Vision Aid: Real-Time Object Detection System for Assisting Visually Impaired Individuals

---

Assessment 3: Final Report

Project-Based Learning and Technology (PBT205)

Prepared by:
1. Smriti Parajuli (MDS3000025)
2. Gloria Hawkins-Roberts (MDS2000662)

Submitted to:
Ranpreet Kaur, Lecturer, BSE-AI
Media Design School

November 14th, 2024

Abstract

This report explores the development and evaluation of **"*Vision Aid*"**, a computer vision system utilising YOLOv8 for real-time object detection to assist visually impaired individuals in outdoor navigation. The system identifies and classifies objects such as vehicles, pedestrians, and obstacles, providing auditory feedback to improve situational awareness and independence. Utilising the 26 Class Object Detection Dataset, the project demonstrates the potential of AI-driven solutions for mobility support. Key findings include robust object detection under diverse environmental conditions, with areas for improvement in complex or cluttered settings. Vision Aid exemplifies the application of assistive AI technology, fostering inclusivity and autonomy for visually impaired users.

1. Introduction

1.1 Background and Context

Navigating outdoor spaces presents a fundamental yet challenging aspect of daily life for visually impaired individuals. Devoid of access to visual cues, these individuals face heightened risks from both stationary obstacles, such as benches and poles, and dynamic hazards like moving vehicles and pedestrians. Research indicates that visually impaired individuals experience a higher incidence of injuries in public spaces, significantly impacting their independence and quality of life (Högner, 2015). Traditional mobility aids, such as white canes and guide dogs, provide some support but have inherent limitations. For instance, white canes only detect obstacles directly in the user's path and cannot identify elevated or fast-approaching hazards, while guide dogs, though effective, are costly and inaccessible for many due to financial and logistical barriers. Additionally, both tools require significant user training and often struggle to adapt to dynamic or unfamiliar environments, leaving critical gaps in mobility support.

1.2 Vision Aid Solution Overview

Vision Aid was developed to address these challenges by leveraging advancements in artificial intelligence (AI) and computer vision. Powered by YOLOv8, a state-of-the-art object detection model, Vision Aid offers real-time identification and classification of various

objects, providing actionable auditory feedback to guide visually impaired users. Key features of Vision Aid include directional audio cues, proximity alerts, and adaptability to diverse environmental conditions, such as varying lighting and weather. The latest version of Vision Aid builds upon previous iterations by introducing an intuitive graphical user interface (GUI), voice-controlled settings, and customizable object detection preferences. These enhancements enable seamless user interaction, delivering precise, context-aware guidance tailored to each user's immediate environment. By leveraging YOLOv8 for robust real-time object detection and focusing on accessibility, Vision Aid aims to set a new standard in assistive technology, reinforcing a commitment to improving the quality of life for visually impaired individuals.

1.3 Problem Domain

The Vision Aid project addresses challenges within the intersection of assistive technology and computer vision. Specifically, it aims to support visually impaired individuals in navigating outdoor environments by detecting and identifying objects such as vehicles, pedestrians, and stationary obstacles. In doing so, Vision Aid enhances users' situational awareness and mobility. Traditional navigation aids like white canes and guide dogs, while useful, are often inadequate in dynamic and unfamiliar environments. Canes cannot detect elevated hazards or fast-approaching objects, and guide dogs are not accessible to many due to their high cost and logistical challenges. Vision Aid aims to provide a scalable, cost-effective alternative by utilising AI-powered real-time object detection and audio guidance to enhance user safety and independence.

1.4 Motivations

The primary motivation behind the Vision Aid project is the critical need to improve accessibility and safety for visually impaired individuals in outdoor settings. Navigating public spaces can be hazardous, as obstacles such as vehicles, pedestrians, and stationary objects pose constant challenges. Traditional aids like canes and guide dogs offer limited support, especially in complex or dynamic outdoor environments. Recent advancements in AI, particularly in object detection models such as YOLOv8, provide an opportunity to create a robust assistive tool that delivers real-time, precise information about a user's surroundings. Through these technologies, Vision Aid aims to offer visually impaired users increased autonomy and security in their daily lives. Additionally, the project aligns with broader social

goals, including enhancing inclusivity, reducing accident rates, and empowering visually impaired individuals to engage more fully in society.

---

2. Literature Review

2.1 Existing Assistive Technologies

Various assistive technologies have been developed over the years to support visually impaired individuals in navigating their environments. Traditional aids, such as white canes and guide dogs, have long been relied upon for mobility support. However, while these tools are helpful, they have limitations in complex, dynamic environments where the user must identify a wide range of objects and respond to moving obstacles.

In response to these limitations, technology-driven solutions have emerged. One notable example is the *WeWALK* smart cane, which combines traditional cane functionality with modern technology (*WeWALK Smart Cane – Smart Cane for the Visually Impaired*, n.d.). The WeWALK cane integrates an ultrasonic sensor, GPS navigation, and smartphone connectivity, providing users with vibration alerts for nearby obstacles, turn-by-turn navigation, and even connection to public transportation information (*WeWALK Smart Cane – Smart Cane for the Visually Impaired*, n.d.). The device aims to enhance both safety and independence for visually impaired users, especially in urban settings.

Similarly, Microsoft's *Seeing AI* app is another innovation in assistive technology. Using computer vision, Seeing AI provides real-time auditory descriptions of the user's surroundings, including text recognition, face identification, and object detection (*Seeing AI - Talking Camera for the Blind*, n.d.). This app exemplifies the trend towards AI-based solutions that enhance spatial awareness and enable greater independence (*Seeing AI - Talking Camera for the Blind*, n.d.). Despite its versatility, Seeing AI's reliance on smartphone-based input limits the immediacy of its feedback, especially in high-stakes environments where hands-free interaction is essential (*Seeing AI - Talking Camera for the Blind*, n.d.).

2.2 Advances in Object Detection for Assistive Technologies

Object detection models have significantly advanced over recent years, becoming foundational for AI-based assistive tools. Early object detection models, such as Haar cascades and Histogram of Oriented Gradients (HOG), provided the groundwork for later developments but were limited in speed and accuracy (Sun et al., 2024). The advent of deep learning models like the *Single Shot Multibox Detector (SSD)*, *Faster R-CNN*, and *You Only Look Once (YOLO)* revolutionised object detection with improved accuracy and real-time capabilities (Sun et al., 2024). The YOLO family of models, in particular, has gained prominence for its speed and efficiency, which are critical for assistive applications where real-time response is crucial (Dey, 2023). YOLO models are designed to process entire images in a single pass, enabling rapid detection of multiple objects in complex scenes (Dey, 2023) The latest versions, including YOLOv8, further improve upon previous iterations with enhanced accuracy, adaptability to diverse object types, and reduced latency, making them ideal for mobile and wearable applications(Dey, 2023). YOLO-based models have been applied in various assistive technologies, offering the potential to detect vehicles, people, and obstacles with minimal delay.

2.3 AI-Based Assistive Tools

beyond object detection, AI-based tools for the visually impaired have expanded to include comprehensive systems that integrate machine learning, computer vision, and natural language processing. For example, *OrCam MyEye* is a wearable device that attaches to eyeglasses and provides real-time audio feedback by reading text, recognizing faces, and identifying products (*OrCam MyEye 3 Pro - Revolutionize Your Vision With Cutting-Edge AI Technology*, 2024). OrCam MyEye's focus on audio feedback allows visually impaired individuals to interact with their surroundings more independently, yet it does not offer extensive navigational guidance or real-time detection of dynamic obstacles in outdoor environments (*OrCam MyEye 3 Pro - Revolutionize Your Vision With Cutting-Edge AI Technology*, 2024).

Another example is *NavCog*, a navigation app developed by Carnegie Mellon University specifically for visually impaired users. NavCog uses Wi-Fi and Bluetooth beacons to provide audio navigation cues in indoor settings, helping users navigate buildings, museums, and other spaces where GPS is unavailable (*CAL - NavCog*, n.d.). However, NavCog's reliance on infrastructure-based beacons limits its usability in outdoor, unstructured

environments, where visually impaired individuals face the most significant challenges. (*CAL - NavCog*, n.d.)

2.4 Vision Aids Position in the Current Landscape

Within the current landscape of assistive technologies for visually impaired individuals, many existing solutions focus on either basic obstacle detection or smartphone-based functionality. Vision Aid stands out by addressing key gaps in these existing technologies and providing a more comprehensive, accessible approach to outdoor navigation.

Most assistive programs, such as WeWALK and Seeing AI, lack support for users with partial vision loss (*WeWALK Smart Cane – Smart Cane for the Visually Impaired*, n.d.). Vision Aid addresses this limitation by offering customization options specifically designed to enhance accessibility for those who retain some level of vision. For instance, Vision Aid includes a high-contrast, colourful GUI, which makes it easier for users with partial vision to navigate the system's interface. This added visual support is especially beneficial in environments where lighting conditions or background clutter might otherwise make interaction with the device challenging.

