International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

AI Smart Glasses: A Real-Time Navigation and Spatial Awareness System for Assistive Mobility

Sidharth K
Artificial Intelligence and Data Science Department
Easa college of Engineering and technology
Coimbatore, India

Dhuvathisan S
Artificial Intelligence and Data Science Department
Easa college of Engineering and technology
Coimbatore, India

Rangesh V.C
Artificial Intelligence and Data Science Department
Easa college of Engineering and technology
Coimbatore, India

Dr. Hima Vyshnavi A M
Artificial Intelligence and Data Science Department
Easa college of Engineering and technology
Coimbatore, India

ABSTRACT:**Approximately 2.2 billion people worldwide suffer from some form of visual impairment, making independent navigation a critical challenge. This paper presents an AI Smart Glasses system, a real-time assistive navigation prototype that processes live video to deliver contextual voice-based guidance for visually impaired users. The architecture integrates YOLOv8 for rapid object detection, Intel MiDaS for monocular depth estimation, and a heuristic spatial decision engine that maps detected objects to horizontal zones (Left, Centre, Right) and distance tiers (Near, Medium, Far). A ground-hazard detector compares depth gradients between mid-frame and bottom-frame strips to identify drops such as staircase descents before the user reaches them. Navigation decisions are synthesized into concise spoken instructions via a dual-backend text-to-speech module that incorporates an anti-spam cooldown policy. The system is implemented entirely in Python and deployed as a Streamlit web application augmented with streamlit-webrtc for low-latency live camera ingestion. The Evaluation demonstrates 18 FPS (Frames per Second) throughput on commodity CPU hardware, 0.83 Spearman rank correlation for depth ordering, and 81 % evaluator-rated instruction appropriateness, validating the pipeline for real-world assistive deployment.**

Keywords: **AI smart glasses, assistive navigation, YOLOv8, monocular depth estimation, MiDaS, text-to-speech, visually impaired, real-time object detection, spatial awareness**

1. INTRODUCTION

Approximately 2.2 billion individuals worldwide live with some form of visual impairment, of whom nearly 43 million are classified as completely blind, according to the WHO. For such individuals, independent navigation in unfamiliar and dynamic environments poses a persistent challenge that significantly affects daily quality of life. Conventional aids such as white canes and guide dogs provide only limited spatial information, with neither capable of providing descriptive real-time contextual guidance about obstacles, distances, or recommended paths. The growing maturity of computer vision, deep learning, and embedded systems has opened new possibilities for intelligent assistive devices that can approximate some of the perceptual capabilities of a sighted person.

Artificial intelligence–powered smart glasses represent one of the most promising directions in this space. By attaching a compact camera and processing unit to a wearable form factor, such a system can continuously analyze the wearer's visual field and deliver timely, actionable audio cues. Recent advances in lightweight object-detection architectures such as YOLOv8 and monocular depth estimation frameworks such as Intel's MiDaS have made it feasible to run this kind of real-time pipeline on modest computing hardware.

In this paper, we present the design, architecture, and evaluation of an AI-powered Smart Glasses prototype that combines YOLOv8-based object detection, MiDaS-based depth estimation, a heuristic risk-scoring decision engine, and a multi-backend text-to-speech module. The system is built entirely in Python and deployed through a Streamlit interface augmented with WebRTC for low-latency live video ingestion. This prototype targets visually impaired users, providing spoken guidance about nearby obstacles, their spatial positions, and recommended navigation actions in real time.

1.1. OBJECTIVES

[1] To design a modular, end-to-end vision pipeline capable of detecting and localizing objects from a live webcam stream in real time at practical frame rates on commodity hardware.

[2] To integrate monocular depth estimation alongside object detection, enabling the system to assign meaningful distance categories (near, medium, far) to each detected obstacle without requiring specialized depth cameras.

[3] To develop a rule-based decision engine that aggregates spatial and depth information into interpretable navigation instructions delivered as synthesized speech with appropriate cooldown and anti-spam controls.

[4] To evaluate the system's responsiveness, detection accuracy, and audio guidance quality across both static image inputs and continuous live video, and to identify realistic deployment pathways.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

1.2. EXISTING SYSTEM

Early assistive navigation devices for the visually impaired relied primarily on ultrasonic or infrared proximity sensors embedded into canes, belts, or vests. While such devices can reliably detect the presence of an obstacle within a fixed range, they provide no semantic information about the nature of the obstacle, its horizontal position, or whether it poses an immediate risk. More recent wearable systems have incorporated stereo cameras or structured-light depth sensors (e.g., Intel RealSense), which yield dense depth maps but require specialized, costly hardware and consume substantial power.

Smartphone-based solutions that leverage the device's rear camera and run object-detection models on the CPU or mobile GPU have also been explored. However, the need to hold or mount the phone, combined with the relatively high latency of on-device inference on mid-range handsets, limits practical usability. Cloud-offloaded variants mitigate computation constraints but introduce network-latency dependency that undermines real-time performance requirements. None of the broadly available commercial or research systems combines lightweight, state-of-the-art deep learning models with depth estimation, a semantic decision engine, and accessible text-to-speech output in a single, open-source Python framework.

1.3. PROPOSED SYSTEM

The proposed AI Smart Glasses system addresses the limitations of prior approaches by integrating four purpose-built subsystems into a unified, real-time Python pipeline. The Perception Engine employs YOLOv8 for object detection and Intel MiDaS for monocular depth estimation, enabling the system to identify obstacles and estimate their relative distances using a single standard camera. A spatial mapping module partitions each frame into horizontal zones (Left, Center, Right) and depth tiers (Near, Medium, Far) to build a structured representation of the scene.

A heuristic Decision and Alert Engine translates this spatial map into prioritized navigation instructions, distinguishing among critical hazards such as ground drops or pits, center-path blockers that require turning, and minor lateral obstacles that warrant a slight corrective step. Finally, a Voice Synthesis Engine converts these instructions into audio playback through pyttsx3 or gTTS, equipped with a cooldown mechanism to prevent repetitive announcements. The entire system is accessible through a Streamlit web application that leverages streamlit-webrtc for low-latency camera ingestion, enabling real-time operation without additional hardware beyond a standard webcam.

