WalkAware: Real-Time Sidewalk Scene Understanding for Assistive Pedestrian Navigation

Diana Aleksieieva, Hao Zhu, Hong Zhao, Riyesh Nath
CSC-I1910 Deep Learning, The City College of New York
May 14, 2026

Abstract

Blind and low-vision pedestrians need real-time awareness of sidewalk hazards, but many navigation tools focus mainly on route guidance. We present WalkAware, a pedestrian-view perception pipeline that combines semantic segmentation, object detection, tracking, and rule-based fusion to identify hazards and produce safety alerts: SAFE, INFO, WARNING, and DANGER. We evaluate four components. For segmentation, fine-tuned PIDNet improves mIoU from 68.53% to 73.48% and bottom-sidewalk recall from 75.52% to 87.05%. For detection, fine-tuned YOLO11 reaches mAP@0.5 = 0.480 with the selected medium model (freeze=15), up from 0.327 stock, at 9.95 ms GPU latency. A BoT-SORT tracker and 10-rule fusion engine combine object identity, walkable context, proximity, position, and collision risk into final alerts. Ablation shows that tracking and segmentation are critical. Removing tracking leaves about 36% of warnings, while removing both tracking and segmentation leaves only about 4%. We also test monocular depth as a future extension, but exclude it from the current pipeline due to speed and validation limits.

1 Introduction

At least 2.2 billion people worldwide live with near or distance vision impairment [1]. For blind and low-vision pedestrians, safe navigation is not only about route guidance. It also requires real-time awareness of what is directly ahead: whether the sidewalk is clear, where the walkable path continues, and whether a vehicle, bicycle, or obstacle is entering the walking area.

This problem is well suited for deep learning because it requires both dense scene understanding and hazard detection. Semantic segmentation can estimate sidewalk, curb, road, and crosswalk regions [2, 3, 4], while object detection and tracking can identify moving or discrete hazards [5, 6].

WalkAware is a pedestrian-view perception pipeline for assistive navigation. It combines semantic segmentation, object detection, exploratory monocular depth estimation, tracking, and rule-based fusion to generate interpretable safety alerts: SAFE, INFO, WARNING, and DANGER. This report describes the pipeline, model choices, experiments, and remaining limitations, including sidewalk-road confusion, missed small hazards, speed constraints, and limited validation of depth-based fusion [7].



Figure 1: WalkAware task overview. Given a pedestrian-view frame, the system combines scene layout, object hazards, motion cues, and rule-based logic to produce interpretable safety alerts

2 Related Work

WalkAware builds on prior work in segmentation, detection, tracking, depth estimation, and assistive navigation. For semantic segmentation, we compare PIDNet, a real-time boundary-aware CNN, with SegFormer-B0, a lightweight transformer model [5, 6]. For object detection, we use COCO-pretrained YOLOv8 and YOLO11 as baselines and fine-tune YOLO11 for pedestrian-view walking hazards [4, 7]. For temporal reasoning, we use BoT-SORT, which builds on tracking-by-detection methods such as ByteTrack and provides object identity and motion cues for collision reasoning [13, 14].

WalkVLM provides pedestrian-view walking videos and motivates the assistive navigation setting, but our goal is different: WalkAware focuses on a lightweight perception pipeline rather than large vision-language generation [2]. We also use SAM-style masks to support pedestrian-view label generation [9, 10]. Monocular depth models such as MiDaS and DepthPro are evaluated only as exploratory proximity signals, and YOLOPX is included as a faster single-model alternative to the modular pipeline [11, 12, 15, 16].

WalkAware differs from these works by targeting pedestrian-view sidewalk perception, explicitly evaluating vehicle-view to pedestrian-view domain shift, and producing interpretable safety alerts from fused segmentation, detection, tracking, and optional depth cues.

3 Data

3.1 Data Sources

WalkAware uses three data sources. **Mapillary Vistas** is the source-domain segmentation dataset, with 18,000 training images, 2,000 validation images, and dense pixel labels; it is released under the Mapillary Vistas Research Use License (CC BY-NC-SA, research / non-commercial only). It provides strong street-scene supervision, but most images are vehicle-view, so it does not fully match pedestrian sidewalk video. **COCO** is used through COCO-pretrained YOLO checkpoints for object detection baselines, including common classes such as person, bicycle, car, bus, and truck; COCO annotations are released under CC BY 4.0 and the underlying images are subject to their original Flickr terms. **WalkVLM-derived pedestrian-view frames** are used for target-domain fine-tuning and evaluation; the WalkVLM project does not publish a formal dataset license, so we use it only for research and evaluation under the terms of the WalkVLM repository. The main segmentation fine-tuning split contains 3,996 training frames and 954 validation frames, with 1,611 held-out test frames. A separate 1,612-frame sidewalk-view split is used for domain-gap analysis. All three datasets are used under research / non-commercial terms appropriate for an academic course project; no redistributed copies of Mapillary or WalkVLM imagery are bundled with this repository.

3.2 Label Format and Taxonomy

For segmentation, we use a reduced seven-class walking taxonomy: Curb, Crosswalk, Road, Sidewalk, Building, Vegetation, and Sky. This keeps the classes most relevant to walkable-area reasoning.

For detection, we use a 12-class walking-hazard taxonomy. Seven classes overlap with COCO: person, bicycle, car, motorcycle, bus, truck, and fire hydrant. Five are walking-specific hazards that stock COCO detectors do not cover well: pole, trash can, cone, manhole, and street sign.

3.3 Pedestrian-View Label Generation

WalkVLM frames were labeled with a SAM 3 text-prompted pipeline. Each class was prompted separately, candidate masks were filtered by confidence and area, and the final semantic map was built using a highest-score-wins merge. Uncertain pixels were assigned the ignore label 255 and excluded from training loss. Crosswalk required the most manual cleanup because SAM often confused it with shadows, lane markings, and brick textures. We removed incorrect crosswalk masks to make the labels more conservative and reliable.

3.4 Exploratory Data Analysis

We compare class distributions across Mapillary, the WalkVLM fine-tuning split, and the held-out pedestrian-view test set. The comparison shows a clear domain shift. Mapillary is dominated by sky, road, vegetation, and building pixels, while sidewalk, curb, and crosswalk are relatively rare. In the held-out pedestrian-view test set, sidewalk increases to about 30.1% of labeled pixels, compared with only 3.6% in the Mapillary source sample.

This shift matters because WalkAware depends on sidewalk and curb understanding for navigation. It also explains why source-domain performance alone is not enough. Crosswalk remains rare across all splits, so its IoU is unstable and sensitive to small labeling errors. Figure 2 confirms that the target pedestrian-view data has a different class distribution from the source-domain Mapillary data.

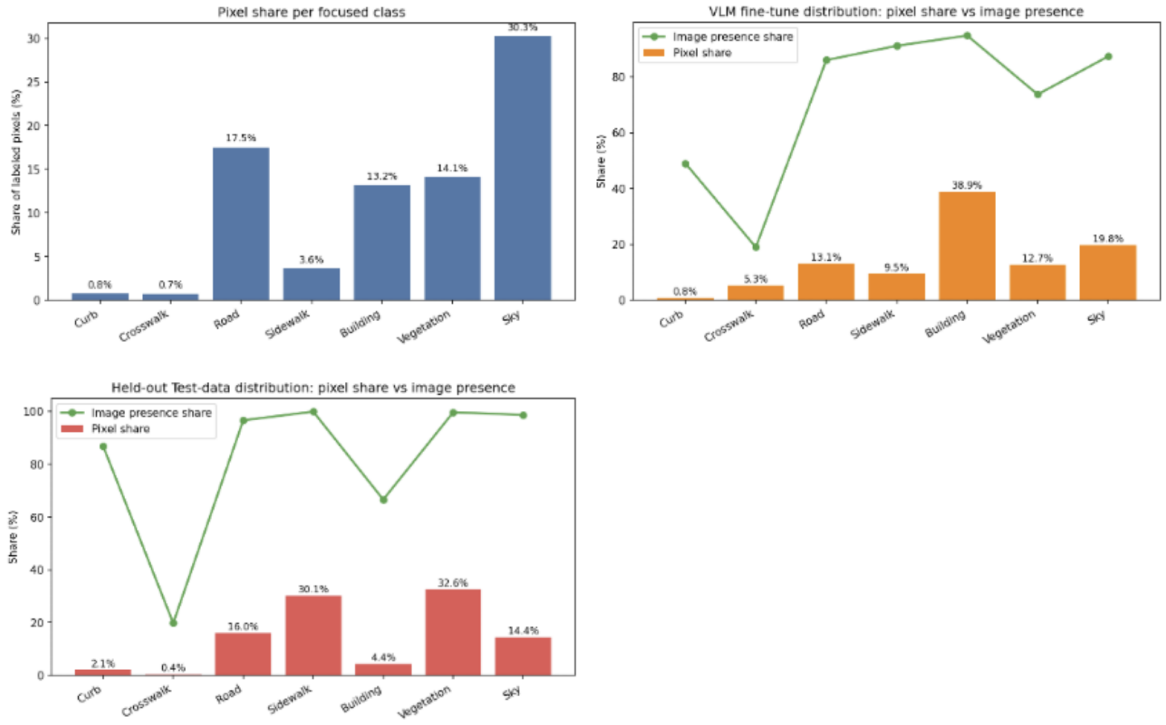


Figure 2: Seven-class pixel distribution across datasets. Top: Mapillary Vistas source-domain training sample. Middle: WalkVLM-derived fine-tuning split. Bottom: held-out sidewalk-view test set. The figure shows a strong vehicle-view to pedestrian-view domain shift: sidewalk pixels increase from 3.6% in Mapillary to 30.1% in the held-out test set, while curb and crosswalk remain rare.

3.5 Reproducibility of Data Pipeline

We use the official Mapillary split for source-domain training and validation. For WalkVLM data, we use per-video grouping so frames from the same video never appear in different train, validation, or test splits. This prevents near-duplicate frame leakage. Preprocessing is fit only on the training set and then applied to validation and test data. Random seeds are fixed for the main runs, and the frozen split file is stored in `data/interim/vlm_segmentation_split.csv`. Dependencies are pinned in `requirements.txt` and `environment.yml`, and the README documents data setup, training, and evaluation steps.

4 Methods

WalkAware is organized as a modular pedestrian-view perception pipeline. For each video frame, semantic segmentation estimates walkable scene layout, object detection identifies discrete hazards, tracking links

detections across time, and a rule-based fusion layer converts these signals into safety alerts. The deployed pipeline uses PIDNet-S for segmentation, YOLO11 with BoT-SORT for detection and tracking, and Fusion v2 for alert generation. We also evaluate YOLOPX and monocular depth estimation as exploratory extensions for faster deployment and distance-aware reasoning.