Unlike WeWALK, which primarily focuses on obstacle detection and basic smartphone connectivity (*WeWALK Smart Cane – Smart Cane for the Visually Impaired*, n.d.), Vision Aid leverages YOLOv8's advanced object detection capabilities to provide detailed information on a wide range of outdoor objects, both stationary and moving. This robust object detection enables Vision Aid to identify potential hazards and significant objects in real time, offering situational awareness that goes beyond what traditional aids like canes or smart canes can provide. Furthermore, Vision Aid's system can detect objects from varying distances and directions, alerting users to the presence of obstacles like cars, bicycles, or crosswalks, thereby enhancing outdoor navigation safety and independence.

Additionally, Vision Aid's hands-free design differentiates it from applications like Seeing AI, which requires the user to hold and operate a smartphone camera. By providing audio feedback and enabling voice-activated controls, Vision Aid allows for uninterrupted navigation without the need for manual handling, making it more convenient and accessible in real-world outdoor settings.

Customization is another essential aspect of Vision Aid. Users can tailor the device to their specific needs and preferences by adjusting detection sensitivity, enabling or disabling voice control, and filtering out objects of lesser interest. This adaptability makes Vision Aid versatile across different environments and user conditions, allowing it to function effectively in various outdoor scenarios. The ability to customise settings like sensitivity and object filters also means Vision Aid can be used effectively by a broad range of users, from those with complete vision loss to individuals with partial vision impairments who benefit from enhanced visual cues.

In summary, Vision Aid's position in the current landscape is defined by its emphasis on inclusivity, hands-free functionality, and the ability to provide a more detailed, personalized, and versatile approach to assistive navigation. It fills significant gaps left by existing technologies, supporting users with both complete and partial vision impairments and empowering them to navigate complex environments with greater independence and confidence.

2.5 Vision Aids Contributions

Vision Aid's contributions to the field of assistive technology are threefold:

1. **Real-Time, Multi-Object Detection and Feedback**: Leveraging YOLOv8, Vision Aid offers high-speed detection of multiple objects in real-time, including dynamic and stationary hazards. This capability enables visually impaired users to receive instant auditory cues about their surroundings, enhancing their spatial awareness and safety in outdoor environments.

2. **Customizable and Accessible Interface**: Vision Aid includes a user-friendly GUI with high-contrast options, voice-controlled settings, and scalable UI elements for ease of interaction. This accessibility-focused design ensures that users with varying levels of technical familiarity and visual acuity can benefit from the system.

3. **Hands-Free Guidance Support**: By providing real-time feedback without requiring hands-on device interaction, Vision Aid empowers users to navigate independently while minimising physical and cognitive load. This hands-free approach enhances

usability, especially for visually impaired individuals who may rely on mobility aids or require constant situational awareness.

In summary, Vision Aid expands the capabilities of AI-based assistive technologies for the visually impaired, offering a comprehensive tool that addresses both navigational and accessibility challenges. By integrating state-of-the-art object detection, customizable settings, and accessible design, Vision Aid exemplifies the potential for AI-driven solutions to improve quality of life, foster independence, and promote safety for visually impaired individuals in outdoor settings.

---

3. Dataset Overview

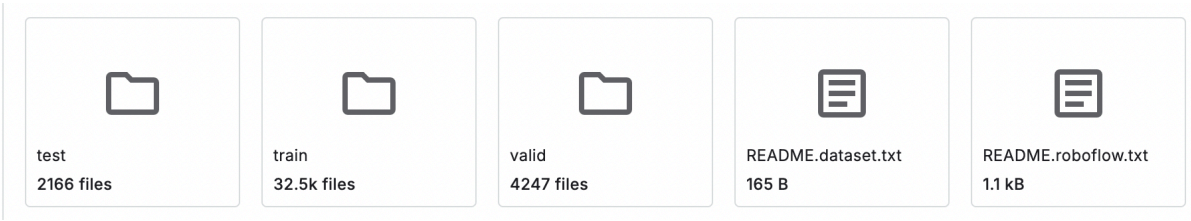


Figure 1. *Screenshot depicting the 26 Class Object Detection Dataset file organization. Retrieved from:https://www.kaggle.com/datasets/mohamedgobara/26-class-object-detection-dataset*

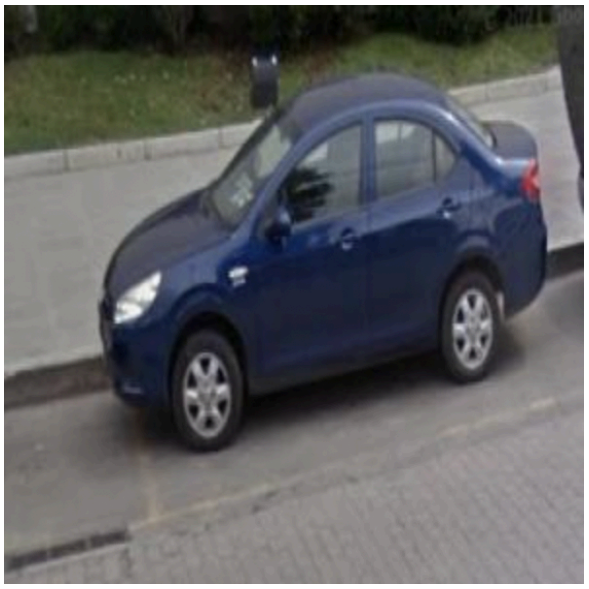


Figure 2. *Image -13AD8372-CDDB-4DCF-B8F9-DC10D210064B-png_jpg from the dataset. Retrieved from: https://www.kaggle.com/datasets/mohamedgobara/26-class-object-detection-dataset*

The dataset utilised for this project, known as the 26 Class Object Detection Dataset, includes 38,952 images annotated across 26 object classes, providing a comprehensive range of labelled images crucial for training and evaluating object detection models. Published in November 2023, this dataset includes classes such as bench, bicycle, branch, bus, bushes, car, crosswalk, door, elevator, fire hydrant, green light, gun, motorcycle, person, pothole, rat, red light, scooter, stairs, stop sign, traffic cone, train, tree, track, umbrella, and yellow light (26 Class Object Detection Dataset, 2024). This diversity enables the Vision Aid project to detect a broad spectrum of outdoor objects, ensuring that visually impaired users are well-informed about various obstacles in their environment.

The dataset is organised into three folders—train, validation, and test—containing 32,500, 4,247, and 2,166 files, respectively, as shown in Figure 1 (26 Class Object Detection Dataset, 2024).. Additionally, two README files provide essential dataset information, such as class descriptions and labelling conventions (26 Class Object Detection Dataset, 2024). Figure 2 showcases an example image from the dataset, labelled with the class “car,” highlighting the dataset’s detailed labelling for object identification. The structured format and diversity in object types offer an ideal foundation for the Vision Aid project, supporting real-time, accurate object detection critical for aiding visually impaired individuals in outdoor navigation.

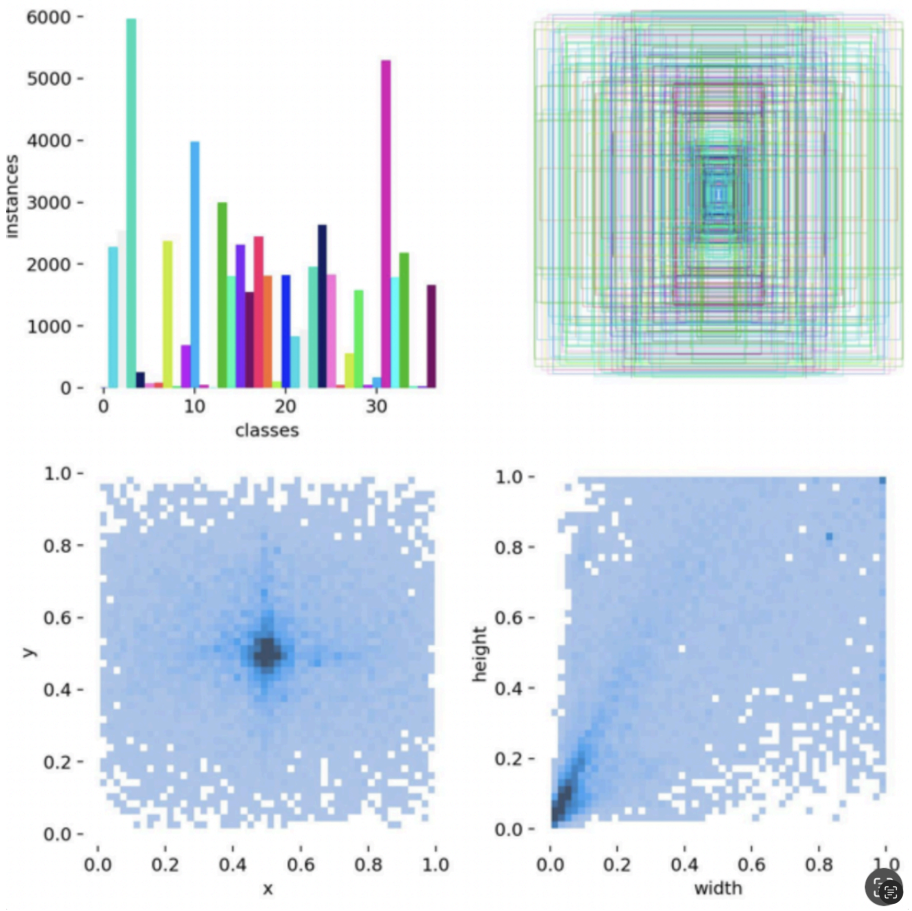
3.1 Pre-Processing by Roboflow

The dataset underwent the following pre-processing steps, ensuring its readiness for training:

- **Auto-Orientation of Pixel Data**: Corrected image orientations by removing EXIF metadata.
- **Resizing**: All images were resized to **300x300 pixels**, maintaining consistency across the dataset.
- **Augmentation**: Each source image was augmented to create three variations using:

- **Salt-and-Pepper Noise**: Applied to 15% of pixels to simulate real-world camera noise and environmental artefacts.

