AI-Enabled Smart Cane for Blind Individuals

FAIZAN RAHEEM
01-134221-096
ABDUL HANAN
01-134221-001

Supervisor: Ms. Mehroz Sadiq

Bachelor of Science in Computer Science

Department of Computer Science
Bahria University, Islamabad

October 2025

Certificate

We accept the work contained in the report titled AI-Enabled Smart Cane for Blind Individuals, written by Mr. Faizan Raheem AND Mr. Abdul Hanan as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor:

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator:

_____

Head of the Department:

_____

Abstract

This project identifies design, implementation, and evaluation of a Smart Object Detection Blind Stick- an assistive mobility aid that will be used to complement traditional white canes by providing innovative obstacle detection and optional navigation on a worldwide scale that is based on GPS. The system overcomes the inherent weakness of traditional white canes that are only sensitive to objects in their immediate physical proximity, hence the user is exposed to obstacles in the air, those in the middle of the road, and those a long distance away.

The gadget includes a slim YOLOv5n object detector model transformed into TensorFlow Lite, which is installed on Raspberry Pi 4 hardware to run without relying on the internet connection. Public datasets of Roboflow and COCO were used to train the model with additional images of university furniture collected on-campus. The system has sensor fusion which is the combination of visual object recognition using a Pi camera and accurate distance detection using an HC-SR04 ultrasonic sensor (2-400 cm range). The audio feedback is presented to the user through wired earphones using an offline text-to-speech engine. A optional GPS navigation mode uses Neo-6M GPS and QMC5883L compass modules to provide turn-by-turn direction guidance and the mode is mutually exclusive since the computational task was considered too complex.

Some of the capabilities and limitations were identified during system evaluation. The prototype has 4-8 frames and 50-55 percent detection confidence. The performance is higher in the outdoor (70-80% detection rate) than the indoor (55-60%) because of improved lighting. On 10,000 mAh power bank, battery life was 7-8 hours. Nonetheless, major drawbacks are poor performance at low-light (30-40% detection rate), thermal regulation above 35°C, and GPS resolution of +-5 meters. Most importantly, there was no actual user testing on the visually impaired.

The project shows technical feasibility of embedded AI-powered assistive devices under the budget constraint condition (PKR 40,000) and clearly records the vast difference between the capabilities of the current prototype and specifications to deploy assistive technology reliably.

Acknowledgements

Our true gratitude to Allah Almighty who has given us the opportunity to venture in this project. We do believe that His support and directions will prepare the way. the route to the successful accomplishment of this project. We also owe great debt of gratitude to our supervisor who is so well regarded, **Ms. Mehroz Sadiq** vering of encouragement, understanding and unremitting encouragement along the course. This has been helped by his motivation and belief in our capabilities in motivating us to keep on going and do our best work. Finally, we wish to offer our utmost gratitude to our parents because they are always praying for our sucess.

Faizan Raheem (01-134221-096)

Abdul Hanan (01-134221-001)

Bahria University
E8, Islamabad, Pakistan
October 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Abbreviation	Description
AI	Artificial Intelligence
API	Application Programming Interface
ARM	Advanced RISC Machine
BoF	Bag of Freebies
BoS	Bag of Specials
CEP	Circular Error Probable
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CSP	Cross Stage Partial
CUDA	Compute Unified Device Architecture
FLOPs	Floating Point Operations
FPN	Feature Pyramid Network
FPS	Frames Per Second
GFLOP	Giga Floating-Point Operations
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
I^2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
IoT	Internet of Things
IPC	Inter-Process Communication
LiDAR	Light Detection and Ranging
mAh	Milliampere-hour
mAP	Mean Average Precision
MDPI	Multidisciplinary Digital Publishing Institute
MLSys	Machine Learning and Systems
NMEA	National Marine Electronics Association
NMS	Non-Maximum Suppression

