SAFE STEP: REAL-TIME AI-ENABLED NAVIGATION AID FOR THE VISUALLY IMPAIRED

A Project Report

Submitted by

HARINI SRI B (210171601017)

Under the guidance of

Dr.N.Sabiyath Fatima

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

APRIL 2025

SAFE STEP: REAL-TIME AI-ENABLED NAVIGATION AID FOR THE VISUALLY IMPAIRED

A Project Report

Submitted by

HARINI SRI B (210171601017)

Under the guidance of

Dr.N.Sabiyath Fatima

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

APRIL 2025

B.S. Abdur Rahman®
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

BONAFIDE CERTIFICATE

Certified that this project report "**SAFE STEP: REAL-TIME AI-ENABLED NAVIGATION AID FOR THE VISUALLY IMPAIRED**" is the Bonafide work of "**HARINI SRI B (210171601017)**" who carried out the project work under my supervision. Certified further that to the best of our knowledge, the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. N. Sabiyath Fatima
Supervisor
Professor
Department of CSE
B S Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600048

SIGNATURE
Dr. Aisha Banu W
Head of the Department
Professor
Department of CSE
B S Abdur Rahman Crescent
Institute of Science and Technology
Vandalur, Chennai – 600048

B.S. Abdur Rahman®
Crescent
Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956

VIVA VOCE EXAMINATION

The viva voce examination of the project work titled "**SAFE STEP: REAL-TIME AI-ENABLED NAVIGATION AID FOR THE VISUALLY IMPAIRED**", submitted by **HARINI SRI B – 210171601017** is held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely express our heartfelt gratitude to Prof. **Dr T. MURUGESAN**, Vice-chancellor and **Dr N. THAJUDDIN**, Pro-Vice Chancellor, B.S. Abdur Rahman Crescent Institute of Science and Technology, for providing us with an environment to carry out our course successfully.

We sincerely thank **Dr. N. RAJA HUSSIAN**, Registrar for furnishing every essential facility for doing our project.

We thank **Dr. SHARMILA SANKAR**, Dean, School of Computer, Information and Mathematical Sciences for her motivation and support.

We thank **Dr. AISHA BANU W,** Professor and Head, Department of Computer Science and Engineering, for providing strong oversight of vision, strategic direction and valuable suggestions.

We express our sincere thanks to the Project Review Committee members, from the Department of Computer Science and Engineering, **Dr. X. ARPUTHA RATHINA**, Associate Professor and **Dr. S. SUBHASHINI,** Assistant Professor (Senior Grade) for their valuable suggestions and support.

We obliged our project supervisor, **Dr. N. SABIYATH FATIMA,** Professor, Department of Computer Science and Engineering for her professional guidance and continued assistance during our project.

We thank our class advisor and project coordinator, **Dr. J. BRINDAMERIN,** Assistant Professor (Senior Grade), Department of Computer Science and Engineering for her guidance and encouragement throughout our project.

We thank all the **faculty members** and the **System staff** of the Department of Computer Science and Engineering for their valuable support and assistance at various stages of project development.

HARINI SRI B

ABSTRACT

Assistive technology for visually impaired individuals has made significant strides with innovations like smart glasses, smart canes, and wearable devices. These solutions have improved individuals' lives and help them navigate, which gives them a sense of self-reliance. These solutions use artificial intelligence, computer vision, and sensor technology to provide real-time assistance. However, despite the advancement, a major limitation persists - these technologies are prohibitively expensive. The high cost of these devices makes them out of reach for a large proportion of people, limiting their accessibility.

This is a common issue; these solutions should not be accessible to those who can afford premium assistive devices. Many visually impaired people struggle with financial constraints, making it impossible for them to access these advanced technologies.

The primary focus of this proposed system is to develop lightweight software that can be integrated into low-cost devices instead of relying on expensive hardware. By making the software platform independent, visually impaired individuals can use the technology on a device they already own, which reduces the cost significantly.

Additionally, the current system primarily focuses on alerting users about the presence of obstacles there is a major gap, and they do not offer adequate guidance on how to overcome them. The proposed system unlike conventional systems that only detect and alert users to obstacles, this system goes a step further by analysing the surroundings to determine how to navigate around them safely.

After detecting an obstacle, the system evaluates spatial information, such as the distance, position, and possible alternate paths. It then plans the safest and most efficient route to bypass the obstacle. Once the path is determined, real-time auditory guidance is provided, informing the user about the nature of the obstacle and instructing them on how to proceed.

In conclusion, the proposed system presents a novel approach to assistive technology for the visually impaired, focused on both obstacle detection and navigation guidance. By using widely available and low-cost devices, this solution has the potential to significantly improve the quality of life for individuals with visual impairments, enabling them to move independently and confidently in their surroundings.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This project proposes a real-time navigation assistance system for visually impaired individuals. The system processes live video input, detects objects using YOLOv11 trained by transfer learning, and estimates proximity with Monocular Size-based Proximity Estimation. To navigate around obstacles, a heuristic-based decision-making approach is used, analysing the surroundings and determining the safest path forward. Finally, real-time audio feedback is provided to guide users step by step. Designed for affordability and adaptability, the system can be deployed on low-cost devices, making advanced navigation assistance more accessible.

1.1. OVERVIEW

The proposed navigation assistance system addresses a crucial challenge faced by visually impaired individuals: guidance for safe outdoor navigation. While existing assistive technologies provide obstacle detection, they often lack actionable navigation instructions, leaving users uncertain about how to move around detected obstacles. Additionally, many solutions are expensive and inaccessible.

This project leverages deep learning and heuristic-based decision-making to create an affordable, real-time audio navigation system. The system processes video input from a camera, detects obstacles using a YOLOv11 trained by transfer learning, and estimates proximity with Monocular Size-based Proximity Estimation. A heuristic-based navigation module analyses the surroundings and determines the safest path forward. Finally, audio feedback is generated, guiding the user with clear step-by-step instructions.

The overall system architecture is designed for lightweight deployment on low-cost devices such as mobile phones or wearable cameras. The input video undergoes preprocessing before being processed by the YOLOv11 model for object detection. Detected obstacles are then evaluated using Monocular Size-based Proximity Estimation to determine spatial positioning. The heuristic-based module interprets the obstacle’s surroundings to generate a safe path, ensuring that users can not only detect obstacles but also receive directional guidance on how to navigate safely.

Unlike traditional solutions that focus solely on alerting users about obstacles, this system actively provides actionable navigation assistance, making it a more intelligent and accessible alternative. By reducing dependency on costly hardware and ensuring real-time performance, this project aims to improve the mobility, safety, and independence of visually impaired individuals in outdoor environments.

1.2. DESCRIPTION

Navigating outdoor environments independently is a major challenge for visually impaired individuals. While existing assistive technologies offer basic obstacle detection, they often fail to provide real-time guidance on how to navigate around obstacles, making them less effective in real-world scenarios. Additionally, most of these solutions rely on expensive hardware, limiting accessibility for a wider population.

This project aims to develop an affordable, real-time navigation assistance system that utilizes computer vision, deep learning, and heuristic-based navigation to help visually impaired individuals move safely in outdoor environments. The system processes live video input from a mounted camera, detects obstacles using YOLOv11 trained by transfer learning, and estimates proximity with Monocular Size-based Proximity Estimation. Instead of merely alerting users about obstacles, the system employs a heuristic-based decision-making approach to analyse the surroundings and generate an optimal navigation path. The final output is a real-time audio feedback system that guides the user step by step, ensuring a safe and smooth navigation experience.

The core components of the system include:

- Object Detection – Uses a YOLOv11 trained by transfer learning to identify obstacles in the user's path.
- Depth Estimation – Utilizes Monocular Size-based Proximity Estimation to determine the relative distance of detected objects.
- Navigation Module – Implements a heuristic approach to determine the safest way to move around obstacles.
- Audio-Based Feedback – Converts navigation instructions into real-time auditory guidance for the user.

This project is designed to be lightweight and adaptable, making it deployable on low-cost devices, including smartphones or wearable cameras. Unlike traditional assistive technologies, this system provides actionable guidance rather than simple alerts, significantly improving the mobility, safety, and independence of visually impaired individuals.

1.3. OBJECTIVE

The primary objective of this project is to develop a real-time, cost-effective navigation assistance system for visually impaired individuals that goes beyond mere obstacle detection by providing intelligent auditory guidance for safe navigation. Unlike existing solutions that only alert users about obstacles, this system aims to analyse the environment, determine an optimal path, and provide step-by-step instructions to help users move around obstacles independently.

- **Develop an Affordable and Scalable Solution** - Create a software-based navigation system that can be deployed on low-cost devices like smartphones and wearable cameras. Reduce dependency on expensive specialized hardware like smart canes and AR glasses.
- **Enhance Obstacle Detection Accuracy** - Improve object detection performance to ensure high precision in dynamic environments (e.g., streets, sidewalks).
- **Implement Depth Estimation for Distance Awareness** - Provide users with better situational awareness by integrating distance information into navigation guidance.
- **Enable Intelligent Path Planning for Safe Navigation** - Ensure the system not only detects obstacles but also guides users on how to avoid them effectively.
- **Provide Real-Time Audio Feedback for Intuitive Interaction** - Convert navigation instructions into natural language audio guidance, allowing users to receive step-by-step movement directions. Ensure the system responds dynamically to real-time environmental changes.
- **Optimize for Real-Time Performance and Low Computational Overhead** - Ensure the model runs efficiently on devices with limited processing power without compromising performance. Minimize latency

in object detection, depth estimation, and audio output to provide a seamless user experience.

This project ultimately seeks to enhance mobility, independence, and safety for visually impaired individuals by providing an intelligent, real-time navigation system that is both affordable and practical for everyday use.

1.4. ABOUT THE PROJECT

The motivation behind this project stems from the gaps in existing assistive technologies for visually impaired individuals. While many solutions exist for detecting obstacles, very few guide overcoming them, leaving users uncertain about how to navigate safely. Additionally, these solutions often require specialized hardware, making them prohibitively expensive for a large percentage of visually impaired individuals.

This project addresses these challenges by offering a cost-effective, software-based alternative that can be deployed on smartphones, wearable cameras, or other portable devices. By integrating deep learning models for object detection and depth estimation, along with a heuristic navigation system, it provides real-time, intelligent guidance rather than just alerts. The voice-based feedback mechanism ensures accessibility, enabling users to interact with the system naturally without relying on visual or tactile inputs.