These pre-processing steps by Roboflow provided a baseline, allowing Vision Aid to build on a structured and uniform dataset.



3.2 Class Distribution Analysis

An examination of the 26 Class Object Detection Dataset reveals a notable imbalance in class representation. Certain object classes, such as 'person' and 'car,' are more frequently represented, while others like 'rat' and 'pothole' appear less often. This uneven distribution can impact the model's performance, as it may become biassed toward detecting more prevalent classes, potentially leading to reduced accuracy for underrepresented objects. Addressing this imbalance is crucial to ensure the Vision Aid system reliably detects a wide range of objects in diverse real-world scenarios.

3.3 Significance of Roboflow's Contributions

Incorporating the **Senior Design VIAD - v4** dataset ensured:

- **Diversity**: The dataset's 26 object classes and varied scenes provided the model with robust training data for outdoor navigation.
- **Augmented Realism**: The salt-and-pepper noise and resizing simulated conditions likely to be encountered in real-world scenarios.
- **Scalability**: The dataset's compatibility with YOLOv8 and its pre-prepared annotations streamlined the training pipeline.

---

4. YOLOv8 Model Selection and Training

The Vision Aid project leverages YOLOv8 for real-time object detection, providing a robust and efficient solution to help visually impaired individuals navigate outdoor environments safely. YOLO, which stands for "You Only Look Once," is a deep learning model known for its speed and accuracy in object detection. Unlike multi-stage models, YOLO's architecture processes the entire image in a single forward pass through the neural network, allowing for faster detection, which is crucial for real-time applications like Vision Aid. This enables the system to promptly alert users of nearby objects—such as vehicles, pedestrians, crosswalks, and obstacles—via audio feedback, ensuring immediate awareness of their surroundings.

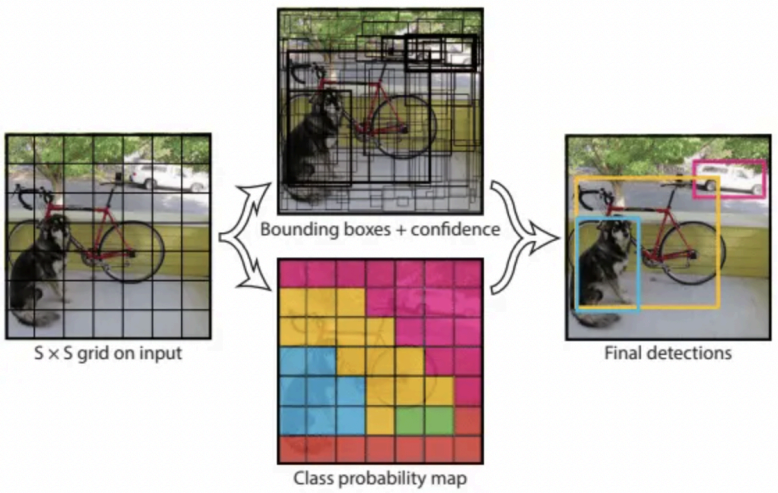


Figure 3. *Diagrams depicting the YOLO architecture. Retrieved from:* *https://www.labellerr.com/blog/evolution-of-yolo-object-detection-model-from-v5-to-v8/*

4.1 Understanding YOLO's Architecture (Figure 3)

Figure 3 illustrates the YOLO architecture, consisting of a streamlined sequence of convolutional layers followed by a detection layer. YOLO's one-stage design is a key factor in its speed and efficiency, distinguishing it from two-stage models like Faster R-CNN, which first generate region proposals and then classify objects. Here's a breakdown of YOLO's architecture:

1. **Convolutional Layers**: The image is first processed through several convolutional layers, which extract essential features such as edges, textures, and patterns. In the initial layers, YOLO captures low-level features, while deeper layers detect more complex structures.
2. **Detection Layer**: After passing through the convolutional layers, the features are consolidated in the detection layer, which is responsible for predicting object classes and bounding boxes. YOLO predicts multiple bounding boxes per cell, assigning a confidence score to each prediction, indicating the likelihood of an object's presence.
3. **Grid Cell Assignment**: YOLO divides the image into a grid, with each cell predicting bounding boxes and class scores for objects within its region. This grid-based approach allows YOLO to detect multiple objects in a single image, even if they overlap or vary in size.
4. **Bounding Box and Class Prediction**: Predicted bounding boxes are visualised with coloured boxes, each associated with an object class. As shown in Figure 3, YOLO can successfully detect objects like a bicycle and a dog, each with a distinct bounding box.

This efficient, one-stage approach not only enhances detection speed but also minimises computational resources. By performing the entire detection in one forward pass, YOLO can provide real-time feedback, critical for Vision Aid's application. The immediate feedback empowers visually impaired users with timely information about their surroundings, enabling them to navigate more safely and confidently.

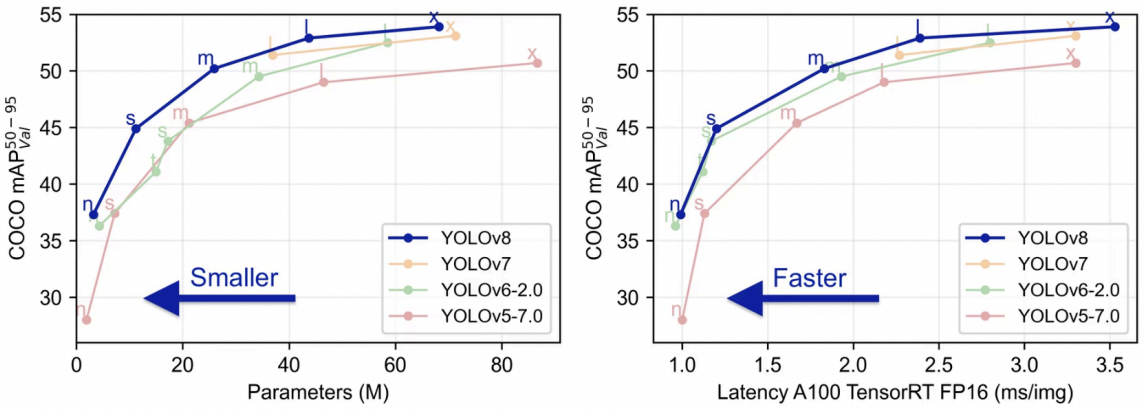


Figure 4. *Graphs depicting the accuracy of different YOLO versions. Retrieved from:* https://docs.ultralytics.com/models/yolov8/

4.2 Choosing YOLO for Vision Aid (Figure 4)

For the Vision Aid project, YOLOv8 was selected due to its advancements over previous versions, including YOLOv5, YOLOv6, and YOLOv7. As shown in Figure 4, YOLOv8 offers significant improvements in model efficiency and processing speed, making it an ideal choice for real-time applications.

The left graph in Figure 4 demonstrates that YOLOv8 achieves higher COCO mean Average Precision (mAP) scores across various model sizes (“n,” “s,” “m,” “l,” “x”), even with smaller models. This efficiency in parameter usage allows YOLOv8 to deliver high detection accuracy with a reduced model size, which is essential for real-time applications running on devices with limited computational resources. In the context of Vision Aid, this smaller, more efficient model ensures quick processing without compromising detection accuracy, enabling the system to keep users informed about their surroundings in real-time.

The right graph in Figure 4 highlights YOLOv8’s improvement in latency compared to previous versions, achieving faster detection times when running on an A100 TensorRT FP16 setup. Faster detection times are vital for Vision Aid, as they ensure that identified objects, like approaching vehicles or pedestrians, are immediately relayed to the user, enhancing safety and situational awareness.