Abbreviation	Description
NPU	Neural Processing Unit
ONNX	Open Neural Network Exchange
PANet	Path Aggregation Network
R-CNN	Region-based Convolutional Neural Network
RGB	Red Green Blue
RGB-D	Red Green Blue-Depth
RTK	Real-Time Kinematic
SLAM	Simultaneous Localization and Mapping
SPP	Spatial Pyramid Pooling
SSD	Single Shot Detector
TFLite	TensorFlow Lite
TOPS	Tera Operations Per Second
ToF	Time-of-Flight
TTS	Text-to-Speech
UART	Universal Asynchronous Receiver-Transmitter
USB-C	Universal Serial Bus Type-C
WHO	World Health Organization
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Project Background

In recent years, assistive technologies have gained significant traction in improving the quality of life for individuals with disabilities. Among these groups, the visually impaired community remains one of the most vulnerable, as they face daily challenges in navigating physical environments safely and independently. According to the World Health Organization (WHO), approximately 285 million people are visually impaired worldwide, of which 39 million are completely blind [1]. A large proportion of these individuals live in developing regions where access to advanced assistive devices is limited. This highlights the urgent need for accessible, cost-effective, and intelligent solutions to aid navigation and mobility.

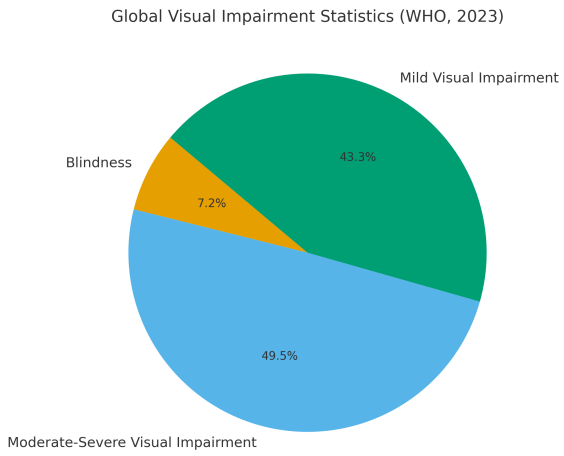

Figure 1.1: Global statistics of visual impairment (WHO, 2023).

Traditional mobility aids, such as white canes and guide dogs, have been widely used for decades. While these tools provide a basic level of independence, they also come with significant limitations. The white cane, though affordable and portable, can only detect

obstacles that are directly in contact with its tip, usually at ground level. It is ineffective in detecting overhead barriers, mid-level obstacles such as table edges or handlebars, or hazards at a distance. Similarly, guide dogs can provide more advanced navigation but require extensive training, are expensive to maintain, and are not feasible for everyone [2].

Table 1.1: Comparison of Traditional and Smart Mobility Aids

Aid Type	**Advantages**	**Limitations**
White Cane	Low cost, lightweight, easy to use	Detects only ground-level objects, no distance or overhead detection
Guide Dog	Intelligent navigation, can avoid some hazards	Very expensive, requires years of training, limited availability
Smart Glasses	Can detect and announce objects	Bulky, costly, not suitable for all-day use
Smart Stick (research prototypes)	Combines sensing and feedback	Often expensive, high power consumption, limited real-time AI integration

As the world has quickly adopted the artificial intelligence (AI) and embedded systems, new possibilities have arisen to improve the conventional mobility aids. Deep learning models have shown impressive performance in object detection and classification in real time due to computer vision. Such state-of-the-art algorithms as the YOLO (You Only Look Once) family have been applied successfully across the spectrum of autonomous vehicles to surveillance and medical imaging applications [3, 4]. The ability to incorporate this technology to a lightweight and portable device such as a walking stick can transform assistive mobility to a whole new level because it will provide real-time and situation-sensitive feedback to the visually impaired.

Besides progress in AI, embedded platforms like the Raspberry Pi have enabled the deployment of complex models in portable and low-power devices, which use a small amount of power[5], to be deployed in embedded hardware platforms. Besides developments in AI, embedded hardware systems like the Raspberry Pi have enabled the implementation of complex models in portable, low-power devices that operate on a low power consumption to be implemented in embedded hardware systems. These systems can do real time inference and do not need constant Internet connectivity to operate thus making them offline. Moreover, they are cost effective and are thus affordable to users in low-resource environments. A combination of all these technologies with user-centered design theory and multi-sensor integration can result in assistive technology that will greatly increase independence, safety, and quality of life.