The ultimate goal of this project is to empower visually impaired individuals with a greater sense of independence by making outdoor navigation more intuitive and safe. The system's emphasis on affordability, adaptability, and real-time processing sets it apart from existing solutions and makes it a practical choice for widespread adoption. Future developments could include expanding navigation capabilities to indoor environments, integrating scene understanding, and improving path optimization techniques to further enhance usability.

This project is a step forward in making assistive technology more accessible, intelligent, and effective**,** bridging the gap between obstacle detection and real-time navigation guidance.

1.5. ORGANISATION OF THE PROJECT REPORT

This report is organized into several chapters, each addressing a key aspect of the real-time navigation assistance system for visually impaired individuals.

- **Chapter 1: Introduction**
  The introduction chapter outlines the challenges faced by visually impaired individuals in navigating their surroundings and the limitations of existing assistive technologies. It explains the necessity for an affordable, AI-driven navigation system that not only detects obstacles but also provides real-time guidance to help users navigate safely. This chapter sets the stage for the project by highlighting the integration of computer vision, deep learning, and heuristic-based navigation techniques to develop an effective and accessible mobility aid.
- **Chapter 2: Literature Survey**
  This chapter reviews existing research and technologies in the field of assistive navigation for visually impaired individuals. It examines previous studies on object detection, depth estimation, and audio-based guidance systems, identifying the strengths and limitations of current solutions. The survey highlights the gaps in traditional obstacle detection systems, particularly the lack of intelligent path planning, which this project aims to address through a real-time, AI-driven guidance approach.
- **Chapter 3: System Requirements and Design**
  Chapter 3 details the technical and operational requirements for implementing the navigation assistance system. It discusses the hardware and software prerequisites, outlining the specifications for processing units, memory, storage, and development environments. The design section elaborates on the system architecture, highlighting key modules such as video input processing, object detection, proximity estimation, heuristic-based navigation, and real-time audio feedback for user guidance.
- **Chapter 4: System Methodologies**
  This chapter thoroughly explains the methodologies used to develop and deploy the system. Additionally, it details the heuristic approach for navigation, which analyses detected obstacles and determines the safest path for the user.

The integration of real-time audio feedback ensures that users receive clear instructions on how to navigate around obstacles. This chapter also discusses the evaluation metrics used to assess the system's accuracy, efficiency, and real-world applicability.

- **Chapter 5: Implementation**
  This chapter details the actual implementation of the proposed system. It covers how the models and techniques described earlier were applied in practice, including development tools, testing environments, and performance evaluation.

- **Chapter 6: Result and Discussion**
  This chapter analyses the working and the test results of each module in a real-world environment. From this chapter, the performance of the overall system is evaluated and tested on how well it can be adapted to real-life scenarios

- **Chapter 7: Conclusion and Future Enhancement**
  This final chapter summarizes the outcomes of the project, emphasizing key results and the overall impact of the proposed solution. It reflects on the effectiveness of the system and suggests directions for future enhancements.

CHAPTER 2

LITRATURE SURVEY

Kaipeng Hong, et. al [1] has proposed a solution with lightweight modules which can be deployed easily by using IoT-based cameras. This proposed approach uses multi learning algorithm to simultaneously perform
Scene classification and path detection. Though this approach gave 91.7% accuracy it only analyses paths and signals but not real-world obstacles which limits its usage to real-world scenarios.

Rakesh Chandra Joshi, et. al [2] has proposed a portable wearable assistive device that integrates computer vision and sensor-based technologies to provide real-time auditory information about obstacles. This system offers multiple operational modes and real-time alerts which effective even in low light conditions. However, Ultrasonic sensors are not effective in identifying transparent or low reflective obstacle which is a challenge in this system.

Mukhriddin Mukhiddinov and Jingo Cho [3] proposed a system utilising computer vision and deep learning. It has features like low light enhancement, Object Recognition and audio feedback and text-to-speech and tactile graphic. As tactile feedback interpretation requires training it is difficult for the Visually impaired to adapt to it.

Yahia Said, et. al [4] proposed a wearable assistive device for visually impaired by integrating Raspberry Pi, a camera module and a pretrained CNN into smart glasses. This system identifies objects, estimates distances and provides a real-time auditory or haptic feedback to the user. However, the system effectively detects objects and distances, it relies on advanced hardware that limits implementation.

Bineeth Kuriakose, et. al [5] proposed DeepNAVI, a smartphone-based navigation assistant which uses deep learning to provide real-time audio feedback on obstacle type and details about it. Although this system gives detailed obstacle awareness without requiring extensive training the lack of contextual information makes it difficult for users.

Yasuhiro Nitta, et. al [6] proposed a novel importance rank learning method that identifies the obstacle by prioritising it. It uses a neural network-based ranking estimation to estimate the importance of the obstacles detected. However, the study's dependency on accurate optical flow estimation introduces challenges on dynamic scenarios.

Mooseop Kim, et. al [7] proposed a deep learning-based approach that optimises visual-auditory sensory substitution systems. This study has an improved alignment with human sensory perception. But this study focusses only on auditory substitution which might not work well under all condition.

Fahad Ashiq, et. al [8] has proposed a navigation system that ensures mobility assistance and safety of the visually impaired. This system integrates voice-based navigation and a web application for the family to track the user. The CNN model used in the system achieves 83.3% accuracy in object detection. Although its location sharing feature is extremely useful the system dictates every object in the system which rises confusion and might be misleading for the visually impaired.

Usman Masud, et. al [9] has proposed a system that integrates a Raspberry Pi, camera, an ultrasonic sensor and an Arduino mounted on a walking stick. Using algorithms like Viola-jones algorithm and tenor flow object detection, the framework classifies scenes and detects obstacles. However, the use of an ultrasonic sensor may struggle to detect obstacles like glass or low-height objects which limits the accuracy and reliability of the system.

Alice Lo Valvo, et. al [10] proposed an augmented reality-based navigation system that uses smartphone to provide virtual paths and real-time object recognition through machine learning. This system eliminates the need for physical markers. However, as it relies on pre-recorded paths it limits the performance and usage in the real-world environment.

Jagadish K. Mahendran, et. al [11] proposed a computer vision-based assistance system that utilises deep learning and edge AI for real-time scene understanding. Their approach integrates the OpenCV AI Kit-Depth (OAK-D) sensor for obstacle detection and segmentation. Although this system enhances mobility, more accurate

and efficient navigation experience performance is constrained by the computational limitations of edge AI.

Salvador Martínez-cruz, et. al [12] proposed an assistive device that helps the blind and visually impaired with public transport navigation. The proposed system utilises BLE technology for location tracking and communication. Although the system demonstrated 97.6% accuracy, its reliance on BLE which requires effective placement and maintenance is one of its limitations.

Minnan Leong and R. Kanesaraj Ramasamy [13] have proposed an assistive device integrated into smart glasses. This study focuses on improving spatial awareness using Computer Vision and Machine Learning. The output of the system is auditory and haptic which is easily interpretable by the users. However, the hardware used in the system has processing limitations which causes latency in object detection.

Rakesh Chandra Joshi, et. al [14] proposed a fully automatic assistive technology by recognising objects and providing real-time auditory feedback. With an average accuracy of 95.15%, the system performs well in recognising the objects. However, further testing in diverse environments is needed.

Shubham Melvin Felix, et. al [15] proposed an Android-based mobile application that has features such as voice assistance, object and currency recognition, text analysis etc. This application enables users to navigate, read and engage with the world. However, the application’s performance relies on the quality of the smartphone’s camera quality which affects the accuracy.

CHAPTER 3

SYSTEM REQUIREMENTS AND DESIGN

This section provides a detailed account of the requirements and design of the proposed navigation assistance system for visually impaired individuals, which is built to enhance mobility and safety with precision. The system addresses the inherent challenges of existing assistive technologies by leveraging deep learning, computer vision, and real-time audio feedback to provide intelligent obstacle detection and guided navigation.

3.1 PROBLEM DEFINITION

Assistive technology for visually impaired individuals has made significant strides with innovations like smart glasses, smart canes, and wearable devices. These solutions have improved the lives of individuals by helping them navigate independently and fostering a sense of self-reliance. However, despite these advancements, a significant limitation persists—the high cost of these devices makes them inaccessible to a large proportion of people. Many visually impaired individuals struggle with financial constraints, making it difficult to access advanced technologies.

The primary challenge is the lack of affordable solutions that can provide real-time navigation assistance. Current assistive devices primarily focus on alerting users about obstacles but fail to guide them around these obstacles effectively. This gap in functionality can create difficulties in real-world navigation, where users need both detection and guidance. The objective of this project is to develop a cost-effective, lightweight, and platform-independent solution that not only detects obstacles but also provides intelligent navigation assistance.

3.2 EXISTING SYSTEM

Several assistive technologies have been developed to aid visually impaired individuals in navigating their surroundings. These solutions vary in terms of technology, cost, and effectiveness.

- **Smart Canes**

  Example: WeWALK Smart Can; Technology Used: Ultrasonic sensors, gyroscope, accelerometer, Bluetooth connectivity; Functionality: Detects obstacles using ultrasonic sensors and provides haptic feedback. Some models connect with smartphones for voice-guided navigation; Limitations: Only detects obstacles at cane level, and missing overhead objects (e.g., signboards, hanging branches). Limited navigation assistance—users are only alerted about an obstacle but are not guided around it. High cost (WeWALK costs ~$600+).

- **Smart Glasses and AI-Based Wearables**

  Examples: Envision Glasses, OrCam MyEye; Technology Used: AI-based object recognition, OCR (Optical Character Recognition), text-to-speech, cameras; Functionality: Recognizes objects, faces, and text, providing spoken descriptions to the user. Some models integrate with smartphones for cloud-based processing; Limitations: Extremely expensive (OrCam MyEye costs ~$4,000). Limited scene understanding—detects and describes objects but does not offer real-time navigation assistance. Dependence on cloud processing can cause latency issues.