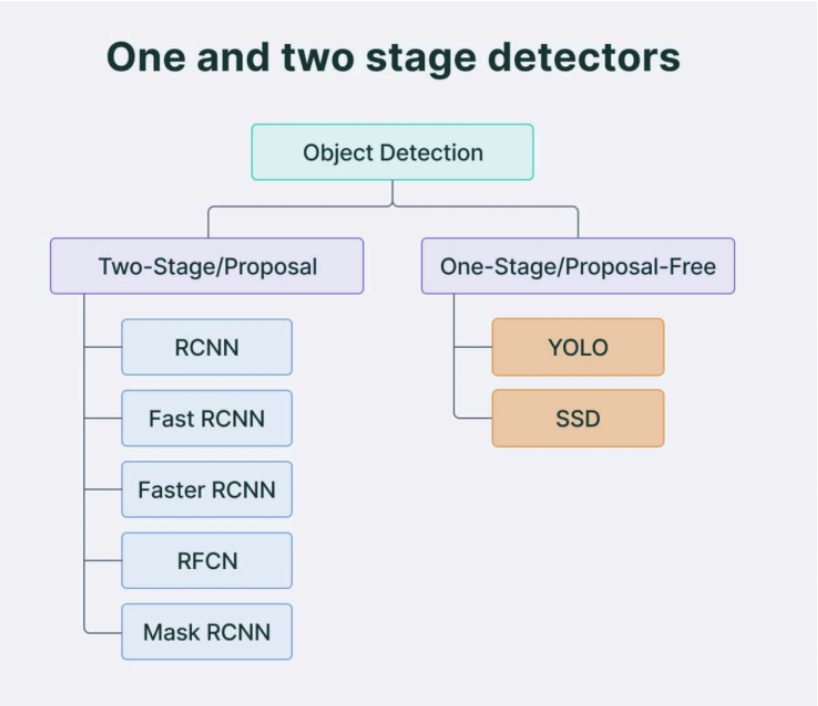


Figure 5. *Flow chart depicting the different types of detection models considered for Vision Aid. Retrieved from: https://www.v7labs.com/blog/yolo-object-detection*

4.3 YOLO Compared to Other Models

In selecting YOLOv8, we considered several other object detection models, as seen in Figure 5, including SSD (Single Shot Multibox Detector) and two-stage models like Faster R-CNN . Here is why YOLOv8 was chosen over these alternatives:

- **Compared to Two-Stage Models**: Two-stage models, such as Faster R-CNN, are structured to deliver high accuracy by first generating region proposals and then performing object classification within each proposed region (Figure 5). This two-step approach allows two-stage detectors to achieve precise localization and classification, especially for complex and densely populated scenes. However, this accuracy comes at the cost of speed and computational efficiency. Two-stage detectors must run separate processes for region proposal and object classification, making them computationally intensive and slower than single-stage models like YOLO. This inherent latency and resource demand make two-stage models less suited for real-time applications, where even slight delays could impact effectiveness and safety. For the Vision Aid project, which relies on immediate audio feedback to assist visually impaired users in navigating their surroundings, the speed of detection is crucial. A delay in identifying nearby obstacles like vehicles or pedestrians could pose a safety risk. Thus, while two-stage

models excel in accuracy, they are not practical for real-time applications like Vision Aid that prioritise responsiveness over precision (Ultralytics, n.d.).

- **Compared to SSD**: YOLO and SSD (Single Shot Multibox Detector) are both one-stage object detection models, designed to detect objects quickly in live video streams. However, they have distinct approaches that make them suitable for different applications (YOLO Vs SSD: Which One Is a Superior Algorithm | Algoscale, n.d.).YOLO, developed by Joseph Redmon in 2015 (Redmon et al., 2016), is designed to process the entire image in a single forward pass, dividing it into a grid where each cell predicts bounding boxes and object classes simultaneously, achieving impressive speeds of up to 65 FPS. This makes YOLO ideal for real-time detection in scenarios requiring rapid object identification, such as live traffic monitoring or tracking fast-moving objects.

Its grid-based structure and use of Intersection over Union (IoU) allow YOLO to manage multiple overlapping objects, though it can sometimes struggle with smaller objects or close-up details (YOLO Vs SSD: Which One Is a Superior Algorithm | Algoscale, n.d.). In contrast, SSD also performs object detection in a single forward pass but uses a distinct bounding box regression method with pre-calculated anchor boxes refined during training to match detected objects. This method can offer slightly better localization accuracy than YOLO, making SSD well-suited for applications that require precision, such as video forensics or legal applications. However, SSD can be slower than YOLO and is less optimized for crowded scenes, making it more suitable for static or low-motion scenarios (YOLO Vs SSD: Which One Is a Superior Algorithm | Algoscale, n.d.).

Both models offer advantages, but YOLO's speed and efficiency make it the preferred choice for Vision Aid, where real-time feedback is essential for user safety in dynamic outdoor environments. The rapid detection of multiple object classes at high speed allows YOLO to handle complex, crowded scenes, such as busy intersections, where visually impaired users need immediate information about various objects around them.

4.4 Benefits of YOLOv8 for Vision Aid

In summary, YOLOv8 enhances Vision Aid's mission by providing a powerful tool for accessible, responsive, and reliable navigation assistance. The integration of this model allows the system to offer real-time, accurate, and efficient detection, aligning with Vision Aid's goal of improving accessibility, safety, and independence for visually impaired individuals. By providing accurate detection across a wide range of obstacles, Vision Aid empowers visually impaired users to engage with their surroundings confidently, increasing their autonomy and ability to navigate outdoor environments safely.

4.5 Model Implementation & Fine Tuning

For Vision Aid, fine tuning was chosen as the primary approach for implementing the YOLOv8 model. Fine-tuning meant that the pre-trained YOLOv8 model (in this case, the nano version, YOLOv8n) could be leveraged, having been originally trained on the COCO dataset. This approach was more efficient than training from scratch because it used the pre-existing object detection capabilities learned from a larger, general-purpose dataset, while tailoring the model to the specific objects and conditions of the 26 class object detection dataset.

1. **Data Preparation:** The chosen dataset was structured to fit the YOLO formatting requirements, with separate folders for training, validation, and test sets, as seen in Figure 1. The labels were cleaned and any invalid annotations were removed to maintain the datasets integrity.

2. **Training and Configuration:**
    - **Learning Rate:** A learning rate of 0.001 was selected to balance training speed and stability. A higher learning rate risks overshooting the optimal solution, particularly with complex models such as YOLOv8. Conversely, a lower learning rate provides more precise updates but would require more epochs to reach a similar level of convergence. The rate of 0.001 allows the model to progress steadily without any major fluctuations.
    - **Batch size:** A batch size of **16** was chosen, balancing memory efficiency and the statistical reliability of the gradient estimates. Given the resource demands of YOLOv8, which processes large amounts of data at each layer, a batch size of 16 is optimal for training on a GPU. Larger batch sizes could exceed GPU memory capacity, resulting in slower training or even crashes. With 16

samples per batch, the model can make efficient use of memory while processing multiple images in parallel, allowing for faster training iterations.

- **Optimizer:** The **Adam optimizer** was selected for its adaptive learning rate properties, which enhance training efficiency and convergence stability. Unlike traditional optimizers like Stochastic Gradient Descent (SGD), which uses a fixed learning rate, Adam adjusts the learning rate for each parameter individually. This adaptability is especially useful for fine-tuning, where some parameters may need smaller adjustments while others need larger ones to fit the new dataset. Adam is also known for faster convergence compared to other optimizers, as it leverages both the moving average of the gradients (momentum) and an adaptive learning rate.
- **Loss Functions:**
    1. **Bounding Box Regression Loss:** This optimises the accuracy of the bounding box predictions.
    2. **Classification Loss:** Ensures correct categorization of the detected objects.
    3. **Objectness loss:** Minimises the false positives by distinguishing objects from background noise.
- **Epochs:** The model was fine-tuned for 50 epochs, this number was selected to ensure convergence whilst minimising the risk of over-fitting. Throughout the fine-tuning process, different numbers of epochs were experimented with such as 20 epochs, 15 epochs and so on. However, lower epochs did not yield optimal results. By extending the training to 50 epochs, a significant improvement in model performance was observed.

4.6 Model Evaluation Metrics

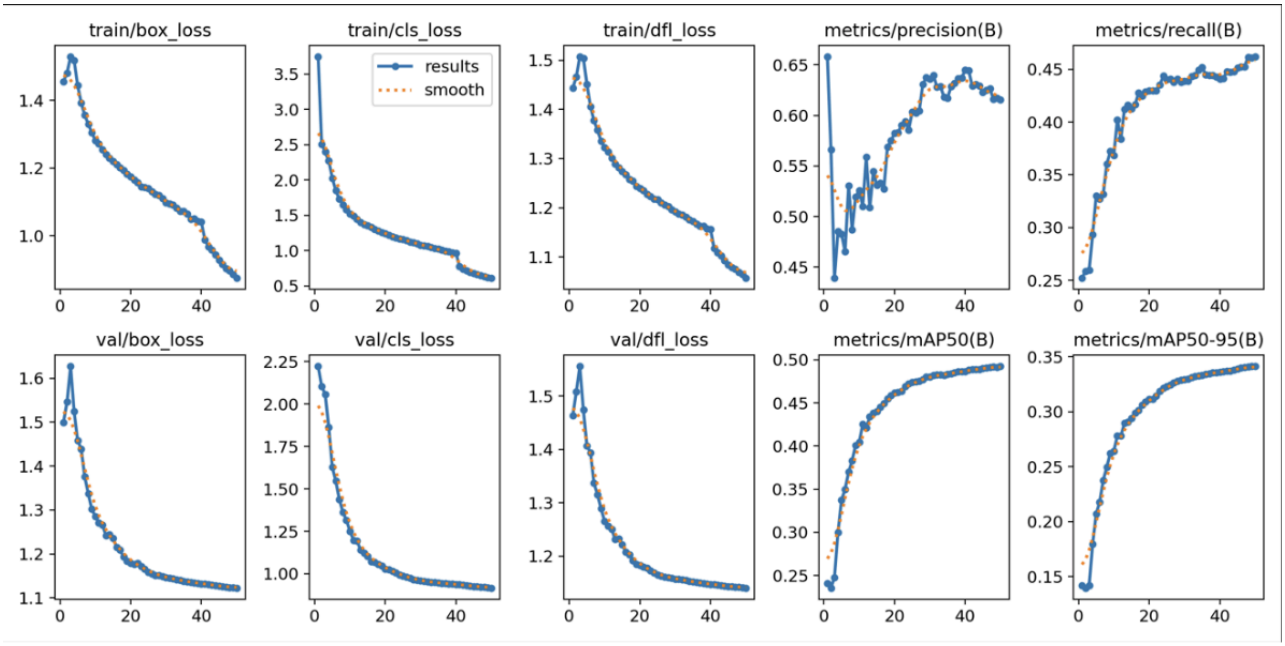


Figure 6. *Graphs depicting the models training and evaluation metrics over 50 epochs.*

