimensions vary considerably: widths range from 512 to 4,096 pixels, and heights from 512 to 4,624 pixels. The mean resolution is 3,113 × 3,869 pixels, with a mean aspect ratio of 0.84, reflecting a prevalence of portrait-oriented images consistent with a first-person viewpoint. Normalized bounding-box analysis shows that blind roads occupy a substantial portion of the frame, with a mean normalized width of 0.457, a mean normalized height of 0.527, and a mean normalized area of 0.257. This scale suggests that the target objects are generally central and large enough to be reliably detected by standard segmentation architectures.

Visualizations of the dataset—including histograms of image dimensions, aspect ratios, and instance-per-image distribution—were generated using Python scripts with libraries such as Matplotlib and OpenCV. These plots confirm the dataset's bias toward single-instance, portrait-format images and highlight the need for preprocessing steps such as resizing with letterbox padding to maintain aspect ratios during training. The analysis also revealed no class-imbalance issues, given the single-class nature of the task, but underscored the importance of data augmentation to improve model robustness given the limited sample size (120 images).

4. Methodology

4.1 Overview


Data Preparation
Blind Path Dataset
Pedestrian、 Vehicles、 Street objects、 Traffic signals
Customized Dataset
Evaluation
Bilibili video (China)
On-site footage in Pittsburgh (US)
Fine-tune
Fine-tune
YOLOv8-seg
YOLOv8-det
Input
Camera Video Stream
1. Walking Mode
2. Crossroad Mode
3. Chatting Mode
Rule-based Event Detection
Path following / Obstacle avoidance
LLM-assisted Guidance
Signal detection / Direction guidance / LLM-assisted scenario description
BLIND NAVIGATION ASSISTANT
BLIND PATH
ROAD CROSSING
OBSTACLE
MESSAGES
DON'T WALK 0.45
WALK 0.61
I see fruits on the left and vegetables on the right. There are shelves with bread and milk.

The figure illustrates the overall architecture of our blind navigation assistant. The system is designed around three recurrent pain points reported by visually impaired pedestrians: staying aligned with tactile paving during routine walking, crossing signalized intersections safely, and understanding unfamiliar surroundings in daily life. To address these needs, we provide three corresponding modes: Walking Mode, Crossroad Mode, and Chatting Mode. Each mode shares the same perception backbone but applies different event rules and language templates, so the system can transition smoothly between path following, road crossing, and general scene description.

At the perception layer, the camera stream from the user's phone or wearable device is fed into two fine-tuned YOLOv8 models in parallel. The first is a YOLOv8-seg model trained on tactile paving data. Its output is a pixel-wise mask that delineates the blind path and highlights any objects that lie directly on or very near this path. The second is a YOLOv8-det model trained on our merged street-scene dataset to detect vehicles, pedestrians, traffic lights, pedestrian signals, and other relevant objects. Running both networks concurrently allows the system to maintain an accurate estimate of the user's position relative to the tactile path while also monitoring dynamic agents and infrastructure in the broader scene.

Downstream of these models, a rule-based event detection layer converts raw masks and bounding boxes into discrete navigation events. For Walking Mode, rules focus on whether the user is centered on the blind path, whether the path is temporarily lost, and whether obstacles block the way. For Crossroad Mode, rules monitor combinations of traffic light states, pedestrian signals, and nearby vehicles to infer higher level events such as "safe to cross," or "wait, vehicle approaching from left." In Chatting Mode, the same detections are grouped into a more descriptive scene graph that captures object types and coarse locations without enforcing strict safety logic.

Finally, a local vision-language pipeline transforms these events into spoken feedback. For highly structured events, such as path deviation or a red pedestrian signal, we use short templates that generate commands like "Shift right to stay on the path" or "Please wait; red light ahead." For richer descriptions in Chatting Mode, the structured scene summary is passed to a local LLM that produces short, safety-aware sentences about nearby objects. Because the perception and event logic are shared across modes, we can evaluate the full system on diverse video sources, including Bilibili recordings from Chinese cities and our own footage from Pittsburgh, and observe how the same architecture generalizes across different urban environments.

At the system level we intentionally organize perception and feedback into two complementary workflows. For safety-critical behaviors such as staying on the tactile paving, detecting path-blocking obstacles, and deciding when to start crossing, YOLO outputs feed into a compact rule-based state machine that converts detections into discrete navigation events and short template sentences. For more open-ended questions in Chatting Mode, the same YOLO outputs are first compressed into a structured scene summary, which is then passed to a local language model. This split design keeps the rule-based workflow predictable and easy to debug, while the LLM workflow can generate richer descriptions without influencing the timing of core safety decisions.

4.2 YOLO Model Fine-tuning

Before describing the training details, we briefly motivate our choice of a YOLO-based backbone. Our system requires a single architecture that supports both bounding-box detection and pixel-wise segmentation, runs close to real time on CPU-class edge devices, and has stable open-source tooling for training and deployment. YOLOv8 provides off-the-shelf variants for detection and segmentation that share a common codebase, which allows us to fine-tune a compact YOLOv8n detector and a YOLOv8-seg model using similar scripts and augmentation pipelines. Compared with heavier two-stage detectors or recent transformer-based models, a small YOLOv8 variant offers a better tradeoff between latency and accuracy under our resource constraints and simplifies integrating perception into the rest of the Python prototype.

4.2.1 Class Imbalance Mitigation

After consolidating five public datasets into a unified YOLO training set, we found that the distribution of classes was heavily imbalanced. Vehicle-related classes contained thousands of samples, while important accessibility-related categories such as Walk, Don't Walk, Crosswalk, and tactile-guide-path appeared only occasionally. Training directly on this distribution would cause the model to focus on high-frequency classes and achieve very low recall on minority classes that are essential for safe navigation. To address this issue, we applied two practical data-level strategies: class-reweighted sampling and hard-example mining.

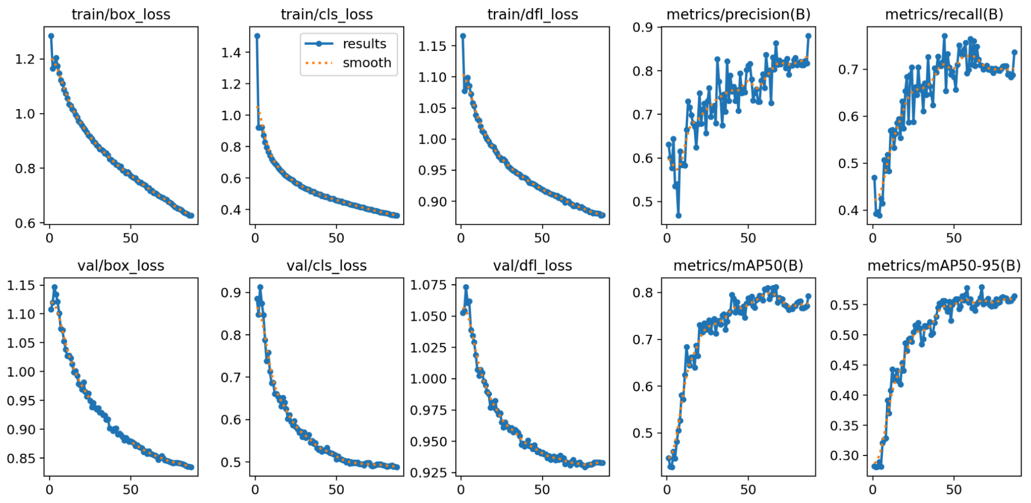
Class-reweighted sampling was implemented through offline oversampling. Because the YOLOv8 training pipeline does not provide hooks for custom sampling weights, we increased the representation of minority classes by duplicating images that contained these categories and inserting them multiple times into the training split. This ensures that low-frequency classes appear more often during each epoch and that the model receives enough gradient signal to learn them effectively. This approach works smoothly with YOLOs built-in augmentations such as Mosaic, color jitter, and scale jitter.