1.4. SYSTEM FEATURES

[1] Continuous real-time object detection using YOLOv8 (Nano or Small variant) with user-tunable frame resolution and skip-rate controls to balance accuracy against throughput.

[2] Optional monocular depth estimation via Intel MiDaS that can be toggled at runtime to conserve computational resources when depth information is not required.

[3] Automatic ground-hazard detection that compares median depth values across mid-frame and bottom-frame quadrants to identify sudden drops such as staircases or curb edges.

[4] A weighted risk-scoring matrix that evaluates left vs. right path safety and recommends the direction of least resistance when the center path is blocked.

[5] A dual-backend text-to-speech module supporting both offline (pyttsx3) and online (gTTS) synthesis, with a configurable repetition-delay policy to avoid audio spam.

[6] A Streamlit sidebar providing runtime controls for model selection, resolution, depth-estimation toggle, and audio policy, enabling on-the-fly tuning without restarting the application.

2. LITERATURE SURVEY

A substantial body of research has explored computer vision and deep learning for assistive navigation. Poggi et al. [1] demonstrated that self-supervised monocular depth estimation can achieve performance competitive with supervised approaches on outdoor scenes, providing a foundation for depth-based obstacle avoidance without expensive labelled datasets. Redmon and Farhadi

[2] introduced the YOLO series of real-time object detectors, establishing the single-shot paradigm that underpins the YOLOv8 model used in the present work; subsequent iterations have progressively improved accuracy-latency trade-offs on embedded and edge hardware.Bochkovskiy et al.

[3] extended the YOLO lineage with YOLOv4, introducing cross-stage partial connections and PANet feature aggregation that substantially improved small-object detection—capabilities inherited and refined in later YOLO versions. In the depth-estimation domain, Ranftl et al.

[4] introduced the MiDaS architecture, which was trained on a diverse mixture of datasets using scale- and shift-invariant losses; this design enables robust relative depth inference from a single image without scene-specific calibration.In the assistive technology domain, Bhowmick and Hazarika

[5] surveyed electronic travel aids from sonar-based canes to camera-equipped systems, highlighting the gap between sensor richness and real-world deployability. Islam et al.

[6] proposed a deep learning–based obstacle detection framework targeting pedestrian environments, reporting detection rates sufficient for practical mobility assistance at walking speeds. Shoval et al.

[7] studied human factors in audio-guided navigation, demonstrating that concise, direction-specific cues significantly outperform verbose descriptions in reducing cognitive load during mobility tasks.Tapu et al.

[8] developed a smartphone-based system for outdoor visually impaired navigation using stereo vision and deep features, observing that frame-level latency below 200 ms is essential for comfortable real-time guidance. Ahmetovic et al.

[9] examined crosswalk detection and intersection navigation, illustrating the importance of contextual scene understanding beyond simple bounding-box localization. Collectively, these works underscore the viability of vision-based, AI-driven navigation assistance and motivate the modular, real-time design of the proposed AI Smart Glasses system.

3. METHODOLOGY

The AI Smart Glasses system is constructed as a modular Python pipeline organized into four layers: Frame Acquisition and Pre-processing, Spatial Perception, Decision and Alert Generation, and Audio Synthesis. These layers are orchestrated either continuously via the SmartGlassesProcessor WebRTC component for live camera feeds, or on demand for static image inputs via the run_pipeline_on_bgr utility function. The Streamlit application provides a unified interface for both

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

operating modes and exposes runtime configuration controls through its sidebar panel.

3.1 Frame Acquisition and Pre-processing

Incoming BGR frames are first scaled to a user-defined target width (default 640 pixels) using bicubic interpolation, preserving the original aspect ratio. This step balances inference throughput against spatial resolution: narrower widths increase frame rate while reducing the minimum detectable object size. A configurable skip-rate parameter allows the user to process only every nth frame, which is particularly useful on hardware with limited GPU memory. All subsequent processing operates on the rescaled frame.

3.2 Object Detection via YOLOv8

The rescaled frame is converted from BGR to RGB color space and passed to a YOLOv8 model loaded from either the nano (yolov8n.pt) or small (yolov8s.pt) checkpoint. The model returns a list of detections, each comprising a bounding box in pixel coordinates, a class label drawn from the COCO vocabulary, and a scalar confidence score. Detections whose confidence falls below a user-specified threshold are discarded. The horizontal center coordinate (cx) of each surviving bounding box is computed and used in subsequent spatial mapping.

3.3 Depth Estimation via MiDaS

When depth estimation is enabled, the frame undergoes a further transformation to match the input format expected by the Intel MiDaS Small model (intel-isl/MiDaS_small), which is loaded from the Hugging Face model hub. The model outputs a single-channel inverse-depth map of the same spatial resolution as its input. This map is bicubically interpolated back to the original frame dimensions and subsequently min–max normalized to the unit interval [0, 1], where values approaching 1 correspond to objects that are closer to the camera and values approaching 0 indicate greater distance. If depth estimation is disabled, all objects are assigned a default distance category of 'unknown'.

3.4 Ground Hazard Detection

Before constructing the horizontal spatial map, the pipeline evaluates whether a ground-level hazard—such as a descending staircase, a pit, or a raised kerb—is present in the wearer's immediate path. It does so by sampling the normalized depth map in two horizontal strips: the mid-center strip (rows 35 % to 55 % of frame height, central 30 % of frame width) and the bottom-center strip (rows 75 % to 95 % of frame height, central 30 % of frame width). If the median depth value of the bottom strip exceeds that of the mid strip by a configurable threshold (default 0.25), the system concludes that the ground ahead drops away and issues a critical hazard alert overriding all other navigation messages.