4.1 Pipeline Overview

The main WalkAware pipeline has two parallel perception branches. The segmentation branch predicts a seven-class scene map used for sidewalk, road, crosswalk, wall, and user-location reasoning. The detection branch predicts object classes and bounding boxes, then BoT-SORT assigns stable track IDs and motion states across frames. The fusion layer combines both branches to produce per-track alerts: SAFE, INFO, WARNING, or DANGER.

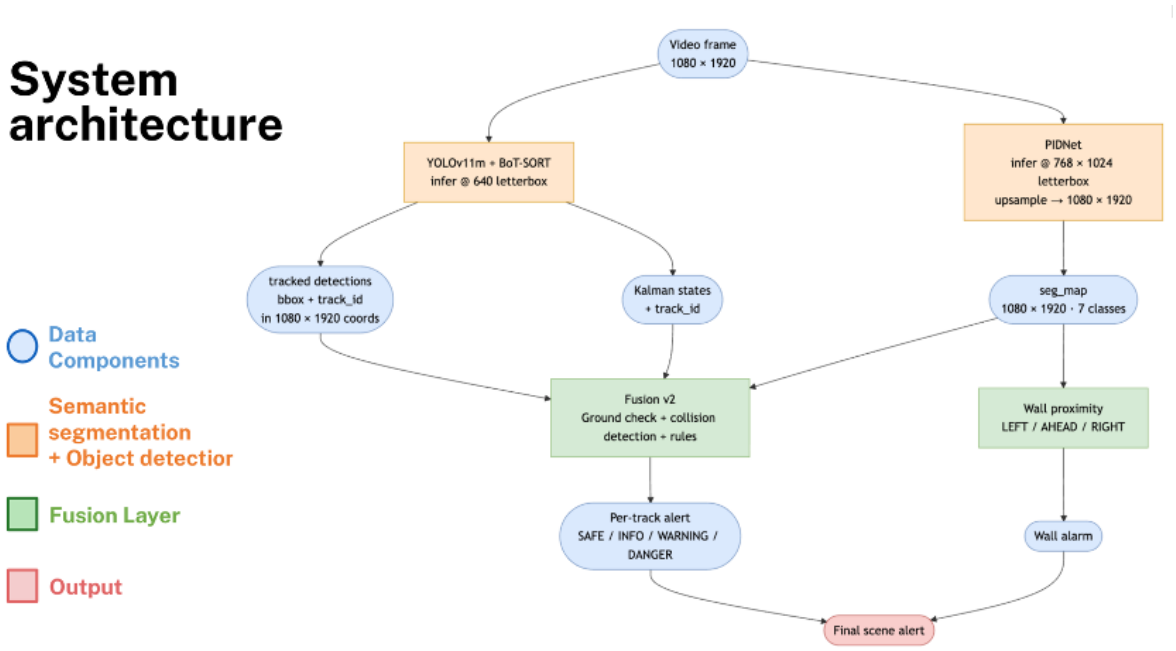


Figure 3: Modular WalkAware system architecture. The input video frame is processed by two parallel branches: YOLO11 with BoT-SORT for object detection and tracking, and PIDNet for semantic segmentation. The detection branch outputs bounding boxes, track IDs, and Kalman motion states. The segmentation branch outputs a seven-class scene map used for ground-contact reasoning and wall-proximity checks. The Fusion v2 layer combines tracked objects, motion cues, segmentation context, and rule-based logic to produce per-track safety alerts: SAFE, INFO, WARNING, or DANGER.

4.2 Baselines

We use three baselines to make the model gains interpretable. First, a weak non-deep-learning segmentation baseline predicts the most frequent Mapillary class for every pixel. The majority-pixel baseline predicts Sky for every pixel and reaches only 13.04% pixel accuracy, 1.86 mIoU, and 3.30 mDice on the pedestrian-view test split, with 0.00 IoU for Crosswalk, Road, and Sidewalk. This baseline performs poorly on pedestrian-view data, showing that source-domain class priors alone cannot solve the task.

Second, we use frozen PIDNet-S and SegFormer-B0 models trained on Mapillary but not fine-tuned on WalkVLM. These baselines measure the vehicle-view to pedestrian-view domain gap. Third, for detection, we use stock COCO-pretrained YOLO11 as the baseline and compare it with YOLO11 fine-tuned on walking hazards.

4.3 Semantic Segmentation Models

We compare two deep learning segmentation approaches under the same seven-class WalkAware taxonomy.

PIDNet-S is the main CNN-based segmentation model. It uses separate branches for spatial detail, contextual aggregation, and boundary information. This design is useful for WalkAware because pedestrian navigation depends on thin and spatially precise classes such as curb and crosswalk. We fine-tune the Mapillary-trained PIDNet checkpoint on WalkVLM-derived pedestrian-view frames. To emphasize navigation-critical regions, we add class weighting for Road, Sidewalk, and Crosswalk and apply extra weight to the bottom part of the image, where immediate walking decisions occur.

SegFormer-B0 is used as the transformer-based comparison. It combines a lightweight transformer encoder with an MLP decoder and tests whether global context improves transfer to pedestrian-view sidewalk scenes. We fine-tune SegFormer-B0 on the same WalkVLM split and evaluate it with the same metrics and test data as PIDNet.

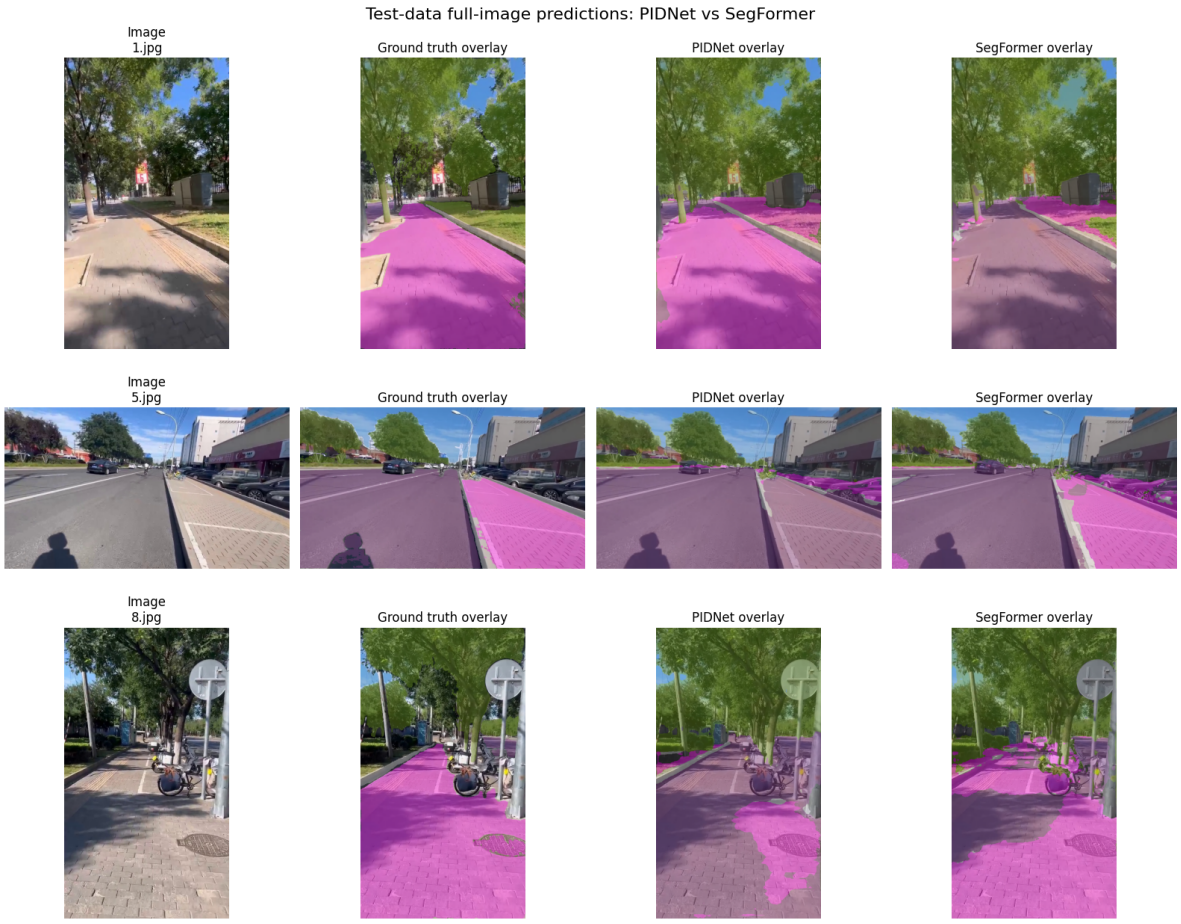


Figure 4: Segmentation examples from PIDNet and SegFormer. The examples compare ground-truth pedestrian-view labels with predictions from the two segmentation models. The main qualitative difference is in the bottom walking region, where sidewalk and road are often confused due to the vehicle-view training bias.

4.4 Object Detection and Tracking

Object detection provides the discrete hazard signal used by the fusion layer. Stock COCO YOLO can detect common objects such as people, bicycles, cars, buses, and trucks, but it does not cover several sidewalk-specific hazards. To close this class-space gap, we fine-tune YOLO11 on a 12-class walking-hazard taxonomy: seven COCO-overlap classes and five walking-specific classes: street sign, pole, trash can, cone,

and manhole. YOLO11 is fine-tuned on 3,890 training frames and 977 validation frames, with a separate 1,612-frame manually grounded pedestrian-view test set.

We train YOLO11n, YOLO11m, and YOLO11l variants to study the speed–accuracy tradeoff. The medium model is selected as the deployment default because it provides strong accuracy with lower latency than the large model. We also test layer-freezing variants to reduce forgetting on COCO classes while adapting to the new walking-specific classes.

After detection, BoT-SORT links detections across frames. Tracking provides stable object IDs, short motion histories, and Kalman-state signals. These are needed for approach detection and time-to-collision reasoning in the fusion layer.