In addition to increasing exposure to minority classes, we also needed to help the model learn difficult samples that it often failed to recognize. To accomplish this, we used a two-stage hard-example mining process. After running an initial training stage, we examined validation outputs and identified images where the model consistently made mistakes. These cases often involved small or visually subtle elements, such as distant pedestrian signals or partially occluded tactile paths. We duplicated these hard samples and added them back into the training set, then performed a short fine-tuning stage starting from the best checkpoint of the first run.

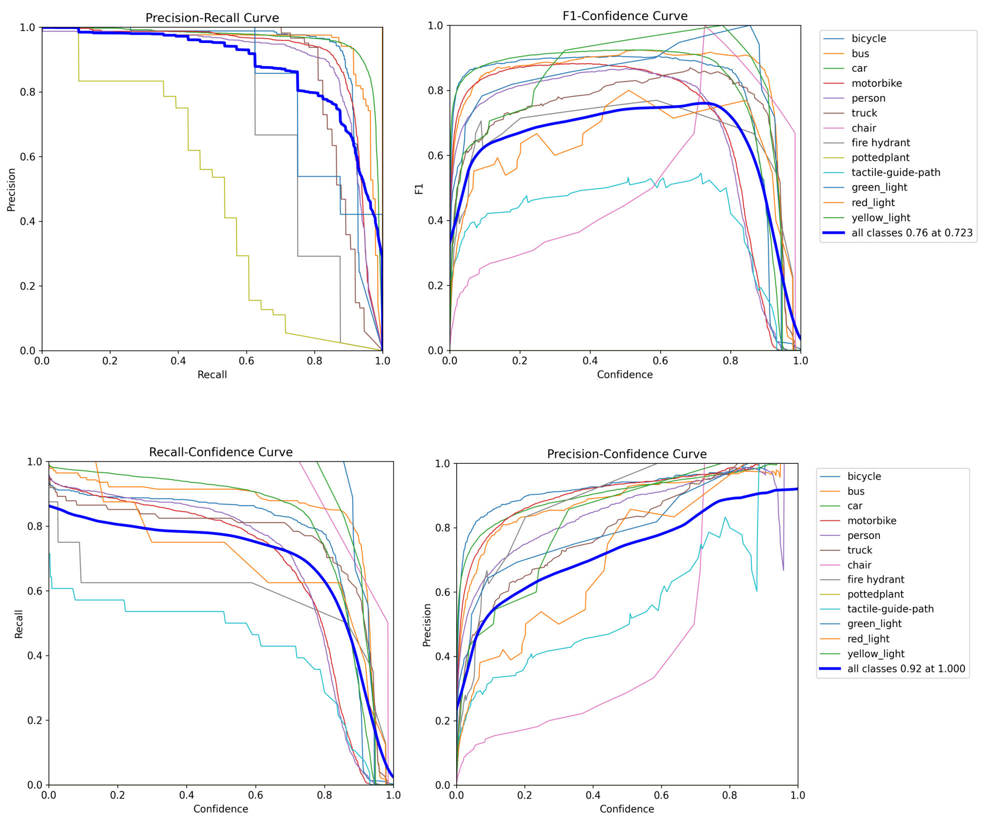
Together, these two strategies provide complementary benefits. Class-reweighted sampling increases the presence of minority classes at the dataset level, and hard-example mining strengthens the models ability to recognize challenging samples. Combined with YOLOv8s augmentation pipeline, these adjustments significantly improved recall for low-frequency but safety-critical classes, creating a more dependable input foundation for subsequent spatial encoding and language-based navigation guidance.

4.2.2 Object Detection Model Training and Performance

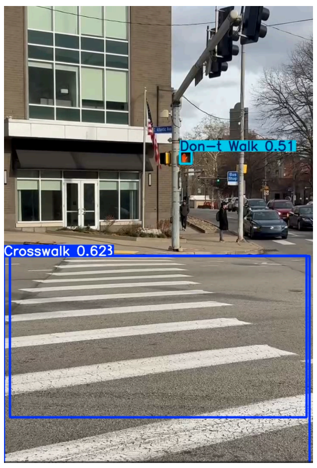
In the detection module, we trained a YOLOv8n model, chosen for its lightweight architecture and ability to support real-time inference. The model was trained for 150 epochs at a resolution of 640 with a batch size of 4, and we fixed the random seed to ensure reproducibility. To stabilize training under a small batch regime, we set the initial learning rate to 0.0008 and applied a learning-rate decay factor of 0.01 to allow smoother convergence in later stages. We added three warmup epochs to soften early gradient fluctuations. To reduce overfitting risks in a long-tailed dataset, we applied L2 regularization with weight_decay=0.0005, and we enabled early stopping with a patience of 20. Mosaic and other strong augmentations were disabled in the final ten epochs to allow fine-grain tuning on more realistic images. Checkpoints were saved every ten epochs for later inspection.

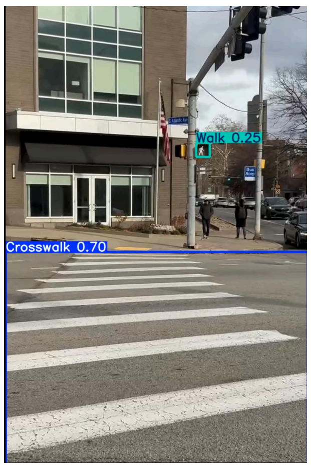


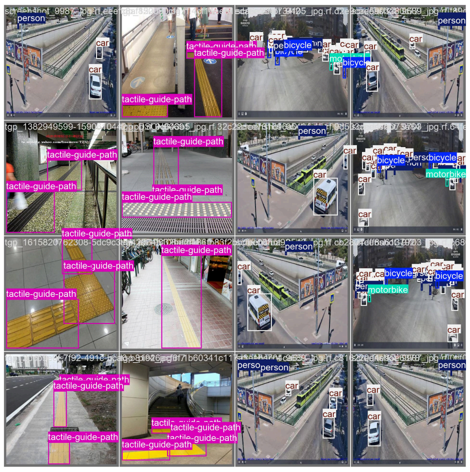
The training and validation loss curves show a consistent downward trend across box_loss, cls_loss, and dfl_loss, with only small gaps between the two sets of curves. This suggests the model did not overfit. Both mAP50 and mAP50–95 increased rapidly during the first 30 epochs before reaching a plateau, ending at approximately mAP50≈0.81 and mAP50–95≈0.56—reasonable for a lightweight model operating on a diverse street-scene dataset. Precision and recall also improved steadily, with recall showing a more notable rise, reflecting the earlier class-reweighted sampling and hard-example mining strategies that increased exposure to minority classes.



