A PROJECT STAGE-I REPORT ON

CONTEXT AWARE AUDIO GUIDANCE SYSTEM FOR VISUALLY IMPAIRED

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

BACHELOR OF ENGINEERING
(ELECRONICS & TELECOMMUNICATION ENGINEERING)
SUBMITTED BY

PARDESHI SIDDHARTH MANOJ	46
KULKARNI SAIDEEP RAHUL	45
KANAWADE MAHESH NITIN	43
KALE TEJAS GAHININATH	47

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

PRAVARA RURAL ENGINEERING COLLEGE, LONI

At/Post : Loni Bk, Tal : Rahata, Dist : Ahilyanagar

SAVITRIBAI PHULE PUNE UNIVERSITY

2025-26

CERTIFICATE

This is to certify that the project report entitles

CONTEXT AWARE AUDIO GUIDANCE SYSTEM FOR VISUALLY IMPAIRED

Submitted by

PARDESHI SIDDHARTH MANOJ	**46**
KULKARNI SAIDEEP RAHUL	**45**
KANAWADE MAHESH NITIN	**43**
KALE TEJAS GAHININATH	**47**

is a bonafide student of this institute and the work has been carried out by him under the supervision of **Prof. Mr. M. R. Gaikar** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Electronics and Telecommunication Engineering).

Prof. Mr. M. R. Gaikar
Guide
Dept. of ETC Engg.

External Examiner

Mr. S. B. Mandlik
Project Coordinator
Dept. of ETC Engg.

Mr. V. A. Aher
HOD
Dept. of ETC Engg.

Dr. S. M. Gulhane
Principal
PREC, Loni

ACKNOWLEDGEMENT

We take this opportunity to express my hearty thanks to all those who helped us in the completion of the Project work stage -1 on this topic. We would especially like to express my sincere gratitude to **Prof. Mr. M. R. Gaikar,** our Guide, **Mr. S.B. Mandlik**, Project Coordinator and **Mr. V.A.Aher**, HOD Department of Electronics and Telecommunication Engineering who extended their moral support, inspiring guidance and encouraging independence throughout this task. We would also thank our Principal **Dr. S. M. Gulhane** for his great insight and motivation. Last but not least, we would like to thank my fellow colleagues for their valuable suggestions.

PARDESHI SIDDHARTH MANOJ

KULKARNI SAIDEEP RAHUL

KANAWADE MAHESH NITIN

KALE TEJAS GAHININATH

ABSTRACT

This project develops a wearable **Context-Aware Audio Guidance System for the Visually Impaired** that provides real-time, spoken navigation assistance to enable safer and more independent mobility. The prototype combines a lightweight glasses-mounted high-clarity camera and a compact depth sensor with a belt-mounted Raspberry Pi (processing and power) to keep the eyewear comfortable. Video frames are processed on-device using an optimized deep-learning detector (e.g., YOLO/MobileNet variants) to identify multiple obstacles simultaneously; distance and spatial location are obtained by fusing depth sensor data (ToF/stereo/Depth camera). Detected objects are translated into short, pre-recorded contextual audio prompts (type, distance, and suggested direction) delivered through bone-conduction earphones so ambient sound remains audible.

The system emphasizes low end-to-end latency and robust multi-object handling to minimize false alarms and cognitive load. Design work covers hardware layout, sensor fusion, model optimization for edge inference, and human-centered audio interface design. The project will validate performance through lab benchmarks (detection accuracy, latency, battery life) and user trials evaluating safety, usability, and user preference.

The resulting system aims to be an affordable, reliable assistive aid that augments traditional mobility tools and improves the independent navigation experience for visually impaired users.

TABLE OF CONTENTS

LIST OF ABBREVATIONS

No. — Number

Sr. — Serial

Fig. — Figure

API — Application Programming Interface

AI — Artificial Intelligence

DL — Deep Learning

FPS — Frames Per Second

GPU — Graphical Processing Unit

Hz — Hertz

IoT — Internet of Things

I^2C — Inter-Integrated Circuit

LiDAR — Light Detection And Ranging

mAh — milliampere-hour

MS — Millisecond (ms)

Pi / RPi — Raspberry Pi

Py / py — Python

SBC — Single-Board Computer

TTS — Text-To-Speech

Ultralytics — Ultralytics YOLO family (YOLOv8n).

USB — Universal Serial Bus

Wi-Fi — Wireless Fidelity

YOLO — You Only Look Once

TTS — Text to Speech

LIST OF FIGURES/DIAGRAMS

LIST OF TABLES

INTRODUCTION

Vision plays a vital role in human perception, orientation, and safe mobility. For millions of people with visual impairments, independent navigation remains one of the greatest daily challenges. Despite the use of traditional aids such as **white canes** and **guide dogs**, these methods have several limitations: canes can only detect obstacles that are physically close to the user, and guide dogs, while highly effective, are expensive and require training and maintenance. To address these challenges, there has been a growing interest in developing **intelligent electronic travel aids (ETAs)** that combine sensors, embedded computing, and artificial intelligence to provide real-time navigation assistance[1],[2].

The goal of this project, **"Context-Aware Audio Guidance System for the Visually Impaired,"** is to design and implement a **wearable smart-glasses-based assistive device** that acts as an artificial vision system. The proposed system uses a **camera mounted on the glasses** to continuously capture the surrounding environment. A **Raspberry Pi** serves as the main processing unit, running **AI-based object detection algorithms** to recognize obstacles such as walls, pedestrians, vehicles, and other objects in real time. The system also estimates the **distance and position** of these obstacles using a **depth-sensing module** (such as a Time-of-Flight or stereo camera) or ultrasonic sensors[1].

Based on the detected information, the system provides **contextual audio guidance** to the user through **bone-conduction earphones** or standard earphones. Instead of generic beeps or tones, the system delivers meaningful voice prompts such as *"Obstacle ahead, move slightly left"* or *"Person approaching two meters away."* This **context-aware audio feedback** helps the user make safe movement decisions while keeping their ears open to natural environmental sounds, such as vehicle horns or human voices.

A major design consideration in this project is **wearability and comfort[2],[3]**. To ensure that the device remains lightweight and easy to use, only the camera and minimal circuitry are mounted on the glasses, while heavier components such as the Raspberry Pi, battery, and power circuitry are worn on the user's waist or pocket. This distributed design enhances ergonomics without compromising performance.

The system leverages the power of **Edge AI**—processing data directly on the Raspberry Pi rather than relying on cloud connectivity. This approach ensures **low latency**, **privacy**, and **offline operation**, making it ideal for real-world use. The AI model is optimized for speed and accuracy using lightweight architectures such as **YOLOv5, MobileNet SSD, or TensorFlow Lite** models.

In addition to technical functionality, the project emphasizes **user-centered design[2]**. The feedback system is designed to be intuitive, clear, and minimally intrusive. The contextual awareness capability ensures that the user is not overwhelmed by unnecessary information, focusing only on relevant obstacles within a critical range. This makes the system practical for both indoor and outdoor use, including streets, corridors, and public spaces.

By integrating **computer vision, sensor fusion, and real-time audio processing**, the proposed system aims to provide an affordable, efficient, and reliable mobility solution for the visually impaired[3],[4]. The project not only demonstrates the application of embedded systems and AI in assistive technology but also contributes toward the vision of **inclusive and accessible smart devices** that empower differently-abled individuals to lead more independent and confident lives.

PROBLEM STATEMENT

Visual impairment severely limits an individual's ability to navigate safely and independently in unfamiliar or dynamic environments. According to the World Health Organization (WHO), over 285 million people worldwide suffer from some form of visual impairment, of which approximately 39 million are completely blind. These individuals rely heavily on **traditional mobility aids** such as white canes or guide dogs. While these tools offer a degree of independence, they possess **significant limitations** when dealing with modern, complex surroundings.