- **Wearable Navigation Devices**

  Examples: Sunu Band, Strap, Sonic Eye; Technology Used: Ultrasonic sensors, LiDAR, haptic feedback; Functionality: Uses vibration feedback to alert users of obstacles. Some devices scan the environment using LiDAR and ultrasonic waves; Limitations: No clear directional guidance—users only feel vibrations but must figure out where to move. Limited range (some devices detect obstacles only within 2-4 meters). Wearability discomfort—constant vibrations can be distracting or overwhelming.

- **Mobile-Based Navigation Apps**

  Examples: Seeing AI (Microsoft), BlindSquare, WayAround; Technology Used: GPS, AI-based object detection, OCR, text-to-speech; Functionality: Uses AI to describe surroundings, read text, and recognize objects. GPS-based apps

provide navigation instructions for outdoor environments; Limitations: GPS is unreliable for close-range navigation (e.g., avoiding small obstacles). It does not detect physical obstacles in real-time (it mostly helps with location-based information). Many features require internet connectivity for them to work.

- **LiDAR-Based Navigation Systems**

  Examples: iPhone LiDAR for Accessibility, Guide Cane; Technology Used: LiDAR sensors, AI-based depth perception; Functionality: Uses LiDAR sensors to create a 3D map of the environment. Provides detailed depth perception to assist with mobility; Limitations: Lidar-based devices are very expensive (they often cost thousands of dollars) and limited in availability—only high-end smartphones (like iPhone Pro models) and specialized equipment have LiDAR; By addressing these issues, this project aims to provide an accessible, intelligent, and real-time navigation assistance system for visually impaired individuals, offering both obstacle detection and step-by-step navigation guidance.

3.3 PROPOSED SYSTEM

The proposed system is a real-time navigation assistance solution designed for visually impaired individuals, leveraging deep learning models for obstacle detection and spatial awareness. It functions entirely offline and is optimized for lightweight deployment on standard devices like smartphones, webcams, or wearable cameras, ensuring accessibility without the need for specialized hardware or internet connectivity.

At the core of the system is a custom-trained YOLOv11 object detection model, which identifies real-world obstacles such as vehicles, pedestrians, poles, and traffic cones from live video input. The object detection model is fine-tuned on a targeted dataset relevant to road and pedestrian environments to maximize practical utility during outdoor navigation.

To estimate how close an object is to the user, the system uses a bounding box–based depth approximation technique. It calculates the relative height of each detected object’s bounding box with respect to the frame height and categorizes the distance.

This distance classification is lightweight and fast, making it ideal for edge devices without requiring computationally heavy depth estimation networks.

Each detected object is also analysed to determine its position within the frame—categorised as left, centre, or right—based on the horizontal midpoint of the bounding box. The system assigns impact scores to objects based on their type (e.g., cars and buses are considered high-risk, pedestrians are medium-risk), which are further weighted depending on their proximity. This results in a risk score for each direction (left or right), allowing the system to intelligently decide the safest direction of movement.

The navigation logic uses this risk analysis to provide real-time movement suggestions such as:
“Move right” or “Move left” – when an obstacle is very close to the centre.
“Proceed Forward” – when no immediate threat is present.
These instructions are conveyed using text-to-speech audio feedback powered by pyttsx3, which allows spoken directions even in offline mode.
The overall architecture of the system consists of five primary components:

- **Live Video Capture** – from a webcam or external camera.
- **Object Detection** – using YOLOv11 trained on custom classes.
- **Distance Estimation** – based on bounding box height.
- **Navigation Decision** – using object position, proximity, and impact score to assess risk.
- **Audio Feedback** – through speech instructions for intuitive guidance.

This system stands out as a cost-effective, offline-capable, and user-friendly assistive solution that empowers visually impaired individuals to navigate unfamiliar environments with increased confidence and autonomy.

3.4 MODULE IDENTIFICATION

The proposed system is designed with several interconnected components that work together seamlessly.

- **Dataset Acquisition and Preprocessing**

  This module captures real-time video input from a camera and streams frames into the system for processing. It ensures a smooth and continuous video feed while adjusting settings like brightness and frame rate to improve clarity under different lighting conditions.

- **Model Training**

  - **Object Detection and Recognition**

  This module identifies obstacles and objects in the user's surroundings by analysing video frames. It detects various objects such as pedestrians, vehicles, stairs, poles, and curbs, providing contextual awareness for safe navigation.

  - **Distance / Proximity Calculation**

  This module estimates the distance of detected objects from the user, helping differentiate between nearby and distant obstacles. Calculating depth information ensures an accurate assessment of potential hazards.

- **Navigation Assistance and Path Planning**

  This module determines the best path to avoid obstacles and provides real-time guidance. Instead of simply alerting the user about an obstacle, it evaluates different routes and suggests the safest way forward based on obstacle placement and movement patterns.

- **Real-Time Audio Feedback System**

  This module converts processed information into voice-based alerts, guiding users with step-by-step navigation instructions. It informs users about detected

obstacles, suggests safe movement directions, and provides contextual awareness of the environment.

- **Real-Time Integration Module**

  The object detection, distance estimation, and navigation modules work together in a seamless pipeline. Real-time video frames are processed continuously, updating detection and navigation paths dynamically. The integrated system ensures smooth, real-time interaction with minimal latency, improving user experience.

Each of these modules plays a crucial role in delivering an efficient and intelligent navigation system, ensuring that visually impaired individuals can move independently and confidently in their surroundings.

3.5 SYSTEM REQUIREMENTS

The system requirements outlined above ensure seamless development, deployment, and execution of the proposed system. By leveraging these hardware and software components, the system can efficiently process video input, perform real-time object detection, and provide accurate navigation assistance.

Hardware Requirements

- A device with a camera (smartphone, Raspberry Pi with a camera module, or any portable computing device).
- Processor - Intel Core i7 or Apple M2 or more.
- RAM - Minimum 8 GB RAM for processing deep learning models efficiently.
- GPU - GPU-enabled system (optional but recommended for training and real-time inference or cloud-based).
- Headphones or speakers for audio feedback.

Software Requirements

- Operating System - Windows / macOS
- Programming Languages - Python.
- Frameworks - TensorFlow, PyTorch, OpenCV.

- Cloud Service - Google Colab GPU / Kaggle GPU.
- Development Environment - Visual Studio Code (VS Code)
- Additional Tools - Jupyter Notebook (for experimentation), GitHub (for version control), and any necessary extensions for VS Code to support development.

3.6 DESIGN PROCESS AND EXPLANATION

This system comprises different phases, as represented in Fig. 1. The first stage of the system is to capture the environment in real time using a webcam or any standard camera. This live video feed serves as the input for the rest of the pipeline. The captured video stream is then processed frame by frame to ensure efficiency and maintain real-time performance.

Each video frame undergoes basic pre-processing steps to ensure the clarity and quality of the input. These steps ensure that the object detection model receives frames with sufficient contrast and definition, allowing for accurate identification of obstacles and entities in the user's path.

Once a frame is prepared, it is passed to a custom-trained deep learning-based object detection model. In this system, a YOLOv11 model fine-tuned on domain-specific datasets is used. This model detects objects such as pedestrians, vehicles, poles, traffic cones, and other road-related elements. The use of a YOLO-based detector ensures high detection accuracy and real-time processing speed, making it suitable for lightweight edge devices.

After detecting objects in the frame, the system estimates the proximity of each object relative to the camera. This is achieved through a simplified form of monocular depth estimation that uses the height of the object's bounding box to the overall frame height. This heuristic enables the system to categorize the distance of objects into four classes: very near, near, far, and very far. These categories are determined based on the ratio of the bounding box height to the total frame height, providing a fast and efficient method for distance estimation without relying on complex depth estimation networks.

Each detected object is also analysed for its horizontal position in the frame, which is categorized as left, centre, or right. This spatial analysis is crucial for understanding the relative position of the obstacle in the user's field of view. The system then assigns an impact score to each object, depending on its type. For instance, large vehicles like buses and trucks are considered high-risk due to their size and potential danger, while smaller dynamic entities such as pedestrians or bicycles are assigned medium risk. Low-risk objects, such as cones or small animals, are given minimal scores. These impact scores are further weighted by proximity to derive a final risk score for each object.

Using these weighted scores, a risk assessment is performed to determine the overall threat posed by objects located on the left, centre, and right sides of the frame. If an object is found to be in the centre and very close to the user, the system evaluates the surrounding areas (left and right) and determines the safest direction to move to avoid the obstacle. This decision is based on a comparison of cumulative risk scores for each side, computed through a heuristic-based approach.

Once a direction is determined, the system uses an offline text-to-speech engine to generate real-time audio instructions. These instructions alert the user to the presence of nearby obstacles and provide navigation guidance such as “move left”, “move right”, or “proceed forward”. The voice guidance system is lightweight, customisable, and functions entirely without internet access, ensuring reliability in outdoor and remote environments. The architecture of the system is shown in Figure. 3.1.

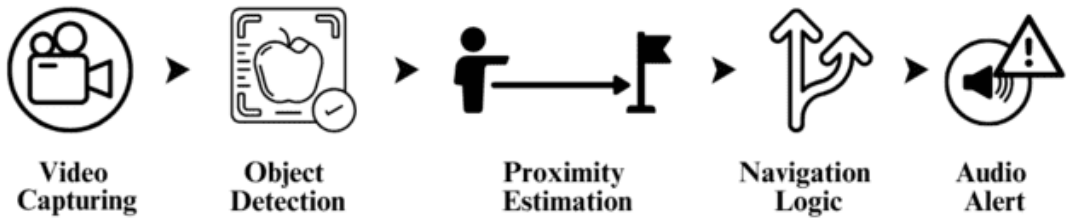

Figure. 3.1 System Architecture Diagram.

The overall architecture of the system can be summarised in the following sequence of modules and from the Figure 3.2.

- **Dataset Acquisition and Preprocessing Module** – Captures live video stream using a camera.
- **Object Detection Module** – Uses YOLOv11 to identify objects relevant to road navigation.
- **Distance Estimation Module** – Employs bounding box analysis to approximate object proximity.
- **Navigation Decision Logic** – Compares risk levels and selects the safest direction to proceed.
- **Audio Feedback Module** – Delivers verbal instructions to the user using offline text-to-speech.