The F1-confidence curve shows that common classes reach strong overall performance, while rare categories such as tactile-guide-path and pedestrian-signal classes remain more challenging. Even so, their F1 scores are clearly improved compared to the initial unbalanced experiments, indicating that the balancing strategies helped the model learn more stable representations for minority classes. The Precision–Recall curve further illustrates this pattern. High-frequency categories including car, bicycle, bus, motorbike and person achieve high AP values that align with real navigation needs, with green_light and red_light performing especially well for safety-critical decisions. Tactile-guide-path reaches a moderate AP that reflects both its visual complexity and limited sample size. The Recall–Confidence curve shows that the model can achieve high recall at low confidence thresholds, which is important for minimizing missed detections in a safety-focused system. Recall declines as the threshold increases, and rare classes drop more sharply, consistent with their inherent difficulty. The Precision–Confidence curve complements this by showing that overall precision steadily increases with confidence and reaches a high level at the upper end of the threshold range. Vehicle, signal and pedestrian classes respond most strongly to higher confidence levels, while minority classes improve more slowly but still maintain identifiable patterns when the model is confident.







4.2.3 Semantic Segmentation Model Training and Performance

The core component for the walking mode is a semantic segmentation model tasked with precisely identifying tactile guide paths. We began with a pre-trained YOLOv8-seg model. Its initial performance on our dedicated test set showed a solid baseline, with an mAP50 of 0.887 and a precision of 0.971. However, the recall of 0.859 indicated that the model was missing a portion of the actual blind path area, a significant concern for user safety.

To improve recall without sacrificing precision, we conducted a hyperparameter optimization grid search. The optimal configuration identified used a Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01 and a reduced data augmentation strength of 0.1. Fine-tuning with these parameters yielded a model with an mAP50 of 0.899. More importantly, recall improved to 0.912 while precision remained exceptionally high at 0.993. This balanced improvement ensures the model is both highly accurate and more reliable in detecting the complete blind path, directly addressing the safety requirement of minimizing missed detections.

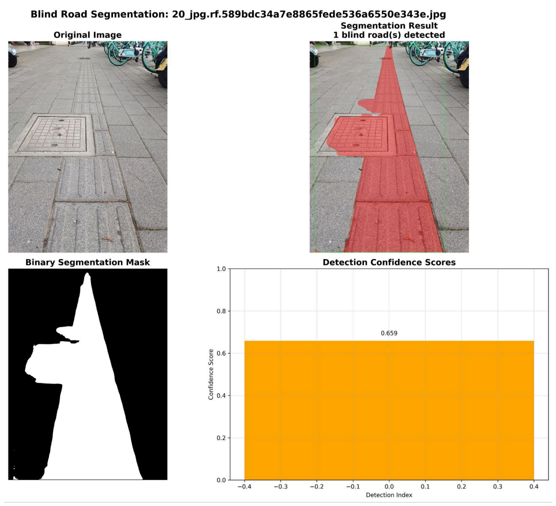


Fig: Before Fine-tuning

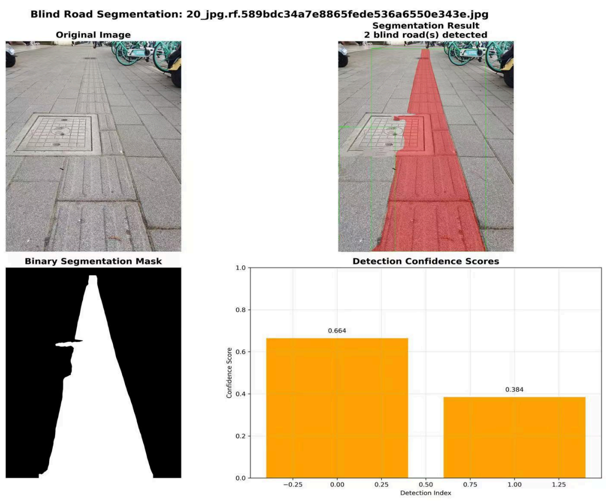


Fig: After Fine tuning

4.4 Feature 1: Always-on Blind Path Guidance + Obstacle Alert

We implement an always-on walking assistant that keeps the user centered on the tactile paving while monitoring obstacles that block or approach the path. The whole loop is designed to be frame-wise, stable over time, and interruption-safe, so that guidance does not flicker when detections are noisy.

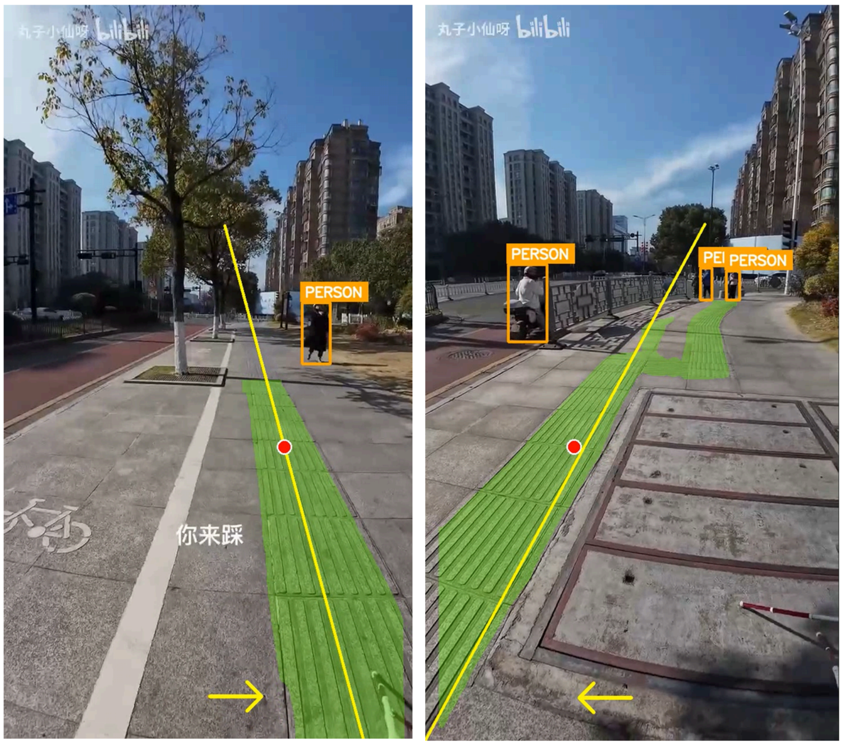
4.4.1 Blind path activation and stability

For each frame, the segmentation model produces a blind-path mask. We treat the path as “detected” if the mask covers more than a small ratio of the image (default 0.5% of pixels). When this condition holds for 2 seconds continuously, the system marks the blind path as stably found and switches the workflow state to BLIND_PATH_ACTIVE. If the mask disappears for 2 seconds, the state flips to “stably lost.” In that case, we exit Walking Mode, raise a high priority warning (“LOST BLIND PATH – PLEASE STOP”), and stop issuing directional commands until a new stable path is found. This temporal hysteresis makes the mode robust to short occlusions and noisy frames.