Figure 6 provides detailed insights into the model's performance across various metrics during training on the Senior Design VIAD dataset over 50 epochs. The train/box_loss and val/box_loss graphs (top left and bottom left) display the decrease in bounding box localization errors, indicating that the model is progressively learning to identify the object boundaries more accurately (Torres, 2024). Similarly, the train/cls_loss and val/cls_loss graphs (second column) show the reduction in classification errors, demonstrating improved accuracy in identifying object classes (*What Is Loss_Cls and Loss_Bbox and Why Are They Always Zero in Training*, n.d.).

The train/dfl_loss and val/dfl_loss graphs (third column) track the model's distributional focal loss, which also declines consistently, suggesting enhanced confidence in the model's predictions (Brownlee, 2019 & *YOLOv8 Dfl_Loss Metric*, n.d.). The metrics/precision(B) graph (top right, fourth column) and metrics/recall(B) graph (top right, fifth column) reveal the steady increase in precision and recall, respectively, signifying that the model is correctly identifying and detecting relevant objects in the dataset with higher accuracy as training progresses.

Lastly, the metrics/mAP50(B) and metrics/mAP50-95(B) graphs (bottom right two panels) reflect the model's mean Average Precision (mAP) at different Intersection over Union (IoU)

thresholds (Wood & Chollet, 2022). These metrics aggregate the model's precision across recall levels, with mAP50(B) indicating performance at a 0.5 IoU threshold and mAP50-95(B) showing the average across multiple IoU thresholds from 0.5 to 0.95. The steady increase in both mAP metrics demonstrates that the model achieves increasingly reliable detection performance (Wood & Chollet, 2022). Overall, these graphs illustrate the model's improved learning across localization, classification, and overall detection performance, culminating in robust accuracy by the final epoch.

4.7 Addressing Class Imbalance

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%| ▌▌▌▌▌▌▌▌▌▌ | 266/266 [00:
all	4246	7242	0.616	0.463	0.493	0.341
Bus	203	275	0.8	0.749	0.792	0.651
Bushes	207	437	0.815	0.661	0.654	0.473
Person	191	1023	0.462	0.101	0.147	0.0634
Truck	37	45	0.696	0.333	0.418	0.321
backpack	3	3	0	0	0	0
bench	2	3	1	0	0	0
bicycle	290	358	0.646	0.606	0.645	0.392
boat	2	4	0	0	0	0
branch	21	62	0.558	0.387	0.467	0.269
car	228	463	0.75	0.611	0.665	0.477
chair	1	1	0	0	0	0
clock	2	2	0	0	0	0
crosswalk	291	354	0.891	0.875	0.918	0.731
door	176	196	0.827	0.964	0.965	0.787
elevator	200	409	0.789	0.686	0.731	0.574
fire_hydrant	200	220	0.955	0.973	0.986	0.682
green_light	297	362	0.664	0.812	0.769	0.507
gun	217	261	0.796	0.643	0.722	0.49
handbag	3	6	0	0	0	0
motorcycle	59	238	0.578	0.378	0.451	0.215
person	25	43	0.495	0.535	0.561	0.357
pothole	29	95	0.643	0.411	0.478	0.231
rat	191	247	0.856	0.889	0.941	0.697
red_light	221	293	0.912	0.904	0.927	0.534

Figure 7. *Figure illustrating class imbalance within the dataset.*

During the training phase, certain object classes consistently exhibited a mean Average Precision (mAP) of zero, indicating the model's inability to detect these classes effectively. This issue is likely due to the underrepresentation of these classes in the training dataset, leading to insufficient learning during the training process.

To maintain system efficiency and enhance user experience, these underperforming classes, as seen in Figure 7, were excluded from the graphical user interface (GUI) integration. Some of the excluded classes included handbag, chair, and boat. By focusing on classes with higher detection accuracy, the Vision Aid system provides more reliable and relevant feedback to users. This approach highlights the importance of addressing class imbalance in training datasets to improve model performance across all intended object classes.

5. System Architecture and Design

5.1 Overall System Design

Vision Aid's high-level architecture consists of three interconnected modules: Object Detection, Audio Feedback, and Graphical User Interface (GUI). This architecture is designed to support visually impaired individuals by providing real-time feedback about their surroundings, using a streamlined and accessible interface.

1. **Object Detection Module:**

   This module processes live video feed captured by the device's camera using the fine-tuned YOLOv8 trained with 50 epochs. The model is able to detect various outdoor objects that are relevant to visually impaired users, including vehicles, pedestrians, and obstacles. The system determines their type, position, and distance relative to the user. The models real-time performance and accuracy allow vision aid to provide timely alerts, making it suitable for dynamic environments.

2. **Audio Feedback Module:**

   This module translates detected objects into auditory cues, guiding users with relevant information about their surroundings. This module also provides feedback to the user regarding the Vision Aid system, for their understanding without the need for the use of sight. In terms of object detection, each audio cue conveys the type of object, proximity, and direction (left, centre, or right). Priority objects, such as approaching vehicles or nearby pedestrians, generate immediate alerts to help the user respond quickly. The feedback is designed to be clear and concise, providing essential

information without overwhelming the user with unnecessary alerts.

```
def voice_control(self):
    """Voice control for basic commands."""
    while True:
        if not self.voice_control_enabled:
            continue
        with sr.Microphone() as source:
            recognizer.adjust_for_ambient_noise(source)
            try:
                audio = recognizer.listen(source)
                command = recognizer.recognize_google(audio).lower()
                self.add_log(f"Voice command received: {command}")

                if self.awaiting_instruction_response and command in ["yes", "no"]:
                    self.comm.instructions_signal.emit(command)
                elif command == "start detection":
                    self.comm.start_detection_signal.emit()
                elif command == "stop detection":
                    self.comm.stop_detection_signal.emit()
                elif command in ["settings", "settings menu"]:
                    self.comm.settings_feedback_signal.emit()  # Emit signal for settings feedback
                elif command == "exit":
                     self.comm.exit_signal.emit()  # Emit exit signal on voice command
                elif command in ["list commands", "voice help", "commands", "help", "voice commands"]:
                    self.comm.speak_commands_signal.emit()  # Use the signal to call speak_voice_commands
                elif command in ["high contrast on", "contrast on"]:
                    self.comm.toggle_high_contrast_signal.emit(True)
                elif command in ["high contrast off", "contrast off"]:
                    self.comm.toggle_high_contrast_signal.emit(False)
                elif command == "distance on":
                    self.comm.toggle_distance_signal.emit(True)
                elif command == "distance off":
                    self.comm.toggle_distance_signal.emit(False)
                elif command == "guidance on":
                    self.comm.toggle_guidance_signal.emit(True)
                elif command == "guidance off":
                    self.comm.toggle_guidance_signal.emit(False)
                elif command in ["increase volume", "volume up"]:
                    self.comm.volume_up_signal.emit()
                elif command in ["decrease volume", "volume down"]:
                    self.comm.volume_down_signal.emit()
                elif command == "toggle bus":
                    current_state = self.object_filters["bus"]
                    self.comm.toggle_bus_signal.emit(not current_state)
                elif command == "toggle bench":
                    current_state = self.object_filters["bench"]
                    self.comm.toggle_bench_signal.emit(not current_state)
                elif command in["toggle bush", "toggle bushes"]:
                    current_state = self.object_filters["bushes"]
                    self.comm.toggle_bushes_signal.emit(not current_state)
                elif command == "toggle bicycle":
                    current_state = self.object_filters["bicycle"]
                    self.comm.toggle_bicycle_signal.emit(not current_state)
```

Figure 8. Some of *Vision Aid's voice control functions.*

3. **Voice Control Module:**

   Vision Aid's voice control module enables hands-free operation through voice commands, allowing users to control the application without needing visual or manual interaction. As seen in figure 8, users can initiate 15 different commands such as "Start detection," "Stop detection," "Settings," and "Exit." This functionality is especially beneficial for users with complete vision loss, offering an intuitive way to operate the system without the need for sight. This module is particularly valuable for fully blind users, as it reduces reliance on physical or visual interaction with the device.