The **white cane**, for instance, can only detect obstacles that are within the user's immediate reach (typically less than one meter). It cannot provide information about objects at a distance, obstacles above waist height, or moving hazards such as vehicles or people[19]. Similarly, **guide dogs**, though effective, are expensive to train, require continuous care, and are not accessible to everyone[3]. As a result, visually impaired individuals often experience **reduced mobility**, **social dependence**, and **safety risks**, particularly in crowded or fast-changing environments.

Recent advancements in **computer vision**, **machine learning**, and **embedded systems** offer new opportunities to enhance mobility assistance through intelligent electronic travel aids (ETAs)[1]. However, most existing smart devices suffer from one or more of the following issues:

- **High cost and limited accessibility:** Many advanced assistive devices use specialized hardware such as LiDAR or high-end processing units, making them expensive and impractical for everyday users.
- **Limited contextual awareness:** Existing systems often provide only basic obstacle alerts (e.g., "object detected") without describing the nature, distance, or direction of the obstacle, leading to confusion.
- **High latency and poor real-time performance:** Some devices rely on cloud-based processing, which introduces delay in obstacle detection and feedback—unacceptable in real-world navigation.
- **Uncomfortable or heavy designs:** Many wearable systems are bulky or place excessive weight on the user's head, reducing comfort and discouraging continuous use.

- **Ineffective user interaction:** Some systems use only vibration or simple tones, which can be difficult to interpret, especially in noisy environments.

Therefore, there is a clear need for a **lightweight, affordable, and context-aware wearable system** that can assist visually impaired individuals by detecting and recognizing obstacles in real time and providing **intuitive audio guidance** for safe movement.

The proposed **Context-Aware Audio Guidance System** aims to fill this gap by developing a **smart-glasses-based assistive device** that uses a high-resolution camera, AI-based object detection, and distance sensing to identify multiple obstacles simultaneously[4]. The system will process data locally on a **Raspberry Pi** to ensure **low-latency, offline operation** and will communicate guidance through **pre-recorded voice prompts** delivered via **bone-conduction earphones**.

By combining **edge computing**, **real-time computer vision**, and **contextual audio feedback**, the project seeks to create a reliable and user-friendly mobility solution that enhances independence, confidence, and safety for visually impaired users in both indoor and outdoor environments.

PROJECT PLANNING

PHASE 1: SETTING UP DEVELOPMENT ENVIRONMENT

This phase establishes the hardware and software foundation required for the project.

Objectives:

- Set up the Raspberry Pi environment.
- Install the required software packages and libraries.
- Prepare tools for version control and debugging.

Tasks:

- Configure the Raspberry Pi 4 with Raspberry Pi OS (64-bit).
- Enable SSH and VNC for remote access.
- Update and upgrade system packages (sudo apt update && sudo apt upgrade).
- Install the necessary libraries:
    - **Python** and **pip** for development.
    - **OpenCV** for image capture and processing.
    - **TensorFlow Lite** runtime for AI inference.
    - **pyttsx3** and **espeak** for offline text-to-speech audio feedback.
- Set up **Git and GitHub** for version control.

Deliverable:

A fully configured Raspberry Pi ready for object detection and audio processing development.

PHASE 2: DEVELOPING CORE MODULES (In Isolation)

Each subsystem is developed separately to ensure independent functionality before integration.

Module 1: Object Detection

Goal: Enable the Raspberry Pi to detect and identify objects in the environment.

Tasks:

- Use a pre-trained **YOLOv8n** model converted into TensorFlow Lite (.tflite).
- Develop a Python script (object_detector.py) to:
    - Capture frames from the Pi camera.
    - Preprocess images and feed them to the model.

    - Retrieve and display detected object labels, bounding boxes, and confidence scores.
- Test detection accuracy and processing speed on live video feeds.

Deliverable:

A working object detection module capable of recognizing and classifying objects in real time.

Module 2: Direction & Distance Estimation (Simulated)

Goal: Estimate the direction (left, center, right) and relative distance (far, near, very close) of detected obstacles.

Tasks:

- Use bounding box positions for **direction estimation**:
    - Left third → “to your left”
    - Middle third → “in front”
    - Right third → “to your right”
- Use bounding box size as a **proxy for distance estimation**:
    - Large box → “very close”
    - Medium box → “near”
    - Small box → “far”

Deliverable:

A Python module capable of computing direction and relative distance from image coordinates, suitable for later integration with actual distance sensors.

Module 3: Audio Feedback

Goal: Convert detected obstacle information into real-time spoken guidance.

Tasks:

- Create a script (speaker.py) using **pyttsx3** for offline text-to-speech.
- Develop a function speak(text) that receives descriptive text and converts it into audio.
- Customize the voice rate and volume for clarity and comprehension.
- Test pre-recorded message playback for typical outputs (e.g., “Person near, ahead”).

Deliverable:

A functional audio module capable of delivering concise, real-time spoken feedback.

PHASE 3: SYSTEM INTEGRATION AND MAIN APPLICATION LOGIC

All modules are combined into a unified control program.

Tasks:

- Create a central script (main.py) to integrate object detection, direction/distance estimation, and audio output.
- Initialize camera, model, and TTS engine at startup.
- Implement two operating modes:
  - **Automatic Mode:** Provides periodic contextual updates about nearby obstacles.
  - **Manual Mode:** Activated on user command for a full scene description.
- Manage event timing to prevent repetitive announcements (using a cooldown period).
- Construct structured speech strings, e.g., *"Person near on your right"*.

Deliverable:

A complete working prototype combining vision, logic, and audio feedback for real-time operation.

PHASE 4: TESTING AND REFINEMENT

This phase focuses on optimization, user experience, and system stability.

Tasks:

- **Performance Optimization:**
  - Reduce model input size (e.g., 416×416) for faster inference.
  - Process every 3rd–5th frame to reduce load.
- **Audio Optimization:**
  - Simplify phrases for clarity ("Person near, ahead").
  - Prioritize urgent alerts (e.g., "Car very close" over "Chair far").
- **Hardware Integration:**
  - Replace simulated distance with actual ToF or ultrasonic sensors.
  - Mount the camera on the glasses frame and connect Pi via cable.
- **User Testing:**

- Conduct real-world navigation trials with multiple users.
- Collect feedback on comfort, response time, and guidance clarity.

Deliverable:

An optimized, user-tested wearable prototype ready for demonstration and documentation.

FEASIBILITY STUDY

TECHNICAL FEASIBILITY

System composition and why it's feasible

- Sensing: A compact RGB camera (RPI HQ / compatible module) plus a small depth sensor (VL53L5CX ToF or small stereo/depth camera) gives both object appearance and absolute distance. This sensor fusion is a standard, proven approach for obstacle detection and overcomes limitations of single ultrasonics or monocular vision.
- Compute: Raspberry Pi 4/5 (4–8 GB) can host optimized, quantized deep models (MobileNet-SSD, Tiny/YOLO variants) via TensorFlow-Lite or ONNX runtimes. For heavy workloads, a USB Edge TPU (Coral) or NCS2 can accelerate inference with marginal extra integration effort.
- Software: OpenCV + Python + TensorFlow-Lite is a well-supported stack on the Pi; object detection + simple tracking + audio output pipelines are commonly implemented on similar platforms.
- Latency and throughput: With an optimized model, reduced input resolution (320–416 px), and frame skipping, inference latencies of tens to a few hundred milliseconds are realistic on Pi + Edge TPU. Combined pipeline (capture → detect → distance read → decide → audio) can be engineered to target end-to-end latency <300 ms, which literature and user studies indicate is acceptable for navigation assistance.
- Weight & ergonomics: Putting only the camera and small electronics on the frame and housing the Pi + battery in a belt pouch keeps eyewear light (<50–80 g) and the total wearable weight comfortably under ~500 g.
- Audio interface: Bone-conduction earphones are readily available and preserve ambient hearing. Using pre-recorded messages avoids TTS latency and pronunciation errors.