4.4.2 Centerline estimation and direction decision

Given a blind-path mask, we compute a parametric centerline instead of directly reacting to the raw pixels. The mask is scanned row by row from bottom to mid-height. In each row with enough path pixels, we compute the row-wise center of the path, and we split these centers into a “bottom band” and a “mid band”. A straight line is fitted through the average point in each band, which yields a slope and intercept for the path centerline. From this line we derive: (1) an orientation angle relative to the camera vertical and (2) the lateral offset between the target point on the path and the image center.

Navigation commands follow simple but stable rules. We first check lateral offset: if the offset exceeds a fixed ratio (default 0.10 of image width), the system issues “Shift left” or “Shift right” depending on the sign of the offset. When the user is roughly centered, we instead check orientation: if the path tilts more than 10 degrees to the left or right, we issue “Turn left” or “Turn right”. Only when both offset and angle are within thresholds do we say “Continue straight”. The navigation direction must be consistent for several consecutive frames, and at least nav_guidance_interval seconds must pass between two voice prompts, which avoids rapid oscillation of instructions.



4.4.3 Obstacle detection on and near the path

In parallel, the detection model searches for a set of obstacle classes such as pedestrians, bicycles, cars, buses and street furniture. For each detected box, we test two spatial relations with the blind-path mask:

1. On-path obstacles. We intersect the lower part of the box with the blind-path mask and compute the overlap ratio. If more than 5% of the box area overlaps with the path, the object is tagged as "on path".

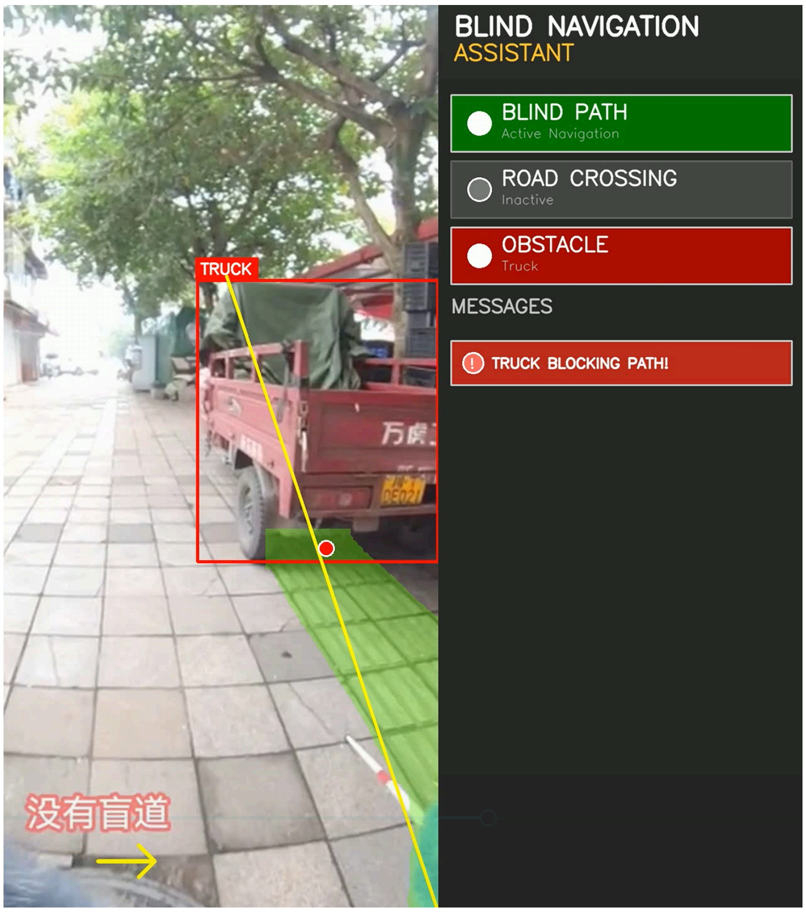
2. In-front obstacles. We also check whether the box center falls inside a central "front zone" in the lower half of the frame. These objects are not directly on the tactile paving but lie in front of the user.

For Walking Mode, we prioritize the nearest obstacle that either lies on the path or, if none is on-path, the nearest object in front. We ignore tiny boxes by enforcing a minimum area ratio. A candidate obstacle must persist over time before it becomes a true blocking obstacle: on-path objects require only 0.8 seconds, while in-front objects require 2.5 seconds. Once confirmed, the obstacle state changes to DETECTED, a sticky warning such as "CAR BLOCKING PATH!" is shown and spoken, and navigation guidance is temporarily suppressed so that the user hears only safety-critical messages. If the obstacle disappears and has not been seen for 3 seconds, we reset the state to CLEAR, remove the warning, and briefly confirm that "Path is clear".

4.4.4 Message and voice integration

All guidance and warnings go through a message manager with a voice queue. Quick guidance messages ("Shift right", "Continue straight") are short lived and rate limited. Persistent warnings ("LOST BLIND PATH", "CAR BLOCKING PATH") remain on the interface until explicitly cleared by the state machine, and they are spoken with higher priority. Duplicate messages of the same category are suppressed if they occur too frequently, which reduces cognitive load for blind users.

Overall, this design makes blind-path navigation truly "always on": as soon as a stable tactile path is visible, the system automatically enters Walking Mode, gives smooth directional cues, and escalates to strong alerts when the path is lost or blocked by obstacles.



TRUCK BLOCKING PATH!

4.5 Feature 2: Verbal Interaction at Crossroads

4.5.1 Real-Time Inference and Spatial Encoding

To bridge the gap between computer vision detections and human-understandable navigation, we developed a custom inference wrapper, implemented in blind_navigation_assistant.py and inference_yolo.py. While the upstream YOLOv8 model successfully outputs class labels and bounding boxes, we evaluated that raw coordinates (e.g., x=200, y=400) impose an unmanageable cognitive load on a blind user who cannot verify the scene visually. Providing raw pixel data does not offer immediate situational awareness.

To address this, our system acts as a translation layer. It calculates the geometric centroid of each detected object's bounding box and maps it to a coarse 3 x 3 spatial grid, dividing the camera frame into nine distinct zones (Left/Center/Right x Top/Middle/Bottom). This "spatial encoding" step transforms precise but abstract pixel data into relative human directions—for example, translating a bounding box in the upper-left quadrant into the instruction "Red light in the top-left". We purposely chose this coarse grid approach over precise metric distance estimation to make the guidance device-agnostic and understandable, ensuring the system prioritizes immediate relative awareness over unnecessary precision. The output of this stage is a structured text summary (JSON) representing the scene's key safety elements, which serves as the optimized input for the language model.