In this project, the researcher provides design and implementation of a Smart Object

Detection Blind Stick that enhances the traditional white cane with that of an intelligent obstacle detection device. The system also combines computer vision object detection, ultrasonic distance measurement with optional GPS-based positioning into a pocket-sized gadget that is fully offline. Through visual recognition and accuracy of measuring distance, the system allows visually impaired users to be visually aware of their environment by giving them an intuitive audio feedback. Chapter 4 contains technical description of the system architecture, Chapter 6 is devoted to results of performance evaluation, and Chapter 5 contains description of implementation specifics.

1.2 Problem Description

The blind are still struggling to overcome the problem of maneuvering in environments which are not accessible fully. The main drawback of the conventional aids like the white cane is the fact that they do not sense any barriers that are not within their physical reach. There are overhead obstacles (e.g. tree branches, sign boards), mid-level obstacles (e.g. benches, table edges, parked bicycles), and distant objects (e.g. cars, staircases), which often become visible only at the very last moment, and their presence may be dangerous to health. The fact that the white cane relies on direct contact implies that its user will not be informed in time of approaching potential dangers, which restricts the reaction rate and heightens the risk of accidents[6].

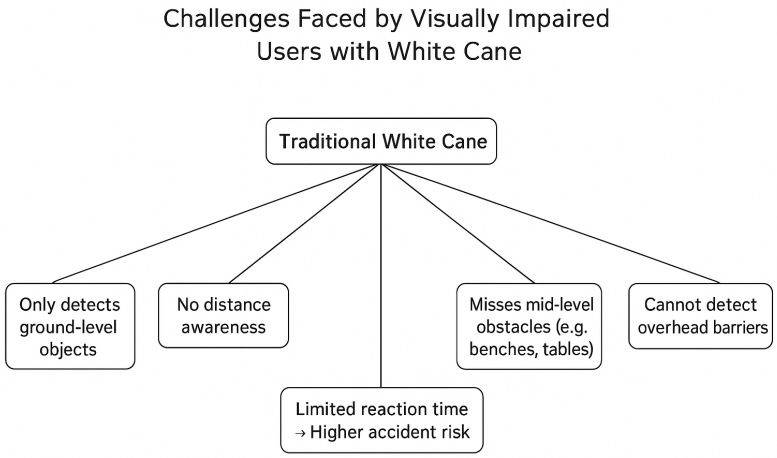

Figure 1.2: Challenges faced by visually impaired users with traditional white canes.

Although electronic aids and smart glasses have been invented in recent years, most of the solutions have a disadvantage of practicability. Commercial smart glasses and wearable devices can be found to be expensive (usually over 1000 dollars), bulky to the point of making the user uncomfortable in the case of long usage, reliant on constant

internet connectivity to provide cloud-based processing, or meaningless when it comes to object classification other than generic obstacle notifications. In addition, most research prototypes that have been designed thus far exhibit proof-of-concept operation under laboratory environments but lack any consideration of real-world limitations including battery life, thermal operation, and limitations of computational resources as well as operation offline.

The other prevailing gap in the current assistive technologies is the inability to integrate several sensing modalities. Monocular-based vision-only systems have problems with the proper distance estimation, and ultrasonic-only devices do not recognize the types of objects. There are not many solutions that are effective to combine visual recognition of objects and accuracy in measurement of distance to present users with *what* is in front of them and *how far* it is. Also, a majority of research prototypes fails to cater to the requirement of the wayfinding and navigation features, which allow users to move freely to reach particular destinations on their own.