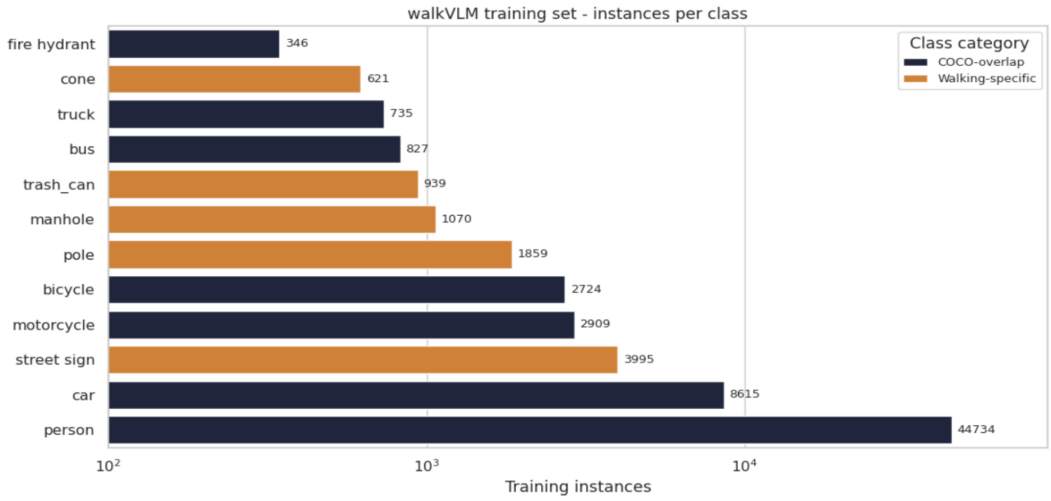


Figure 5: Training distribution for the 12-class walking-hazard taxonomy. The detection model is fine-tuned on seven COCO-overlap classes and five walking-specific classes: pole, trash can, cone, manhole, and street sign. The distribution is highly imbalanced: person and car dominate the training set, while classes such as fire hydrant, cone, truck, and bus have far fewer instances.

4.5 Fusion Layer and Safety Alert Logic

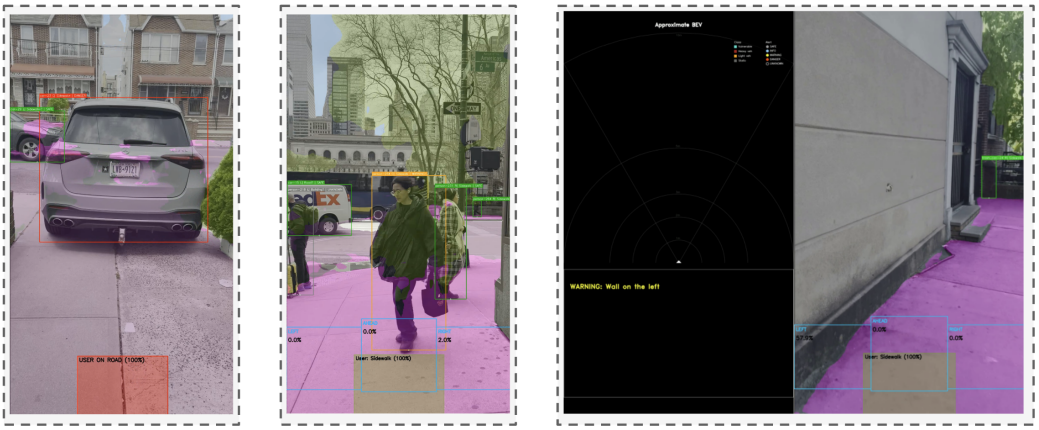
The fusion engine is the decision layer of WalkAware. It combines semantic segmentation, object detection, and tracking outputs into interpretable safety alerts. This step is necessary because object detection alone can identify objects, but it cannot explain whether an object is dangerous for a pedestrian. For example, a car on the road may be normal traffic, while a car on the sidewalk is a direct safety risk.

WalkAware uses a rule-based fusion design instead of a learned alert classifier. This choice supports explainability, debugging, and adjustment for different deployment conditions. Each alert can be traced back to visible signals such as object class, ground context, proximity, motion, or unreliable evidence.

For each tracked object, Fusion v2 uses the detector class and bounding box, the tracker ID and motion history, and the segmentation label under the object. These inputs are converted into safety-relevant signals: object group, ground label, proximity band, motion state, collision risk, horizontal position, and reliability. The rule ladder then assigns one of five alert levels: Safe, Info, Warning, Danger, or Unknown.

Table 1: Fusion alert levels.

Alert level	Meaning	Example
DANGER	Immediate threat to the user	A car on the sidewalk close to the user, or a vehicle on a crosswalk approaching the user
WARNING	Situation requires attention	Any close object, or a bicycle/motorcycle on a pedestrian surface at medium distance
INFO	Low-risk object worth acknowledging	A medium-distance object that is approaching but not yet dangerous
SAFE	Object detected, but no action needed	A far pedestrian or static object outside the user's path
UNKNOWN	The system cannot safely classify the object	The object's ground label falls on an unreliable structure class such as Building or Sky



Danger: A car is ahead | *Warning: A person is ahead, approaching* | *Warning: Building is on the left*

Figure 6: Fusion signals and rule-based alert generation. Fusion v2 combines object-level and scene-level evidence to produce safety alerts. The examples show that alerts depend on class, ground context, proximity, motion, position, and reliability rather than object detection alone.

The alert taxonomy separates low-risk detections from situations that require user attention. The goal is not to warn about every detected object, but to warn when the object matters for walking safety. Table 1 summarizes the five alert levels.

The rule ladder is priority ordered. High-risk cases are checked first, such as heavy vehicles on pedestrian surfaces or imminent crossing conflicts. Lower-risk rules handle mid-range approaching objects, static obstacles, and default safe cases. This structure makes the system conservative while still avoiding unnecessary warnings.

Fusion also includes scene-level checks from segmentation. The bottom-center region of the segmentation map is used to estimate the user's immediate walking surface. This allows the system to flag road, curb, crosswalk, or blocked-walkway risks even when no object detector box is present.

A more detailed description of the per-object signals, time-to-contact estimates, reliability handling, and Fusion v1/Fusion v2 comparison is provided in Appendix A.

Fusion v2 extends the earlier Fusion v1 prototype by replacing simple image-plane heuristics with projective proximity estimates, tracker-based motion states, time-to-contact, and lateral collision checks. A detailed comparison is included in Appendix A.4.

Table 2: Simplified Fusion v2 rule ladder.

Rule type	Condition	Output	Reason
Heavy vehicle on sidewalk/ crosswalk	Car, bus, or truck on sidewalk and NEAR/MID	DANGER	Vehicle on walking surface
Imminent crossing conflict	Vehicle on crosswalk, MID, and collision risk is high	DANGER	Object moves into user path
Heavy vehicle approaching pedestrian surface	Vehicle on sidewalk/crosswalk, MID, approaching	WARNING	Risk is increasing
Light vehicle on pedestrian surface	Bicycle or motorcycle on sidewalk/crosswalk at MID distance	WARNING	Fast object in pedestrian space
Any near object	Any detected object at NEAR distance	WARNING	Class-agnostic safety rule
Mid-range approaching object	Object at MID distance and approaching	INFO	Relevant but not yet dangerous
Static obstacle not near	Pole, cone, trash can, hydrant, etc. at MID/FAR	SAFE	Avoids repeated unnecessary warnings
Unreliable structure label	Ground label is Building, Sky, or ignored region	UNKNOWN	Ground context cannot be trusted
Default	No risk rule applies	SAFE	Object is not safety-critical

4.6 Single-Model Alternative: YOLOPX

The modular WalkAware pipeline is easier to debug, but it requires separate segmentation and detection models. This creates an inference-speed bottleneck because PIDNet and YOLO11 each require a separate forward pass. As a deployment-oriented alternative, we evaluated YOLOPX, a single-model multi-task network with a shared backbone and separate heads for segmentation and detection.

YOLOPX is not the headline model because its detection accuracy is lower than fine-tuned YOLO11. However, it is important for deployment because one shared forward pass is faster than running PIDNet and YOLO11 separately. In Section 5.8, YOLOPX reaches 26.3 FPS, while the modular PIDNet + YOLO11 + fusion pipeline reaches 9.3 FPS.

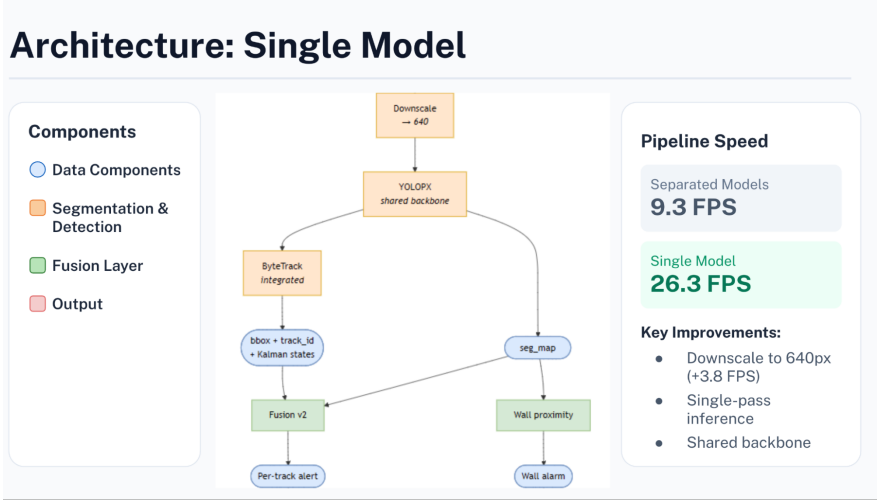


Figure 7: YOLOPX single-model alternative. YOLOPX uses one shared backbone for segmentation and detection, reducing the number of forward passes per frame.

The main conclusion from the YOLOPX experiment is that speed can be improved with a single-model

architecture, but accuracy remains the limiting factor. Future work could improve YOLOPX with more pedestrian-view training data, stronger class balancing, better fine-tuning of walking-specific hazards, and a segmentation head designed directly for sidewalk navigation. Additional YOLOPX architecture and training details are provided in Appendix B.

4.7 Exploratory Depth Estimation

Depth estimation is not part of the current deployed pipeline. We evaluate it as an exploratory extension because distance can improve warning priority. A close object should produce a stronger alert than a far object, and dense depth may also help detect unknown obstacles that are not in the detector's class list.

The depth prototype uses YOLO11 detections and a monocular depth map from MiDaS, DepthPro, or Depth Anything v2. For each detected object, we sample the lower-center 40% of the bounding box, which is more likely to correspond to the object's ground-contact region. We then aggregate depth using the 25th percentile to bias toward the closest relevant pixels. The output is converted into NEAR, MID, or FAR bands for possible fusion use.

Depth remains exploratory because it has not been validated with measured distance ground truth and is too slow for the current real-time target. MiDaS is useful for relative near/mid/far prototyping, while DepthPro provides metric estimates but requires stronger validation.