4.5.2 Local LLM Integration and Prompt Engineering

For the verbal interaction layer, we integrated a Large Language Model (LLM) to generate natural language guidance. We specifically chose to deploy the Phi-3 Mini model running locally via Ollama, rather than relying on cloud-based APIs. This architectural decision was driven by an evaluation of three critical factors:

1. Latency: Cloud API calls introduce variable network lag. By running the model locally on the CPU, we eliminate network dependency, ensuring the low latency required for real-time street crossing.
2. Privacy and Offline Reliability: A blind navigation assistant must function reliably in areas with poor cellular service and should not stream continuous video of the user's surroundings to a third-party server.
3. Compute Efficiency: Phi-3 Mini is lightweight enough to perform inference on a standard CPU, demonstrating the feasibility of deploying this system on wearable or mobile hardware without heavy GPU requirements.

Furthermore, we evaluated that standard "image captioning" prompts often produced verbose, conversational descriptions (e.g., "I see a nice tree and a red light") that delayed critical warnings. To improve the user experience (UX), we implemented strict prompt engineering to act as a "Safety Filter". We tuned the system prompt to enforce a "Single Imperative Sentence" constraint, instructing the model to completely ignore background objects and strictly prioritize safety cues such as traffic signal states and approaching vehicles. For instance, if the spatial encoder detects a red light and a potted plant, the prompt ensures the model outputs only "Stop; red light ahead," filtering out the irrelevant plant to maximize user safety.

Because LLM inference on a CPU-only device is still slower and less predictable than running YOLO and simple rules, we restrict the use of the language model to workflows where short delays are acceptable. In our design, all time-critical checks – for example losing the tactile path, detecting a vehicle blocking the crosswalk, or noticing that the pedestrian signal has turned red again – are handled entirely by the rule-based pipeline and short template messages. The LLM receives aggregated scene summaries at a lower rate and is primarily used when the user explicitly asks for more context, such as "What is around me?" in a supermarket or "What is happening in front of me?" before deciding to cross. Street crossing itself also separates into a decision phase and an execution phase. During the decision phase the user can safely wait a moment to hear a concise description. Once the user confirms and the system switches into Crossroad Mode, continuous route guidance and obstacle alerts come directly from the rule-based module, so occasional LLM latency does not compromise safety.

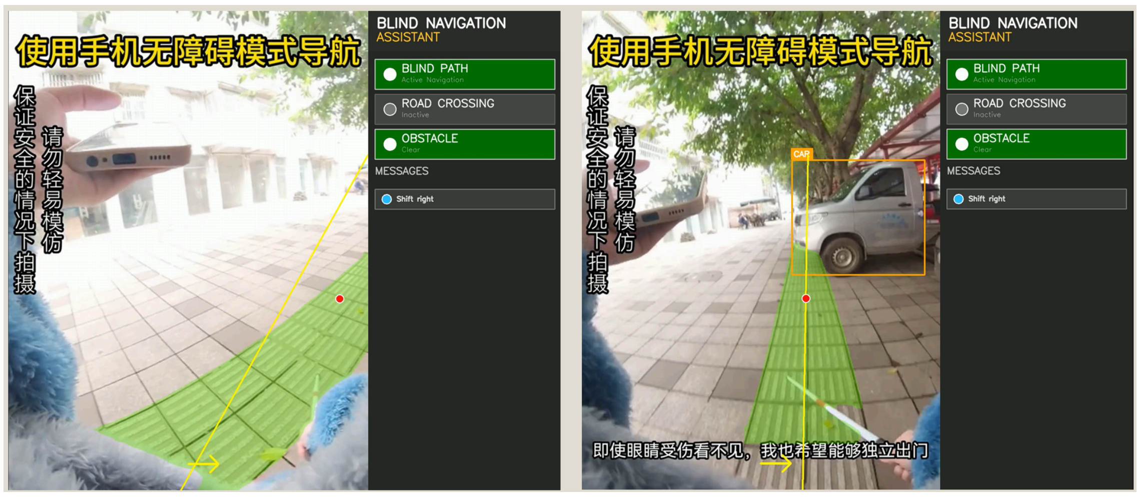
5. Case Study

To understand how the integrated system behaves in realistic settings, we conducted three pilot case studies using recorded egocentric video. Each case highlights one of the three modes and illustrates how low-level detections, rule-based events, and language generation interact in real time.

5.1 Blocked Blind Path by Random Parking

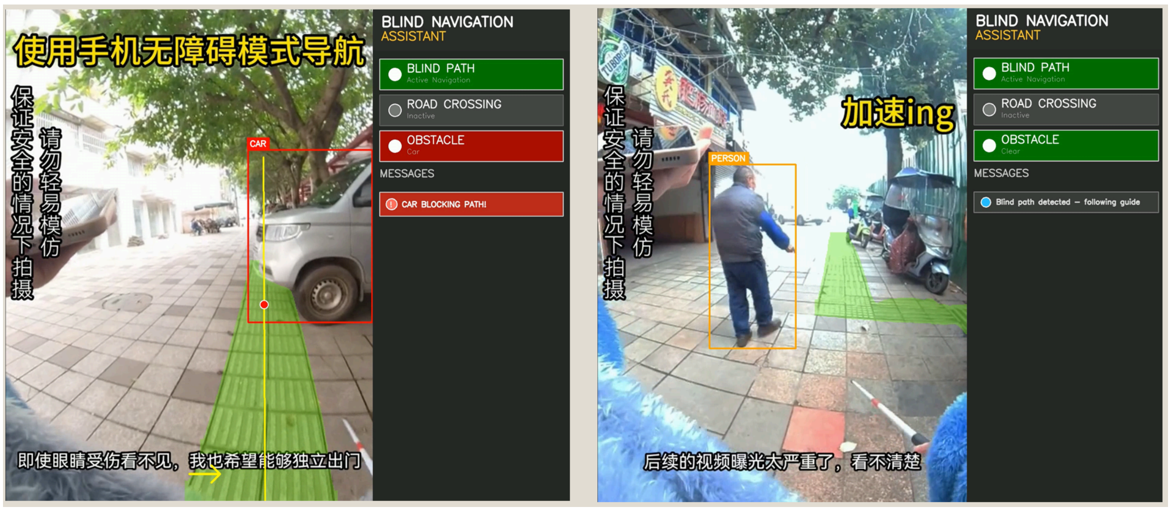
The first case focuses on Walking Mode in a residential street in China where a car is parked directly on the tactile paving. At the beginning of the clip, the YOLOv8-seg model locks onto the tactile path, and the system automatically activates blind path guidance. As the user walks forward, the event layer continuously checks whether the center of the camera frame overlaps with the segmented blind path. Since the alignment remains acceptable, the voice guidance periodically confirms “On blind path, walk straight,” rather than issuing unnecessary corrections.