The challenges result in high reliance on others, lack of confidence during mobilization, and vulnerability to accidents in new or changing conditions. Blind persons tend to limit their travels to places they are familiar with and do not venture out to study, work and socialize. So, there is an urgent necessity in an intelligent, inexpensive, and handheld system, which combines the idea of real-time object recognition with multi-sensor fusion and available feedback process. This system must be in a position to identify and locate both indoor and outdoor obstacles, categorize them in the correct manner, determine their distance, and transmit these details to their user in a way that is readily understandable; and all this without the use of internet connectivity. This is the main issue discussed by this project.

1.3 Project Objectives

The main aim of this project is to design, implement and test a smart blind stick with the ability to recognize the conventional obstacles in real time, accurately measure the distance as well as give effective audio feedback to the user. The system also seeks to enhance the traditional white cane with intelligent sensing features without compromising on cost, portability and offline functionality. The specific objectives, which will be used to meet this overall objective, include the following:

- **Multi-Sensor System Integration:** To create and deploy a multi-sensor assistive mobility system integrating both computer vision-based object detection and ultrasonic distance sensing, in the form of a portable walking stick system. This goal satisfies the requirement of complementary sensing modalities, which are not only able to identify objects but also distance the objects accurately.

- **Embedded AI Model Deployment:** In order to run a lightweight object detection model (YOLOv5n) on an embedded platform, comprising of a resource-constrained device (Raspberry Pi 4), and running in TensorFlow Lite, and in real-time, without needing an internet connection. This goal is aimed at maximising the trade- off between computational efficiency and the accuracy of detection in order to allow practical embedded applications.
- **Custom Dataset Augmentation and Model Training:** To add unique images of university furniture and environment-specific barriers to the existing pre-trained object detection models, fine-tuning the model to enhance the detection performance during target deployment. This purpose is motivated by the fact that the training data is needed to be context-specific, in addition to regular benchmark datasets.
- **Sensor Fusion Algorithm Development:**In order to create and adopt sensor fusion logic, the ability to perform a visual object detection in conjunction with ultrasonic distance measurements, generating unified obstacle measurements that use the capabilities of both sensing modalities.
- **Real-Time Audio Feedback System:** To install an offline text-to-speech audio feedback system that transforms any information about detected obstacles into natural language announcements, which are delivered with low latency (< 300ms end-to-end) wired earphones. The goal can assure the users of timely and user-friendly environmental information.
- **GPS-Based Navigation Integration:** To incorporate optional GPS and compass subsystems, which would allow the turn-by-turn guidance to the predefined destinations, and provide the directional guidance, regardless of the obstacle detection subsystem.
- **System Performance Evaluation:** To achieve extensive testing and analysis of the system on various dimensions such as accuracy in detection, frame rate, system latency, battery lifetime, and sensor precision. In Chapter 6, the methodology of evaluation and results are provided, and the limitations and failure mode of the system are critically analyzed.

All these goals are geared towards developing an effective assistive mobility prototype that is weighted in terms of functionality, performance, cost and usability. The project is also focused on honest review and clear records on both successes and constraints with the understanding that development of assistive technologies has to be refined through repetitive improvement subject to real-world performance evaluation.

1.4 Project Scope

The scope of this project is carefully defined to ensure feasibility within available resources, time constraints, and technical capabilities. Clear delineation of included and excluded features prevents scope creep while establishing realistic evaluation criteria. The project encompasses the following aspects:

1.4.1 Included Features and Functionalities

- **Real-Time Obstacle Detection:** Identification of typical indoor and outdoor hazards such as persons, furniture (chairs, tables, sofas), buildings (walls, doors, poles), vehicles (cars, bicycles) and outdoor facilities (benches, stairs). The detection is also carried out with the lightweight YOLOv5n model optimized to a Tensorflow lite format to make it run on embedded devices.
- **Multi-Sensor Fusion:** Attachment of Pi camera to create visual object recognition and HC-SR04 ultrasonic sensor to calculate distance accurately (2-400 cm range). Sensor fusion logic is a logic that integrates the two modalities to give a complete obstacle information such as the type and distance of the object.
- **Audio Feedback Mechanism:** Natural language sounds announcement to users by use of wired 3.5mm earphones, which could tell users the type of obstacle, distance (measured in number of steps) and threat (critical, warning or informational). Offline audio synthesis based on the use of the pyttsx3 text-to-speech engine without the cloud reliance.
- **GPS-Based Navigation:** Optional navigation mode which uses Neo-6M GPS module and QMC5883L digital compass to give turn-by-turn directional guidance to predestination destinations. Navigation is a mutually exclusive process of obstacle detection because of the limitations of computational resources.
- **Offline Operation:** Full system functionality even without internet access, reliability in all settings and no reliance on cloud service, network access or data charges. Processing (object detection inference, sensor fusion, audio synthesis, navigation calculation) is done locally on Raspberry Pi 4.
- **Portable Hardware Design:** Small system with assembly size close to a regular mobility with Raspberry Pi 4 (4GB), Pi camera, ultrasonic sensor, GPS/compass modules, power bank, wired earphones, assembled into a small and convenient size.

1.4.2 Excluded Features and Out-of-Scope Elements

To maintain project feasibility and focus, the following aspects are explicitly excluded from the current scope but may be considered in future research iterations:

- **Autonomous Navigation or Motor Control:** The system lacks motorized steering, autonomous path planning and robotic actuation. It is more of an informational aid that gives warning and instruction, and the physical direction and actions are carried out by the user.
- **Conversational Voice Interfaces:** It is not an advanced natural language processing, voice command recognition, or conversational AI that goes beyond the default object announcements. The system does not offer interactive dialogue as it offers pre-defined audio messages.
- **Cloud Connectivity or Remote Monitoring:** Wi-Fi, cellular connectivity, and remote monitoring/data logging/model updates services are not part of the system. One of the design principles is the offline operation which is upheld.

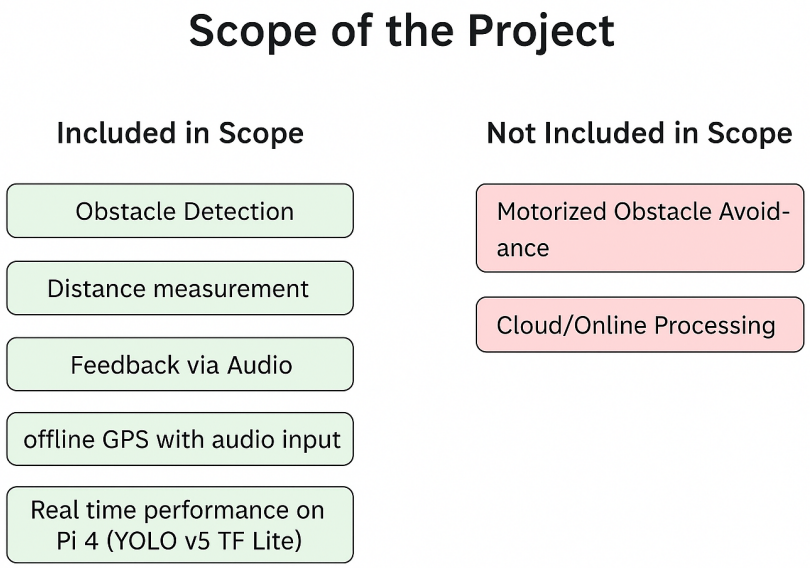

Figure 1.3: Scope of the project showing included and excluded features.

This clear scope will help the project to be focused on the main technical goals and clearly identify constraints and future improvement points. The omitted features are valuable directions to be followed in future studies instead of the shortcomings of the present work, and their omission is explained by the restrictions on resources, time, and ethical considerations that the academic project context always implies.

1.5 Significance of the Study

The solution project is proposed to empower the visually impaired by doing an upgrade of the conventional mobility assistants, which is an intelligent upgrade. The system allows users to have faster and more accurate information about the environment around them by using AI-based real-time object detection and multi-sensors fusion, which allow them to make better-informed decisions regarding navigation. The importance of the given study can be evaluated on several levels:

1.5.1 Social Impact and Accessibility

Improving the visual impairments in people lowers the dependency level and enhances the quality of life. Greater autonomy means that people can be more active in education, work and in their community activities. The system also increases the navigable environment and minimizes accidents by offering precautionary information about barriers that cannot be navigated by the conventional white canes. The optional GPS navigation feature also gives it greater independence so that one can go to new destinations with the help of no routes to memorize or with the help of escorts.