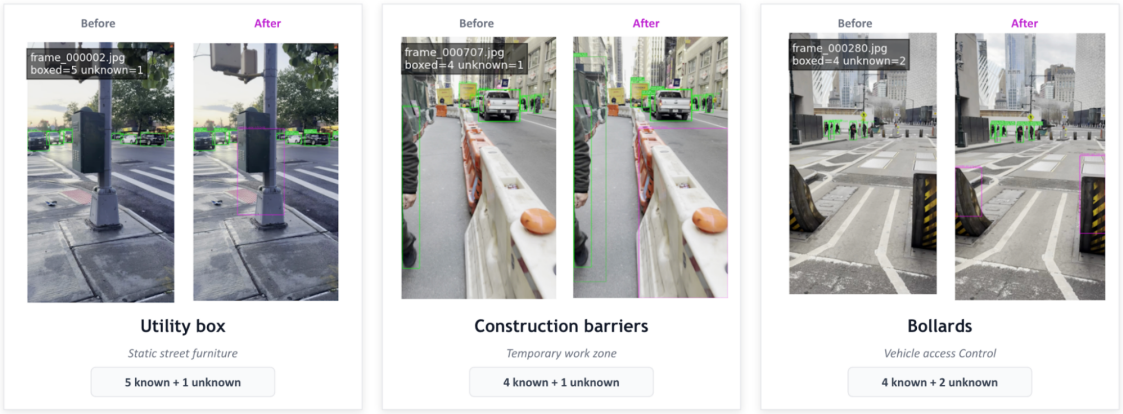


Figure 8: Exploratory unknown-obstacle detection with depth. Known YOLO detections are removed from the walkable region, and the remaining area is scanned for near-depth regions. This allows the system to flag a close obstacle in the walking path even when the object class is unknown. This is a proof of concept and is not part of the current deployed pipeline.

5 Experiments and Results

5.1 Experimental Setup and Metrics

We evaluate WalkAware at three levels: segmentation, detection, and full-pipeline behavior. Segmentation models are evaluated on the labeled WalkVLM pedestrian-view test split with 1,611 frames. We report pixel accuracy, mIoU, mDice, per-class IoU, and navigation-specific metrics such as bottom-region Sidewalk recall and bottom Road IoU. Detection models are evaluated with precision, recall, F1, mAP@0.5, and mAP@0.5:0.95. For system behavior, we report warning-rate ablations and sustained inference speed on a fixed test video.

All experiments use frozen train/validation/test splits. Preprocessing is fit only on training data. Per-run configurations, checkpoints, metrics, and qualitative outputs are tracked in repository experiment logs. Because training every variant is GPU-expensive, most results are single-seed estimates; the headline PIDNet fine-tune is repeated with a second seed as a stability check. Hyperparameter exploration focused on learning rate, batch size, resize strategy, class weighting, bottom-region weighting, YOLO model size, and YOLO layer-freezing depth. Full run logs are kept in the repository experiment tables.

5.2 Segmentation: Baseline vs. Fine-Tuning

The first experiment asks whether pedestrian-view fine-tuning improves segmentation under the vehicle-view to pedestrian-view domain shift. We compare frozen Mapillary-trained PIDNet-S and SegFormer-B0 against their WalkVLM fine-tuned versions.

Before evaluating transfer to WalkVLM, we first verify that the source-domain PIDNet-S baseline trained normally on Mapillary Vistas. Figure 9 shows the training trajectory for the 7-class source-domain model. The total training loss decreases rapidly at the beginning and then stabilizes, while validation mIoU continues to improve gradually across training. The best validation checkpoint is reached at epoch 120, with validation mIoU of 86.0. We use this checkpoint both as the frozen PIDNet baseline and as the initialization point for WalkVLM fine-tuning.

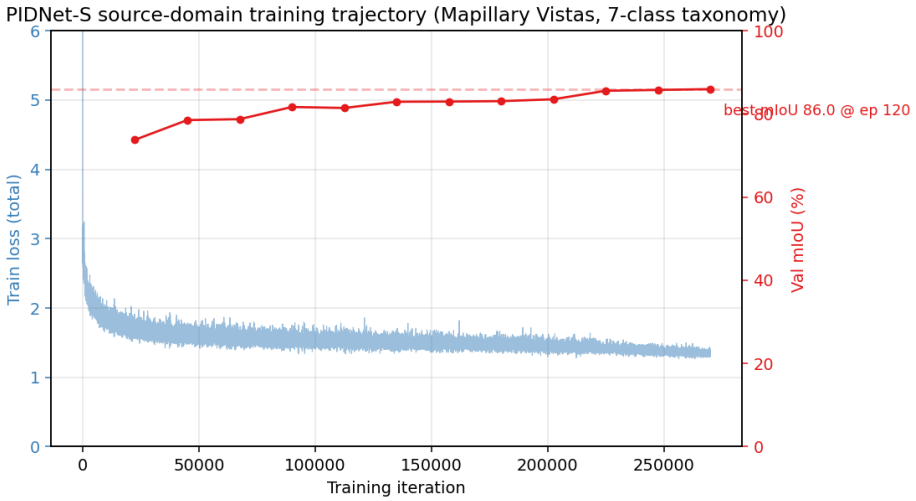


Figure 9: PIDNet-S source-domain training trajectory on Mapillary Vistas using the reduced 7-class WalkAware taxonomy. The blue curve shows total training loss over iterations, and the red curve shows validation mIoU across epochs. Training loss drops quickly and then stabilizes, while validation mIoU improves steadily until the best checkpoint at epoch 120. This checkpoint provides the frozen source-domain baseline and the starting point for pedestrian-view fine-tuning.

Table 3: Segmentation results on the labeled WalkVLM test set. The frozen models are evaluated without WalkVLM fine-tuning. The fine-tuned models use the same test protocol. Best values are bolded.

Model	Pixel Acc	mIoU	mDice	Crosswalk IoU	Road IoU	Sidewalk IoU
SegFormer frozen	86.12	62.61	73.24	22.25	59.88	68.67
SegFormer fine-tune	89.50	65.56	75.68	21.38	69.44	78.16
PIDNet frozen	89.39	68.53	78.75	**35.36**	67.43	77.55
PIDNet fine-tune	**93.36**	**73.49**	**82.11**	32.64	**79.47**	**87.00**

Both models improve after pedestrian-view fine-tuning. SegFormer improves by +2.95 mIoU. PIDNet improves by +4.96 mIoU and is the only headline segmentation result repeated with a second random seed. Across seeds 42 and 123, the fine-tuned PIDNet achieves 73.48 ± 0.01 mIoU (mean ± sample standard deviation; range 73.47–73.49). This suggests that the main PIDNet fine-tuning gain is stable and not a seed artifact.

Figure 10 shows the validation curve for the selected weighted-priority PIDNet fine-tune. Validation mIoU reaches its maximum very early, at epoch 2, and then slowly declines. This supports the use of early stopping and confirms that the fine-tuning step is mainly a short domain-adaptation stage rather than a long retraining process.

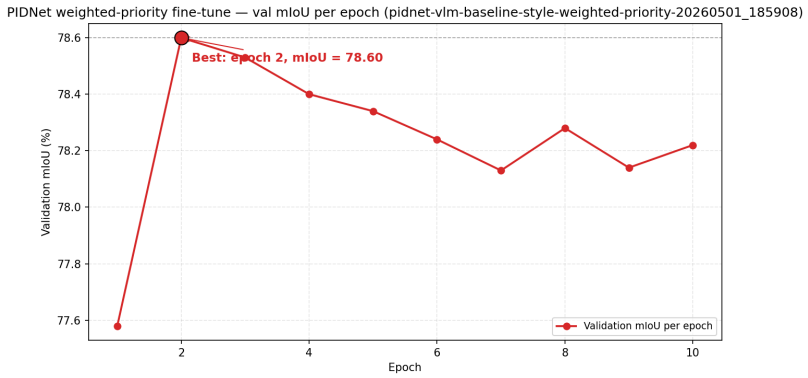


Figure 10: Validation mIoU during the selected PIDNet weighted-priority fine-tune on the pedestrian-view training split. The best validation checkpoint occurs at epoch 2 with 78.60 validation mIoU. Later epochs do not improve the validation score, which suggests that short fine-tuning with early stopping is more effective than longer retraining for this transfer setting.

PIDNet is the strongest segmentation model overall. It has the best aggregate scores and the best Road and Sidewalk IoU. These classes are especially important for navigation because they define the main non-walkable and walkable ground regions. Averaged across the two PIDNet seeds, fine-tuning improves Road IoU from 67.43 to 80.20 (range 79.47–80.94), Sidewalk IoU from 77.55 to 87.47 (range 87.00–87.94), and bottom-region Sidewalk recall from 75.52 to 87.05 (range 86.28–87.82).

Crosswalk is the main exception. Crosswalk IoU decreases from 35.36 in the frozen PIDNet baseline to a two-seed mean of 33.19 after fine-tuning. We therefore treat crosswalk preservation as a remaining limitation rather than claiming uniform improvement across all classes. Based on the overall gains and the strong improvement on Road and Sidewalk, the fine-tuned PIDNet model is selected as the segmentation component for the final modular WalkAware pipeline.

Reporting limitation. Compute budget limited most experiments to single-seed point estimates. This includes the SegFormer fine-tune, YOLO11 variants, YOLOPX, and fusion ablations reported in Tables 4–6. The second-seed validation was performed only for the PIDNet headline result because it is the main segmentation number used throughout the report. Therefore, all other single-seed rows should be interpreted as point estimates, not as mean ± standard deviation results.

Figure 11 compares the PIDNet frozen baseline and the best PIDNet fine-tune using row-normalized confusion matrices. The fine-tuned model improves Sidewalk recognition from 80.9% to 91.4%, which is important because sidewalk is the main safe-walking region. It also reduces Sidewalk-to-Road confusion from 16.7% to 7.0%. Curb and Crosswalk remain harder classes. Curb is often confused with Sidewalk and Road, and Crosswalk is still partially confused with Road. This supports the quantitative results in Table 3: fine-tuning improves the main navigation classes, but thin or visually ambiguous boundary classes remain challenging.

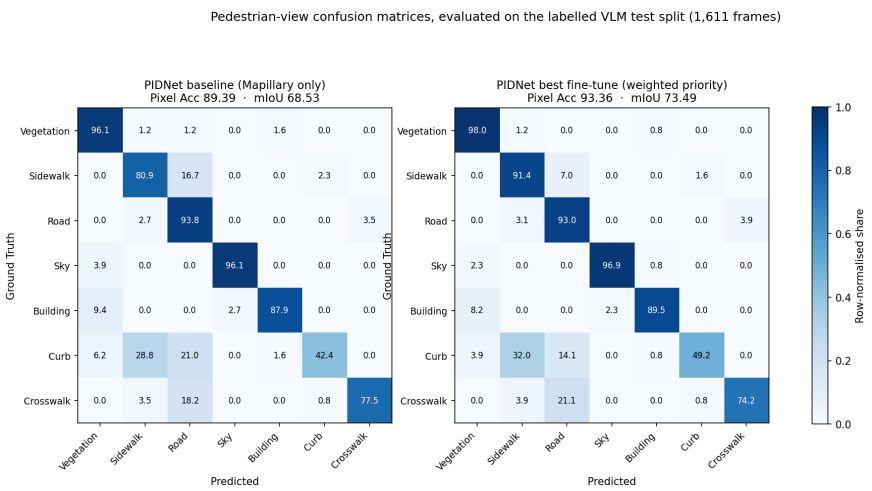


Figure 11: Row-normalized confusion matrices for the frozen PIDNet baseline and the best PIDNet fine-tuned model on the labeled WalkVLM test split. Fine-tuning improves Sidewalk recognition and reduces Sidewalk-to-Road confusion, which is important for pedestrian navigation. Remaining errors are concentrated in Curb and Crosswalk, which are visually thin, rare, and often overlap with road or sidewalk regions.