As the user gradually drifts away from the center line, the segmentation mask shifts within the frame. The event rules interpret this as a lateral deviation and trigger corrective commands such as “Shift right.” These prompts are issued only when deviation persists for several frames, which avoids overreacting to small head movements while still steering the user back toward the path.



Shift right

When the user approaches the parked car that blocks the tactile paving, the segmentation mask and detection outputs intersect. The car’s bounding box overlaps with the blind path region, and the event layer marks this as an “obstacle blocking path” event. The interface highlights the obstruction and the language module switches from pure alignment guidance to a warning such as “Obstacle ahead on the blind path, slow down.” After the user passes the car and the overlap disappears, the system automatically reacquires the path and resumes standard “walk straight” prompts. This case shows that Walking Mode can maintain path following while also reacting to transient obstacles that break the tactile paving.



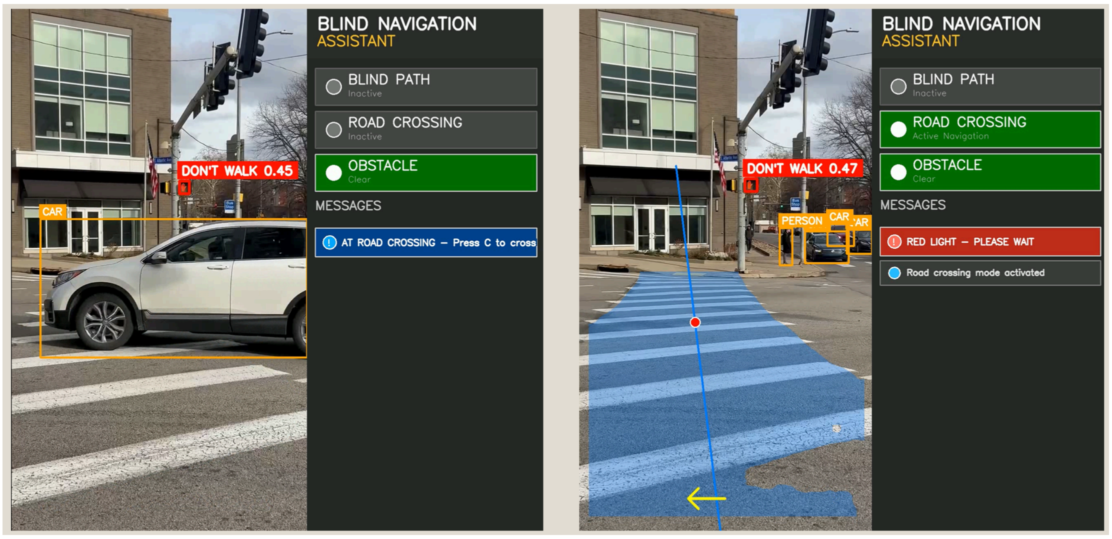
CAR BLOCKING PATH!

Blind path detected – following guide

5.2 Crossing a Street in Pittsburgh

The second case study illustrates Crossroad Mode at a signalized intersection in Pittsburgh. At the start of the clip, the user approaches the curb while still in Walking Mode. The detection model identifies a pedestrian signal and traffic lights ahead. When the event layer confirms the presence of a crossing signal within a predefined region of interest, the interface notifies the user that a crossing opportunity is available and offers to switch to Crossroad Mode. After the user confirms, the system focuses on traffic signals, pedestrian lights, and vehicles near the crosswalk.

While the pedestrian signal is red, the system aggregates detections into a “do not cross” state. The display shows a text prompt such as “PLEASE WAIT,” and the speech output reinforces this instruction. During this waiting period, detections of cars moving through the intersection are logged but not converted into detailed voice descriptions, in order to keep the message short and clear.

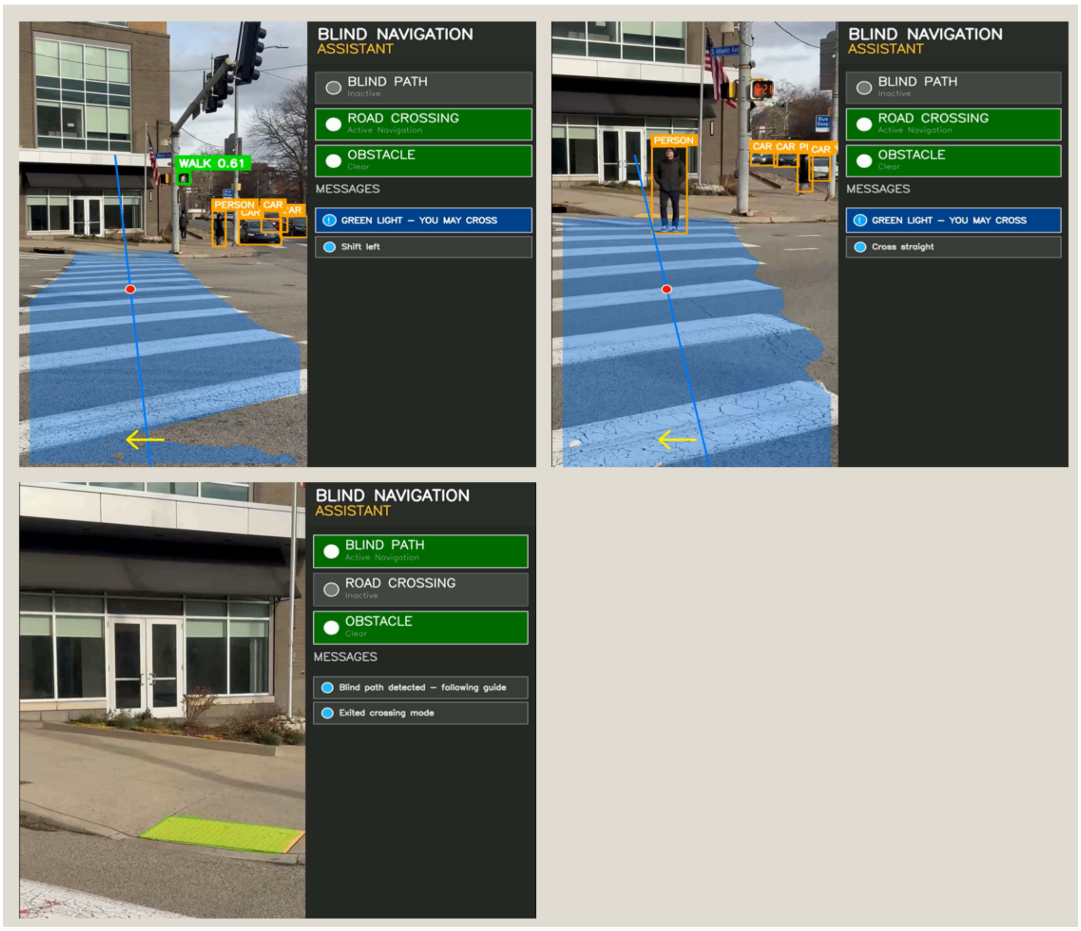


AT ROAD CROSSING – Press C to cross

RED LIGHT – PLEASE WAIT

Road crossing mode activated

When the pedestrian signal turns green, the state changes to “safe to initiate crossing,” subject to simple safety checks on nearby vehicles. The system announces “You may cross” and begins issuing short route-keeping instructions, for example “Cross straight” or “Shift left slightly.” These cues are updated about once per second based on the relative position of the crosswalk markings and curb ramps in the camera view. This update rate was chosen to match a natural walking cadence and to avoid flooding the user with messages. After the user reaches the far curb and the system no longer detects crosswalk stripes in front of the camera, Crossroad Mode automatically ends, and Walking Mode resumes. The model re-locks onto the tactile paving on the far side, demonstrating seamless mode switching within one continuous journey.