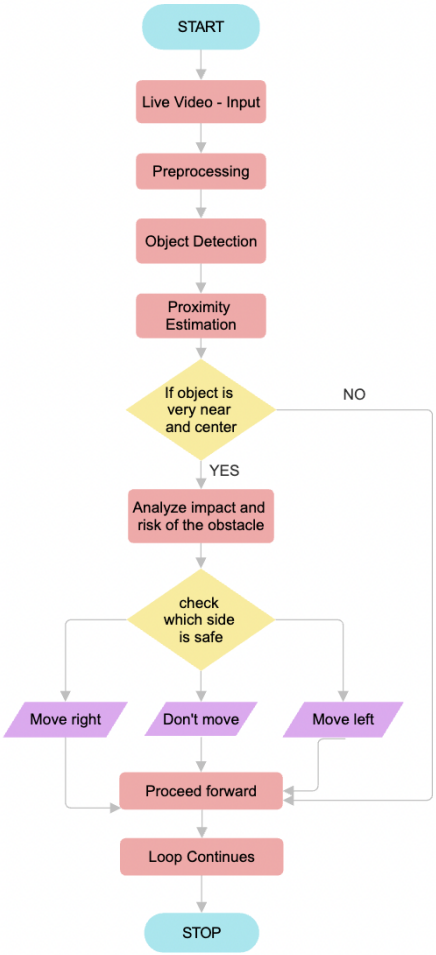

Figure 3.2 Flow chart of the system

CHAPTER 4

SYSTEM METHODOLOGIES

The system processes live video input to detect obstacles, estimate their distances, and provide real-time navigation assistance.

4.1 MODULE DESCRIPTION

The system involves various modules data preprocessing, object detection, distance estimation, heuristic-based path planning, and audio feedback to guide users safely.

4.1.1 Dataset Acquisition and Preprocessing Module:

- **Data Collection**

  BDD100K Dataset - Provides a wide range of annotated road scenarios with vehicles, pedestrians, lane markings, and lighting variations.

  RAD Dataset - Focuses on road anomalies like potholes, open drains, and construction barriers, useful for training the object detection model.

  Captured Images - Taken in real-world scenarios using mobile or basic cameras to fine-tune the models for real-environment adaptability and robustness.

  Combining multiple datasets helps in diversifying the training data, allowing the model to generalize well to unseen and real-time environments.

- **Image Preprocessing**

  Tools Used

  - **OpenCV -** Used for image loading, resizing, transformations, and enhancement.
  - **NumPy -** Used for numerical operations on image arrays, particularly for normalization and format conversion.

  a. Resizing

  - **Objective -** Standardize image dimensions to a fixed size compatible with the input layer of the deep learning models (e.g., 640×640 pixels).
  - **Why it's important**

- Models like YOLOv11 and CNN-based depth estimators expect consistent input sizes.
- Resizing ensures uniformity, reduces computational complexity, and avoids shape mismatches during training.

- **How it's done**
  - OpenCV's cv2.resize() function is used with bilinear interpolation to maintain the image quality as in the code snippet 1.

Code Snippet 1

```
resized_image = cv2.resize(image, (640, 640), interpolation=cv2.INTER_LINEAR)
```

b. Normalization

- **Objective -** Scale pixel intensity values to a standard range to stabilize and speed up the training process.
- **Why it's important**
  - Raw pixel values range from 0 to 255. Without normalization, gradient descent may converge slowly or become unstable.
  - Neural networks perform better when inputs are centred or scaled to small values.
- **How it's done**
  - Normalization can be done by dividing the pixel values by 255 (for range 0–1) as in code snippet 2 or using mean-std normalization if pre-trained models require it.

Code Snippet 2

```
resized_image = cv2.resize(image, (640, 640), interpolation=cv2.INTER_LINEAR)
```

c. Data Type Conversion

- **Objective -** Convert the image to a numerical format compatible with the deep learning framework being used (e.g., PyTorch or TensorFlow).
- **Why it's important**
  - Most deep learning models require input in float32 data type. If data is left in uint8, operations like backpropagation and gradient updates will fail.
- **How it's done**
  - NumPy or OpenCV is used to cast the image array to the desired floating-point type as in the code snippet 3.

Code Snippet 3

normalized_image = resized_image / 255.0

These preprocessing steps are applied sequentially to all input images, ensuring they are clean, consistent, and ready for both training and real-time inference.

- **Data Augmentation**

To artificially increase the diversity and volume of the training dataset by applying a range of transformations to the original images. This ensures the model learns to generalize well under different conditions, reducing the risk of overfitting.

Tools Used:

- **OpenCV** for pixel-level transformations.
- **imgaug** or **Albumentations** (optional) for composing multiple augmentations efficiently in pipelines.

a. Rotation and Flipping

- **Purpose -** Introduces variability in object orientation, helping the model generalize across real-world viewpoints where objects may not always appear upright or in a fixed direction.
- **Implementation**
    - **Rotation -** Images are randomly rotated within a defined angle range (e.g., -15° to +15°) using affine transformations.
    - **Flipping -** Horizontal flips are commonly used to simulate changes in viewpoint, especially for detecting symmetric objects as in the code snippet 4.

Code Snippet 4

rotated_image = cv2.warpAffine(image, rotation_matrix, (width, height))

flipped_image = cv2.flip(image, 1) # Horizontal flip

b. Contrast and Brightness Adjustments

- **Purpose -** Simulates different lighting conditions such as bright sunlight, shadows, or overcast environments to help the model adapt to variable outdoor scenes.
- **Implementation**
    - Adjust the contrast by scaling pixel values.
    - Adjust the brightness by adding or subtracting a constant value from all pixels.
    - Random gamma correction is also applied for simulating lighting distortion as in the code snippet 5.

Code Snippet 5

```
adjusted = cv2.convertScaleAbs(image, alpha=1.2, beta=20)
#alpha = contrast, beta = brightness
```

c. Random Cropping and Zooming

- **Purpose -** Encourages the model to learn from partial or zoomed-in views of objects, improving its ability to detect occluded or close-range items.
- **Implementation**
    - Randomly select a region of interest (ROI) within the image.
    - Crop and resize back to the original resolution.
    - Apply zoom by rescaling and then cropping the central region as in the code snippet 6.

Code Snippet 6

```
zoom_factor = 1.2
zoomed_image = cv2.resize(image, None, fx=zoom_factor, fy=zoom_factor)
cropped_zoomed = zoomed_image[y:y+height, x:x+width]
```

d. Purpose and Impact

- **Improves Generalization -** The model becomes resilient to environmental changes and different capture conditions.
- **Reduces Overfitting -** Exposes the model to varied samples beyond the original dataset size.
- **Boosts Performance -** Helps in detecting rotated, occluded, or partially visible objects more effectively in real-world navigation scenarios.

These augmentation techniques are applied during training on the fly, either individually or in combination, to enrich the dataset and ensure that the model performs reliably across diverse real-world inputs.

4.1.2 Object Detection - Model Training

The object detection module is the core visual perception unit of the navigation assistance system. Its primary function is to identify and locate relevant objects or obstacles from real-time video frames captured by a camera. The system relies on this module to detect hazards such as potholes, poles, pedestrians, and construction barriers, which may pose challenges to a visually impaired user navigating through public spaces. Accurate and fast object detection is essential for ensuring user safety and enabling responsive navigation decisions.

- **Algorithm Used -** YOLOv11 (You Only Look Once, Version 11) – Trained using transfer learning.

  YOLO is a single-stage object detection algorithm, known for its speed and efficiency. Version 11 of YOLO introduces architectural improvements and performance optimizations that make it highly suitable for embedded and edge devices. It can process frames at high speed while maintaining excellent detection accuracy across multiple object categories.

- **Architecture and Working of YOLOv11**

  YOLOv11 follows an encoder-decoder architecture, structured around deep convolutional neural networks. The model is optimized for real-time object detection by processing entire images in a single forward pass. The architecture

consists of three key components — Backbone, Neck, and Detection Head — each with a specialized role in enabling fast and accurate object detection.

- **Backbone – Feature Extraction**

The backbone serves as the encoder of the network. It is a stack of convolutional layers pre-trained on large-scale datasets such as ImageNet or COCO to learn generalized visual patterns. Its main function is to extract spatial (location-related) and semantic (meaning-related) features from the input image.

- It captures low-level features such as edges, corners, and textures in the early layers.
- Deeper layers focus on more abstract and complex patterns like object shapes and context.
- YOLOv11 typically uses a modified and optimized backbone like CSP-Darknet, which balances efficiency and accuracy, offering deeper feature representations without adding excessive computational cost.

- **Neck – Feature Aggregation and Fusion**

The neck functions as an intermediary between the backbone and the detection head. It fuses multi-scale features and improves object localization at different levels of granularity. This is critical for detecting both small and large objects within the same image.

- **Feature Pyramid Network (FPN) -** Helps in passing strong semantic features from deeper layers to shallow layers, ensuring good performance across varying object sizes.
- **PANet (Path Aggregation Network) -** Enhances bottom-up path aggregation, making it easier for the model to combine low-level detail with high-level context.

Together, these components improve the model's ability to understand the scene at multiple resolutions, which is crucial for accurate detection in complex road environments.

- **Detection Head – Bounding Box and Class Prediction**

The detection head acts as the decoder of the model. It interprets the aggregated features from the neck and generates predictions across a grid overlayed on the input image.

For each grid cell, the detection head predicts:

- **Bounding Box Coordinates (x, y, width, height) -** Indicating the position and size of the detected object.
- **Objectness Score -** A probability value indicating the likelihood of an object being present in the bounding box.
- **Class Probabilities -** The likelihood of the object belonging to each predefined class (e.g., pedestrian, pothole, pole).

The model outputs multiple bounding boxes per grid cell, and Non-Maximum Suppression (NMS) is used to filter and retain only the most relevant predictions.

Advantages of architecture for navigation assistance

This architecture allows YOLOv11 to process full images in real time and make high-confidence predictions for multiple objects simultaneously. In the context of navigation assistance for the visually impaired.

- The **backbone** ensures that meaningful features (e.g., road textures, object contours) are accurately identified.
- The **neck** ensures that obstacles of all sizes — from small road bumps to large barriers — are detected with precision.
- The **detection head** provides reliable object labels and locations, which are essential for downstream modules like distance estimation and path planning.