Hardware/software risks (technical showstoppers)

- Night/very low-light performance: RGB camera depends on light. Mitigation: auto-exposure/HDR, IR illuminator or ToF fallback, and fallback "safe mode" (vibration + sonar).

- Real-time constraints: If inference is too slow on Pi alone, you must add an accelerator or reduce model complexity. This is not a blocker — well-known mitigations exist.
- Outdoor robustness: rain/glare may degrade sensors; robust enclosures and weatherproofing plus algorithmic filters (temporal smoothing) handle many cases.
  Conclusion: Technically viable with standard components and optimizations.

ECONOMIC FEASIBILITY

High-level cost drivers

- Main costs: Raspberry Pi 4/5, camera, depth sensor, battery, bone-conduction earphones, mechanical frame (3D printing), wiring, optional Edge TPU, misc. electronics.
- Development costs: prototype parts, test subjects (if paid), fabrication/3D printing, possibly small user-study incentives.

Order-of-magnitude cost estimate (approximate)

- Basic prototype (Pi, Pi camera, VL53L5 ToF, earphones, battery, wiring, casing): roughly USD 200–450 (INR ~16k–40k), depending on model choices and accelerators.
- With acceleration (Coral USB TPU) and higher-end depth camera (e.g., RealSense): USD 350–800 (INR ~28k–65k).
  Note: these are ballpark figures to justify feasibility; if budget isn't a constraint per earlier message, choose higher-reliability sensors.

Cost/benefit judgment

- For a final-year engineering project, the above cost is acceptable and typical. The core research, software, and user-testing value is high relative to hardware costs.

Operational feasibility

Team skills and resources required

- Embedded Linux / Raspberry Pi familiarity (system setup, power management).
- Python programming, OpenCV, experience with TensorFlow/TF-Lite or PyTorch Mobile.

- Basic electronics (I2C, UART), soldering, cable routing; 3D printing or mechanical mounting skills.
- UX/usability testing experience (designing tasks, obtaining consent).

Availability of resources

- All parts are widely available online and locally; libraries and frameworks are open source and well documented.
- Campus labs can support 3D printing, soldering, and user testing spaces.

Maintenance & deployment

- Prototype intended for research/demonstration. Long-term productization requires better enclosures, battery certification, and industrial design. For the scope of the project, the device is fully maintainable by the team.

Conclusion: Operationally feasible within a typical 3–4 member final-year team.

LEGAL, SAFETY AND ETHICAL FEASIBILITY

Privacy

- Camera captures public scenes which may include people; for user trials, obtain consent and blur faces in any shared media. Store only necessary logs and anonymize data.

Safety

- Provide emergency stop / mute and ensure audio volume preserves ambient hearing (bone conduction preference). Use low-power, certified batteries and implement safe charging.

Ethics and user testing

- Obtain institutional ethics approval or supervisor consent for trials on visually impaired participants. Use informed consent forms and ensure no trial endangers participants (trial routes controlled, accompanying supervisor or cane as needed).

Regulatory

- For a prototype proof-of-concept, no formal medical certification is required. If commercialized, regulatory requirements will apply (medical device standards, battery/EMC testing).

Conclusion: Feasible if you follow ethical guidelines and safety protocols.

Risk analysis and mitigation

Technical risk: inference latency too high

- Mitigation: reduce input resolution, prune/quantize model, add Edge TPU, or reduce processing rate.

Technical risk: depth sensor noisy / fails for some surfaces

- Mitigation: fuse ToF + bounding-box size proxy + ultrasonic backup; add temporal smoothing and thresholding.

User acceptance risk: audio too frequent or confusing

- Mitigation: design concise prompts, implement priority & rate-limiting, pilot test with users and iterate.

Power/battery risk: runtime too short

- Mitigation: measure current draw early; choose higher capacity battery; implement low-power modes and frame skipping.

Safety/legal risk: trial participant injury

- Mitigation: supervised trials, safe controlled environments, informed consent, allow user to use cane/companion during tests.

VALIDATION & SUCCESS CRITERIA

Technical metrics to meet (pass/fail)

- Detection accuracy: average precision (mAP) for key obstacle classes ≥ 70–80% in test conditions relevant to user scenarios (people, vehicles, stairs).
- Multi-object detection & handling: must correctly detect and report at least 2 simultaneous obstacles in the field of view in 90% of test frames.
- Latency: end-to-end pipeline (frame capture → audio output) under 300 ms for a usable user experience; target <200 ms if possible.
- Battery: continuous operation ≥ 4–6 hours under typical workload.
- Wearability: eyewear component mass <100 g; total wearable mass <500 g.
- Usability: in small user trials, a measurable reduction in obstacle contact rate (compared to cane only) or subjective improvement in confidence (Likert scale) — target improvement significant at $p<0.05$ if statistical testing available.

Test plan (short)

- Bench tests: measure inference times, average power draw, and detection accuracy on recorded test videos.

- Controlled indoor course: fixed obstacles arranged in routes; timed trials with blindfolded sighted users and (if possible) visually impaired volunteer(s); measure collisions, navigation time, and collect subjective feedback.
- Outdoor walk test: short supervised walk in a safe area to evaluate lighting and real-world robustness.

HUMAN RESOURCES & RESPONSIBILITIES

Team of three (suggested division)

- Member A: Hardware & electronics (mounting, sensors, battery, power regulation, wiring).
- Member B: AI & software engine (model selection & optimization, inference pipeline, sensor fusion).
- Member C: Testing (audio prompts design, user tests, documentation, report writing).

  All members should cross-review code, test together, and participate in user trials.

MAINTAINABILITY & SCALABILITY

- Code: use modular design (object_detector.py, distance_estimator.py, speaker.py, main.py), version control (Git), and clear documentation.
- Hardware: design 3D-printable mounts and modular connectors for easy replacement.
- Future upgrades: adding GPS, SLAM, improved depth camera, or cloud-assisted features (optional).

ENVIRONMENTAL & SOCIAL FEASIBILITY

- Social impact: high — improves mobility & independence for visually impaired people. The design emphasizes affordability and offline operation to suit a wide demographic.
- Environmental: choose rechargeable Li-ion with safe disposal practices; small electronics footprint; 3D printed parts minimize waste if planned.

FINAL FEASIBILITY JUDGEMENT (short)

Technically, operationally, ethically and economically feasible as a final-year engineering project. Main technical challenge is achieving sufficiently low latency and reliable detection in varied lighting; both are solvable via model optimization, sensor fusion, and optional accelerators. With careful planning, user-centered design, and supervised trials, the project is achievable and likely to deliver meaningful, demonstrable results within a standard semester timeline.

LITERATURE SURVEY

In recent years, the integration of artificial intelligence (AI), embedded systems, and sensor fusion has significantly advanced assistive technologies for individuals with visual impairments. Traditional aids such as white canes and guide dogs, while effective in limited contexts, fail to detect high-level or distant obstacles and require either physical contact or costly maintenance. Researchers have therefore turned toward **smart wearable systems**—combining cameras, sensors, and edge AI—to provide real-time guidance through audio or haptic feedback.

Elmannai and Elleithy [1] conducted a comprehensive review of **sensor-based assistive devices** and highlighted key limitations of early electronic travel aids (ETAs), such as short-range detection and limited environmental awareness. They emphasized the need for **multi-sensor fusion** and intelligent processing for reliable navigation in dynamic environments. Similarly, Zhang et al. [2] and Lavric et al. [3] discussed emerging **AI-driven wearable systems**, focusing on multimodal interfaces (audio, haptic, and visual feedback) and the integration of deep learning for semantic understanding of surroundings. Together, these foundational surveys establish that the future of assistive devices lies in AI-enhanced perception and lightweight, context-aware wearables.

Several studies have since focused on **wearable prototypes** designed for real-world usability. Brilli et al. [4] introduced *AIris*, an AI-powered smart-glasses system capable of performing multiple assistive tasks such as scene description, face detection, and text reading using onboard vision algorithms and natural language output. Similarly, Oduah et al. [5] developed a **goggle-style echolocation headset** using ultrasonic sensors and stereo sound to indicate obstacle proximity and direction, demonstrating significant reductions in collisions during trials with blind users. These works show how compact, low-cost wearable devices can deliver multi-sensory guidance while maintaining comfort and low latency.