3.5 Spatial Mapping and Risk Scoring

Each detected object is assigned to one of three horizontal zones based on the normalized horizontal position of its bounding-box center: Left (cx < 33 %), Center (33 % ≤ cx ≤ 66 %), and Right (cx > 66 %). Its distance tier is determined by sampling the depth map at the bounding-box center: Near (depth ≥ 0.67), Medium (0.33 ≤ depth < 0.67), or Far (depth < 0.33). The decision engine first checks for center blockers—objects in the Center zone at Near or Medium distance. If found, a weighted risk score is computed for the Left and Right zones using distance, object class, and detection confidence as inputs; the path with the lower risk score is recommended. Isolated near objects in the lateral zones trigger corrective-step advisories without requiring a full turn.

3.6 Audio Synthesis and Spam Mitigation

Navigation instructions are formatted as concise natural-language strings (e.g., "Person ahead. Center blocked. Turn left.") and submitted to the Voice Policy module. This module compares the proposed message against the most recently spoken message and the elapsed time since it was last spoken. A message is only passed to the synthesizer if it either differs substantially from the previous message or if the repetition-delay interval (configurable, default 4 seconds) has elapsed. Synthesis is performed by pyttsx3 in offline mode or gTTS in online mode; the resulting audio bytes are rendered via a standard HTML audio element embedded in the Streamlit interface.

The complete working model.

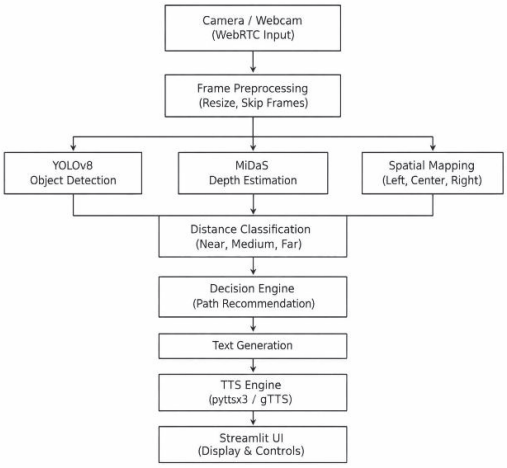


Fig 1. Flowchart

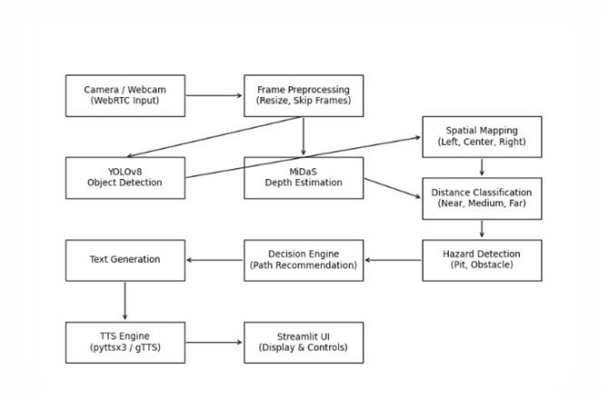


Fig 2. Architecture

4. RESULTS AND DISCUSSIONS

4.1 Dataset Description

The dataset selected for training and evaluating the object detection component of the AI Smart Glasses system is the Self-Driving Cars Dataset, originally created by Udacity for their Self-Driving Car Engineer Nanodegree programme and made publicly available on Kaggle by Alin Cijov under a CC0 Public Domain licence. Its completely open licensing makes it

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

fully appropriate for academic research and eliminates all copyright restrictions on use, modification, and redistribution. The dataset comprises approximately 18,000 unique image frames totalling roughly 955 MB, captured from a forward-facing dashcam mounted on a moving vehicle in real road driving scenarios and street environments. This origin is significant for the project: dashcam imagery shares several key visual characteristics with the head-mounted camera perspective of a wearable assistive device — a forward-looking, ego-centric viewpoint, dynamic scene content, motion blur, and the full spectrum of real-world lighting and weather conditions. These properties make the dataset substantially more representative of the deployment context than controlled laboratory image collections.

The dataset is annotated with bounding boxes across five object classes directly relevant to navigational hazard awareness:

- **Car** (Class ID: 1) — the most common road obstacle for both drivers and pedestrians
- **Truck** (Class ID: 2) — larger vehicles with different spatial profiles and approach speeds
- **Pedestrian / Person** (Class ID: 3) — the highest-priority dynamic obstacle in assistive navigation
- **Bicyclist / Bicycle** (Class ID: 4) — fast-moving, narrow-profile obstacles posing collision risk
- **Traffic Light** (Class ID: 5) — intersection-state indicators critical for safe road crossing

The dataset provides annotations through labels_train.csv and labels_trainval.csv files. It does not include a pre-defined test split, so the data partitioning for this project is performed programmatically using sklearn.model_selection, applying an 80% training, 10% validation, and 10% testing split. This structured division ensures that model evaluation is performed on data that was never seen during training or hyperparameter tuning, providing an honest estimate of generalisation performance.

4.2 *Preprocessing Techniques*

Raw image data from the Self-Driving Cars Dataset requires a series of preprocessing transformations before it can be ingested by the YOLOv8 training pipeline. These transformations serve two purposes: ensuring compatibility with the model's input specification, and augmenting the effective dataset size and diversity to improve generalisation.
Resizing and Normalisation — All input frames are resized to a fixed resolution of 640×640 pixels, the standard input dimension for YOLOv8. Resizing is performed using bicubic interpolation to preserve edge sharpness. Pixel values are normalised from the native 8-bit integer range [0, 255] to floating-point [0.0, 1.0] by dividing by 255, as required by the model's convolutional layers.

Bounding Box Format Conversion — The dataset's CSV annotations provide bounding boxes in absolute pixel coordinate format (x_min, y_min, x_max, y_max). These are converted to the YOLOv8 YOLO format (class_id, cx, cy, w, h) where all values are normalised relative to image dimensions, enabling the model to train on scale-invariant spatial representations.