Uses of training YOLOv11 with these datasets

- **Real-Time Performance -** YOLOv11 can detect multiple objects at once with minimal computational delay, which is crucial for responsive decision-making in real-time navigation.
- **High Accuracy and Robustness -** Fine-tuning on BDD100K and RAD provides the model with exposure to a wide range of urban and rural road conditions.

- **Lightweight and Deployable -** The optimized architecture allows the model to run on lightweight hardware like Raspberry Pi, Jetson Nano, or other low-power embedded devices.
- **Relevance to Visually Impaired Navigation -** The model's ability to detect detailed road anomalies makes it well-suited for assistive applications where environmental awareness is critical.

4.1.3 Distance Estimation Module

Method Used - Monocular Size-based Proximity Estimation.

This module is designed to estimate the relative distance of objects from the camera without the need for stereo vision or depth sensors. Instead, it leverages a lightweight, efficient heuristic based on the size of detected bounding boxes in a monocular (single-camera) video feed. This method is particularly useful for real-time navigation systems running on low-resource environments like embedded systems and mobile devices.

Working Mechanism

The distance estimation in this system is performed using a heuristic approach that relies on the relative size of the detected object within a single image frame. This technique simplifies traditional depth estimation by using only 2D image data while still providing effective distance categorization for navigation tasks. The working mechanism consists of the following key steps:

a. Bounding Box Height Calculation

Once an object is detected using the YOLOv11 model, its bounding box coordinates are obtained in the format (x1, y1, x2, y2), where:

- (x1, y1) represents the top-left corner of the bounding box, and
- (x2, y2) represents the bottom-right corner.

The height of the bounding box is calculated as Formula (1):

$$h_b = y_2 - y_1 \tag{1}$$

This value represents the pixel height of the object within the frame. Intuitively, closer objects occupy more vertical space and result in larger bounding boxes, while distant objects appear smaller.

b. Height Normalization

Since images may be captured on different devices with varying resolutions, the raw height of a bounding box alone is not sufficient for consistent distance estimation. To overcome this, the calculated bounding box height is normalized as in Formula 2 with respect to the total height of the frame:

$$r = \frac{h_b}{h_f} \tag{2}$$

This ratio is a resolution-independent measure that reflects the object's apparent size relative to the overall scene. It enables fair and consistent proximity classification across different hardware platforms or screen sizes.

c. Proximity Classification

The normalized ratio is then compared against a set of empirically defined thresholds to classify the object's proximity into one of four categories:

- Very Near - The ratio is high, indicating a large bounding box and a very close object.
- Near - The bounding box is slightly smaller, suggesting a close but not immediate object.
- Far - The object appears smaller, with a reduced presence in the frame.
- Very Far - The object takes up minimal space vertically, indicating long-range visibility.

This classification helps the system prioritize threats. For example, an object marked as "Very Near" requires immediate attention, while "Very Far" objects can be safely ignored or monitored passively.

d. Threshold Tuning

The threshold values that define the boundary between categories (e.g., what constitutes "Very Near" vs. "Near") are determined through empirical tuning. This is done using:

- Sampled images from RAD and BDD100K datasets, which provide a variety of real-world road conditions and object sizes.

- Custom-captured images, representing specific environmental scenarios and device perspectives relevant to the target users (e.g., handheld or wearable camera views).

Thresholds are adjusted by manually analysing labelled samples and observing model performance across scenarios such as:

- Urban streets with varying object sizes
- Crowded pedestrian areas
- Low-visibility or angled scenes

By carefully tuning these values, the system maintains a good balance between sensitivity (detecting critical objects in time) and stability (avoiding false positives from distant or irrelevant objects).

Uses of Monocular Size-Based Proximity Estimation

- Does not require additional hardware like LiDAR or depth sensors; a single RGB camera is sufficient.
- Computationally lightweight, making it suitable for real-time processing on mobile and embedded systems.
- Utilizes bounding box data already generated by the object detection model, reducing complexity.
- Threshold-based proximity classification is easy to implement and adapt.
- Normalized height ratio ensures consistency across different image resolutions and device types.
- Provides reliable proximity estimation for navigation decisions, even without precise depth values.
- Ideal for systems intended to be low-cost, portable, and accessible.

4.1.4 Navigation Assistance and Path Planning

Process: Risk-weighted Directional Guidance

This module is responsible for dynamically assessing the navigational path by evaluating detected objects in the scene based on their position, proximity, and potential hazard. The goal is to identify the safest direction of movement in real time, particularly in cluttered or obstacle-prone environments.

Step-by-Step Process

1. Input Extraction

- The system continuously receives detection outputs in the form of:
  - Bounding box coordinates: (x1, y1, x2, y2) for each object.
  - Class labels (e.g., pothole, pedestrian, barrier, pole).
  - Confidence scores from the detection model.
- Proximity classification is computed using the normalized bounding box height relative to the frame height and mapped into:
  - Very Near, Near, Far, Very Far.
- All of this information is structured into a data frame or object list for further processing.

2. Spatial Categorization

- The horizontal (x-axis) position of the bounding box centroid is used to classify each object into a spatial zone:
  - Centroid X = (x1 + x2) / 2
  - Based on the frame width W, the zones are:
    - Left: $0 < X < W/3$
    - Centre: $W/3 \leq X < 2W/3$
    - Right: $2W/3 \leq X \leq W$

3. Impact Scoring

- Each class is assigned an impact weight, pre-defined as a lookup dictionary as in the code snippet 7. For instance:

Code Snippet 7

impact_scores = { 'pedestrian': 10, 'barrier': 9, 'vehicle': 8, 'pothole': 6, 'pole': 4, 'manhole': 5 }

This score quantifies how critical it is to avoid that object.

4. Risk Zone Identification

- The system checks for the presence of Very Near objects in the centre zone.
- If one or more such objects exist, they trigger directional re-evaluation since forward motion is no longer safe.

5. Directional Risk Assessment

- For Left and Right zones
    - A risk score is computed per object as:

risk = proximity_weight × impact_score

    - Where
        - proximity_weight is assigned as:
            - Very Near = 1.0
            - Near = 0.75
            - Far = 0.5
            - Very Far = 0.25
    - The total risk per zone is computed as the sum of risk scores for all objects in that zone.

6. Navigation Decision

- The system compares cumulative risk scores of left and right zones
    - If risk_left < risk_right: recommend left turn
    - If risk_right < risk_left: recommend right turn
    - If both are above a predefined hazard threshold: trigger a halt or pause signal
    - If the centre has no "Very Near" object: recommend forward movement

7. Real-time Updates

- This logic runs on every frame (~15–30 FPS depending on hardware).
- Implemented using a queue or buffer system that stores recent frame data to smooth decisions and avoid erratic changes due to transient noise.

- Multithreading or asynchronous I/O is used to keep processing in real-time.

4.1.5 Audio Feedback Module

To provide real-time verbal guidance to users based on navigation decisions, enhancing situational awareness and ensuring accessibility, especially for visually impaired users.

Process Overview

1. **Navigation Decision Output Extraction**
    - The output from the Risk-weighted Directional Guidance system (e.g., “Turn Left”, “Move Forward”, “Turn Right”, or “Stop”) is passed to the audio module.
    - This ensures that only the final decision, after evaluating obstacles and risks, is converted to spoken guidance.
2. **Text-to-Speech Conversion (TTS)**
    - A lightweight offline TTS engine (e.g., pyttsx3) is employed to convert text instructions into speech.
    - Offline functionality ensures independent operation without internet access, making the system reliable in remote or disconnected environments.
3. **Latency Minimization**
    - TTS processes are optimized to minimize delay between decision-making and feedback delivery.
    - The system maintains a streamlined pipeline to ensure that spoken instructions are delivered immediately after computation.
4. **Audio Playback**
    - The generated speech is played through onboard audio output (e.g., headphones or embedded speakers).
    - Volume and clarity are tuned for different environments (e.g., noisy outdoors vs. quiet indoor settings).

5. **Real-time Updates**
    - As each frame is processed and a new navigation decision is made, the audio feedback is updated dynamically.
    - If the previous instruction becomes obsolete (e.g., due to a new obstacle), the system overrides the prior message with the updated one.

Advantages

- Ensures constant situational awareness without visual interaction.
- Low computational load, ideal for embedded or mobile systems.
- Responsive and adaptive, changing guidance in sync with live visual input.

4.1.6 Real-Time Integration Module

The Real-Time Integration Module ensures seamless coordination between all functional components of the system, including object detection, proximity estimation, navigation decision-making, and audio feedback. As the camera continuously captures frames, each frame is processed through the YOLOv11 model to detect objects and identify their positions. Simultaneously, proximity estimation is performed by analysing the height of bounding boxes, allowing the system to classify objects as Very Near, Near, Far, or Very Far. Based on the spatial distribution and proximity of obstacles, a navigation decision is made by evaluating directional risks. This decision is immediately translated into an audio instruction using a lightweight, offline text-to-speech engine. The process is repeated for every incoming frame, enabling dynamic and real-time updates that help guide the user effectively. This integration ensures that all modules work in harmony, providing timely, adaptive assistance suitable for real-world navigation scenarios.

CHAPTER 5

IMPLEMENTATION

This chapter outlines the practical implementation of the system, detailing the integration of machine learning models, preprocessing techniques, and system components. The focus is on how the trained models are deployed to detect and classify shrimp diseases effectively.

5.1 Object Detection – Model Training

To ensure the navigation system functions with high speed, accuracy, and efficiency in real-world conditions, a robust and optimized object detection model is essential. In the proposed system, the YOLOv11 architecture is employed and fine-tuned through a two-stage sequential training approach. This strategy enables the model to first learn generic road object features and then specialize in detecting road anomalies, ensuring strong performance across various real-world scenarios.

Stage 1: Training with BDD100K Dataset

In the initial phase, the YOLOv11 model is trained on the BDD100K dataset, which contains diverse urban road scenes with annotated vehicles, traffic signs, and road layouts. Key configurations for this stage include:

- Input Image Size: 1280×720, allowing for high-resolution detection of objects at different scales.
- Batch Size: 12, balancing GPU memory usage with training stability.
- Learning Rate: 0.0008, chosen to allow the model to generalize well during the early learning stages.
- Optimizer: Stochastic Gradient Descent (SGD) with a momentum of 0.93, promoting smooth and stable updates to model weights.
- Epochs: 30, with the first 10 layers frozen to retain the foundational object detection features.