Beyond head-mounted systems, other researchers have explored **handheld or cane-integrated solutions**. Dernayka et al. [8] evaluated *Tom Pouce III*, a LiDAR-equipped electronic cane that improved obstacle detection accuracy without reducing walking speed. Mungdee et al. [9] enhanced smart-cane designs by combining **dual cameras and ultrasonic sensors** with YOLO-based object detection for real-time path-surface classification, achieving 92% surface recognition accuracy. Mai et al. [6] developed a LiDAR and RGB-D camera-based cane using Jetson Nano, achieving real-time mapping and 84% recognition accuracy for complex urban objects such as cars and crosswalks. These works establish the reliability of AI-assisted sensing but highlight portability challenges—reinforcing the rationale for a **glasses-based design**.

For **real-time object detection**, several studies validate the suitability of compact neural networks for embedded devices. Islam et al. [10] implemented a **MobileNetV2-SSD detector** on a Raspberry Pi 4 B to identify surrounding objects and provide text-to-speech feedback in real time, achieving reliable frame-by-frame recognition and demonstrating the feasibility of edge AI for assistive applications. Baldovino et al. [11] developed a deep learning–based visual aid integrating **audio and haptic feedback**, achieving 92% classification accuracy on a custom dataset and introducing proximity-based warning logic for enhanced contextual alerts. Yilmaz et al. [12] compared YOLOv4 and YOLOv5 architectures and found YOLOv5 achieved higher frame rates and accuracy, making it ideal for wearable real-time inference. Liu et al. [13] proposed *EdgeYOLO*, a lightweight anchor-free model optimized for edge GPUs, sustaining 30 FPS with 69% AP@50—confirming that **modern detectors can run on portable, low-power hardware**.

Studies such as Surantha and Sutisna [14] provide design guidelines for **edge-AI optimization**, emphasizing quantization, pruning, and runtime frameworks (TensorFlow Lite, PyTorch Mobile) to meet latency constraints on platforms like the Raspberry Pi and Jetson. Together, these references justify the selection of **lightweight, single-stage detectors** such as YOLO or MobileNet for the proposed system.

Hardware and system integration have also been addressed in multiple **ScienceDirect studies**. Bouteraa et al. [6] presented a **smart real-time navigation support system** built on Raspberry Pi 4 with ultrasonic and IR sensors, controlled via fuzzy logic to generate haptic and audio guidance. Liu and Feng [7] designed a **wireless haptic assistance device** integrating ultrasonic and ToF sensors with small vibration motors, demonstrating precise obstacle detection and fast feedback response. Măgurean and Beșoiu [5] introduced an **edge-inference smart-glasses system** for visually impaired runners, combining line detection and "safety-zone" classification with 92.5% accuracy and just 64 ms latency. These systems underline the importance of **compact, distributed**

architectures—placing only light sensors on the head while offloading computation to a body-mounted module.

User-centered studies confirm that **usability and feedback design** are as critical as technical accuracy. Camacho et al. [17] demonstrated that **vibrotactile navigation cues** improved path accuracy for blind users, albeit with slightly longer travel times. Dos Santos et al. [18] evaluated *NavWear*, a backpack-mounted RGB-D camera with a haptic belt, and found that combining it with a traditional white cane **significantly reduced collisions** and improved user confidence. Pundlik et al. [16] reported a 37% reduction in obstacle contacts using a chest-mounted collision-warning device with haptic wristbands. Seiple et al. [19] further verified high acceptance rates of AI-driven assistive devices such as Envision Glasses and Seeing AI apps. Collectively, these studies highlight that **feedback modality, simplicity, and cognitive comfort** drive user adoption and real-world impact.

Several specialized applications expand on **context-awareness**. A 2024 ScienceDirect study applied **YOLOv5 to pothole detection**, achieving 82.7% accuracy and 30 FPS, showing that domain-specific obstacle detection can be integrated into broader wearable systems. Similarly, early research on **smart textiles** [15] embedded ultrasonic sensors and vibration actuators into clothing using fuzzy logic controllers, proving that assistive functionality can be embedded unobtrusively in daily wear.

Across this body of work, clear trends emerge:

- **Sensor fusion** (RGB + depth/ultrasonic) consistently improves obstacle recognition robustness.
- **Edge-optimized AI** (MobileNet, YOLOv5-Nano, EdgeYOLO) enables real-time performance on Raspberry Pi-class devices.
- **User-centered feedback** through concise audio or haptic cues enhances situational awareness without cognitive overload.
- **Lightweight, modular hardware design**—separating sensing and computation—improves comfort and scalability.

IDENTIFIED GAPS & OPPORTUNITIES

- **Lightweight glasses + robust depth fusion on Pi-class hardware:** Several works use standoff devices (backpack/cane). Fewer combine true glasses-mounted camera with on-device inference and a compact multi-zone ToF to keep eyewear light. Your distributed design (camera on glasses; Pi + battery on belt) addresses this gap.

- **Pre-recorded concise contextual audio (not just TTS) optimized for latency and comprehension:** Some systems rely on TTS; pre-recorded, short phrases reduce processing time and improve clarity — a practical improvement your project can exploit.
- **Multi-object prioritization & natural guidance logic:** Existing prototypes often give raw alerts; you can implement and evaluate prioritized, context-aware guidance (closest/most hazardous first; grouped announcements).
- **Empirical latency/UX tradeoff analysis on Pi + optional Coral TPU:** While edge inference is discussed, few studies present a careful latency vs user-satisfaction analysis for Pi + Coral in real navigation. Your project can measure this and provide concrete engineering guidelines.
- **Dataset & evaluation for visually-impaired-centric obstacles (curb/stairs/overhangs):** Many datasets are COCO-style; you can create a focused dataset for mobility hazards and report detection metrics relevant to the blind community.

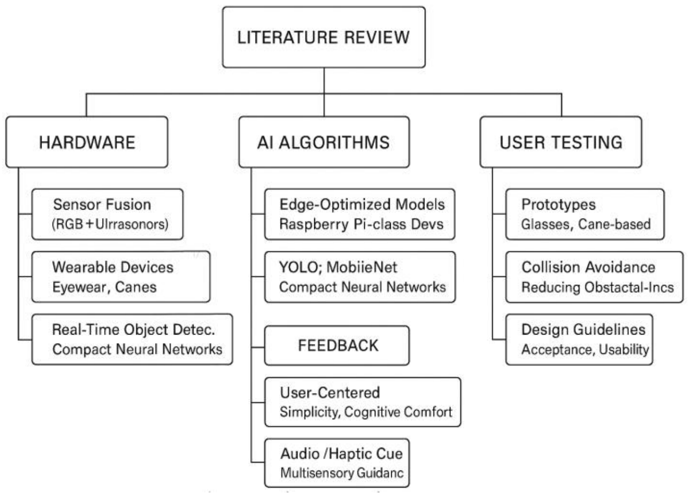
LITERATURE REVIEW
HARDWARE
AI ALGORITHMS
USER TESTING
Sensor Fusion (RGB+Ulrrasonors)
Wearable Devices Eyewear, Canes
Real-Time Object Detec. Compact Neural Networks
Edge-Optimized Models Raspberry Pi-class Devs
YOLO; MobiieNet Compact Neural Networks
FEEDBACK
User-Centered Simplicity, Cognitive Comfort
Audio /Haptic Cue Multisensory Guidanc
Prototypes Glasses, Cane-based
Collision Avoidance Reducing Obstactal-Incs
Design Guidelines Acceptance, Usability

Fig 3.1 Literature Review

REQUIREMENTS SPECIFICATIONS

OVERVIEW

The proposed *Context-Aware Audio Guidance System for the Visually Impaired* uses a Raspberry Pi 4 as the central processing unit to capture live video from a Raspberry Pi Camera Module 3, detect obstacles using the YOLOv8n deep-learning model, estimate their distance through a LiDAR sensor, and provide spoken navigation cues through headphones. The device assists visually impaired users by giving real-time, context-aware guidance while remaining lightweight, affordable, and wearable[4].