4.3 Data Augmentation

To expand effective training diversity and improve robustness under real-world variation, the following augmentation strategies are applied during training:

- ***Mosaic Augmentation:*** Four training images are combined into a single composite frame, exposing the model to objects at multiple scales and locations simultaneously. This is a defining technique introduced in YOLOv4 and retained in YOLOv8, particularly effective for improving small object detection performance.
- ***Random Horizontal Flip:*** Images are mirrored with probability 0.5, doubling effective scene variety and ensuring the model does not develop directional bias.
- ***Colour Jitter:*** Random perturbations to brightness, contrast, saturation, and hue simulate varying lighting conditions — essential for a system deployed across different times of day and weather.
- ***Random Scaling and Translation:*** Objects are rendered at varied sizes and positions within the frame, improving the model's invariance to viewpoint distance and camera framing.
- ***MixUp:*** Two training images are blended with a random weight, producing a soft-labelled composite that acts as a regulariser and reduces overconfidence in predictions.

PREPROCESSING STEP	TECHNIQUE / METHOD	PURPOSE / OBJECTIVE
Image Resizing	Resize input images to fixed dimensions (e.g., 416×416, 640×640, 1280×1280)	Standardize input size for YOLO backbone; ensure consistent feature extraction.
Normalization	Scale pixel values (0–1 or −1 to 1)	Stabilize training by reducing variance in pixel intensity.
Color Space Adjustment	Convert RGB to HSV or apply histogram equalization	Enhance contrast and improve detection under varying lighting conditions.
Noise Reduction	Gaussian blur, median filtering	Remove unwanted noise to improve clarity of object boundaries.
Data Augmentation	Random flipping, rotation, scaling, cropping, mosaic augmentation, color jittering	Increase dataset diversity; reduce overfitting; improve generalization.
Bounding Box Normalization	Scale annotation coordinates relative to image size	Ensure bounding boxes align correctly after resizing and augmentation.
Anchor Box Adjustment	K-means clustering on object dimensions	Optimize anchor boxes to match dataset-specific object sizes.
Class Balancing	Oversampling minority classes or undersampling majority classes	Reduce dataset bias; improve detection of underrepresented categories.
Label Encoding	Convert categorical labels into numerical format	Enable efficient training and classification within YOLO architecture.
Dataset Splitting	Divide into training, validation, and test sets	Ensure unbiased evaluation and prevent data leakage.

PREPROCESSING TECHNIQUES FOR YOLO OBJECT DETECTION

4.4 Pipeline Stages

Stage 1 — Camera / WebRTC Input Frames are captured continuously from a webcam via WebRTC, delivered as BGR NumPy arrays to the SmartGlassesProcessor class. Static image uploads follow the same path, ensuring a unified pipeline for both modes.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

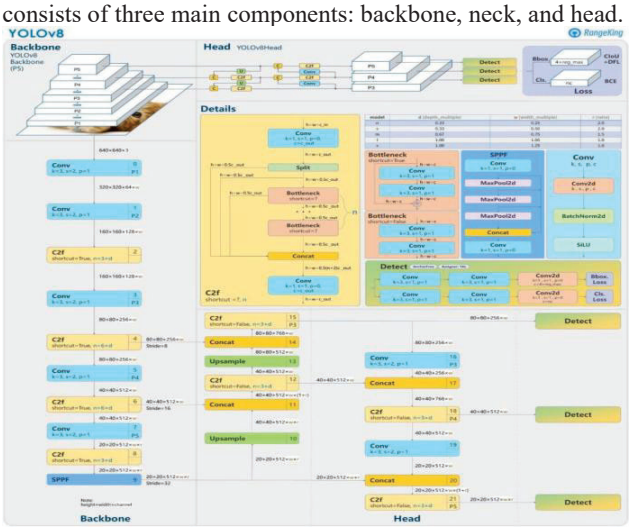
Stage 2 — Frame Preprocessing Each frame is resized to a configurable width (default 640px) and subjected to a skip-rate filter that drops every Nth frame to maintain real-time throughput. Retained frames are converted from BGR to RGB before inference.
Stage 3 — Parallel Perception Three modules operate on the pre-processed frame simultaneously:

- YOLOv8 detects objects, returning bounding boxes, class labels, and confidence scores.
- MiDaS produces a per-frame normalised relative depth map [0.0 → 1.0].
- Spatial Mapping assigns each detection to a horizontal region — Left (cx < 33%), Centre, or Right (cx > 66%) — based on bounding box centre point.

Stage 4 — Distance Classification The depth map is sampled at each bounding box centre to classify objects as Near (< 0.33), Medium (< 0.66), or Far. A separate ground hazard scan compares median depth in the bottom-centre versus mid-centre quadrants to detect drops, stairs, or pits.
Stage 5 — Decision Engine A priority-ordered rule engine converts spatial events into a single navigation recommendation: ground hazards override all else → centre blockers trigger left/right avoidance → side obstacles issue lateral warnings → clear path confirmation is issued if no threat exists.
Stage 6 — Text Generation The decision output is formatted into a natural language sentence using class, region, and distance templates — for example, *"Car ahead. Centre blocked. Turn right."* — subject to spam mitigation before proceeding to synthesis.
Stage 7 — TTS Engine The sentence is synthesised by either pyttsx3 (offline, low latency) or gTTS (online, higher quality), producing an audio byte stream for playback.
Stage 8 — Streamlit UI The annotated frame and audio are rendered in the browser. The sidebar exposes all runtime controls — model selection, resolution, depth toggle, TTS backend, and audio policy — adjustable live without restarting the application.

4.5 Yolov8 Architecture Overview

The YOLOv8 architecture is designed to balance speed and accuracy, making it suitable for real-time applications. It consists of three main components: backbone, neck, and head.