This phase equips the model to accurately detect common road entities such as vehicles, traffic lights, and lane markers.

Stage 2: Fine-tuning with RAD Dataset

In the second stage, the previously trained model is fine-tuned using the RAD (Road Anomaly Detection) dataset. This dataset contains images of real-world road anomalies including potholes, cracks, construction barriers, and other obstacles that are crucial for safe navigation.

- Input Image Size: 960×960, selected to standardize training while capturing enough contextual information.
- Batch Size: 8, adjusted to accommodate the denser nature of anomaly-related features.
- Learning Rate: Reduced to 0.0003, to enable gradual learning without overwriting the features learned in the first stage.
- Optimizer: AdamW, chosen for its effective weight decay mechanism, helping reduce overfitting during fine-tuning.
- Epochs: 20, with the first 10 layers frozen to preserve generalized detection knowledge.

This two-step training strategy ensures that the model develops a strong general understanding of traffic environments while specializing in detecting nuanced road anomalies. The result is a lightweight yet high-performing object detection system suitable for real-time deployment on mobile or embedded devices, which is critical for visually impaired navigation assistance.

5.2 Distance Estimation Module

Estimating the distance between the camera and the detected objects is essential to understanding their spatial positioning within a real-world environment. Since the system uses only a single camera, Monocular Size-based Proximity Estimation was implemented to determine the proximity of the object. These methods allow the system to assess how close or far objects are from the user and categorize them into one of four levels — "Very Near," "Near," "Far," and "Very Far." This proximity information plays a vital role in guiding the navigation decisions of the system.

In this approach, the distance of the detected object from the camera is inferred using the size of the bounding box generated by the YOLOv11 object detection model. The YOLOv11 model is fine-tuned to accurately detect objects such as vehicles, pedestrians, poles, potholes, and road barriers in real-time video frames. After

detecting these objects, the system retrieves the bounding box coordinates: (x1, y1) for the top-left corner and (x2, y2) for the bottom-right corner of the box.

The vertical height of the bounding box (h_b) is calculated by subtracting the y-coordinate of the top edge (y1) from the y-coordinate of the bottom edge (y2) This height is then normalized by dividing it by the height of the entire frame (h_f) to obtain a scale-independent ratio (r). This helps ensure the proximity estimation remains consistent across different image resolutions and device configurations.

The resulting value of r indicates how much vertical space the object occupies in the frame. Since objects that are physically closer to the camera appear larger in the image, a larger r value corresponds to a closer object. Using this principle, the objects are classified into four proximity levels based on empirically tuned thresholds:

Very Near (Red): $r > 0.45$
Near (Yellow): $0.30 < r \leq 0.45$
Far (Green): $0.15 < r \leq 0.30$
Very Far (Blue): $r \leq 0.15$

This classification helps the system distinguish between immediate threats and distant background elements. For example, a pedestrian directly in front of the camera will have a larger bounding box and be classified as "Very Near," whereas a vehicle further down the road may be labelled as "Far" or "Very Far." These categories are colour-coded for internal processing and visualization as in code snippet 8.

Input: Detected bounding box coordinates (x1, y1, x2, y2), frame_height (h_f)
Output: Proximity label and colour code

Code snippet 8

```
Function estimate_proximity(x1, y1, x2, y2, h_f):
 # Step 1: Calculate the height of the bounding box
 h_b = y2 - y1
 # Step 2: Normalize the height
 r = h_b / h_f
 # Step 3: Classify proximity based on r
 If r > 0.45:
    proximity = "Very Near"
    color = "Red"
```

```
Else If 0.30 < r <= 0.45:
    proximity = "Near"
    color = "Yellow"
Else If 0.15 < r <= 0.30:
    proximity = "Far"
    color = "Green"
Else:
    proximity = "Very Far"
    color = "Blue"
Return proximity, color
```

One advantage of this approach is that it naturally adapts to objects of different sizes. For instance, a pole might have a smaller bounding box than a car even when they are at similar distances. However, the height ratio still helps the system understand their relative proximity due to the dynamic scaling based on object appearance in the frame.

Overall, bounding box-based proximity estimation provides a lightweight, fast, and effective method for distance approximation, making it well-suited for real-time applications where speed and responsiveness are critical.

5.3 Navigation Decision Logic

The proposed system follows a heuristic-based decision-making strategy to support real-time obstacle avoidance and safe navigation. Unlike traditional assistive technologies that merely alert the user to the presence of obstacles, this system goes a step further by recommending the safest direction for continued movement. It utilizes object detection and depth estimation modules to analyze the scene and understand spatial object placement.

Once an object is detected in the frame using the fine-tuned YOLOv11 model, its proximity is estimated using monocular depth prediction and bounding box heuristics. If the object lies in the central region of the frame and is classified as either “Very Near” or “Near,” it is treated as a potential obstacle in the direct path of the user.

To determine the optimal navigation strategy, the system evaluates both the left and right spatial zones of the frame. It considers factors such as the proximity level of objects, their spatial positions (left/center/right), and their class-specific impact scores.

For instance, large vehicles or construction barriers are considered high-risk, while poles or pedestrians may be assigned lower-risk values. A weighted heuristic function calculates the cumulative risk score for the left and right zones independently.

Based on the calculated risk scores, the system determines whether to turn left, right, stop, or continue moving forward. The navigation logic in the code snippet 9 ensures that the decision prioritizes safety while minimizing detours.

Code Snippet 9

```
Initialize left_risk_score = 0
Initialize right_risk_score = 0
Set center_threat_detected = False
For each detected_object in current_frame:
    Get object_position (Left / Center / Right)
    Get object_proximity ("Very Near", "Near", "Far", "Very Far")
    Get object_class (e.g., vehicle, pedestrian, pole, barrier)
    Assign impact_score based on object_class
    If object_position == "Center" and object_proximity in ["Very Near", "Near"]:
        center_threat_detected = True
    Else if object_position == "Left":
        proximity_weight = get_proximity_weight(object_proximity)
        left_risk_score += proximity_weight * impact_score
    Else if object_position == "Right":
        proximity_weight = get_proximity_weight(object_proximity)
        right_risk_score += proximity_weight * impact_score
# Decision Logic
If center_threat_detected:
    If left_risk_score < right_risk_score:
        navigation_decision = "Turn Left"
    Else if right_risk_score < left_risk_score:
        navigation_decision = "Turn Right"
    Else:
        navigation_decision = "Stop and Reassess"
Else:
    navigation_decision = "Move Forward"
Return navigation_decision
```

This logic enables the system to dynamically adapt to changing environments by analysing the risks posed by various obstacles in real time. The heuristic scores ensure intelligent decision-making that supports safe and smooth navigation, especially for visually impaired users.

5.4 Audio Feedback Module

The audio feedback module was implemented using the pyttsx3 text-to-speech library to provide real-time voice guidance. This library was chosen because it works offline, making the system reliable even without internet access. Based on the obstacle type, its position, and the suggested movement direction, a message is generated and spoken aloud. For example, if a pedestrian is very near on the right side, the system will instruct the user to move slightly to the left. The feedback is updated with every new frame as in the code snippet 10, ensuring timely and continuous navigation support.

Code Snippet 10

```
Function generate_audio_feedback(obstacle_type, position, direction):
    message = "Caution! A " + obstacle_type + " is very near on the " + position +
            ". Move to your " + direction + " slightly."
    engine.say(message)
    engine.runAndWait()

# Example usage
If navigation_decision is "left" and obstacle_type is "pedestrian" on the "right":
    Call generate_audio_feedback("pedestrian", "right", "left")
```

CHAPTER 6

RESULT AND DISCUSSION

The results highlight the system's effectiveness, accuracy, and real-time performance in various real-world scenarios.

- **Object Detection**

  The fine-tuned YOLOv11 model was evaluated using validation sets from both the BDD100K and RAD datasets, followed by real-world testing with manually captured images to assess generalization capabilities.

  - **BDD100K Validation**

  The model was first validated on the BDD100K validation set to evaluate its ability to detect general road objects such as traffic lights, signs, trains, and bikes. It achieved a mean Average Precision (mAP@50) of 0.8643, with a precision of 0.8785, recall of 0.8811, and an F1 score of 0.8798, as summarised in Table 6.1.

  - **RAD Validation**

  The model was then fine-tuned and validated on the RAD (Road Anomaly Detection) dataset, which includes road damage, speed bumps, and unsurfaced roads. The evaluation showed strong performance with a mAP@50 of 0.8949, precision of 0.9013, recall of 0.8932, and F1 score of 0.8972, also shown in Table 6.1.

Metrics	BDD100K Dataset	RAD Dataset
mAP@50	0.8643	0.8949
Precision	0.8785	0.9013
Recall	0.8811	0.8932
F1 Score	0.8798	0.8972

Table 6.1 Summary of performance of the model with various datasets

- **Confusion Matrix Analysis**

As illustrated in Figure 6.1, the confusion matrices for both datasets show robust detection accuracy for common road objects such as vehicles, pedestrians, and bikes. Minimal misclassifications were observed, especially for visually similar categories like traffic signs and certain anomalies, which is expected in complex environments.