GREEN LIGHT – YOU MAY CROSS

Shift left

GREEN LIGHT – YOU MAY CROSS

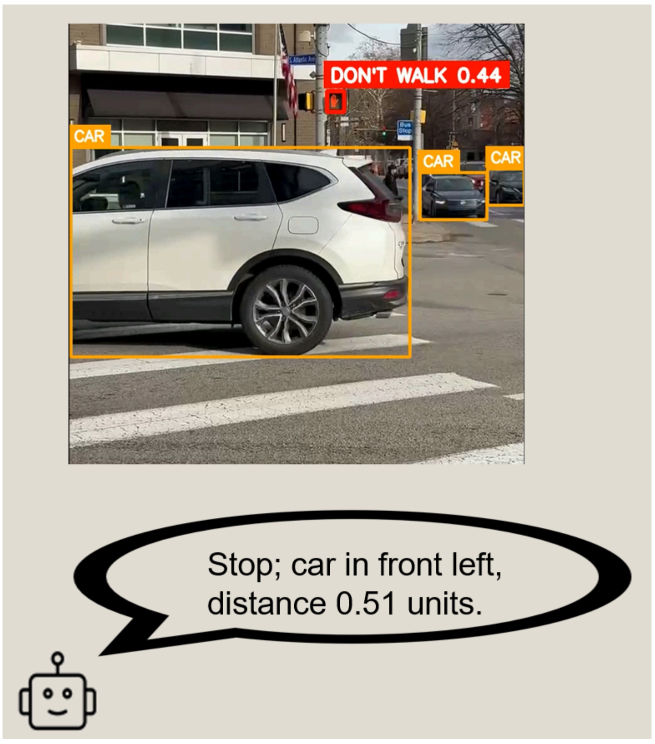
Cross straight

Blind path detected – following guide

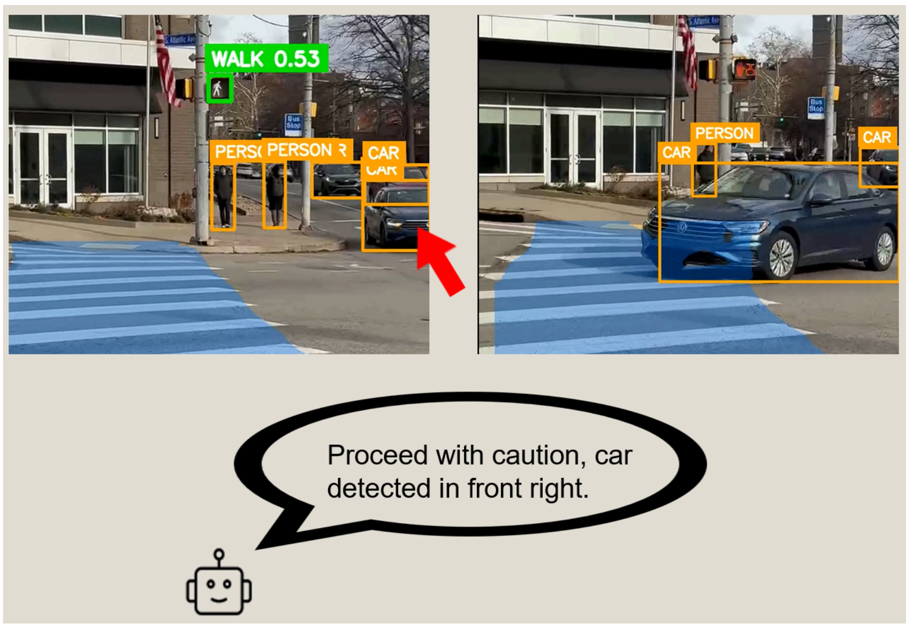
Exited crossing mode

5.3 Real-Time Hidden Risk Insights

The third case revisits the same intersection but activates Chatting Mode to explore the system's ability to surface subtle safety risks. At the start, the pedestrian signal is red, and several vehicles are waiting in different lanes. The spatial encoder converts detections into a structured description that specifies object types and coarse positions such as "car front left" or "bus far right." The prompt to the LLM instructs it to prioritize safety and maintain a single short sentence. As a result, the system announces "Stop; car in front left," summarizing the key risk while ignoring less relevant background details.



When the signal turns green and the crosswalk appears open, many sighted pedestrians might consider the situation safe. However, the detection model continues to track vehicles that could turn across the crosswalk. In this clip, a car to the right begins moving toward the pedestrian trajectory. The event rules interpret this pattern as a potential conflict, and the LLM receives an updated scene summary that emphasizes the moving vehicle. The generated guidance becomes "Proceed with caution, car in front right." A few moments later, the car indeed turns through the crosswalk without yielding.



This case illustrates that even though the system does not perform full trajectory prediction, combining persistent object detection with safety-focused language can help highlight risks that human observers might underestimate. It also demonstrates the value of Chatting Mode as more than a convenience feature; when tuned with strict safety prompts, it can complement the discrete event logic used in Walking Mode and Crossroad Mode and provide an additional layer of situational awareness for blind users.

6. Criteria for Success

We established four principal criteria to evaluate the success of our blind navigation system, addressing both technical robustness and practical utility. First, detection quality was measured through standard object detection and segmentation metrics. Our fine-tuned model was required to achieve a mean Average Precision at 50% Intersection over Union (mAP50) of at least 0.90 and a recall above 0.90, ensuring both high accuracy and minimal missed detections of critical elements like blind paths and traffic signals. Inference speed was also critical, with a target latency under 50 milliseconds per frame to guarantee real-time responsiveness on standard CPU hardware.

Second, we assessed behavior quality by evaluating whether the system could consistently generate concise, actionable, and contextually appropriate voice instructions from the detected objects, rather than merely listing them. Third, system behavior was evaluated for end-to-end stability; the integrated pipeline from camera input through YOLO detection and LLM-based instruction generation must maintain consistent performance without failure under typical urban environmental variations.

Finally, and fundamentally, success was defined by human-centered understanding. The system's design and outputs were grounded in the documented navigation challenges faced by blind and low-vision individuals. This meant prioritizing safety-critical scenarios, clearly understanding the limitations of each model component, and ensuring the technology provided genuine utility rather than just technical demonstration.