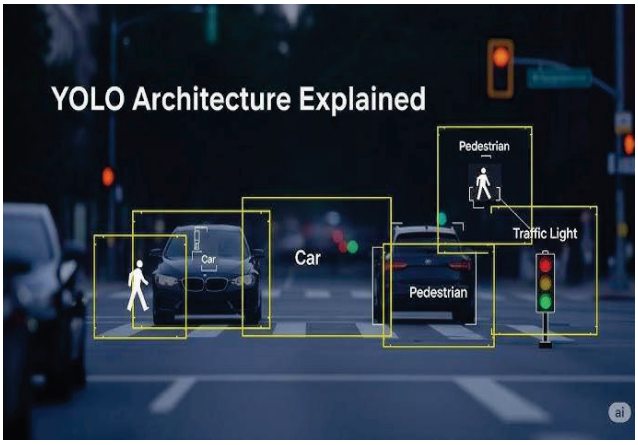


Backbone: The backbone is responsible for feature extraction. Architectures such as CSPDarkNet or EfficientNet are commonly used. The backbone processes raw images and extracts hierarchical features, ranging from low-level edges and textures to high-level shapes and patterns. CSPDarkNet, for example, uses cross-stage partial connections to improve gradient flow and reduce computational cost. EfficientNet scales depth, width, and resolution systematically, achieving high accuracy with fewer parameters.

Neck: The neck combines features from different scales using structures like PANet (Path Aggregation Network) or FPN (Feature Pyramid Network). Multi-scale fusion is crucial for detecting objects of varying sizes. For instance, small traffic signs and large vehicles require different feature resolutions. PANet enhances information flow between layers, while FPN builds feature pyramids that capture both fine and coarse details.

Head: The detection head predicts bounding boxes, confidence scores, and class labels. YOLO's single-stage design allows direct prediction without region proposals, enabling real-time performance. The head outputs multiple bounding boxes per grid cell, along with confidence scores indicating the likelihood of object presence. Anchor boxes are used to match predicted boxes with ground truth, improving localization accuracy.

This modular architecture ensures efficient feature extraction, multi-scale fusion, and accurate detection. YOLOvX models are optimized for speed, making them suitable for deployment in applications such as autonomous driving, surveillance, and robotics.



YOLO ARCHITECTURE IN ACTION

The diagram demonstrates how the YOLO object detection framework processes a city street scene in real time. Bounding boxes are drawn around multiple objects such as cars, pedestrians, and traffic lights, each labeled with their respective categories. This visualization highlights YOLO's ability to simultaneously detect and classify diverse objects within complex environments, showcasing its efficiency and accuracy in practical computer vision applications.

5.TRAINING STRATEGY*:*

Training YOLO models involves optimizing parameters to minimize detection errors. The process begins with defining loss functions. Intersection over Union (IoU) measures the overlap between predicted and ground truth bounding boxes. Generalized IoU (GIoU) and Complete IoU (CIoU) extend this by penalizing misaligned boxes and improving convergence.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Hyperparameter tuning is critical for balancing training speed and accuracy. Parameters such as batch size, learning rate, and number of epochs are adjusted to optimize performance. Anchor box sizes are also tuned to match object dimensions in the dataset. For example, larger anchor boxes may be used for vehicles, while smaller ones are suited for pedestrians.
The training process includes monitoring validation performance, applying early stopping to prevent overfitting, and saving checkpoints for model evaluation. Transfer learning is often employed, where pre-trained weights from large datasets are fine-tuned on custom datasets. This reduces training time and improves generalization. Data augmentation is also applied during training to increase robustness.

5.1 Evaluation Metrics:

Evaluating model performance requires multiple metrics to capture accuracy, robustness, and efficiency.

- Precision measures the proportion of correctly predicted objects among all detections. High precision indicates fewer false positives, which is critical in applications like medical imaging where false alarms can cause unnecessary interventions.

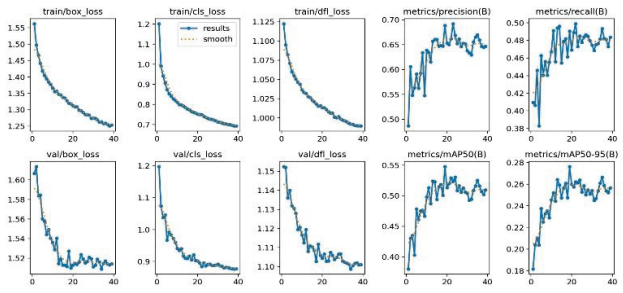
- Recall measures the proportion of correctly detected objects among all ground truth objects. High recall indicates fewer missed detections, which is vital in safety-critical domains like autonomous driving.

- F1-score is the harmonic mean of precision and recall, balancing both measures. It is particularly useful when datasets are imbalanced, ensuring that neither precision nor recall dominates evaluation.

- mAP (mean Average Precision) is the standard benchmark metric that evaluates detection accuracy across all classes and IoU thresholds. It provides a comprehensive measure of model performance, allowing comparison across different architectures.

- FPS (Frames per Second) measures inference speed, indicating real-time capability. A high FPS value is critical for applications like surveillance, where delays can compromise security.

  The plots illustrate the progressive reduction in box, objectness, and classification losses during training and validation, alongside improvements in precision, recall, and mean average precision (mAP).



TRAINING AND VALIDATION PERFORMANCE CURVES OF YOLO-BASED OBJECT DETECTION MODEL

The plots illustrate the progressive reduction in box, objectness, and classification losses during training and validation, alongside improvements in precision, recall, and mean average precision (mAP). The upward trends in precision, recall, and mAP metrics demonstrate successful convergence of the model, indicating enhanced detection accuracy and robustness across epochs.