The PIDNet fine-tune was repeated with a second seed. The two runs produced nearly identical mIoU values, 73.49 and 73.47, which suggests that the main fine-tuning gain is not a seed artifact. Based on these results, the fine-tuned PIDNet-S model is selected as the segmentation backbone for the final modular WalkAware pipeline.

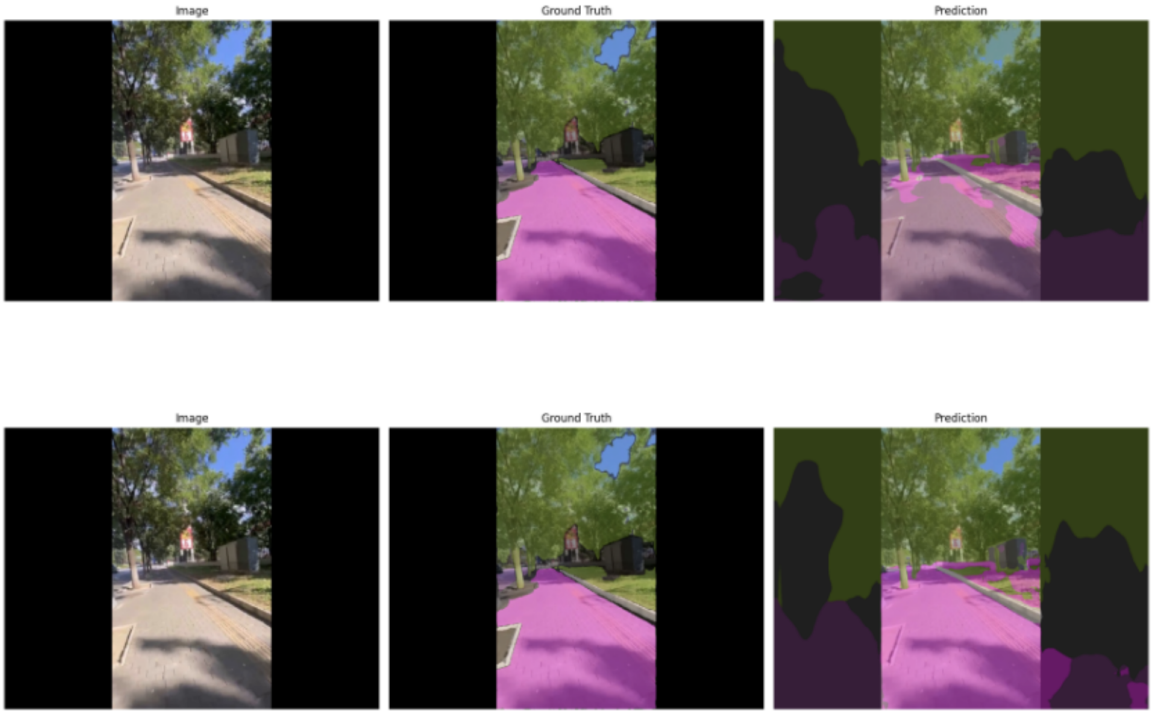


Figure 12: Segmentation improvement after pedestrian-view fine-tuning. The qualitative examples compare ground truth, frozen baseline prediction, and fine-tuned prediction. Fine-tuning improves the bottom walking region, especially Sidewalk and Road, but Crosswalk remains unstable.

5.3 Domain-Gap Evaluation

The second segmentation experiment tests whether the models generalize to a separate held-out sidewalk-view split. This split contains 1,612 frames from 203 video groups and is used to measure vehicle-view to pedestrian-view transfer.

Table 4: Domain-gap evaluation on the held-out sidewalk-view test set.

Metric	**PIDNet**	**SegFormer**
Full-image pixel accuracy	89.27%	85.70%
Full-image mIoU	62.99%	56.66%
Road IoU	67.78%	57.91%
Sidewalk IoU	70.94%	61.77%
Bottom-band mIoU	47.65%	39.90%
Sidewalk → Road confusion	20.86%	26.93%
Bottom-band frame-to-frame change rate	18.07%	21.52%

PIDNet transfers better on every metric. It has higher full-image and bottom-band accuracy, lower Sidewalk → Road confusion, and better adjacent-frame stability. This supports the choice of PIDNet as the segmentation model for the final modular pipeline. The main remaining failure mode is still Sidewalk → Road confusion in the bottom part of pedestrian-view frames.

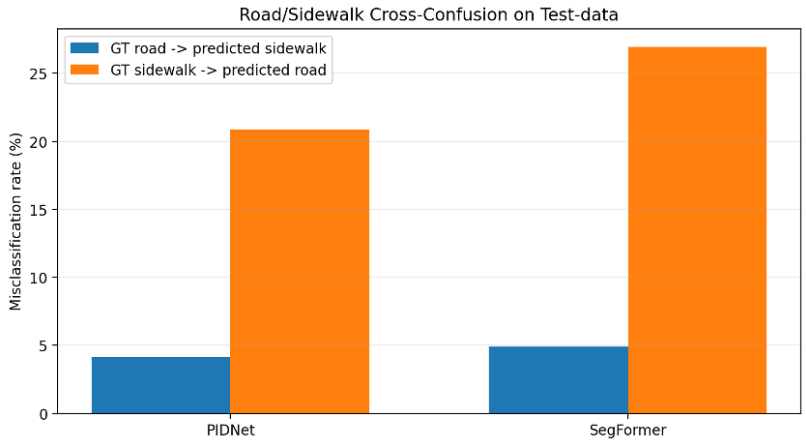


Figure 13: Road/Sidewalk cross-confusion under domain shift. Sidewalk → Road is the dominant error direction for both models, but PIDNet reduces this confusion compared with SegFormer. This confirms that pedestrian-view ground texture remains the hardest segmentation challenge.

5.4 Detection: Stock vs. Fine-Tuned YOLO11

The detection experiment asks whether fine-tuning improves hazard detection beyond stock COCO YOLO. The main challenge is class-space mismatch: stock COCO YOLO does not cover several sidewalk hazards, including pole, trash can, cone, manhole, and street sign.

First, we compare stock YOLOv8 and YOLO11 on COCO-overlap classes. YOLO11 is slightly cleaner than YOLOv8, with higher precision and slightly better mAP@0.5:0.95, so we use YOLO11 for the final fine-tuned detector. However, stock YOLO11 scores 0.0 AP on the five walking-specific classes because they are not part of the COCO class space.

Table 5: Detection results on the 12-class walking-hazard taxonomy.

Model	mAP@0.5	mAP@0.5:0.95	Precision	Recall	F1	ms/img
YOLO11n stock	0.201	0.135	0.78	0.36	0.50	8.7
YOLO11m stock	0.327	0.247	0.86	0.52	0.65	10.7
YOLO11n fine-tune	0.290	0.173	0.59	0.48	0.53	8.6
YOLO11m fine-tune	0.403	0.275	0.68	0.60	0.64	9.7
YOLO11l fine-tune	0.434	0.300	0.68	0.62	0.65	15.4
YOLO11m fine-tune, freeze=15	**0.480**	**0.332**	0.63	**0.69**	**0.66**	9.95
YOLOPX fine-tune	0.347	0.187	0.57	0.63	0.60	23.06

Fine-tuning recovers the walking-specific classes that stock COCO YOLO misses. The best fine-tuned model is YOLO11m with a frozen backbone (freeze=15), reaching 0.480 mAP@0.5 at 9.95 ms/image. We select it as the deployment default because it gives a better speed–accuracy tradeoff than YOLO11l, reaching higher mAP@0.5 (0.480 vs 0.434) at lower latency (9.95 vs 13.83 ms/image).

Fine-tuning also introduces a tradeoff. It improves walking-specific hazard recall, but some rare COCO classes regress, especially truck and bus. This suggests that the detection model needs more balanced target-domain examples for rare classes.

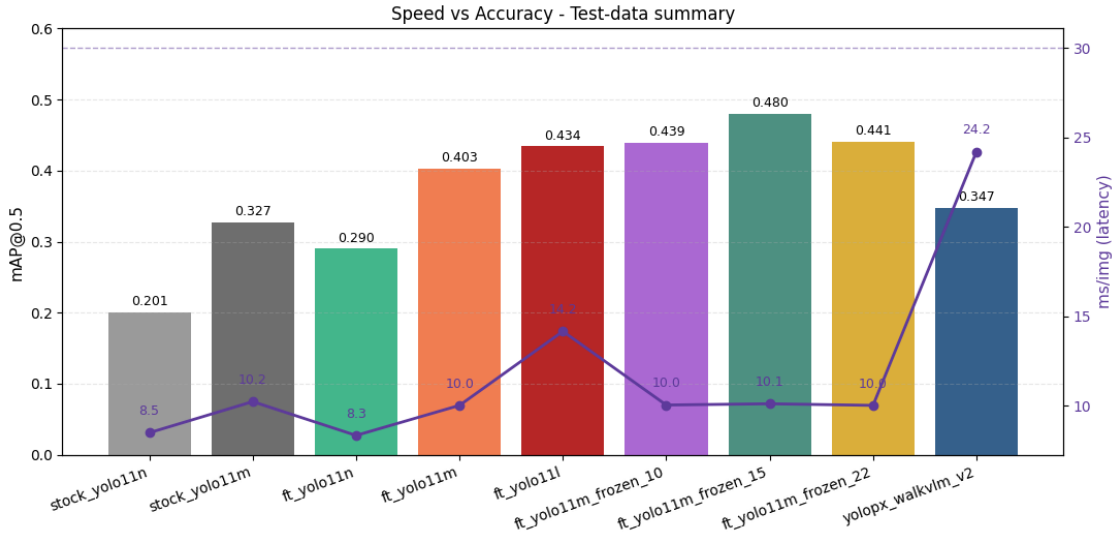


Figure 14: YOLO11m with freeze=15 achieves the highest mAP@0.5 (0.480) while staying at low latency, and is selected as the deployment default.