7. Limitations and Future Work

7.1 Limitations

Nevertheless, our current system has several limitations. Firstly, our data coverage remains narrow. A major portion of our training and test data remains to be collected from a few urban areas only. The conditions are more likely to be the same in those regions. Certain critical classes, such as tactile paths with unusual designs, uncommon objects, and pedestrian signals used in other countries, might be observed only a few times.

Second, there are several limits regarding the models and system performance. YOLOv8-seg and YOLOv8-det models may fail to detect small objects far away or objects that are partly hidden and visible only in a few frames. Additionally, sunlight exposure, shades, and motion blurs may affect model precision. Moreover, this model fails to detect many potentially hazardous street elements, such as small bumps on pedestrian paths and low street curbs. A blind user may trip over such features even when the main path and obstacles are handled correctly, so this gap is a serious safety limitation.

Third, our evaluation and use scenarios are limited. We perform most of our experiments on Bilibili recordings and on short routes in Pittsburgh instead of using our system for real-world walking. We also do not have a formal user study with visually impaired individuals and, therefore, are unable to measure workload-related tasks. The current system is a research prototype and not a certified safety device. Using our system instead of a cane and a guide dog could potentially pose a hazard for a user under particular boundary conditions.

Fourth, our evaluation is system-oriented rather than model-oriented. We do not conduct extensive ablation studies on architecture choices or hyperparameters, and we do not compare our YOLO backbone against a wide range of alternative detectors. We also do not aim to reach state-of-the-art benchmark numbers on any single dataset. Within the scope of a semester project and under limited compute, we prioritize building and debugging a complete end-to-end pipeline that runs in real time on edge hardware and supports three integrated modes. A more thorough experimental study – including controlled ablations of the rule-based event logic and comparisons between different perception backbones – is an important direction for future work once the core pipeline is stable.

7.2 Future Work

Future research could explore broadening where the system can be used and improving usability. One important direction is to extend to indoor and transit scenarios. Many pain points for blind users happen at elevators, escalators, and subway entrances. Our walking mode and crossroad mode could be adapted to detect doors, stairs, platform edges, and vehicle entrances, while chatting mode could help with tasks such as reading signs or finding the correct bus.

A second direction is to add more rule-based logic and events. The current event layer deals with a small set of events, such as path lost, obstacle ahead, and safe to cross. In future work, the model is expected to be able to handle more complex behaviors, such as tips about public transit and responses to unusual pedestrian behavior. Also, coding more low-confidence-level responses, such as “not sure, double check with your cane”, could be helpful. This would bring the language more in line with real-world orientation and would give a clearer and safer path for the language model to follow.

A third direction is to deploy the pipeline as a containerized service that can run on real devices. Right now, the system runs as a research demo. By packaging the YOLO-to-event-to-LLM stack into a container, we could test it on mobile phones or wearable devices and measure true end-to-end latency, battery use, and robustness. This would also make it easier to integrate with existing assistive apps that blind users already trust. Alongside this system’s development, we plan to test controlled studies with blind or low-vision participants and experts. These studies would help researchers understand how well the three modes, walking, crossroad, and chatting, fit into real daily routines, and what changes are needed before any future real-world deployment.

A fourth direction is to reuse the same perception and event pipeline to detect more subtle surface hazards. As highlighted in feedback from our instructor, one critical risk for low-vision pedestrians in Pittsburgh is not only missing curb ramps or obvious obstacles, but also uneven sidewalks where one slab is raised slightly above the next. Such small vertical discontinuities can cause serious trips and falls, yet they are not explicitly modeled by our current detector or segmentation modules. Extending the system to capture these hazards might require combining higher resolution patches near the ground plane, lightweight depth sensing, or learning visual cues such as shadows and texture changes that correlate with bumps. Designing a reusable “surface irregularity” module that plugs into the existing workflow could benefit many scenarios beyond outdoor blind-path following, including indoor corridors and transit platforms.

8. Conclusion & Reflection

This project aims to build an AI assistant that can turn first-person video into safety-focused guidance for blind pedestrians. We developed a multi-mode pipeline that begins with YOLOv8-based detection and segmentation, passes through a rule-based event layer, and ends with short voice messages generated through a small language model. Walking Mode keeps the user aligned with tactile paving and warns about obstacles on the path. Crossroad Mode detects signals and traffic to separate “wait” and “cross now” phases at intersections. Our experiments show that, with a comprehensive dataset and hyperparameter fine-tuning, lightweight models can reach usable accuracy, around 0.81 mAP50 for detection and 0.899 mAP50 for tactile-path segmentation, and support a prototype that operates on CPU.

At the same time, our work highlights several important limitations and lessons. The system still has a few limitations: it struggles with subtle surface hazards such as bumps and low curbs, and scenes in other countries that had different traffic signals or blind paths. Evaluation is based on recorded videos rather than live walks with blind users, so we cannot yet conclude about cognitive trust and long-term usability. These constraints led us to focus more on system behavior than on pushing model benchmarks. We found that explicit spatial encoding and strict constraints on language output were crucial for producing clear guidance. Looking ahead, we see the main priorities as expanding and diversifying datasets, modeling surface irregularities more directly, packaging the pipeline for on-device deployment, and working with blind users and orientation-and-mobility specialists to evaluate and refine the three modes in real-world settings.

Reference

Bourne, Rupert R. A., et al. “Trends in Prevalence of Blindness and Distance and Near Vision Impairment over 30 Years: An Analysis for the Global Burden of Disease Study.” *The Lancet Global Health*, vol. 9, no. 2, 2021, pp. e130–e143.

Jadhav, Aishwarya, et al. “AI Guide Dog: Egocentric Path Prediction on Smartphone.” *Proceedings of the AAAI Symposium Series*, vol. 5, no. 1, 2025, pp. 220–27.

Takano, Tsubasa, et al. “Tactile Paving Detection and Tracking Using Tenji10K Dataset.” *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 19, no. 10, 2024, pp. 1661–72.

Code Release

Our training dataset and implementation code are publicly available at:

https://drive.google.com/file/d/1Isqc7mHGZR8vmFGpl7AzRizI4IINzica/view

https://github.com/tantansir/BlindNav

Team Member Contributions

Kaizhen Tan: methodology overview; implementation of Feature 1; case study analysis; additions to the limitations and future work sections; code integration.

Nicole Lyu: implementation of Feature 2 (YOLO inference, spatial encoding logic, and local LLM integration); prompt engineering for real-time safety constraints; end-to-end pipeline verification.

Yixiao Li: introduction, problem definition, and motivation; preliminary literature review; limitations and future work; conclusion & reflection; reference.

Yufan Wang: dataset consolidation; implementation of YOLO training configuration and hyperparameter tuning for the detection part; EDA and detection sections.

Hanzhe Hong: EDA on semantic model datasets; training, inference test, fine tuning and evaluation on semantic segmentation model; implementation of criteria for success.

AI Appendix

We used AI tools only for translation, grammar correction, and wording refinement.