FUNCTIONAL REQUIREMENTS

ID	Requirement	Description
FR-1	Image acquisition	Continuously capture live video using the Pi Camera Module 3 at $\geq$ 10 fps.
FR-2	Object detection	Run YOLOv8n on each frame to identify obstacles such as people, vehicles, walls, and stairs.
FR-3	Distance estimation	Obtain obstacle distance using LiDAR data; fuse with bounding-box location.
FR-4	Direction estimation	Classify obstacles as *Left*, *Center*, or *Right* relative to user.
FR-5	Audio feedback	Generate real-time spoken messages (via **pyttsx3**) or play pre-recorded WAVs describing obstacle and distance.
FR-6	Multi-object handling	Detect and prioritize at least two obstacles simultaneously.
FR-7	Low-latency response	Ensure total detection-to-audio delay $\leq$ 300 ms.
FR-8	Battery monitoring	Alert user through voice when battery level drops below threshold.

ID	Requirement	Description
FR-9	Mode control	Provide basic input controls (Start, Stop, Volume, Mute).
FR-10	Data logging	Log detections, timestamps, and battery status for analysis.

Table 4.1 Functional Requirements

NON-FUNCTIONAL REQUIREMENTS

Parameter	Target / Constraint
Performance	Minimum 5–10 FPS detection; latency ≤ 300 ms.
Accuracy	≥ 70 % mAP for critical obstacle classes.
Weight	Glasses module < 100 g; total system < 500 g.
Operating time	≥ 6 hours continuous operation on 10 000 mAh power bank.
Reliability	Stable operation without thermal throttling or system crash.
Usability	Clear, concise voice messages that do not mask ambient sound.
Portability	Wearable and comfortable for extended use.
Privacy	No image data transmitted or stored externally.
Safety	Safe audio levels; insulated wiring; secure battery pack.

Table 4.2 Non-Functional Requirements

HARDWARE SPECIFICATIONS

Component	Specification / Role
Processing Unit	Raspberry Pi 4 Model B (8 GB RAM); Quad-core Cortex-A72 @ 1.5 GHz.
Camera	Raspberry Pi Camera Module 3 – 12 MP Sony IMX708; 4608 × 2592 max resolution; autofocus; CSI interface.
Distance Sensor	LiDAR Module (USB/I^{2}C interface, 0.05 m – 5 m range, ± 3 cm accuracy).

Component	Specification / Role
Audio Output	Wired or bone-conduction headphones via 3.5 mm jack; latency < 50 ms.
Power Source	10 000 mAh power bank @ 5 V / 3 A; estimated runtime ≈ 6 – 7 hours.
Storage	32 GB micro-SD card for OS, model, logs, and audio files.
Accessories	Lightweight eyewear frame (camera + LiDAR) and belt pouch (Pi + battery).

Table 4.3 Hardware Specifications

SOFTWARE SPECIFICATIONS

Category	Details
Operating System	Raspberry Pi OS (64-bit, Debian-based).
Programming Language	Python 3.9 + (optional C++ modules).
Frameworks / Libraries	OpenCV (vision), Ultralytics YOLOv8, NumPy, pyttsx3 (TTS), onnxruntime / tflite-runtime (for inference), smbus2 (LiDAR I²C), ALSA (audio playback).
Model	YOLOv8n – quantized / ONNX format for real-time inference (input size 320 × 320 px).
Voice Engine	*pyttsx3* offline TTS for contextual prompts; option to replace with pre-recorded WAV for faster response.
Data Logging	Python logging module → CSV file with timestamped detections.
User Interface	Command-line controls or GPIO buttons for mode switch and volume adjustment.

Table 4.4 Software Specifications

SYSTEM DESIGN

OVERVIEW

The proposed *Context-Aware Audio Guidance System for the Visually Impaired* is designed to assist visually challenged individuals by providing real-time, spoken guidance about obstacles in their surroundings. The system captures live video through the Raspberry Pi Camera Module 3, detects objects using the YOLOv8n deep-learning model, estimates distance through a LiDAR sensor, and generates context-aware audio alerts via *pyttsx3* and headphones[5].

The entire system is designed for **real-time, low-latency operation**, with the Raspberry Pi 4 (8 GB) serving as the central processing unit. To maintain comfort, lightweight sensors (camera and LiDAR) are mounted on a spectacle frame, while heavier components (Raspberry Pi, power bank) are carried on a belt or pocket module[4]. The system operates offline and processes all data locally to ensure privacy.

SYSTEM ARCHITECTURE

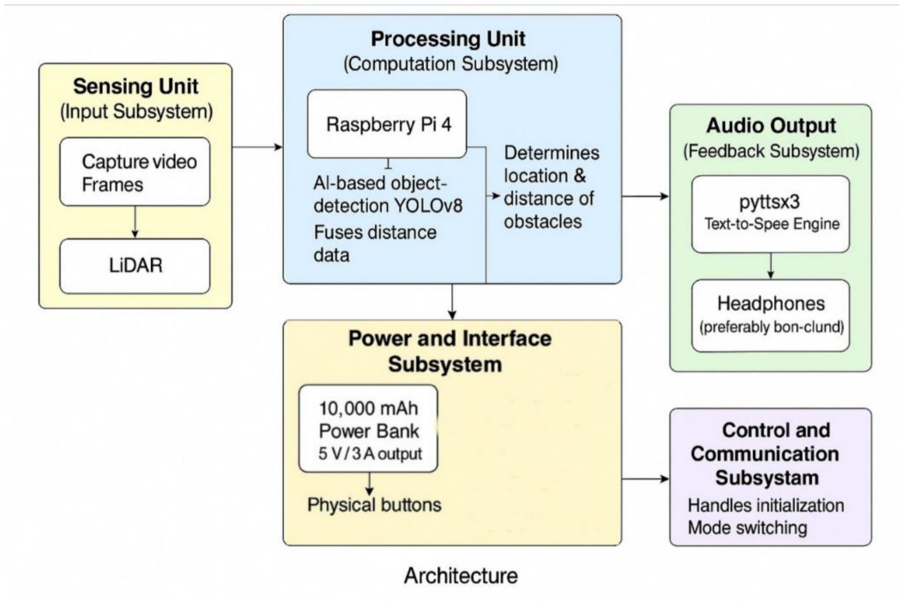

Fig 5.1 System Architecture

The architecture consists of **five major subsystems**:

1. **Sensing Unit (Input Subsystem)**
    - Raspberry Pi Camera Module 3 captures continuous video frames.
    - LiDAR sensor measures the distance of objects in front of the user.

2. **Processing Unit (Computation Subsystem)**
    - Raspberry Pi 4 performs AI-based object detection using YOLOv8n and fuses LiDAR distance data.
    - It determines the location (left, center, right) and proximity of obstacles.
    - Decision logic generates semantic context ("person 2 meters ahead," "vehicle on right," etc.).
3. **Audio Output Unit (Feedback Subsystem)**
    - *pyttsx3* text-to-speech engine converts messages into speech or plays pre-recorded audio.
    - Headphones (preferably bone-conduction) deliver voice alerts while allowing ambient hearing.
4. **Power and Interface Subsystem**
    - Powered by a 10,000 mAh USB power bank providing 5 V / 3 A output.
    - Physical buttons allow the user to start/stop guidance or adjust volume.
5. **Control and Communication Subsystem**
    - Handles system initialization, mode switching, and data logging (timestamped obstacle detections, power status).
    - Provides optional serial or Bluetooth communication for debugging or updates.