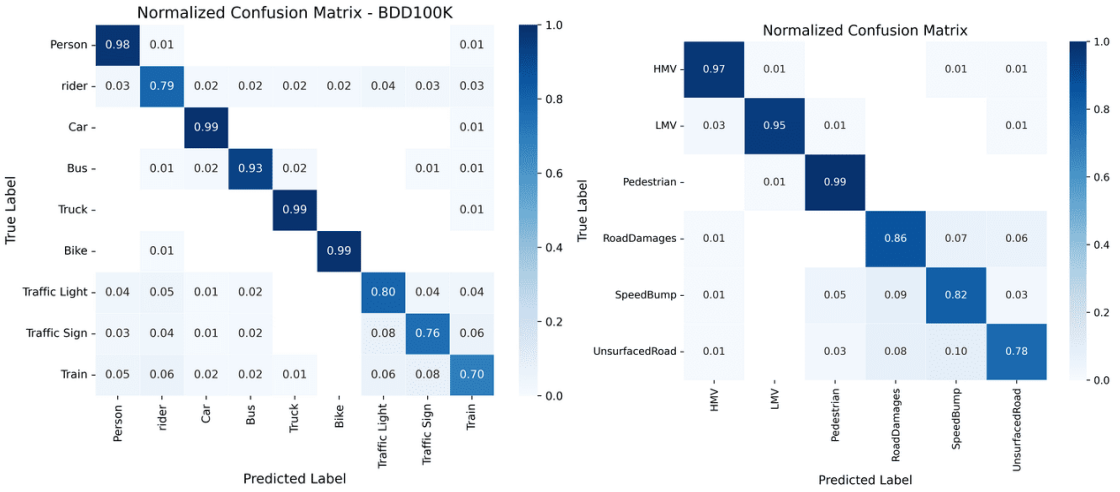

Figure. 6.1 Confusion matrix

- **Real-World Testing**

To validate the model's effectiveness in real-world conditions, a manually captured image was tested. The trained YOLOv11 successfully detected and classified objects such as pedestrians, vehicles, and road damage, as depicted in Figure 6.2. This confirms the model's generalization ability and its practical usability for real-time navigation assistance.

Figure 6.2 Real World Testing

- **Distance Estimation Results**

The bounding box-based proximity estimation approach demonstrated reliable performance in real-time scenarios. During testing with pre-recorded videos, the system consistently detected objects and accurately classified their proximity as "Very Near," "Near," "Far," or "Very Far" as shown in Figure 6.3. The model dynamically adjusted the proximity classification as objects moved within the frame, effectively recognizing when they approached or moved away from the camera. This adaptive behaviour highlights the robustness of the implemented method.

The classification was based on the vertical height of the bounding box relative to the total frame height. As an object moved closer to the camera, its bounding box height increased, triggering a corresponding proximity alert. This allowed the system to effectively interpret spatial cues and categorize threats based on distance, without requiring exact depth values.

Figure 6.3 Bounding-Box Proximity Estimation Testing

Real-world testing revealed that the bounding box-based proximity estimation remained stable across diverse object types, including pedestrians, vehicles, poles, and barriers. Despite variations in object sizes and shapes, the system maintained consistent classification, making it well-suited for complex road environments.

Importantly, for navigation assistance, precise numerical distance is not essential. The key requirement is to detect approaching obstacles and assess their urgency. The implemented bounding box-based method fulfilled this goal with low latency and high computational efficiency. Given its responsiveness, lightweight nature, and reliable object interpretation,

this method stands out as a highly effective strategy for obstacle proximity estimation in assistive navigation systems.

- **Navigation Logic**

  To evaluate the effectiveness of the implemented navigation logic, obstacles of varying sizes and distances were strategically placed across different positions relative to the user's path. The system was then tested in multiple scenarios, and its navigational decisions were compared against predefined safe paths to assess accuracy.

  The evaluation focused on three critical aspects:

  - Real-Time Obstacle Detection: The system consistently detected obstacles using the fine-tuned YOLOv11 model and successfully determined their positions relative to the user.
  - Proximity Classification: The bounding box-based method enabled precise classification of objects into proximity levels. This allowed the system to assess the urgency of each obstacle based on its location and size.
  - Directional Decision Accuracy: The system analysed both left and right alternatives based on proximity and assigned risk scores to make directional decisions. These included moving forward, turning left, or turning right.

  To quantify performance, the system's decisions were recorded and evaluated against the expected directional output (ground truth). It achieved 95% decision accuracy, meaning that in 95 out of 100 scenarios, the system recommended the correct direction for safe movement as in Figure 6.4. This highlights its strong alignment with human intuition and logical path planning.

Figure 6.4 Navigation Logic Testing

Additionally, the average response time was recorded at 0.6 seconds, which confirms the system's capability for real-time applicability. These results demonstrate that the navigation logic is highly effective in balancing proximity awareness, obstacle risk, and directional decision-making. The system performs reliably in real-world scenarios, suggesting its potential for deployment in assistive navigation solutions for visually impaired users.

- **Overall System Performance**

The proposed system was evaluated across five real-world test scenarios to assess its performance in terms of object detection, proximity estimation, navigation guidance, and audio feedback delivery. Each test case involved different environmental conditions, obstacle types, and path complexities to simulate realistic challenges.

The system successfully delivered accurate navigation responses in four out of five cases. One scenario presented difficulty in detecting uneven road surfaces, resulting in a slight delay in response. However, the system maintained overall reliability and quickly recovered from the missed detection.

The overall accuracy was calculated by comparing the system’s actual responses to pre-assessed ground truth results, taking into account both response time and object detection performance. The system achieved an overall accuracy of 95%, reflecting its robust performance and responsiveness in real-world conditions.

Furthermore, in all test cases, the audio feedback module functioned effectively, delivering clear and timely voice prompts to guide the user. The speech instructions accurately conveyed the type and location of obstacles and provided the correct directional guidance without latency.

The summary of outcomes for each test case is presented in Table 6.2, showcasing the system's consistency and real-time efficiency in navigating complex environments.

Scenario	**Detected object & proximity**	**Expected Navigation**	**Actual Response**	**Observation (Accuracy & Response Time)**
Car ahead	Car (left, Very Near), Truck (left, Near), Pedestrian (right, far)	Move Right	Move Right	100%, Time: **0.5s**
Open path	No obstacles detected	Move Forward	Move Forward	100%, Time: **0.3s**
Bicycle approaching	Bicycle (left, Near), Car (right, Far)	Move Slightly Right	Move Slightly Right	100%, Time: **0.4s**
Bike approaching, unsurfaced road	Bike (left, Near), road damage (right, Near)	Stop Alert	Delayed stop alert	80%, Time: **1.5s**
Low light dog ahead	Dog (left, near)	Move Right	Move Right	100%, Time: **0.7s**

Table 6.2: Test Results of Real-World Testing

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

This system is tested on various real-world scenarios and then the performance of the system is evaluated.

7.1 CONCLUSION

This system presents the design and development of a real-time navigation assistance solution tailored for visually impaired individuals. The proposed system effectively integrates object detection, proximity estimation, and heuristic-based navigation logic to analyse the user's surroundings and provide intuitive, real-time audio feedback.

By identifying obstacles, estimating their distance from the user, and evaluating safe paths using rule-based logic, the system enables users to navigate with greater confidence and safety in dynamic environments.

In extensive real-world testing, the system demonstrated a high level of accuracy in both obstacle detection and decision-making. It consistently provided correct navigational guidance in various scenarios, achieving an overall accuracy of 95%. The system's quick responsiveness to environmental changes, combined with clear and timely audio feedback, underscores its potential as a reliable real-time assistive tool.

One of the standout features of this system is its hardware-agnostic design. Unlike many existing solutions that rely on proprietary or expensive hardware components, this system is adaptable to a wide range of devices based on user needs and available resources. This improves affordability and significantly enhances accessibility, making it more inclusive.

Moreover, while the current implementation involves transmitting video data over a network, the system architecture supports fully offline processing when deployed on capable hardware. The integration of the offline text-to-speech module (pyttsx3) ensures uninterrupted audio guidance, even in low-connectivity or rural areas. This makes the system practical and reliable for widespread use.

By combining intelligent perception and decision-making in a modular, scalable framework, this work lays the foundation for next-generation assistive technology. It

enhances the mobility, safety, and independence of visually impaired individuals while remaining open to future improvements.

The modular design also allows developers to extend the system with advanced features such as GPS integration, voice control, and semantic scene understanding. This makes it a promising and forward-thinking step in the field of accessible navigation aids.

7.2 FUTURE ENHANCEMENT

While the current system demonstrates promising results in providing real-time navigation assistance for visually impaired individuals, there are several avenues for future enhancements that can elevate its functionality and make the user experience more holistic and robust. Some potential improvements include:

- **Text Detection for Environmental Awareness**
  Integrating Optical Character Recognition (OCR) can enable the system to detect and read out important textual information from the environment, such as signboards, labels, bus numbers, or shop names, thereby expanding its utility beyond navigation and offering a more contextual understanding of surroundings.
- **Advanced Scene Understanding and Obstacle Recognition**
  Future versions of the system can be extended to recognize a wider variety of dynamic and static obstacles, including temporary barriers, small or low-lying objects, and moving hazards like pets or cyclists. In addition, implementing semantic scene understanding can help the system interpret and explain the surroundings, enabling users to gain a clearer mental map of the environment.
- **Optimization for Edge Devices**
  A key step toward scalability and widespread adoption is to optimize the system for low-power, edge devices such as Raspberry Pi or embedded boards. This would enable fully offline processing, reducing latency, improving real-time performance, and ensuring the system functions effectively without reliance on internet connectivity, which is crucial for deployment in remote or rural regions.

By incorporating these enhancements, the system can evolve into a more intelligent, context-aware, and efficient navigation tool, offering a richer and more empowering experience for visually impaired individuals.

REFERENCE

[1] Hong K, He W, Tang H, Zhang X, Li Q, Zhou B. *SPVINet: A Lightweight Multitask Learning Network for Assisting Visually Impaired People in Multiscene Perception.* IEEE Internet of Things Journal. 2024 Mar 1.

[2] Joshi RC, Singh N, Sharma AK, Burget R, Dutta MK. *AI-SenseVision: A Low-Cost Artificial-Intelligence-Based Robust and Real-Time Assistance for Visually Impaired People.* IEEE Transactions on Human-Machine Systems. 2024 Mar 29.

[3] Leong X, Ramasamy RK. *Obstacle Detection and Distance Estimation for Visually Impaired People*. IEEE Access. 2023 Nov 30;11:136609-29.

[4] Said Y, Atri M, Albahar MA, Ben Atitallah A, Alsariera YA. *Obstacle detection system for navigation assistance of visually impaired people based on deep learning techniques*. Sensors. 2023 Jun 1;23(11):5262.

[5] Kuriakose B, Shrestha R, Sandnes FE. *DeepNAVI: A deep learning based smartphone navigation assistant for people with visual impairments*. Expert Systems with Applications. 2023 Feb 1;212:118720.

[6] Nitta Y, Isogawa M, Yonetani R, Sugimoto M. *Importance Rank-Learning of Objects in Urban Scenes for Assisting Visually Impaired People*. IEEE Access. 2023 Jun 16;11:62932-41.