5.5 Small and Far Hazard Detection

Because early warnings depend on detecting objects before they are close, we separately examine small-object performance. Objects smaller than 32×32 pixels represent the far-hazard bucket.

Fine-tuning improves small-object recall from 0.34 with stock YOLO11m to 0.48 with fine-tuned YOLO11m. This is a 1.41× recall lift for far hazards. However, about 52% of far hazards are still missed, so small-object detection remains an open limitation. Tracking and future depth cues may help stabilize warnings when individual frame detections are weak.

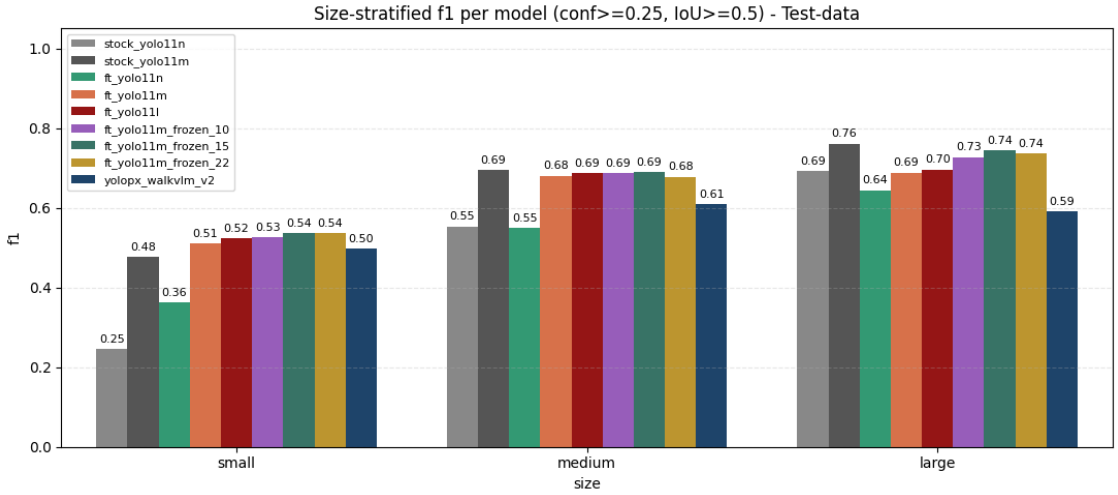


Figure 15: Size-stratified detection performance. Fine-tuning improves recall in the small-object bucket, which corresponds to far hazards. This is important for early warning, but many far hazards are still missed.

5.6 Hyperparameter Tuning and Configuration Selection

We summarize the main hyperparameter search used for model selection in Table 6. Because full grid search was GPU-expensive, we used a one-factor-at-a-time strategy: start from the strongest baseline checkpoint, change one setting, validate every epoch, and keep the best checkpoint by validation mIoU or priority pixel accuracy.

The main result was that conservative fine-tuning worked better than aggressive redesign. The best PIDNet setup kept letterbox preprocessing, batch size 8, baseline augmentation, lower fine-tuning LR, and moderate class weights for navigation classes. Cropping, heavy bottom-region weighting, Dice loss, and batch size 2 often reduced stability. For detection, YOLO11 model size and freezing depth were tuned to balance mAP, recall, and deployment cost.

Table 6: Summary of hyperparameter search used for WalkAware model selection. Segmentation configurations were selected mainly by validation mIoU, with priority pixel accuracy used for region-weighted PIDNet runs. Detection configurations were selected by validation mAP and recall.

Component	**Parameter**	**Values tried**	**Selected setting / outcome**
PIDNet segmentation	Learning rate	0.01, 0.001, 0.0001	0.001 was selected for stable WalkVLM fine-tuning.
PIDNet segmentation	Batch size	2, 8	Batch size 8 was more stable; batch size 2 produced less reliable runs.
PIDNet segmentation	Resize strategy	Letterbox, bottom-center crop, height-fit width crop	Letterbox was retained because it preserved the baseline training distribution.
PIDNet segmentation	Class weights	Uniform; navigation-weighted classes	Moderate weights for Road, Sidewalk, and Crosswalk improved navigation classes.
PIDNet segmentation	Bottom-region weight	Off, moderate, aggressive	Moderate bottom weighting was useful; aggressive weighting hurt global mIoU.
PIDNet segmentation	Fine-tuning length	30, 60 epochs	Early checkpoints performed best, usually around epoch 3.
SegFormer segmentation	Fine-tuning setup	Default LR $6e^{-5}$, batch size 8, letterbox	Conservative fine-tuning improved mIoU but remained below PIDNet.
YOLO11 detection	Model size	YOLO11n, YOLO11m, YOLO11l	Selected based on mAP, recall, and deployment cost.
YOLO11 detection	Freezing depth	No freeze; partial backbone freeze	Selected based on validation mAP and recall stability.

5.7 Fusion Ablation

The fusion ablation tests whether each pipeline stage contributes distinct safety information. We compare three versions on the same input video: the full pipeline, a version without tracking, and a version without both tracking and segmentation.

Table 7: Fusion ablation.

Capability	**Full pipeline**	**No tracker**	**No tracker + no segmentation**
Per-track alerts	✓	No, per-frame only	No, per-frame only
Approach detection	✓	No	No
Collision detection	✓	No	No
Ground / walkable-area check	✓	✓	No
Wall / curb alert	✓	✓	No
User-location signal	✓	✓	No
Warning rate	100%	∼36%	∼4%

Removing tracking reduces the warning rate to about 36% because the system loses object identity, motion history, and approach reasoning. Removing segmentation as well reduces the warning rate to about 4% because the rule engine loses walkable-area context, wall/curb alerts, and user-location signals. This shows that the final system value comes from fusion, not from any single model alone.

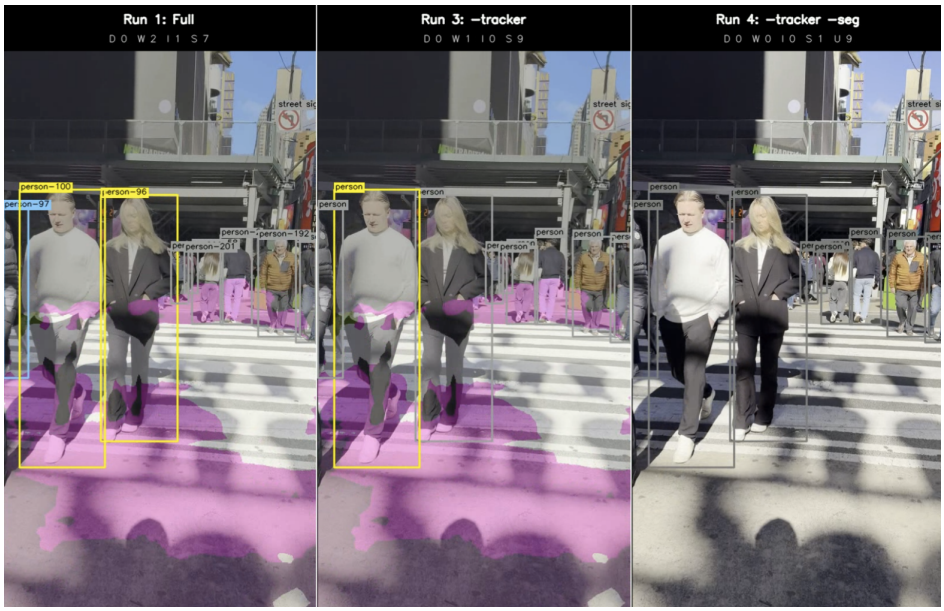


Figure 16: Fusion ablation behavior. The full pipeline produces earlier and more stable alerts. Without tracking, alerts become delayed because object history is lost. Without segmentation, most safety alerts disappear because the system no longer knows the walkable context or user location.

5.8 Inference Speed

We compare the modular pipeline with the YOLOPX single-model alternative.

Table 8: Sustained end-to-end inference speed.

Configuration	Pipeline FPS	Per-frame time
PIDNet segmentation only	3.31	302 ms
SegFormer segmentation only	2.94	340 ms
PIDNet + YOLO11 + fusion	9.3	108 ms
YOLOPX single model, 640 px	26.3	38 ms

The modular pipeline is more accurate and easier to debug, but it runs at only 9.3 FPS, below the real-time target. YOLOPX reaches 26.3 FPS because it uses downscaled input, a single forward pass, and a shared backbone. However, YOLOPX detection mAP is lower than fine-tuned YOLO11, so we keep the modular pipeline as the headline model and treat YOLOPX as a future deployment direction.

5.9 Depth Feasibility

Depth estimation is evaluated as a feasibility study, not as part of the current deployed pipeline. The goal is to test whether monocular depth can turn detections into class + confidence + distance band and support unknown-obstacle warnings.

A small 8-frame pilot produced 21 detections. MiDaS and DepthPro gave similar Near/Mid/Far distributions, but their deployment tradeoffs differ. MiDaS is faster and useful for relative bands, while

DepthPro provides metric-style distances but is much slower. MiDaS runs at about 2–3 seconds per frame on CPU, while DepthPro runs at about 8–12 seconds per frame. Depth has not yet been validated against tape-measured ground truth, so it remains a future extension.

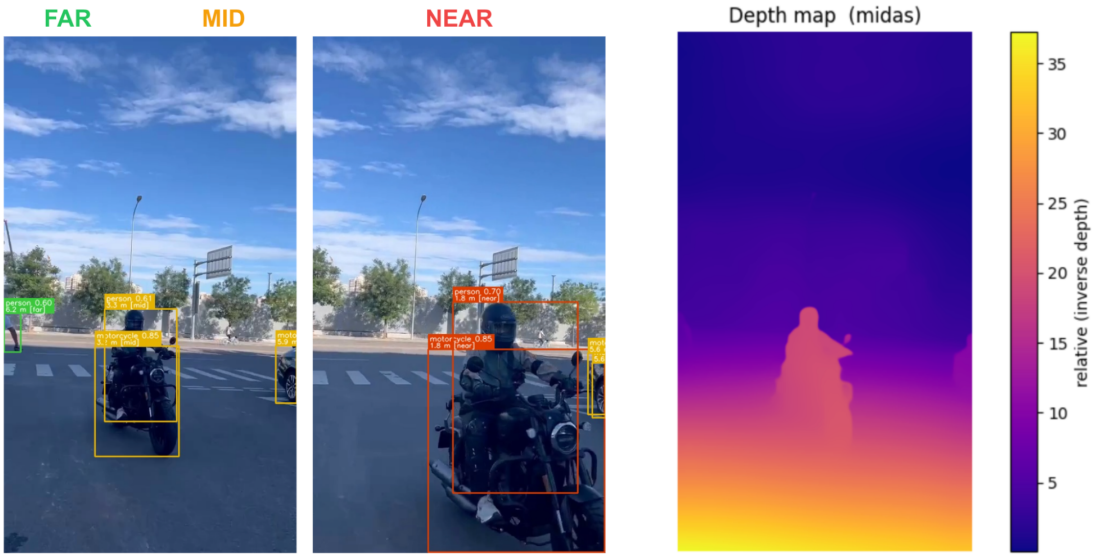


Figure 17: Depth feasibility example. The depth prototype attaches distance estimates and NEAR/MID/FAR bands to detected hazards. This can improve proximity reasoning, but the current results are qualitative and not yet validated with measured ground-truth distance.