4. **Graphical User Interface (GUI):**

   The GUI provides a visually accessible control panel for Vision Aid, allowing users or caregivers to adjust settings, view real-time logs of detected objects, and monitor system status. Key features of the GUI include customizable settings for detection

sensitivity, volume, object filters, and voice control options. The GUI also includes high-contrast themes and UI scaling, making it easier for partially sighted users to navigate and interact with the interface.

Data flows from the video feed to the object detection module, where each detection triggers a customised audio cue based on the detected object's characteristics and user preferences. The GUI displays real-time visual feedback, logging each announcement for easy reference, while the voice control module allows users to manage the application hands-free.

5.2 Real-Time Feedback System

The real-time feedback system is essential to Vision Aid, transforming object detections into actionable auditory guidance. This module uses **Text-to-Speech (TTS)** to announce detected objects along with systematic feedback to the user. This system enhances users' spatial awareness by providing context-aware instructions based on the type, distance, and direction of each object.

```
"bicycle": {
    "announce_distance": 4,
    "ignore": False,
    "approach_only": True,
    "message": "Bicycle approaching.",
    "reactions": {
        "left": "Move to the right to avoid the bicycle.",
        "center": "Stay on the side to avoid the bicycle in front.",
        "right": "Move to the left to avoid the bicycle."
    }
```

Figure 9. *An example of detection thresholds and outcomes.*

1. **Proximity Alerts:**
   Objects detected within a certain (specified) range trigger proximity-based alerts. For example, a car approaching within 4 metres will generate an alert like, "Car in front. Stay on the sidewalk." Detection distance thresholds for various objects are customised as seen in figure 9.

```
def announce(self, message, label):
    """Announce if not recently announced and prioritize important objects."""
    current_time = time.time()
# Prioritize announcement for high-priority objects like "person" and "car"
    high_priority_labels = {"person", "bus", "car","bench", "motorcycle", "bicycle", "crosswalk","Truck", "Scooter"}

    if label not in self.last_announced or (current_time - self.last_announced[label] > ANNOUNCEMENT_INTERVAL):
        self.last_announced[label] = current_time
        self.add_log(message)

    # If it's a high-priority object, announce it immediately
        if label in high_priority_labels:
            with self.tts_lock:
                self.tts_busy = True
                engine.say(message)
                engine.runAndWait()
                self.tts_busy = False
        else:
        # Add lower-priority objects to the queue
            self.announcement_queue.put((message, label))
```

Figure 10. *Vision Aid's priority feedback system.*

2. **Priority Objects:**

   Vision Aid prioritises high-importance objects such as vehicles and pedestrians, which are given precedence in the feedback queue. When multiple objects are detected, the system announces high-priority objects first to ensure user safety, as depicted in figure 10. For example, if there were a car and a bench detected, the car would be announced. This prioritisation helps the user focus on the most critical hazards in complex environments.

3. **Dynamic Feedback:**

   The system continuously monitors the user's surroundings and stops announcing objects once they leave the frame, keeping the feedback relevant to the current context. By ignoring objects that exit the frame, Vision Aid reduces unnecessary information, preventing cognitive overload and helping the user focus on active, nearby hazards.

4. **Guidance Based on Direction:**

   Vision Aid provides guidance instructions based on the detected object's direction relative to the user, stored in the object_rules dictionary (see figure 9). For instance, if a bicycle approaches from the left, the system might announce, "Move to the right to avoid the bicycle." This spatial feedback enhances navigation and safety, especially in busy or complex environments.

Vision Aid's real-time feedback system ensures that users receive only essential, actionable information to navigate safely and confidently.

5.3 Voice Control Module

The voice control module enables users to operate Vision Aid entirely through spoken commands, making it particularly accessible for those with complete vision loss. Key features of the voice control module include:

1. **Voice Commands**

   The system supports 41 different commands, some of which can be seen in figure 7; “yes”, “no”, “start detection”, “stop detection”, “settings”, “list commands”, “exit”, “guidance on”, “guidance off”, “distance on”, “distance off”, “high contrast on”, “high contrast off”, “increase volume”, “decrease volume”, “back to main”, and “toggle” for each individual object. This feature makes Vision Aid highly accessible for users with complete vision loss, who may rely entirely on auditory feedback for interaction.

2. **Command Feedback**

   After each command, Vision Aid provides immediate verbal feedback to confirm the action, such as “Detection started” or “Settings opened.” This feedback loop helps users know that their commands have been successfully registered without needing visual confirmation, creating a more seamless and reassuring experience. This is especially valuable for users with visual impairments, as it prevents uncertainty and reduces potential frustration when interacting with the device.

3. **Enhanced Accessibility**

   Voice control is particularly beneficial for users who may struggle with touchscreen interfaces, which often rely on precise tapping or swiping gestures. The Voice Control Module enables users to operate Vision Aid entirely through speech, removing the need for precise motor skills or sighted assistance. This hands-free capability allows visually impaired users to maintain independence, as they can easily issue commands while navigating their environment. For those who prefer not to use voice commands, Vision Aid’s settings allow voice control to be easily disabled, making the system adaptable to diverse user preferences and needs.

By supporting hands-free operation, providing real-time command feedback, and reducing reliance on visual or physical interaction, the Voice Control Module helps make Vision Aid a powerful and user-friendly tool for visually impaired individuals. This module enhances

accessibility, increases user confidence, and promotes greater autonomy, allowing users to navigate complex environments more safely and efficiently.