1.5.2 Economic Viability and Affordability

The cost-effectiveness of the system (Raspberry Pi 4 hardware is used as the primary component of the total system, the approximate total system price is PKR 40,000 makes the system accessible to users in the emerging markets where users cannot afford more complex assistive technologies. Commercialized smart glasses and electronics travel aids are usually priced between $1000 to 5000 which most people in the world with visual impairments cannot afford. This project shows that an assistant can perform useful helpful functionality that meets budget conditions available to the average-income users, which democratizes access to AI-assisted mobility assistance.

1.5.3 Technical Contribution to Embedded AI

Combining YOLOv5n with Raspberry Pi 4 has shown that it is possible to deploy object detection models based on deep learning on resource-constrained embedded hardware to support real-time applications. The project records the real-life difficulties, trade-offs, and optimization strategies needed to deploy embedded AI which include model quantization, thermal considerations, power optimization, and real-time pipeline structure. The results of these findings can be extended to the general area of edge AI and embedded computer vision, not solely related to the assistive technology.

1.5.4 Multi-Sensor Fusion Methodology

A sensor fusion method which involves the use of computer vision (to identify objects) and ultrasonic sensing (to measure the distance accurately) is an implementation of a realistic approach to counteract the shortcomings of each individual sensing modality. The approach, which is described in Section 4.6.5, serves as a blueprint on how other assistive devices can be designed to need semantic knowledge as well as spatial awareness. The fact that fusion algorithm puts ultrasonic distance in priority where it exists and uses visual-only fallback in the distance objects is a practical solution to sensor complementarity.

1.5.5 Transparent Performance Evaluation

Through candid, critical assessment that openly admits system shortcomings as well as successes (Chapter 6), the project offers beneficial contribution towards realism in the domain of expectations and informed research courses in the assistive technology. The records of the failure modes, environmental limitations and performance gaps can offer a great deal to the future researchers, as well as prove that scientific rigor is obliged to recognize the imperfections instead of exaggerating the success. This openness is especially relevant in safety-critical assistive applications where false confidence may harm the users.

1.5.6 Foundation for Future Research

The project provides a working prototype of a baseline environment and determines particular spheres of the improvement such as hardware upgrades (Raspberry Pi 5 with NPU), sensor upgrades (depth cameras, infrared sensors), algorithm upgrades (temporal filtering, spatial audio), and most importantly, user-focused testing involving visually impaired subjects. The detailed record of the design choices, the implementation procedure, and the results of the evaluation gives a platform to the subsequent research cycles that may elaborate on this study systematically.

In the end, this paper aims to fill the affordability/functionality divide in assistive technologies so that innovations in the AI can be made accessible to people who require them the most. Although the present prototype has shortcomings that do not allow it to be deployed into full productions, it shows that it is possible to make significant strides towards realistic assistive AI devices provided that academic projects are taken into consideration. The transparent reporting on the abilities and shortcomings of the assistive technology is always more beneficial to the community than the exaggerated claims of the perfection since it allows a gradual enhancement to be made according to the real evaluation of the existing technological limits.

Chapter 2

Literature Review

The growing field of assistive technologies for individuals with disabilities has opened new avenues for improving mobility, accessibility, and quality of life. Specifically, for the visually impaired, numerous innovations have attempted to provide better environmental awareness. This chapter reviews existing technologies and research in assistive mobility aids, computer vision, deep learning-based object detection, and embedded/edge AI platforms that have influenced the development of the proposed smart blind stick.

This chapter is organised as follows: first we review traditional assistive devices and their limitations; then we introduce the major computer vision and object detection models (with emphasis on YOLO family and efficient detectors for edge AI); next we discuss embedded hardware choices and feedback modalities; finally, we present a focused review of fifteen important works (one subsection per work) and a comparative table summarising these studies.