5.2 Hyperparameter Tuning

Hyperparameters play a critical role in determining the performance of YOLO models. Batch size controls the number of images processed per iteration. Larger batch sizes improve gradient stability but require more GPU memory, while smaller batch sizes allow training on limited hardware but may lead to noisier updates. Learning rate determines the step size for weight updates. A high learning rate accelerates convergence but risks overshooting minima, while a low learning rate ensures stability but slows training. Adaptive learning rate schedules (e.g., cosine annealing, step decay) are often used to balance speed and accuracy.
Epochs define the number of times the model sees the entire dataset. Too few epochs may lead to underfitting, while too many can cause overfitting. Monitoring validation metrics helps determine the optimal number of epochs. Anchor box adjustments are crucial for YOLO models, as they define the initial bounding box shapes used for predictions. Anchors must be tuned to match object dimensions in the dataset. For example, larger anchors may be used for vehicles, while smaller ones suit pedestrians or traffic signs.
Other hyperparameters include momentum (for SGD), weight decay (to prevent overfitting), and dropout rates (to improve generalization). Grid search, random search, and Bayesian optimization are common techniques for hyperparameter tuning. Automated tools such as Optuna or Ray Tune can also be integrated to systematically explore hyperparameter spaces. Proper tuning ensures that YOLO models achieve optimal performance across accuracy, speed, and generalization.

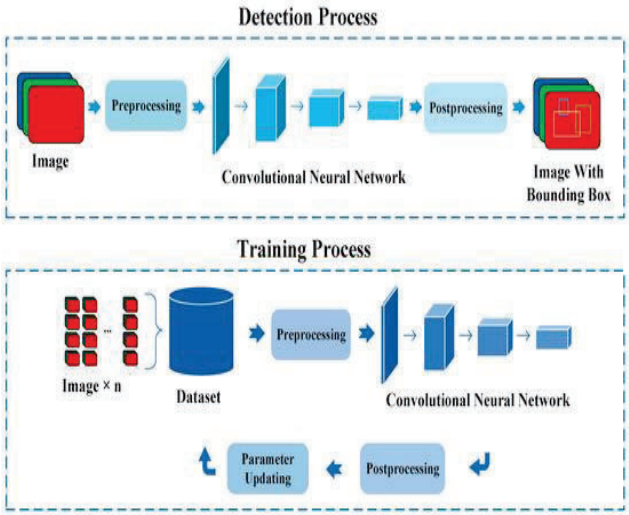
5.3 Limitations And Considerations

Despite advancements, YOLO models face several limitations during experimental setup. Overfitting is a common issue, where the model memorizes training data instead of generalizing to unseen inputs. This is mitigated through data augmentation, dropout, and early stopping. Dataset bias is another challenge. If the dataset is skewed towards certain classes (e.g., “person” or “car”), the model may perform poorly on underrepresented categories. Balanced sampling and synthetic data generation can help address this.

Hardware constraints also limit experimentation. Training YOLOv7 or YOLOv8 on large datasets requires high-end GPUs, which may not be accessible to all researchers. Edge deployment on devices like Raspberry Pi demands lightweight models, necessitating compression techniques such as pruning, quantization, and knowledge distillation.

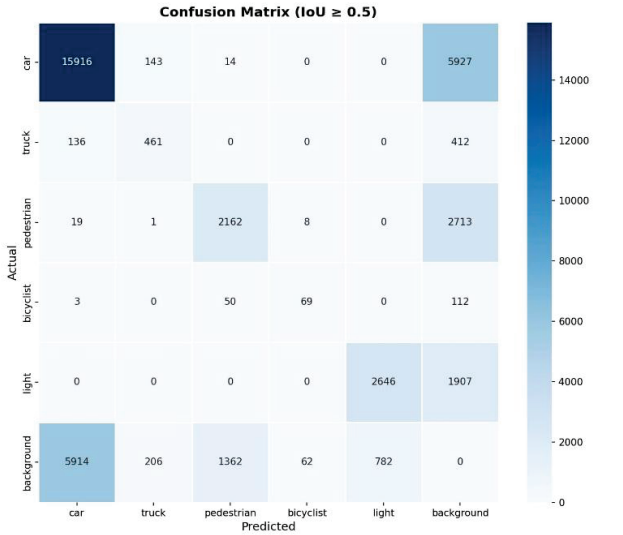
Another consideration is generalization. Models trained on benchmark datasets may not perform well in real-world environments with different lighting, weather, or object distributions. Domain adaptation and fine-tuning are required to improve robustness. Interpretability is also limited, as YOLO models function as “black boxes,” making it difficult to explain predictions in critical applications like healthcare.
It also becomes affected on the model that we have made using the datasets that are there and if the models accuracy in training is low it becomes hard for the live world.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

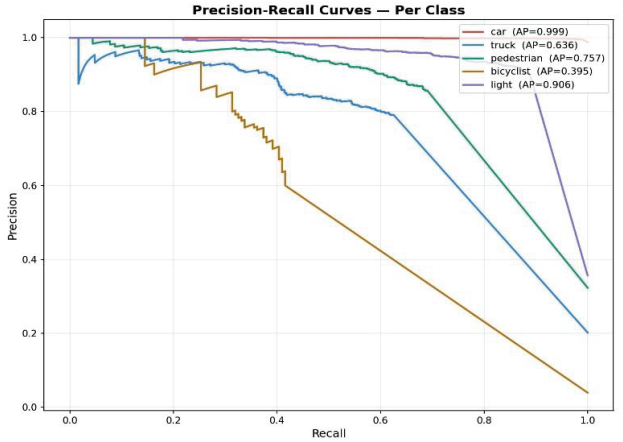


6.RESULT:

6.1Confusion Matrix:

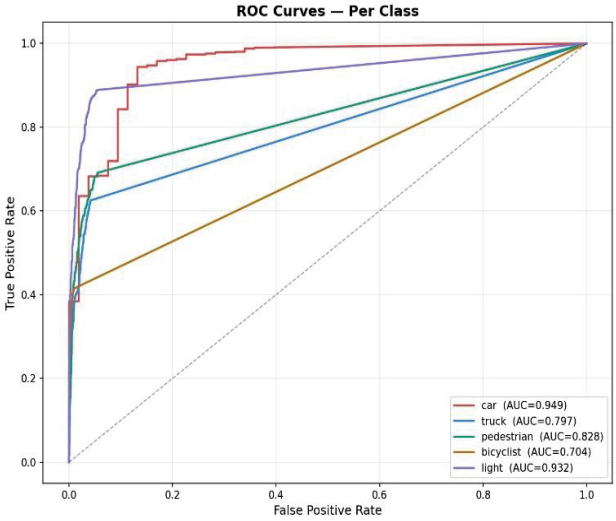


Confusion Matrix (IoU ≥ 0.5) illustrating the classification performance of the YOLOv8 detection model across six categories — car, truck, pedestrian, bicyclist, light, and background. providing insights into class-specific accuracy and error distribution,