5.4 Graphical User Interface (GUI)

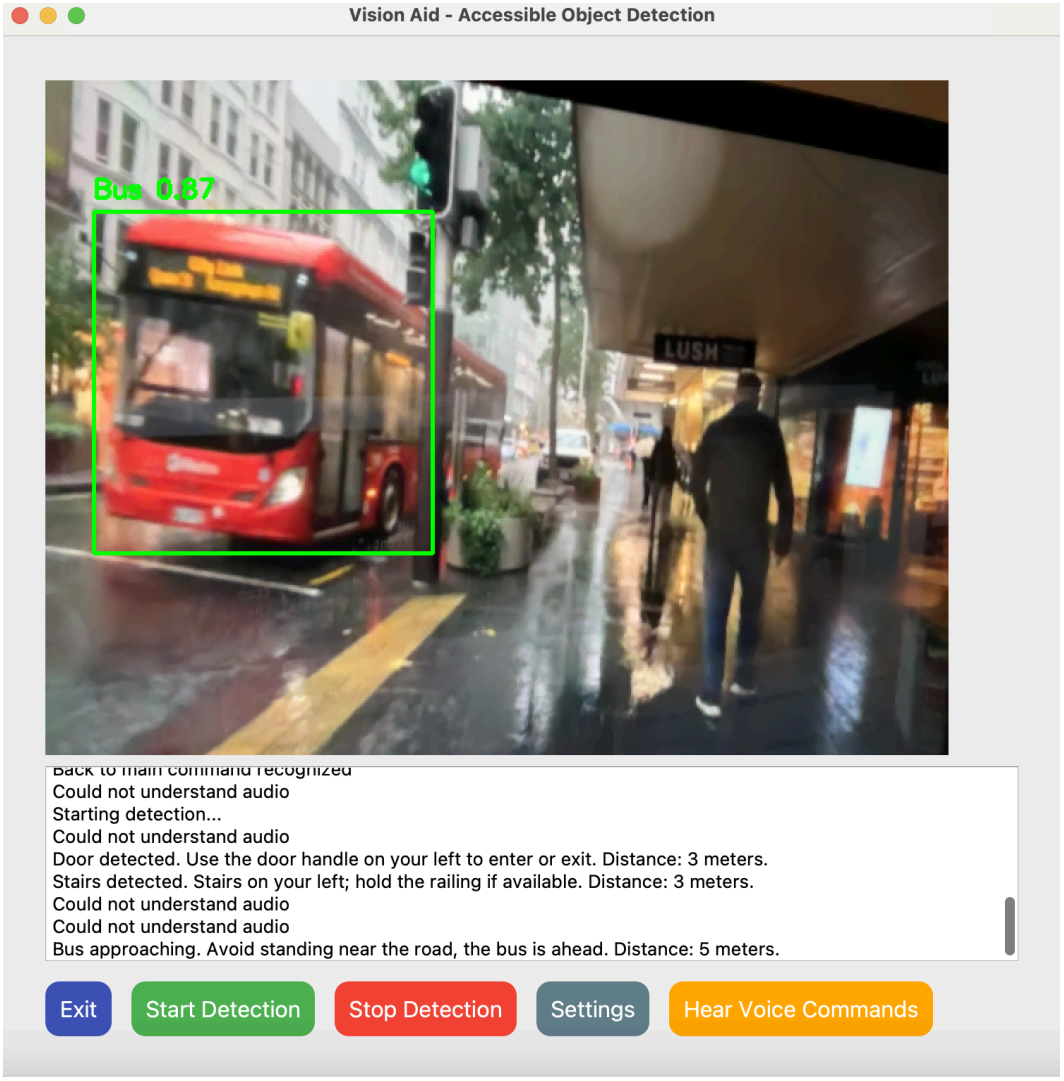


Figure 11. *Vision Aid's GUI during live detection.*

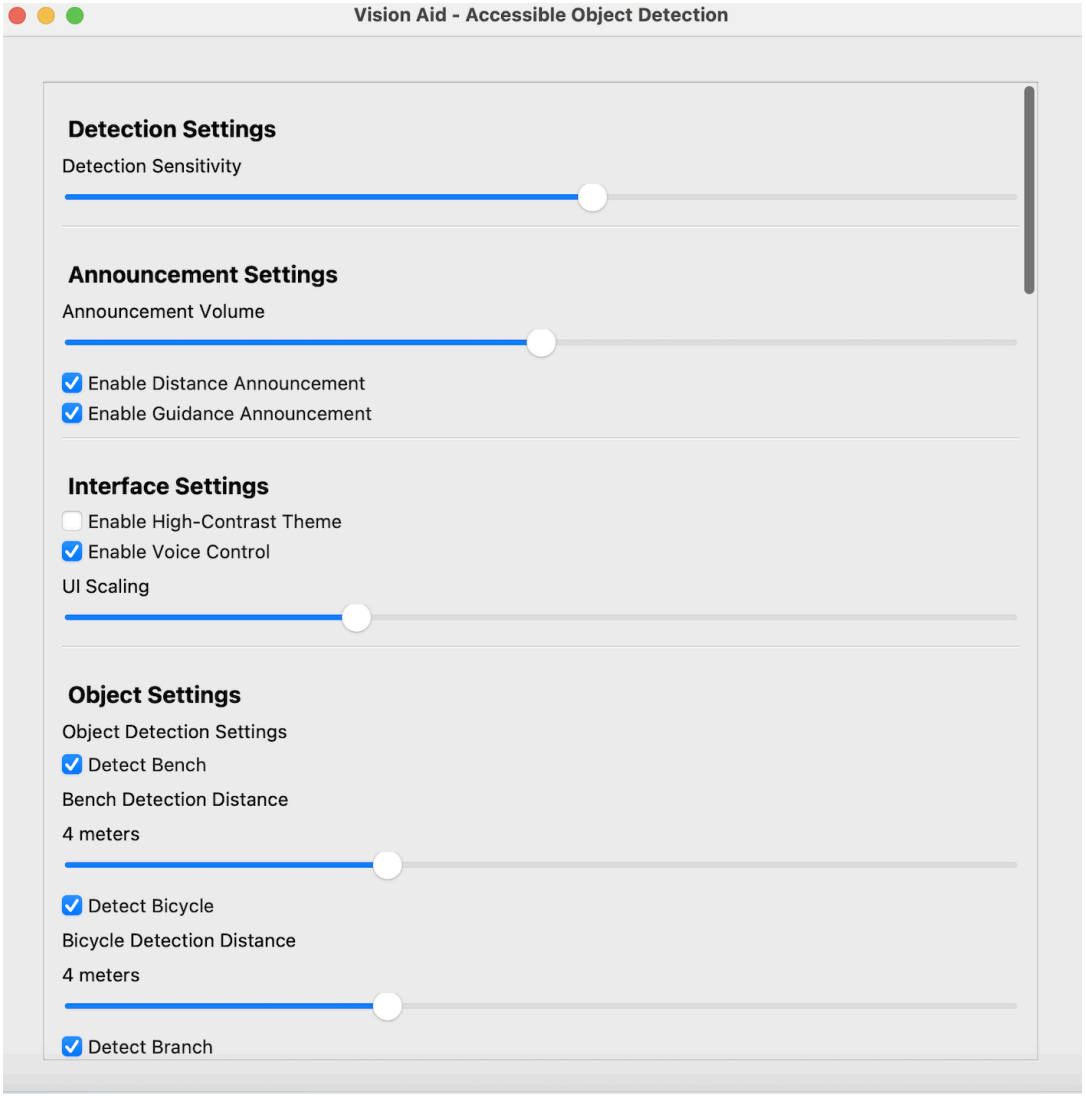


Figure 12. *Vision Aid's Settings menu.*

The GUI is designed with accessibility at its core, offering customization options to cater to the needs of both partially and fully visually impaired users. Key features of the GUI include:

1. **Main Menu:**

   The Main Menu is where users can access essential functions to control and monitor object detection. It provides real-time feedback and object detection results, ensuring that users can make quick decisions based on the detected objects and audio feedback. Key components of the Main Menu include:

   - **Live Detection Feed:** A video display shows the live feed from the camera, with bounding boxes and labels overlaid on detected objects (e.g., "Bus 0.87"

as seen in figure 11). This visual feedback, though primarily for partially sighted users or caregivers, and for validation of the system's detections.

- **Detection Log:** A real-time log below the video feed displays the detected objects, their distances, and any specific guidance (e.g., "Bus approaching. Avoid standing near the road, the bus is ahead. Distance: 5 metres"). This log is particularly useful for partially sighted users who can review alerts visually.
- **Control Buttons:** The Main Menu's design, as seen in Figure 11, enables quick access to essential controls, allowing users to toggle detection and settings based on real-time needs. Control buttons include start detection, stop detection, settings, hear voice commands and exit. These buttons are coloured to aid those with partial vision loss, upon clicking the button the user also receives auditory feedback.

2. **Settings Menu**

The Settings Menu, Figure 12, allows users to personalise Vision Aids behaviour, tailoring the detection and feedback to their specific needs and environment. This customization is crucial for visually impaired users, as it ensures that the system adapts to diverse navigation requirements and a range of degrees of visual impairments. Key sections within the Settings Menu include:

- **Announcement Settings:** Users can set the volume of audio feedback, ensuring it is suitable for both quiet and noisy environments. This can be set via voice command or by using the slider in the settings menu. Both distance and guidance functionality can be toggled on or off in the settings or via voice command, this can also be achieved while using detection.
- **Interface Settings:**
    1. **High Contrast Theme:** Designed for partially sighted users, this option switches the interface to a high-contrast theme (Figure 12). This can be selected via voice command or in the settings menu.
    2. **Voice Control:** Voice control can be toggled on and off (Figure 12)
    3. **UI Scaling:** Adjusts font size and layout scaling, enhancing readability and ease of use for users with low vision.
- **Object Settings:** Users can choose which objects to detect, minimising unnecessary alerts. This section allows toggling individual objects (like benches, bicycles, and cars) on or off. This can be achieved via voice command, also, while using detection. Each object category also has a slider

to customise the detection distance, providing flexibility based on user needs and environmental context.

TheThe Settings menu's organised structure and customizable options, supported by PyQt widgets, ensure Vision Aid is adaptable to various accessibility needs. This approach, with PyQt's interactive interface and OpenCV's robust image processing, creates a comprehensive experience for visually impaired users.

---

6. System Results, Challenges, and Limitations

Vision Aid was rigorously tested across various outdoor settings, from open pathways to challenging environments like crowded sidewalks and urban intersections. The system showcased strong capabilities in detecting a range of outdoor objects, including both moving elements such as pedestrians, bicycles, and vehicles, and stationary hazards like benches, traffic cones, and potholes. In open spaces, Vision Aid provided timely and accurate alerts, enabling users to navigate with minimal disruptions. The YOLOv8 model's object detection capabilities were a significant factor in enhancing users' situational awareness.

However, certain limitations became apparent in more complex environments and under challenging conditions. In cluttered spaces, where multiple objects are densely positioned close together, Vision Aid occasionally generates overlapping or consecutive alerts that could overwhelm users. Despite the object prioritisation algorithm designed to filter and sequence alerts based on object proximity, misidentifications sometimes occurred, particularly with large, similar-looking objects like buses and trucks. This issue likely stems from an imbalanced training dataset, which could benefit from a wider variety of examples for certain object classes. Expanding and balancing the dataset would improve the model's ability to accurately differentiate between similar objects, enhancing reliability in complex environments.

Furthermore, Vision Aid's performance was impacted by very extreme weather conditions, such as heavy rain or snow, and low-light or dark settings. In these environments, the accuracy of object detection decreased, making alerts less reliable. Dark settings, in particular, posed challenges, as the model's ability to identify objects was hindered by insufficient lighting. Incorporating additional data representing varied weather and lighting

conditions could enhance the model's resilience and adaptability, ensuring it performs reliably across a broader range of scenarios.

Users also noted that the GUI, while effective, could be more responsive and stable. Occasional lags and crashes were reported during prolonged use, which can be disruptive in real-world navigation. Optimising the GUI for smooth performance would improve user experience, particularly in situations where quick access to settings or alerts is necessary.
Voice control, while useful, was another area where users expressed a desire for expansion. While Vision Aid includes several essential voice commands, users indicated that a broader range of commands would improve usability, especially in dynamic outdoor settings where hands-free control is critical. Expanding the voice command set to cover more customization options and object-specific interactions would increase user autonomy and adaptability, allowing for on-the-go adjustments tailored to their specific needs and environments.

---

7. Conclusion and Future Improvements

Vision Aid was designed with the objective of empowering visually impaired individuals to navigate complex outdoor environments safely and independently. Through real-time object detection, hands-free audio feedback, and customizable settings, Vision Aid aims to bridge critical gaps in existing assistive technologies. Unlike many available solutions, Vision Aid offers detailed situational awareness by identifying a wide range of objects — from moving elements like pedestrians and bicycles to stationary hazards such as benches and potholes. By enabling users to adjust detection sensitivity, filter objects, and use voice commands, Vision Aid provides a tailored and adaptable navigation tool that meets diverse needs and environmental conditions. Its combination of robust detection, accessible feedback, and user customization signifies a meaningful advancement in assistive technology for the visually impaired community.