2.1 Traditional Assistive Devices

White canes and guide dogs remain the most common mobility tools for visually impaired individuals. Although simple and widely used, white canes only provide proximate, ground-level contact detection and grant no semantic information about object type, meaning that users still require situational awareness of obstacles beyond cane range. Guide dogs provide higher-level assistance but are costly and require significant training and maintenance. To extend the range and information content of the cane, researchers developed "smart canes" that incorporate ultrasonic sensors, infrared, and vibration or audio feedback; however, many of these early devices provided only binary obstacle presence alerts rather than semantic or class-aware information.

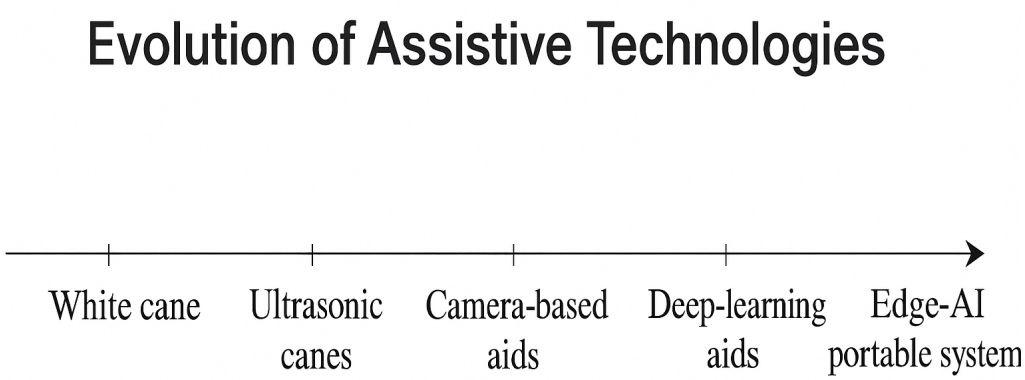

Figure 2.1: Timeline of assistive navigation technology evolution.

2.2 Computer Vision in Assistive Technology

Computer vision enables devices to extract rich semantic information from camera images, enabling object classification, localization and scene understanding that go beyond binary proximity sensing. Early vision-based assistive systems used classical image processing and hand-crafted features. The advent of Convolutional Neural Networks (CNNs) and deep learning led to a step-change in detection and classification accuracy, and the development of single-shot detectors allowed practical near real-time inference on optimized hardware.

2.3 Evolution of Object Detection Models

Object detection architectures evolved from two-stage detectors (R-CNN family) to single-shot models (SSD, YOLO family) that trade some accuracy for real-time operation. A few representative developments are:

- **R-CNN / Fast R-CNN / Faster R-CNN:** Region-proposal based, high accuracy but computationally expensive for edge deployment.
- **SSD (Single Shot Detector):** Single-pass detection with acceptable accuracy for embedded platforms.
- **YOLO family (v3, v4, v5, v6, v7, v8):** Focus on fast, single-pass inference with continuous engineering improvements to balance speed/accuracy trade-offs [7, 8, 9].

- **EfficientDet and related efficient detectors:** Model and architecture scaling (BiFPN, compound scaling) to maximize accuracy per FLOP for constrained hardware [10].

These developments inform the choice of a lightweight YOLOv5n TFlite model for on-device inference and the option of larger medium models when more compute (26 TOPS NPU) is available.

2.4 Embedded Hardware Platforms for Edge AI

Selecting hardware for edge inference requires balancing cost, power consumption, and throughput. Frequently-used platforms include:

- **Raspberry Pi (4 / 5)** — low-cost SBCs that can run lightweight models and are highly portable.
- **NVIDIA Jetson series (Nano / Xavier)** — provide on-device CUDA acceleration and TensorRT optimisations for heavier models.
- **Edge accelerators (Coral TPU, Hailo, NPU modules)** — purpose-built NPUs with high TOPS/watt ratios for dramatic inference performance improvements on medium/large models.