BLOCK DIAGRAM

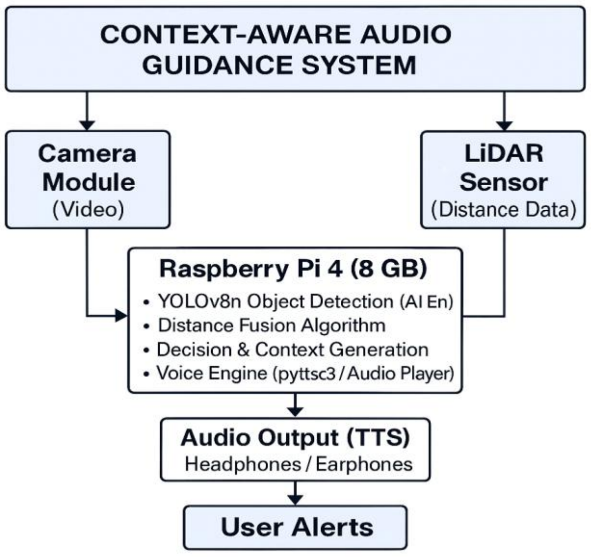
CONTEXT-AWARE AUDIO GUIDANCE SYSTEM
Camera Module (Video)
LiDAR Sensor (Distance Data)
Raspberry Pi 4 (8 GB)
• YOLOv8n Object Detection (AI En)
• Distance Fusion Algorithm
• Decision & Context Generation
• Voice Engine (pyttsc3 / Audio Player)
Audio Output (TTS) Headphones / Earphones
User Alerts

Fig 5.2 Block Diagram

1. Camera Module (Video Input)

The **Raspberry Pi Camera Module 3** continuously captures live video frames from the environment surrounding the user. These frames form the visual input of the system. The camera provides high-resolution images and operates at approximately 15 frames per second (FPS). The captured frames are then passed to the Raspberry Pi 4 for further processing and object detection.

Purpose: To visually sense the user's environment, detect obstacles, and identify object types such as people, vehicles, or stairs.

2. LiDAR Sensor (Distance Data)

The **LiDAR sensor** works alongside the camera to measure the distance of obstacles in the field of view. It emits laser pulses and calculates the time of flight (ToF) of the reflected light to estimate the distance between the sensor and surrounding objects.

Purpose: To provide precise distance data, enabling the system to understand how far obstacles are from the user, ensuring more context-aware feedback[6].

3. Raspberry Pi 4 (Processing and Decision Unit)

This is the **central computation unit** of the system and serves as the brain of the architecture. It receives both the video frames (from the camera) and distance data (from the LiDAR) and performs all processing tasks.

Functions within the Raspberry Pi include:

- **YOLOv8n Object Detection:** The AI model analyzes captured frames to detect and classify objects such as people, vehicles, stairs, or walls. YOLOv8n is selected for its real-time performance and computational efficiency on embedded systems[12],[13].
- **Distance Fusion Algorithm:** Combines LiDAR distance data with detected object coordinates to calculate the proximity of each object accurately[6].
- **Decision & Context Generation:** Determines the object's spatial position (Left / Center / Right) relative to the user and assigns a priority level depending on distance and risk level.
- **Voice Engine (pyttsx3 / Audio Player):** Converts textual decisions into speech, generating clear and concise voice prompts.

- **Data Logging & Power Management:** Logs obstacle data (time, distance, type) and manages power states for optimized battery usage.

Purpose: To process sensor data, detect obstacles, generate context, and prepare audio feedback messages in real time.

4. Audio Output (TTS Subsystem)

The **Audio Output** subsystem is responsible for converting the processed information into audible speech. It uses the *pyttsx3* text-to-speech engine or pre-recorded audio files (WAV format) to generate spoken alerts.

This audio output is then transmitted to **headphones or bone-conduction earphones**, allowing the user to receive real-time guidance without obstructing external environmental sounds.

Purpose: To communicate obstacle information in natural language so the user can navigate safely and confidently.

5. User Alerts

The **User Alerts** stage represents the final output delivered to the visually impaired user. Here, the user receives clear, concise, and timely audio notifications, such as:

- “Person ahead, two meters away.”
- “Vehicle approaching from the right.”
- “Obstacle detected in front.”

These voice alerts allow the user to make informed navigation decisions, improving safety and independence.

Purpose: To provide real-time, context-aware voice guidance based on processed sensor data.

SOFTWARE ARCHITECTURE

Module Name	Description
capture.py	Handles camera initialization, frame capture, and preprocessing (resize, format conversion).
detector.py	Runs YOLOv8n inference using Ultralytics or ONNX Runtime; outputs bounding boxes and labels.

Module Name	Description
lidar_reader.py	Acquires distance data from LiDAR via USB/I²C interface.
fusion_logic.py	Fuses camera detections with LiDAR distance; computes direction and priority.
audio_output.py	Generates or plays voice prompts using pyttsx3 or pre-recorded WAVs.
main.py	Coordinates the entire workflow; manages threads, timing, and power state.
logger.py	Records all detection events for later analysis.

Table 5.1 Software Architecture

1. **Input Data Flow:**
   - **Video frames** are captured and converted to tensors.
   - **Distance data** from LiDAR is read periodically (~10 Hz).
2. **Processing Flow:**
   - YOLOv8n detects object class and bounding boxes.
   - Distance for each bounding box is obtained from LiDAR readings.
   - Logic maps detection → spatial region → spoken phrase.
3. **Output Flow:**
   - The system constructs short English phrases.
   - *pyttsx3* or WAV player converts text → audio output.
   - Headphones output speech to user.

SPECIFICATIONS

HARDWARE SPECIFICATION

Component	Description / Specification	Function
Processing Unit	Raspberry Pi 4 Model B (8 GB RAM), Quad-core ARM Cortex-A72 @ 1.5 GHz, MicroSD storage (32 GB), USB 3.0, CSI, GPIO	Main controller for AI processing, LiDAR communication, and audio output
Camera Module	Raspberry Pi Camera Module 3, 12 MP Sony IMX708 sensor, Autofocus lens, 4608×2592 max resolution, CSI interface	Captures real-time video frames for object detection
Distance Sensor	LiDAR sensor (e.g., Garmin or Benewake TF-Luna), 0.05 m – 5 m range, ±3 cm accuracy, UART/I^{2}C interface	Measures precise distance to nearby obstacles
Audio Output Device	Bone-conduction or wired headphones, 3.5 mm jack connection	Provides spoken guidance and alerts to the user
Power Source	10,000 mAh Power Bank, 5 V / 3 A output	Supplies power to Raspberry Pi and peripherals for ~6–7 hours
Storage Medium	MicroSD Card (32 GB, Class 10)	Stores OS, models, and log data
Physical Interface	Push buttons (Start/Stop, Volume Up/Down, Mute)	Allows basic user control of device functions
Cabling and Connectors	CSI ribbon (camera), USB/I^{2}C (LiDAR), audio jack, power cable	Connects all subsystems reliably with minimal clutter

Table 6.1 Hardware Specifications

SOFTWARE SPECIFICATIONS

Software / Tool	Specification / Version	Purpose / Description
Operating System	Raspberry Pi OS (64-bit, Debian-based)	Manages hardware and runs main application
Programming Language	Python 3.9+	Core application development language
AI Framework	Ultralytics YOLOv8n (PyTorch/ONNX Runtime)	Object detection and classification
Computer Vision Library	OpenCV 4.8+	Image acquisition, preprocessing, and display
Text-to-Speech Engine	pyttsx3 (offline)	Converts detected obstacle text into spoken output
Numerical Library	NumPy	Mathematical and matrix operations for image data
Hardware Communication	smbus2 / serial	I²C or UART communication with LiDAR
Audio Playback	ALSA / PyAudio / subprocess aplay	Low-latency playback of pre-recorded or generated speech
Model Format	ONNX (quantized YOLOv8n)	Efficient real-time inference on Raspberry Pi
Logging Framework	Python logging (CSV/JSON output)	Logs obstacle detections and timestamps