[7] Kim M, Park Y, Moon K, Jeong CY. *Deep Learning-Based Optimization of Visual–Auditory Sensory Substitution*. IEEE Access. 2023 Feb 9;11:14169-80.

[8] Ashiq F, Asif M, Ahmad MB, Zafar S, Masood K, Mahmood T, Mahmood MT, Lee IH. *CNN-based object recognition and tracking system to assist visually impaired people*. IEEE access. 2022 Jan 31;10:14819-34.

[9] Masud U, Saeed T, Malaikah HM, Islam FU, Abbas G. *Smart assistive system for visually impaired people obstruction avoidance through object detection and classification.* IEEE access. 2022 Jan 25;10:13428-41.

[10]Lo Valvo A, Croce D, Garlisi D, Giuliano F, Giarré L, Tinnirello I. *A navigation and augmented reality system for visually impaired people*. Sensors. 2021 Apr 28;21(9):3061.

[11] Mahendran JK, Barry DT, Nivedha AK, Bhandarkar SM. *Computer vision-based assistance system for the visually impaired using mobile edge artificial intelligence*. InProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021 (pp. 2418-2427).

[12] Martínez-Cruz S, Morales-Hernández LA, Pérez-Soto GI, Benitez-Rangel JP, Camarillo-Gómez KA. *An outdoor navigation assistance system for visually impaired people in public transportation.* IEEE Access. 2021 Sep 10;9:130767-77.

[13] Mukhiddinov M, Cho J. *Smart glass system using deep learning for the blind and visually impaired*. Electronics. 2021 Nov 11;10(22):2756.

[14] Joshi RC, Yadav S, Dutta MK, Travieso-Gonzalez CM. *Efficient multi-object detection and smart navigation using artificial intelligence for visually impaired people*. Entropy. 2020 Aug 27;22(9):941.

[15] Felix SM, Kumar S, Veeramuthu A. *A smart personal AI assistant for visually impaired people.* In2018 2nd international conference on trends in electronics and informatics (ICOEI) 2018 May 11 (pp. 1245-1250). IEEE.

APPENDIX

A1- SOURCE CODE

YOLOv11 - Training

```
import ultralytics
from ultralytics import YOLO
import torch
device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')
print(f"Using device: {device}")
data_yaml_path = '/kaggle/input/yamldata/datanew.yaml'
model = YOLO("yolov11n.pt")
# Train on first dataset (BDD100K)
model.train(data="/kaggle/input/datayaml/datayaml/BDD100K.yaml
",
            epochs=30, imgsz=960, freeze=10)
model = YOLO("/kaggle/working/runs/detect/train2/weights/best.p
t")
# Train on first dataset (Pothole Detection)
model.train(data="/kaggle/input/datayaml/datayaml/RAD.yaml",
            epochs=10, imgsz=640, freeze=10)
model_path = "yolo11nbest.pt"
model.save(model_path)
print(f"Model saved as {model_path}")
```

Distance/ Proximity estimation

```
import cv2
import torch
import numpy as np
from ultralytics import YOLO
import tensorflow as tf

def classify_distance(bbox_height, frame_height):
    ratio = bbox_height / frame_height  # Bounding box height a
s fraction of frame height

    if ratio > 0.45:
        return "Very Near", (0, 0, 255)  # Red
    elif ratio > 0.30:
        return "Near", (0, 255, 255)  # Yellow
    elif ratio > 0.15:
        return "Far", (0, 255, 0)  # Green
    else:
        return "Very Far", (255, 0, 0)  # Blue

def get_proximity(y2, y1, frame_height):
    bbox_height = y2 - y1
```

```
    return classify_distance(bbox_height, frame_height)[0]  # E
xtracts distance label
```

Navigation Logic

```
def get_position(x1, x2, frame_width):
    center_x = (x1 + x2) / 2
    if center_x < frame_width * 0.33:
        return "left"
    elif center_x > frame_width * 0.66:
        return "right"
    else:
        return "center"

def calculate_impact(class_label):
    if class_label in ["car", "bus", "truck"]:
        return 10  # High-risk static object
    elif class_label in ["person", "bicycle"]:
        return 5  # Medium-risk dynamic object
    elif class_label in ["traffic cone", "dog"]:
        return 2  # Low-risk
    else:
        return 1  # Default minimal impact

def assess_risk(direction, detected_objects):
    risk_score = 0
    for obj in detected_objects:
        obj_position, obj_distance, obj_impact = obj
        if direction == "left" and obj_position in ["left", "ce
nter"]:
            risk_score += weighted_risk(obj_distance, obj_impac
t)
        if direction == "right" and obj_position in ["right", "
center"]:
            risk_score += weighted_risk(obj_distance, obj_impac
t)
    return risk_score

def weighted_risk(distance, impact):
    if distance == "Very Near":
        return impact * 3
    elif distance == "Near":
        return impact * 2
    elif distance == "Far":
        return impact * 1
    else:
        return 0

def navigation_logic(detected_objects):
    risk_scores = {"left": 0, "right": 0}
```

```
    for obj_position, obj_distance, obj_impact in detected_obje
cts:
        if obj_position == "center" and obj_distance == "Very N
ear":
            risk_scores["left"] += assess_risk("left", detected
_objects)
            risk_scores["right"] += assess_risk("right", detect
ed_objects)
            return "Move right" if risk_scores["left"] < risk_s
cores["right"] else "Move left"
    return "Proceed Forward"
```

Audio Feedback

```
import pyttsx3

# Initialize speech engine
engine = pyttsx3.init()
engine.setProperty('rate', 150)
engine.setProperty('volume', 1.0)

last_instruction = None  # Tracks the last spoken direction

voices = engine.getProperty('voices')
engine.setProperty('voice', voices[132].id)  # Set to voice num
ber 132


cap = cv2.VideoCapture(0)
class_names = model.names  # Get class labels (COCO dataset)
```

Real-time Integration

```
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break  # End of video

    frame_height, frame_width, _ = frame.shape  # Get frame dim
ensions
    results = model(frame)  # Run YOLOv11 detection
    detected_objects = []

    for result in results:
        detections = result.boxes.xyxy.cpu().numpy()
        class_ids = result.boxes.cls.cpu().numpy()

        for i, box in enumerate(detections):
```

```
            x1, y1, x2, y2 = map(int, box[:4])
            class_id = int(class_ids[i])
            class_label = class_names[class_id]
            obj_position = get_position(x1, x2, frame_width)
            obj_distance = get_proximity(y2, y1, frame_height)
            obj_impact = calculate_impact(class_label)
            detected_objects.append((obj_position,  obj_distanc
e, obj_impact))

            # Draw bounding box and label
            _, color = classify_distance(y2 - y1, frame_height)
            display_text = f"{class_label} - {obj_distance}"
            cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
            cv2.putText(frame, display_text, (x1, y1 - 10), cv2
.FONT_HERSHEY_SIMPLEX, 0.6, color, 2)

    # Determine navigation direction
    direction = navigation_logic(detected_objects)
    cv2.putText(frame, f"Direction: {direction}", (50, 50), cv2
.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 0), 2)

    # Speak only if the instruction changes and it's not 'Proce
ed Forward'
    if direction != last_instruction and direction != "Proceed
Forward":
        engine.say(direction)
        engine.runAndWait()
        last_instruction = direction
    elif direction == "Proceed Forward":
        last_instruction = None  # Reset so that next left/righ
t will be spoken

    # Display the processed frame
    cv2.imshow("Navigation Assistance", frame)

    # Press 'q' to exit
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

# Release resources
cap.release()
cv2.destroyAllWindows()
```

Technical Biography

Harini Sri B (210171601017), was born in the year 2003 in Tamil Nadu, India. She completed his higher secondary education in 2021. She is currently pursuing a B. Tech Artificial Intelligence and Data Science at the School of Computer Information and Mathematical Sciences at B.S. Abdur Rahman Crescent Institute of Science and Technology which is located at Vandalur, Chennai, India. Her interests lie in the areas of Data Science, Machine Learning and such. The email for communication is harinesri012@gmail.com.

Batch 9-16AI&DS_2025

by Madhina Banu

Submission date: 16-Apr-2025 08:08PM (UTC+0530)
Submission ID: 2375869117
File name: VIII_AI_DS__Batch_9-16.pdf (2.26M)
Word count: 79671
Character count: 475054

Batch 9-16AI&DS_2025

ORIGINALITY REPORT

13%	9%	10%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.mdpi.com Internet Source	1%
2	R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P. Prasad. "Algorithms in Advanced Artificial Intelligence - Proceedings of International Conference on Algorithms in Advanced Artificial Intelligence (ICAAAI-2024)", CRC Press, 2025 Publication	<1%
3	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	<1%
4	eitca.org Internet Source	<1%
5	fastercapital.com Internet Source	<1%
6	Bui Thanh Hung, M. Sekar, Ayhan Esi, R. Senthil Kumar. "Applications of Mathematics in Science and Technology - International Conference on Mathematical Appplications in Science and Technology", CRC Press, 2025 Publication	<1%
7	Submitted to Liverpool John Moores University Student Paper	<1%
8	arxiv.org Internet Source	<1%

21	www.dsengg.ac.in Internet Source	<1%
22	Submitted to University of Hertfordshire Student Paper	<1%
23	Alboom, Rashed Khalid. "Energy Forecasting Inaccuracies and Their Direct Impact on Grid Performance", Rochester Institute of Technology Publication	<1%
24	www.preprints.org Internet Source	<1%
25	www.fastercapital.com Internet Source	<1%
101	Clash, Shelaniece Qiaunna. "Cyber Reliability Analysis of Autonomous Systems", Morgan State University, 2025 Publication	<1%
102	Dongfeng Ren, Xin Qiu, Zehua An. "A Multi-Source Data-Driven Analysis of Building Functional Classification and Its Relationship with Population Distribution", Remote Sensing, 2024 Publication	<1%
80	Submitted to Queensland University of Technology Student Paper	<1%
164	Sangkyoung Lee, Zhuoxiao Chen, Yadan Luo, Xuliang Li, Mingyuan Lu, Zi Helen Huang, Han Huang. "Enhanced prediction accuracy in high-speed grinding of brittle materials using advanced machine learning techniques", Journal of Intelligent Manufacturing, 2024 Publication	<1%