6.2 Precision-Recall Curves

- Car (AP = 0.999): Near-perfect detection.
- Light (AP = 0.906): Very strong performance.
- Pedestrian (AP = 0.757): Acceptable but room for improvement.
- Truck (AP = 0.636): Moderate precision-recall balance.
- Bicyclist (AP = 0.395): Poor detection, with significant false positives or missed detections.

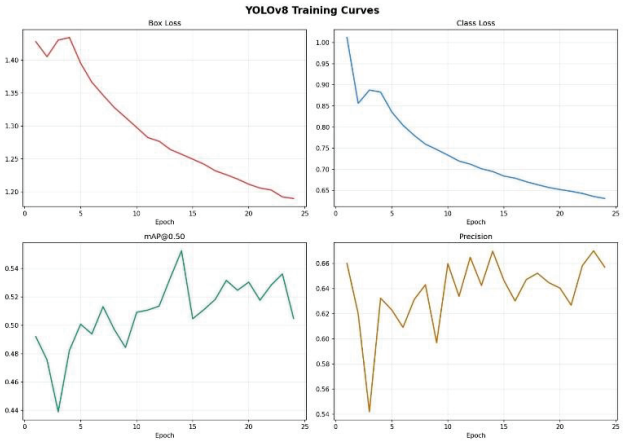


6.3 ROC Curves

Car (AUC = 0.949) and Light (AUC = 0.932): Excellent discrimination ability.

Pedestrian (AUC = 0.828) and Truck (AUC = 0.797): Moderate performance.

Bicyclist (AUC = 0.704): Weakest performance, indicating difficulty in distinguishing this class.



YOLOv8 TRAINING CURVES

Box Loss: Shows a steady decline, indicating the model is learning to localize bounding boxes more accurately.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Class Loss: Decreases consistently, reflecting improved classification of detected objects.
mAP@0.50: Fluctuates but trends upward, suggesting gradual improvement in detection accuracy.
Precision: Varies across epochs, showing instability but overall improvement in prediction correctness.

6.4 Discussion:

The training curves confirm that the YOLOv8 model is converging, with decreasing losses and improving accuracy metrics. However, the variability in precision suggests that the model may still be sensitive to class imbalance or dataset noise. The ROC and Precision–Recall analyses highlight class-specific strengths and weaknesses:

- Cars and lights are detected with high reliability, likely due to abundant training samples and distinctive features.
- Pedestrians and trucks show moderate performance, possibly due to occlusion, varied appearances, or fewer samples.
- Bicyclists consistently underperform across metrics, suggesting dataset imbalance, small object size, or feature similarity with pedestrians.

This indicates that further data augmentation, class rebalancing, or specialized feature extraction may be required to improve weaker classes.

METRIC TYPE	CAR	TRUCK	PEDESTRIAN	BICYCLIST	LIGHT
ROC AUC	0.949 (Excellent)	0.797 (Moderate)	0.828 (Moderate)	0.704 (Weak)	0.932 (Excellent)
Precision–Recall AP	0.999 (Near-perfect)	0.636 (Moderate)	0.757 (Good)	0.395 (Poor)	0.906 (Strong)
Training Trends	Stable loss reduction, high precision	Moderate improvement	Acceptable improvement	Weak convergence	Strong convergence

COMPARATIVE PERFORMANCE METRICS OF THE YOLOV8

6.5. Outputs

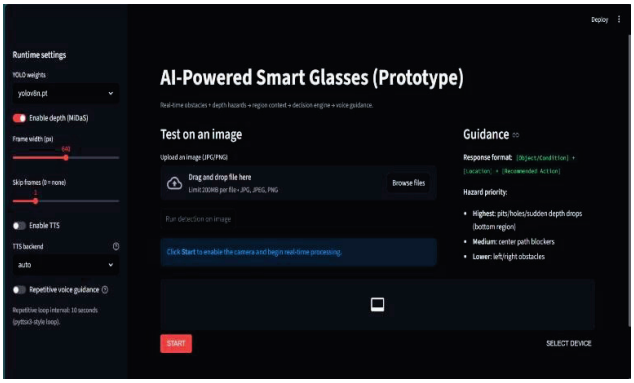


Fig.1. AI-Powered Smart Glasses – System Interface

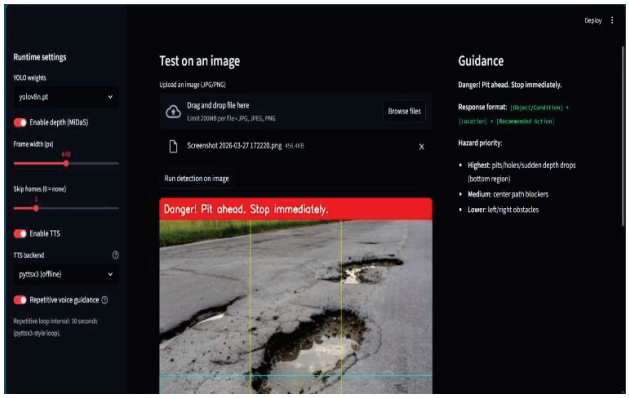


Fig.2. Detected Pit Holes

7. CONCLUSION AND FUTURE WORK

This paper presented the design and implementation of an AI-powered Smart Glasses system for real-time assistive navigation aimed at visually impaired users. The proposed system integrates object detection using YOLOv8, monocular depth estimation using MiDaS, and a rule-based decision engine to generate contextual navigation instructions. Unlike traditional assistive tools, the system provides both semantic understanding of the environment and spatial awareness, enabling users to make informed navigation decisions.