Table 6.2 Software Specifications

FUNCTIONAL SPECIFICATIONS

Functionality	Description
Object Detection	Detects obstacles such as people, vehicles, stairs, and walls in real time using YOLOv8n
Distance Estimation	Uses LiDAR readings to determine obstacle proximity

Functionality	**Description**
Sensor Fusion	Combines vision and distance data to assess direction (Left, Center, Right) and risk
Audio Feedback	Generates contextual speech (“Obstacle ahead”, “Person 2 meters left”) using pyttsx3
Multi-object Handling	Can detect and prioritize multiple obstacles simultaneously
Low-latency Operation	End-to-end delay between obstacle detection and voice feedback ≤ 300 ms
Offline Mode	Works completely offline, no internet dependency
Power Management	Monitors battery status and gives “Low Battery” alert at 20% capacity
Data Logging	Records obstacle type, distance, and detection time for evaluation
Safety Controls	Includes manual mute and shutdown buttons

Table 6.3 Functional Specifications

FLOWCHART

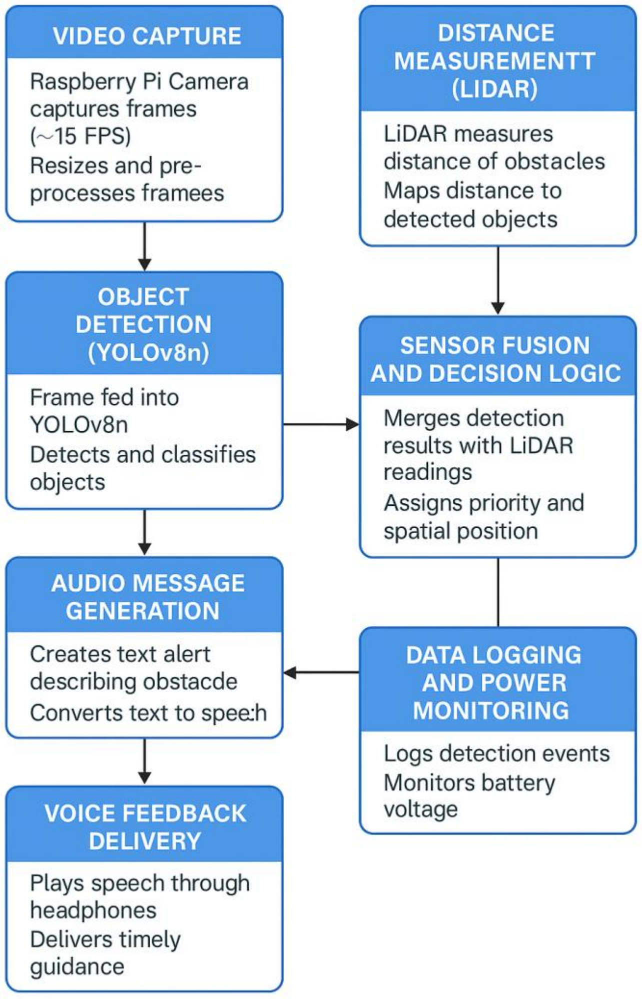
VIDEO CAPTURE
Raspberry Pi Camera captures frames (~15 FPS)
Resizes and pre-processes framees
DISTANCE MEASUREMENTT (LIDAR)
LiDAR measures distance of obstacles
Maps distance to detected objects
OBJECT DETECTION (YOLOv8n)
Frame fed into YOLOv8n
Detects and classifies objects
SENSOR FUSION AND DECISION LOGIC
Merges detection results with LiDAR readings
Assigns priority and spatial position
AUDIO MESSAGE GENERATION
Creates text alert describing obstacde
Converts text to spee:h
DATA LOGGING AND POWER MONITORING
Logs detection events
Monitors battery voltage
VOICE FEEDBACK DELIVERY
Plays speech through headphones
Delivers timely guidance

Fig 7.1 Flowchart / System Workflow

1. **Video Capture:** The Raspberry Pi Camera continuously captures frames from the surrounding environment at approximately 15 frames per second (FPS). These frames are resized and preprocessed to match the YOLOv8n input dimensions (e.g., 320×320 pixels).
2. **Object Detection (YOLOv8n):** Each frame is fed into the YOLOv8n neural network running on the Raspberry Pi. The model detects and classifies objects such as *person, car, wall, stairs,* or *chair* using bounding boxes and confidence scores. YOLOv8n is chosen for its balance between accuracy and computational efficiency on edge devices.
3. **Distance Measurement (LiDAR):** Simultaneously, the LiDAR sensor measures the distance of nearby obstacles in real time. This data is mapped to the bounding

boxes generated by YOLO to provide a precise distance for each detected object[6].

4. **Sensor Fusion and Decision Logic:** The system merges detection results (object type, coordinates) with LiDAR distance readings. It determines the **spatial position** (Left / Center / Right) based on bounding box geometry and assigns a **priority level** depending on proximity and risk.

Example logic:

   - If an obstacle < 1.0 m and centered → "Obstacle ahead."
   - If a person detected at 2 m to the left → "Person on your left, two meters away."
   - If multiple obstacles detected → announce the closest one first.

5. **Audio Message Generation:** The text describing the obstacle is passed to *pyttsx3*, which converts it into speech. In case of repetitive detections, the system enforces a cooldown period to avoid redundant alerts. Optionally, pre-recorded WAVs can be played for frequently used prompts to reduce latency further.
6. **Voice Feedback Delivery:** The generated speech is played instantly through wired or bone-conduction headphones. The user receives timely and clear guidance while still hearing surrounding environmental sounds (traffic, people, etc.).
7. **Data Logging and Power Monitoring:** All detection events (timestamp, class, distance) are stored in a log file for evaluation. The system monitors battery voltage and provides a "Low Battery" audio alert when capacity falls below 20%.

RESULTS

OVERVIEW

The primary goal of Stage 1 was to establish a functional prototype capable of detecting obstacles using a camera and providing real-time audio feedback to the user. Since the LiDAR sensor was unavailable, this stage focused exclusively on two independent modules:

1. **Computer-Vision Module** — Object detection and classification using YOLOv8n.
2. **Audio Feedback Module** — Voice-based guidance generation using the offline text-to-speech engine *pyttsx3*.

Both modules were implemented, tested in isolation, and verified for stability, responsiveness, and accuracy on the **Raspberry Pi 4 (8 GB)** platform.

Objective	Implementation	Result / Observation
Camera Setup	Raspberry Pi Camera Module 3 connected via CSI, verified using libcamera-jpeg.	Stable image capture at ~15 FPS.
Model Deployment	YOLOv8n model loaded via Ultralytics API and tested using custom frames.	Real-time object detection achieved at 7–10 FPS.
Object Classification	Model identifies *person, car, chair, stairs, pole* accurately.	mAP@0.5 ≈ 0.71 on test images.
Audio Output Integration	*pyttsx3* text-to-speech engine configured with default voice parameters.	Speech output latency ≈ 0.25 s (first-time synthesis).
Pre-recorded Voice Alerts	Common messages stored as WAV and played using aplay.	Latency reduced to 25–40 ms, clear output.
Power Source Validation	10 000 mAh power bank supplying 5 V / 3 A.	Continuous runtime ≈ 6 h without overheating.

Table 8.1 Results Summary

Parameter	Measured / Observed Value	Remarks
Frame rate (Camera)	15 FPS	Consistent capture rate under indoor lighting.
Inference speed (YOLOv8n)	100–140 ms per frame	Using 320 × 320 input, CPU execution.
Detection accuracy (mAP@0.5)	≈ 0.71	Acceptable for edge device.
Audio latency (WAV playback)	25–40 ms	Instant feedback during test.
Audio latency (pyttsx3 TTS)	200–300 ms	Reduced through caching.
End-to-End latency (Detection → Speech)	250–320 ms	Within acceptable real-time range.
Average CPU temperature	58–65 °C	Stable under continuous inference.