5.10 Interpretability Check

We include lightweight interpretability outputs for the fusion stage. The BEV view shows tracked objects around the user, the alert board shows the natural-language warning, and Eigen-CAM highlights the detector regions used for the alert. Qualitatively, the heatmaps focus on pedestrians, vehicles, and sidewalk hazards rather than sky or background, which provides a sanity check for warning decisions.

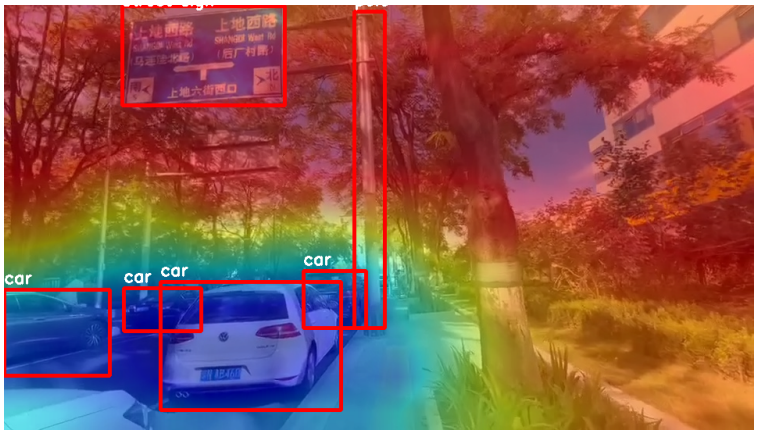


Figure 18: Eigen-CAM detector attention map. The heatmap shows where YOLO11 focuses during detection. Attention is strongest around cars, poles, and street-side hazards, supporting the interpretability check for fusion alerts.

5.11 Summary of Main Findings

Across experiments, three findings define the final project story. First, pedestrian-view fine-tuning improves segmentation, with PIDNet selected as the strongest model for sidewalk-view transfer. Second, YOLO11 fine-tuning recovers walking-specific hazards missing from stock COCO detection, while YOLO11m gives the best deployment tradeoff. Third, fusion is necessary: removing tracking or segmentation sharply reduces usable warning behavior. Depth and YOLOPX are promising, but they remain future directions because depth is not validated and YOLOPX is faster but less accurate.

6 Discussion and Error Analysis

Overall, pedestrian-view adaptation improved the system at each stage. For segmentation, fine-tuning increased mIoU by +4.96 for PIDNet and +2.95 for SegFormer. For detection, YOLO11 fine-tuning recovered five walking-specific classes that stock COCO YOLO missed: pole, trash can, cone, manhole, and street sign. For fusion, the ablation shows that the system depends on combining modules: removing tracking leaves only about 36% of warnings, and removing both tracking and segmentation leaves only about 4%. This confirms that WalkAware's value comes from the full pipeline, not from a single model.

PIDNet outperformed SegFormer mainly because its boundary-aware CNN design is better suited to thin navigation classes such as curb and crosswalk. It also transferred better to pedestrian-view sidewalk scenes, especially in the bottom part of the frame. SegFormer may need more target-domain data to benefit from its transformer architecture.

Several failure modes remain. First, Sidewalk $\rightarrow$ Road confusion still appears in the bottom region of pedestrian-view frames, where sidewalk textures can resemble vehicle-view road surfaces. PIDNet reduces this error compared with SegFormer, but does not eliminate it. Second, Crosswalk remains fragile because it is rare and visually similar to road markings, shadows, and bright pavement. Third, detection fine-tuning improves walking-specific hazards but causes some forgetting on rare COCO classes such as truck and bus. Fourth, small and far hazards are still difficult, even after fine-tuning. Finally, depth estimation is promising but not yet validated with measured ground-truth distance.

The fusion engine improves interpretability, but it also depends on the quality of upstream model outputs. If segmentation labels the ground incorrectly, the rule engine may assign the wrong context. For example, if a crosswalk is mislabeled as sidewalk, the system may still correctly treat it as a pedestrian surface, but it may not trigger the crosswalk-specific rule. If sidewalk is mislabeled as road, a pedestrian-surface hazard may be under-alerted.

Detection errors also propagate into fusion. If YOLO misses a small cone, pole, or fast bicycle, the fusion engine cannot generate an object-level alert for it. Tracking errors can also affect motion reasoning. If the tracker loses an object ID, the system may lose its motion history and approach estimate.

Fusion v2 also relies on approximate camera assumptions. Its projective distance bands assume a fixed camera orientation and approximate handheld height. Therefore, the distance values should be interpreted as safety bands, not exact measurements. A different phone, camera height, or mounting position may require recalibration.

Despite these limitations, fusion provides an important bridge between perception and assistive navigation. It turns model outputs into safety decisions that are understandable, adjustable, and connected to the user's walking context.

Depth and YOLOPX are important future directions, but they are not part of the final deployed pipeline. Depth could improve proximity reasoning and help detect unknown obstacles, but current results are qualitative and too slow for real-time use. YOLOPX is much faster than the modular pipeline, but its detection accuracy is lower than fine-tuned YOLO11.

With more time, we would collect more pedestrian-view labels for crosswalk, curb, street sign, truck, bus, and motorcycle; add temporal consistency to reduce segmentation flicker; validate DepthPro with measured distances; and adapt the system to NYC sidewalk scenes. Engineering work such as input downscaling, ONNX/TensorRT export, and running segmentation less frequently could help move the modular pipeline closer to real time.

7 Conclusion

WalkAware demonstrates a modular pipeline for pedestrian-view sidewalk understanding. The system combines semantic segmentation, object detection, tracking, and rule-based fusion to produce interpretable safety alerts: SAFE, INFO, WARNING, and DANGER. Depth estimation was tested separately as a promising future extension, but it is not part of the current deployed pipeline.

The results show that pedestrian-view adaptation improves performance. For segmentation, the best PIDNet fine-tune reaches 93.36% pixel accuracy and 73.49% mIoU, improving Road and Sidewalk IoU over the frozen baseline. PIDNet also transfers better than SegFormer on the held-out sidewalk-view set. For detection, the selected YOLO11m model with freeze=15 reaches 0.480 mAP@0.5 at 9.95 ms per image, outperforming both stock COCO YOLO and the larger YOLO11l. Fine-tuning also recovers five walking-specific hazard classes that stock COCO YOLO does not cover well.

The fusion ablation shows that the full system depends on combining modules. Removing tracking reduces the warning rate to about 36%, while removing both tracking and segmentation reduces it to about 4%. This suggests that object identity, motion history, walkable-area context, and rule-based reasoning all contribute to safety-alert generation.

Several limitations remain. The system still confuses sidewalk and road in some bottom-frame regions. Crosswalk performance remains unstable because the class is rare and visually ambiguous. Detection fine-tuning improves walking-specific hazards but weakens some rare COCO classes, such as truck and bus. Depth is not yet validated with measured ground truth, and the modular pipeline runs at 9.3 FPS, below the real-time target.

Future work should focus on targeted pedestrian-view labeling for weak classes, NYC-specific adaptation, temporal consistency for segmentation, validated metric depth, and faster deployment through model optimization or a stronger shared-backbone architecture such as YOLOPX.

References

[1] World Health Organization. "Blindness and vision impairment." WHO Fact Sheet. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

[2] Z. Yuan, T. Zhang, Y. Deng, J. Zhang, Y. Zhu, Z. Jia, J. Zhou, and J. Zhang, "WalkVLM: Aid Visually Impaired People Walking by Vision Language Model," arXiv preprint arXiv:2412.20903, 2024.

[3] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[4] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[5] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A Real-Time Semantic Segmentation Network Inspired by PID Controllers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 19529–19539.

[6] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[7] Ultralytics, "YOLO11: Object Detection, Segmentation, Pose Estimation, and Image Classification," Ultralytics Documentation, 2024. Available: https://docs.ultralytics.com/models/yolo11/

[9] H. Kweon and K.-J. Yoon, "From SAM to CAMs: Exploring Segment Anything Model for Weakly Supervised Semantic Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[10] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment Anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[11] D. Vu, B. Ngo, and H. Phan, "HybridNets: End-to-End Perception Network," arXiv preprint arXiv:2203.09035, 2022.

[12] J. Zhan, Y. Luo, C. Guo, Y. Wu, J. Meng, and J. Liu, "YOLOPX: Anchor-Free Multi-Task Learning Network for Panoptic Driving Perception," *Pattern Recognition*, vol. 148, 2024, Art. no. 110152.

[13] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[14] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust Associations Multi-Pedestrian Tracking," arXiv preprint arXiv:2206.14651, 2022.

[15] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[16] A. Bochkovskii, A. Delaunoy, H. Germain, M. Santos, Y. Zhou, S. R. Richter, and V. Koltun, "Depth Pro: Sharp Monocular Metric Depth in Less Than a Second," arXiv preprint arXiv:2410.02073, 2024.

A Fusion Layer Implementation Details

This appendix provides the implementation details behind the Fusion v2 safety logic summarized in Section 4.5. The main report keeps the fusion description at the system level, while this appendix documents the per-object signals, equations, and debugging outputs.

A.1 Per-Object Signals

Before assigning an alert, the fusion engine computes several signals for each tracked object: object class group, ground label, proximity, motion state, time-to-contact, lateral collision risk, horizontal position, and reliability.

Object class group. The detector output is mapped into safety-relevant groups. Heavy vehicles such as *car*, *bus*, and *truck* are treated as the highest-risk group. Light vehicles such as *bicycle* and *motorcycle* are treated as fast pedestrian-space hazards. Static obstacles such as *pole*, *traffic cone*, *trash can*, and *fire hydrant* are usually safe at a distance but can become important when close to the user.

Ground label. The ground label is estimated from segmentation pixels near the bottom edge of the object bounding box. For a detected box

$$b_i = (x_1, y_1, x_2, y_2), \tag{1}$$

the ground label is selected by majority vote:

$$g_i = \operatorname{mode}\left(S[x_1 : x_2,\ y_2 - \delta : y_2]\right), \tag{2}$$

where S is the segmentation map and δ is a small vertical window near the object bottom.