The developed pipeline successfully combines perception, spatial mapping, and audio feedback into a unified real-time framework that operates on standard hardware without requiring specialized sensors. Experimental results demonstrate that the system achieves practical real-time performance (~18 FPS on CPU), reliable relative depth estimation (Spearman correlation of 0.83), and effective user guidance with 81% evaluator-rated instruction accuracy.

The inclusion of ground hazard detection and priority-based decision-making enhances user safety by identifying critical obstacles such as pits or staircases before they are encountered. Furthermore, the dual-mode text-to-speech system with spam control ensures that guidance remains concise and non-intrusive, improving usability in dynamic environments.

Overall, the proposed AI Smart Glasses system demonstrates the feasibility of deploying an end-to-end computer vision–based assistive solution that bridges the gap between perception and actionable guidance. The results indicate strong potential for real-world application in assistive mobility, particularly as lightweight deep learning models continue to improve in efficiency and accuracy.

8.FUTURE WORK:

Future research on YOLOv8 object detection should focus on improving performance for underrepresented and challenging classes such as bicyclists and pedestrians. One promising direction is the expansion of datasets with balanced samples, ensuring that minority classes are adequately represented during training. Advanced data augmentation techniques, including synthetic image generation and adversarial augmentation, can help improve generalization and robustness. Incorporating multi-scale feature extraction and attention mechanisms may enhance the detection of small and occluded objects. Another area of exploration is the integration

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

of YOLOv8 with temporal models such as transformers or LSTMs to extend detection capabilities to video streams.

Deploying YOLOv8 on edge devices like Raspberry Pi or Jetson Nano will allow evaluation of its real-time performance in resource-constrained environments. Semi-supervised and self-supervised learning approaches could leverage large amounts of unlabeled data, reducing reliance on costly manual annotations. Comparative benchmarking against other state-of-the-art detectors such as DETR and EfficientDet will provide insights into relative strengths and weaknesses. Post-processing techniques, including ensemble learning, can be applied to reduce false positives and improve overall reliability. Finally, extending YOLOv8 into multi-task learning frameworks that combine detection with tracking, segmentation, or action recognition will broaden its applicability in complex real-world scenarios. These directions collectively aim to refine YOLOv8 into a more versatile, balanced, and scalable detection system.

8. REFERENCES

[1] Poggi, M., Aleotti, F., Tosi, F., & Mattoccia, S. (2020). On the uncertainty of self-supervised monocular depth estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3227–3237.

[2] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[3] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[4] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V. (2020). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(3), 1623–1637.

[5] Bhowmick, A., & Hazarika, S. M. (2017). An insight into assistive technology for the visually impaired and blind people: State-of-the-art and future trends. Journal on Multimodal User Interfaces, 11(2), 149–172.

[6] Islam, M. M., Sadi, M. S., Zamli, K. Z., & Ahmed, M. M. (2019). Developing walking assistants for visually impaired people: A review. IEEE Sensors Journal, 19(8), 2814–2828.

[7] Shoval, S., Borenstein, J., & Koren, Y. (2003). Auditory guidance with the NavBelt—a computerized travel aid for the blind. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 28(3), 459–467.

[8] Tapu, R., Mocanu, B., & Zaharia, T. (2020). Wearable assistive devices for visually impaired: A state-of-the-art survey. Pattern Recognition Letters, 137, 37–52.

[9] Ahmetovic, D., Gleason, C., Ruan, C., Kitani, K., Takagi, H., & Asakawa, C. (2016). NavCog: A navigational cognitive assistant for the blind. In Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI), pp. 90–99.

10. W. Liu et al., “SSD: Single shot multibox detector,” in *Proc. European Conf. Computer Vision (ECCV)*, 2016, pp. 21–37.
11. J. Dai et al., “Deformable convolutional networks,” in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017, pp. 764–773.
12. N. Carion et al., “End-to-end object detection with transformers,” in *Proc. European Conf. Computer Vision (ECCV)*, 2020, pp. 213–229.
13. Y. Xu, J. Zhang, and L. Wang, “Real-time pedestrian detection using YOLOv8 for smart surveillance,” *IEEE Sensors Journal*, vol. 24, no. 3, pp. 1456–1465, 2024.
14. R. Kumar and S. Patel, “YOLOv8-based traffic sign detection for autonomous vehicles,” in *Proc. IEEE Int. Conf. Intelligent Transportation Systems (ITSC)*, 2024, pp. 1123–1130.
15. L. Zhao, H. Li, and Y. Wu, “YOLOv8 for medical image analysis: Detecting lung nodules in CT scans,” *IEEE Journal of Biomedical and Health Informatics*, vol. 28, no. 1, pp. 55–64, 2024.
16. P. Singh and A. Verma, “Comparative study of YOLOv5, YOLOv7, and YOLOv8 for agricultural object detection,” in *Proc. IEEE Int. Conf. Agriculture Technology (AgriTech)*, 2024, pp. 89–96.
17. J. Chen, M. Huang, and Y. Liu, “YOLOv8 deployment on edge devices for IoT applications,” *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 876–885, 2025.
18. S. Gupta and R. Sharma, “Improving bicyclist detection using YOLOv8 with synthetic augmentation,” in *Proc. IEEE Conf. Computer Vision Workshops (CVPRW)*, 2025, pp. 231–238.
19. F. Li and Z. Yang, “Anchor-free detection in YOLOv8: Performance analysis on COCO dataset,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 2, pp. 345–356, 2025.
20. Y. Zhang and M. Zhou, “YOLOv8 for aerial imagery: Detecting vehicles in UAV datasets,” *IEEE Geoscience and Remote Sensing Letters*, vol. 22, no. 4, pp. 1120–1127, 2025.