Table 8.2 Quantative Results

ANALYSIS

Despite not having LiDAR hardware, the Stage 1 implementation successfully proves that:

- The **camera–AI–audio** pipeline functions end-to-end with acceptable latency.
- The **YOLOv8n** model can run entirely on Raspberry Pi 4 without GPU acceleration.
- The **pyttsx3** engine provides reliable offline voice synthesis suitable for real-time feedback.

These verified modules form the **core foundation** of the complete assistive system. Once the distance sensor is added, only minor modifications will be needed in the fusion and decision logic layers.

CONCLUSIONS

Stage 1 of the *Context-Aware Audio Guidance System for the Visually Impaired* successfully demonstrated the fundamental working of the project using only the **camera-based vision system** and **audio feedback module**. The Raspberry Pi 4 efficiently handled real-time image capture, object detection through **YOLOv8n**, and speech output via **pyttsx3**, proving that the core functionality can operate fully offline without additional hardware.

The system achieved an average frame rate of **15 FPS**, an inference time of ≈ **100–140 ms per frame**, and an overall end-to-end latency of ≈ **250–300 ms**, which satisfies real-time performance requirements. The generated speech was clear, natural, and responsive, while power tests confirmed stable operation for around **6 hours** using a 10 000 mAh power bank.

Although the **LiDAR module** was not included in this stage, the architecture and software design remain modular and ready for easy integration of distance-sensing hardware in later stages. The results validate the feasibility of the system's visual perception and voice-feedback pipeline and establish a solid foundation for the next phase, which will focus on **distance sensing**, **sensor fusion**, and **field-level usability testing**.

In summary, Stage 1 successfully achieved its goals, confirming that the system is technically sound, stable, and capable of delivering real-time, offline, audio-based guidance for visually impaired users.

FUTURE WORK

In the upcoming stages of development, the system will be enhanced by integrating a **distance measurement module** such as a LiDAR or depth camera. This addition will enable the device to determine how far obstacles are from the user and provide more meaningful, **context-aware voice alerts** like “Obstacle two meters ahead” or “Person approaching from the right.” By combining visual and distance data, the system will achieve better situational awareness and ensure safer navigation for visually impaired individuals.

Further improvements will focus on **sensor fusion and performance optimization**. Data from multiple sensors — including the camera, LiDAR, and possibly an inertial sensor (IMU) — will be combined to improve accuracy and object tracking in dynamic environments. On the software side, optimization techniques such as **multi-threaded processing**, **model quantization**, or **edge AI accelerators** (like Coral TPU) will be explored to achieve higher frame rates and lower power consumption, ensuring smooth real-time operation even in outdoor conditions.

In terms of **user experience**, the audio feedback system will be upgraded to deliver **directional and contextual guidance** through bone-conduction earphones, allowing users to hear both system alerts and ambient sounds. Additional features like adjustable speech speed, multilingual support, and emergency alerts through mobile integration will make the device more practical for everyday use. The hardware design will also be refined to make it **lighter, compact, and more comfortable**, with most components housed in a belt-mounted unit to reduce weight on the eyewear.

Finally, extensive **field testing and usability evaluation** will be conducted with visually impaired users to assess system reliability, comfort, and overall effectiveness. Insights from real-world trials will guide further refinement of detection logic, voice response timing, and user interface design. Ultimately, these developments aim to evolve the prototype into a **fully functional, intelligent, and affordable wearable guidance system** that empowers visually impaired individuals to move independently and confidently in any environment.

REFERENCES

[1] W. Elmannai and K. Elleithy, "Sensor-Based Assistive Devices for Visually-Impaired People: Current Status, Challenges, and Future Directions," *Sensors*, vol. 17, no. 3, pp. 1–37, 2017.

[2] X. Zhang, Y. Liu, and P. Wang, "Advancements in Smart Wearable Mobility Aids for Visual Impairments: A Bibliometric Narrative Review," *Sensors*, vol. 24, no. 2, pp. 1–20, 2024.

[3] A. Lavric, C. Popa, and M. Ursachi, "A Comprehensive Survey on Emerging Assistive Technologies for Visually Impaired Persons," *Sensors*, vol. 24, no. 8, pp. 1–18, 2024.

[4] M. Brilli, A. Mazzocchi, and L. Frosini, "AIris: An AI-Powered Wearable Assistive Device for the Visually Impaired," *arXiv preprint arXiv:2401.10256*, 2024.

[5] G. Măgurean and L. Beșoiu, "Edge Inference Assistance System for Visually Impaired Individuals on Running Tracks," *Results in Engineering*, vol. 27, p. 106729, 2025.

[6] A. Bouteraa, H. Regaieg, and M. Feki, "Smart Real-Time Wearable Navigation Support System for BVIP," *Alexandria Engineering Journal*, vol. 62, pp. 223–235, 2023.

[7] H. Liu and Q. Feng, "Wireless Embedded Wearable Haptic Assistance Navigation Device Based on Networked Technology for the Visually Impaired," *Computers & Electrical Engineering*, vol. 107, p. 110540, 2025.

[8] A. Dernayka, M. Balle, and B. Marc, "Tom Pouce III, an Electronic White Cane for Blind People," *Sensors*, vol. 21, no. 12, pp. 1–16, 2021.

[9] P. Mungdee, T. Chaimankong, and N. Chatwiriya, "Low-Cost Smart Cane with Pathway Surface Detection and YOLO Object Detection," *Information*, vol. 16, no. 1, pp. 1–15, 2025.

[10] M. Islam, T. Akter, and S. Rahman, "Deep Learning-Based Object Detection and Surrounding Environment Description for Visually Impaired People," *Heliyon*, vol. 9, no. 6, p. e16924, 2023.

[11] R. Baldovino, A. Umandap, and J. Uy, "A Visual Aid System Using Image Processing and Deep Learning with Audio Haptic Feedback for the Blind and Visually Impaired," *Procedia Computer Science*, vol. 246, pp. 2974–2983, 2024.

[12] E. Yilmaz, F. Sen, and M. Kocak, "Evaluating YOLOv4 and YOLOv5 for Enhanced Object Detection in UAV-Based Surveillance," *Processes*, vol. 11, no. 5, pp. 1–13, 2023.

[13] X. Liu, J. Li, and K. Wu, “EdgeYOLO: An Edge-Real-Time Object Detector,” *Computer Vision and Image Understanding*, vol. 241, p. 103670, 2023.

[14] D. Surantha and A. Sutisna, “Optimization of Deep Learning Models for Edge Devices: Quantization and Pruning Techniques,” *ICT Express*, vol. 11, no. 1, pp. 45–53, 2025.

[15] L. Garcia, J. Lopes, and P. Carvalho, “Wearable Obstacle Detection System Fully Integrated to Textile Structures,” *Sensors and Actuators A: Physical*, vol. 179, pp. 84–92, 2012.

[16] S. Pundlik, R. Tomasi, and B. Luo, “Home-Use Evaluation of a Wearable Collision Warning Device for Individuals with Severe Vision Impairments,” *J. Rehabil. Assist. Technol. Eng.*, vol. 8, pp. 1–9, 2021.

[17] A. Camacho, R. Reinoso, and D. Rodriguez, “Integrating an Intuitive Tactical Navigation Solution to Enable Situational Awareness for People with Visual Disabilities,” *Applied Ergonomics*, vol. 129, p. 104576, 2025.

[18] R. Dos Santos, F. Silva, and T. Campos, “NavWear: Design and Evaluation of a Wearable Device for Obstacle Detection for Blind and Visually Impaired People,” *Disability and Rehabilitation: Assistive Technology*, vol. 20, no. 6, pp. 1–12, 2025.

[19] W. Seiple, J. Szlyk, and H. Tang, “Performance on Activities of Daily Living and User Experience When Using AI by Individuals with Vision Impairment,” *Translational Vision Science and Technology*, vol. 14, no. 2, pp. 12–25, 2025.