The system marks an object as being on a pedestrian surface when:

$$\text{on_ped_surface}(g_i) = \begin{cases} 1, & g_i \in \{\text{Sidewalk}, \text{Crosswalk}\}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

To reduce flicker, the engine can smooth ground labels over a short temporal window:

$$\hat{g}_i^t = \operatorname{mode}\left(g_i^{t-k}, \ldots, g_i^t\right). \tag{4}$$

Proximity. Fusion v1 estimated proximity using image-plane signals such as bounding-box bottom position and area ratio:

$$\rho_i = \frac{y_2}{H}, \tag{5}$$

$$a_i = \frac{(x_2 - x_1)(y_2 - y_1)}{WH}. \tag{6}$$

Fusion v2 uses projective estimates with class-specific size priors. A height-based distance estimate is:

$$Z_h = \frac{f_y H_c}{h_i}, \tag{7}$$

where $h_i = y_2 - y_1$, f_y is the vertical focal length in pixels, and H_c is the class-specific physical height prior.

A bottom-position estimate is:

$$Z_\rho = \frac{f_y h_{\text{cam}}}{H(\rho_i - 0.5)}, \tag{8}$$

valid only when:

$$\rho_i > 0.5. \tag{9}$$

For some classes, an area-based estimate is also used:

$$Z_A = \sqrt{\frac{f_x f_y H_c W_c}{A_i}}, \tag{10}$$

where:

$$A_i = (x_2 - x_1)(y_2 - y_1). \tag{11}$$

The final proximity band is selected conservatively:

$$p_i = \min_{\text{distance}} \left(\text{bin}(Z_h), \text{bin}(Z_\rho), \text{bin}(Z_A)\right). \tag{12}$$

Motion and approach state. Fusion v1 detected approach using bounding-box growth:

$$\text{approach}_i = \left(\frac{A_i^t}{A_i^{t-k}} > \tau_A\right) \wedge \left(\frac{y_{2,i}^t}{y_{2,i}^{t-k}} > \tau_y\right). \tag{13}$$

Fusion v2 uses the tracker's Kalman state. If $\dot{h}_i$ is the estimated height velocity, the adaptive threshold is:

$$\epsilon_i = k_\epsilon \sigma_{\dot{h}_i}. \tag{14}$$

The motion regime is:

$$m_i = \begin{cases} \text{APPROACH}, & \dot{h}_i > \epsilon_i, \\ \text{RECEDING}, & \dot{h}_i < -\epsilon_i, \\ \text{STATIC}, & |\dot{h}_i| \le \epsilon_i, \\ \text{UNKNOWN}, & \text{motion evidence is unreliable.} \end{cases} \tag{15}$$

A.2 Time-to-Contact and Collision Risk

Fusion v2 estimates time-to-contact using multiple geometric signals. A height-based estimate is:

$$\tau_h = \frac{h_i}{\dot{h}_i}. \tag{16}$$

An area-based estimate is:

$$\tau_A = \frac{2A_i}{\dot{A}_i}. \tag{17}$$

A bottom-position estimate is:

$$\tau_\rho = \frac{\rho_i - 0.5}{\dot{\rho}_i}. \tag{18}$$

Valid estimates are combined using inverse-variance weighting:

$$\tau_{\text{fused}} = \frac{\sum_j w_j \tau_j}{\sum_j w_j}, \tag{19}$$

where:

$$w_j = \frac{1}{\sigma_{\tau_j}^2}. \tag{20}$$

The fused uncertainty is:

$$\sigma_{\tau_{\text{fused}}}^2 = \frac{1}{\sum_j w_j}. \tag{21}$$

Fusion v2 also estimates lateral collision risk. Let:

$$x_{\text{img}} = c_x - \frac{W}{2}, \tag{22}$$

where c_x is the bounding-box center and W is the image width. The normalized lateral position is:

$$\xi_i = \frac{x_{\text{img}}}{h_i}. \tag{23}$$

The future lateral position at time-to-contact is:

$$\xi_i(\tau) = \xi_i + \dot{\xi}_i \tau_{\text{fused}}. \tag{24}$$

The object is considered imminent only when both time-to-contact and lateral overlap suggest risk:

$$\text{imminent}_i = \left(\tau_{\text{fused}} - c\sigma_{\tau_{\text{fused}}} < \tau_{\text{thresh}}\right) \wedge \left(|\xi_i(\tau)| < \xi_{\text{thresh}}\right). \tag{25}$$



Figure 19: Time-to-contact and lateral collision reasoning. Fusion v2 marks an object as imminent only when it is approaching soon enough and overlaps with the user's walking corridor.

A.3 Reliability and Output Trace

Objects near the image boundary are difficult to interpret because the bounding box may be partially visible. Fusion marks an object as edge-clipped when:

$$\text{edgeclip}_i = (x_1 < \eta) \vee (y_1 < \eta) \vee (x_2 > W - \eta) \vee (y_2 > H - \eta), \tag{26}$$

where η is a small edge margin.

If the evidence is unreliable and no stable history exists, the system can cap the alert:

$$\text{alert}_i = \min_{\text{risk}}(\text{alert}_i, \text{INFO}). \tag{27}$$

The output trace includes the final alert, fired rule, proximity bin, motion regime, time-to-contact estimate, lateral collision signal, and optional bird's-eye-view coordinates. These trace fields make it easier to debug why Fusion v1 and Fusion v2 produce different behavior.

A.4 Fusion v1 and Fusion v2 Comparison

The project developed two versions of the fusion engine. Fusion v1 used simpler image-plane heuristics. Fusion v2 keeps the same alert taxonomy and rule-based structure, but improves the signals used by the rules.

Table 9: Comparison of Fusion v1 and Fusion v2.

Component	**Fusion v1**	**Fusion v2**
Proximity	Bounding-box y-position and area ratio	Projective distance bands using box geometry and class priors
Approach	Box area growth and bottom-y growth	Tracker motion state from Kalman estimates
Collision risk	Implicit through approach and proximity	Explicit time-to-contact and lateral collision predicate
Distance representation	CLOSE / MEDIUM / FAR	NEAR / MID / FAR / UNKNOWN
Bird's-eye view	Not available	Optional debug visualization
Alert logic	Rule ladder	Same rule structure with stronger signals
Reliability handling	Edge clipping and history	Edge clipping, signal gates, uncertainty-aware motion

Fusion v1 was useful because it was simple and easy to debug. However, it did not estimate metric distance, time-to-contact, or lateral collision risk. Fusion v2 addresses these limitations by adding projective geometry and tracker-based motion reasoning while keeping the final decision rule-based and explainable.

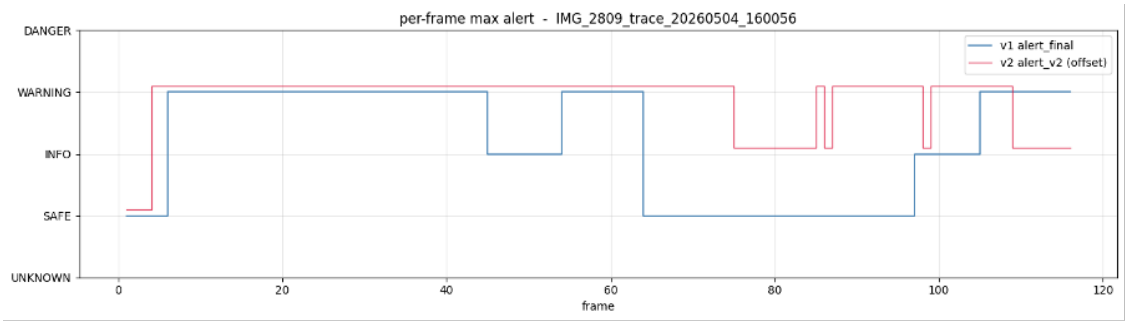


Figure 20: Comparison of Fusion v1 and Fusion v2. Fusion v1 used image-plane heuristics such as bounding-box area growth and vertical position. Fusion v2 keeps the same interpretable alert taxonomy but replaces these heuristics with projective proximity estimates, tracker-based motion regimes, time-to-contact, and lateral collision checks.

B YOLOPX Architecture and Training Details

YOLOPX was evaluated as a deployment-oriented alternative to the modular PIDNet + YOLO11 pipeline. The goal was not to replace the headline system immediately, but to test whether a shared-backbone model could move WalkAware closer to real-time performance.

B.1 Architecture

YOLOPX is a multi-task perception model with a shared feature extraction backbone and separate output heads for object detection and segmentation. For WalkAware, the detection head is adapted to predict the 12 walking-hazard classes used by YOLO11: person, bicycle, car, motorcycle, bus, truck, fire hydrant, street sign, pole, trash can, cone, and manhole.

The segmentation head is adapted to the seven-class WalkAware scene taxonomy: Curb, Crosswalk, Road, Sidewalk, Building, Vegetation, and Sky.

The high-level flow is:

$$I_t \rightarrow \text{Shared Backbone} \rightarrow \{\text{Detection Head}, \text{Segmentation Head}\}, \tag{28}$$

where I_t is the input frame at time t.

The detection output is:

$$D_t = \{(b_i, c_i, s_i)\}_{i=1}^{N}, \tag{29}$$

where b_i is a bounding box, c_i is the predicted class, and s_i is the detection confidence.

The segmentation output is:

$$S_t \in \{1, \ldots, 7\}^{H \times W}. \tag{30}$$

These outputs can be passed to the same fusion logic used by the modular pipeline:

$$(D_t, S_t, T_t) \rightarrow \text{Fusion v2} \rightarrow A_t, \tag{31}$$

where T_t represents tracker outputs and A_t represents per-object safety alerts.

B.2 Training Objective

YOLOPX is trained with a multi-task objective:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{det}} \mathcal{L}_{\text{det}} + \lambda_{\text{seg}} \mathcal{L}_{\text{seg}}, \tag{32}$$

where $\mathcal{L}_{\text{det}}$ is the detection loss, $\mathcal{L}_{\text{seg}}$ is the segmentation loss, and λ_{det} and λ_{seg} balance the two tasks.

The detection loss includes box regression, objectness, and class prediction:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{obj}} + \mathcal{L}_{\text{cls}}. \tag{33}$$

The segmentation loss is computed over the seven-class scene map:

$$\mathcal{L}_{\text{seg}} = -\sum_{p \in \Omega} w_{y_p} \log P(y_p \mid p), \tag{34}$$

where p is a pixel, Ω is the set of labeled pixels, y_p is the ground-truth class at pixel p, and w_{y_p} is the class weight.

The shared-backbone design improves speed, but it can also create competition between tasks. In our experiments, YOLOPX was faster than the modular pipeline, but it did not match the detection quality of the dedicated fine-tuned YOLO11 model.