7.1 Future Improvements

Moving forward, several development opportunities can enhance Vision Aid's usability, accuracy, and accessibility.

1.**Mobile Application Development**: A mobile app version of Vision Aid would increase accessibility, allowing users to run the system on devices they already own. Integrating Vision Aid's functionality into a smartphone app could significantly broaden its reach, making it more accessible for users who may not have access to specialised equipment. Leveraging smartphone sensors such as GPS and accelerometers could add context to object detection, such as guiding users on a predefined route or avoiding specific areas. Additionally, using smartphone cameras as an input source would make Vision Aid more flexible and portable.

2.**Enhanced Training and Model Accuracy**: Expanding the dataset with more balanced, varied data across object classes and training the YOLO model with additional epochs could lead to increased accuracy, especially in distinguishing between similar objects (e.g., bus versus truck). Training with data that represents various weather conditions, lighting levels, and diverse urban and rural environments would also improve performance. With further tuning and longer training durations, Vision Aid could achieve more reliable object detection in complex and challenging settings.

3.**Multi-Sensor Integration**: Incorporating additional sensors, such as LiDAR for depth perception or thermal imaging for low-light environments, could enhance the robustness of Vision Aid. These sensors would help the system navigate in dark settings, where visual data alone may be insufficient. LiDAR, for example, could provide accurate distance measurements, enabling more precise spatial awareness and avoiding collisions even in densely packed or dimly lit areas. Additionally, integrating environmental sensors to detect weather conditions (e.g., rain or snow) could allow Vision Aid to adjust detection sensitivity dynamically.

4.**Improved Feedback Mechanisms**: Expanding the voice command set and refining audio feedback would enhance user experience. A broader array of voice commands, including context-specific actions and setting adjustments, would increase user autonomy. Additionally, implementing tactile feedback, such as vibration patterns for high-priority objects, could provide an alternative alert method when audio may be challenging to hear.

5.**Scaling for Broader Use Cases**: Beyond navigation for visually impaired individuals, Vision Aid's framework could be adapted for other applications, such as

assisting elderly individuals with mobility challenges or providing situational awareness for workers in hazardous environments. This could also be extended to include indoor settings. By fine-tuning the detection model and enhancing environmental adaptability, Vision Aid has the potential to become a versatile assistive tool applicable to a range of safety-critical scenarios.

By focusing on these future developments, Vision Aid could continue to evolve into a powerful, adaptive solution that supports visually impaired individuals and broadens the scope of real-time assistive technology.

---

8. Task List

8.1 Smriti

1. Object Detection System Setup

- Implementation of YOLOv8 model for object detection within the Vision Aid system.
- Experimentation with different models
- Set up and configure object classes, ensuring each object is correctly recognized (e.g., person, bus, truck).
- Test initial model accuracy and make adjustments as needed for high-priority objects (e.g., pedestrians and vehicles).

2. Dataset Inspection and Cleaning

- Inspect dataset structure
- Class imbalance
- Any required cleaning processes

3. Model Optimization

- Train the YOLOv8 model with more epochs to enhance detection accuracy.
- Evaluate and fine-tune detection sensitivity, especially for complex environments (e.g., crowded sidewalks).
- Test distance estimation improvements and adjust model parameters to improve detection in darker settings.

4. System Testing in Real Environments

- Conduct testing in diverse environments (e.g., busy intersections, open areas).
- Document performance issues in cluttered or high-density spaces and note overlapping alerts.
- Analyse detection accuracy during extreme weather or low light, highlighting limitations for future work.

8.2 Gloria

1. **GUI Development and Customization**

- Design and develop a clear, accessible GUI, focusing on high-contrast, colourful options for partial vision loss.
- Implement buttons and controls for system customization (e.g., sensitivity adjustment, object filtering).
- Optimise GUI performance to prevent crashes, especially when navigating between views or settings.

2. **Voice Command Integration**

- Implement voice command functionality for primary commands (e.g., start detection, stop detection).
- Test and add secondary commands for more control options (e.g., toggle individual objects, enable/disable settings).
- Ensure microphone control is managed to prevent the system from recognizing its own feedback.

3. Feedback Mechanisms

- Customise audio feedback for object detection, adjusting the system to give clear, prioritised alerts.
- Test guidance and distance announcements, refining phrasing and timing for user clarity.
- Experiment with audio feedback timing in cluttered environments to prevent overwhelming the user.

4. User Testing and Evaluation

- Conduct user testing sessions in different environments to collect feedback on the overall user experience.
- Collect specific feedback on ease of navigation, effectiveness of alerts, and usability of the GUI.
- Document all feedback and identify improvement areas for future updates (e.g., smoother GUI, expanded voice command options).

5. System Testing in Real Environments

- Conduct testing in diverse environments (e.g., busy intersections, open areas).
- Document performance issues in cluttered or high-density spaces and note overlapping alerts.
- Analyse detection accuracy during extreme weather or low light, highlighting limitations for future work.

8.3 Collaborative tasks

1. Dataset selection

- Selection of an appropriate dataset for Vision Aid

2. Documentation and Reporting

- Collaboratively write the final report, summarising project objectives, methodology, results, and findings.
- Highlight system strengths, limitations, and suggested improvements based on testing feedback.

3. Future Development Planning

- Brainstorm potential future enhancements, such as integrating additional sensor data or scaling to a mobile app.
- Propose strategies for addressing model limitations (e.g., additional data collection, longer training periods).

9. References

1. *26 Class Object detection dataset*. (2024, February 6). https://www.kaggle.com/datasets/mohamedgobara/26-class-object-detection-dataset

2. Brownlee, J. (2019, August 6). *How to use Learning Curves to Diagnose Machine Learning Model Performance*. MachineLearningMastery.com. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/
3. *CAL - NavCog*. (n.d.). https://www.cs.cmu.edu/~NavCog/navcog.html
4. Computer vision engineer. (2023, January 30). *Train Yolov8 object detection on a custom dataset | Step by step guide | Computer vision tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=m9fH9OWn8YM
5. Dey, I. (2023, July 12). What is YOLO algorithm - Ishani Dey - Medium. *Medium*. https://medium.com/@ishudey11032002/what-is-yolo-algorithm-ef5a3326510b
6. Kundu, R. (n.d.). YOLO: Algorithm for Object Detection Explained [+Examples]. *V7*. https://www.v7labs.com/blog/yolo-object-detection
7. Mehra, A. (2024, August 8). *Evolution of YOLO Object Detection Model From V5 to V8*. Labellerr. https://www.labellerr.com/blog/evolution-of-yolo-object-detection-model-from-v5-to-v8/
8. *OrCam MyEye 3 Pro - Revolutionize Your Vision with Cutting-Edge AI Technology*. (2024, February 21). OrCam Techologies. https://www.orcam.com/en-us/orcam-myeye-3-pro?srsltid=AfmBOorkDCsHQCPyMPLMdXitHFYzou0chXDo2kYkgHA5rPBMRW71172L
9. Rath, S., & Rath, S. (2024, August 2). *Train YOLOv8 on Custom Dataset – A Complete Tutorial*. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Code, & Tutorials. https://learnopencv.com/train-yolov8-on-custom-dataset/

10. Skalski, P. (2024, September 23). *How to Train YOLOv8 Object Detection on a Custom Dataset*. Roboflow Blog. https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/
11. Sun, Y., Sun, Z., & Chen, W. (2024). The evolution of object detection methods. *Engineering Applications of Artificial Intelligence*, *133*, 108458. https://doi.org/10.1016/j.engappai.2024.108458
12. Torres, J. (2024, September 9). What is Box Loss in YOLOv8? Understanding Detection Metrics. *YOLOv8*. https://yolov8.org/what-is-box-loss-in-yolov8/
13. Ultralytics. (n.d.). *Two-Stage Object Detectors - Discover the precision of two-stage object detectors in computer vision, ideal for tasks demanding high accuracy in autonomous vehicles and healthcare imaging.* https://www.ultralytics.com/glossary/two-stage-object-detectors
14. *WeWALK Smart Cane – Smart Cane for the Visually Impaired*. (n.d.). https://wewalk.io/en/
15. *What is loss_cls and loss_bbox and why are they always zero in training*. (n.d.). Stack Overflow. https://stackoverflow.com/questions/54977311/what-is-loss-cls-and-loss-bbox-and-why-are-they-always-zero-in-training
16. Wood, L., & Chollet, F. (2022, July 21). *Efficient Graph-Friendly COCO Metric Computation for Train-Time Model Evaluation*. arXiv.org. https://arxiv.org/abs/2207.12120
17. *YOLO vs SSD: Which One is a Superior Algorithm | Algoscale*. (n.d.). Algoscale. https://algoscale.com/blog/yolo-vs-ssd-which-one-is-a-superior-algorithm/
18. *YOLOv8 dfl_loss metric*. (n.d.-a). Stack Overflow. https://stackoverflow.com/questions/75950283/yolov8-dfl-loss-metric