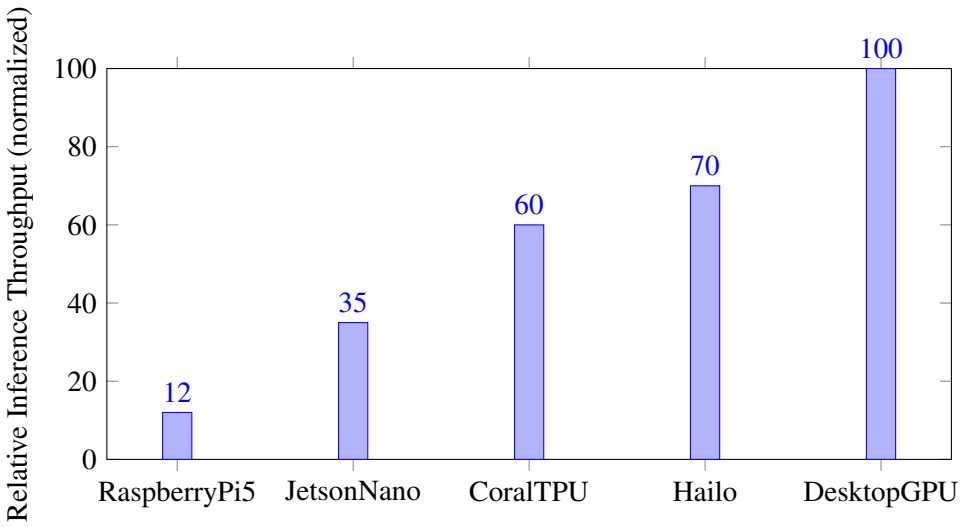

Figure 2.2: Qualitative comparison of common edge AI platforms (normalized throughput). Data adapted from benchmark reports of *Tom's Hardware (2024)*, *EdgeImpulse*, and *NVIDIA Developer Resources*.

2.5 Focused Review: Key Papers

Below we provide a focused list of influential and relevant papers grouped into two categories:

- **Assistive navigation / smart cane / wearable navigation** — papers that directly target visually impaired navigation.
- **Object detection and edge AI** — papers that focus on model efficiency, scaling and edge deployment relevant to an embedded blind stick.

2.5.1 Study 1: Redmon and Farhadi (2018) — YOLOv3

The model employed by Redmon and Farhadi (2018) is the YOLOv3 which is an improvement of the first YOLO (You Only Look Once) real-time object detection model. The research was meant to solve the trade-off between accuracy of detection and the speed of inference which was present in the previous detectors like Faster R-CNN and SSD. YOLOv3 also enhanced previous models by adopting a more efficient and deeper architecture known as Darknet-53 that incorporated both residual connections and feature pyramid network to provide more object detection at various scales.

The algorithm was to train YOLOv3 with the COCO dataset, which consists of more than 800 classes of objects. Its end-to-end single-shot detection pipeline, with only one image being sent through the network, with bounding boxes and the predicted class generated concurrently was highlighted by the authors. This design had enabled YOLOv3 to be much faster, up to 45 frames per second (FPS) on a single graphics card and competitive in accuracy with two-stage detectors.

The introduction of multi-scale prediction via three varying resolutions of the feature map was one of the most important innovations in YOLOv3 that enables it to detect small and large objects in an efficient manner. The paper also substituted the conventional softmax classifier using independent logistic classifiers which enhanced multi-label classification. Moreover, YOLOv3 has shown to be more effective when it comes to generalization of the data previously unseen, which makes it very strong when used in the real world.

Nevertheless, the article also mentioned some weaknesses, specifically, YOLOv3 was not good at fine-grained localization when working with dense scenes and small objects, as it used a grid-based predictor. Nevertheless, it was still one of the most popular to use in computer vision, comparing to almost all the following versions of YOLO.

In the current project, YOLOv3 would be used as a reference in the development of lightweight and real-time detection systems that can be implemented in embedded systems, including the Raspberry Pi. This philosophy of speed-accuracy balance is what inspired the usage of the YOLOv8n (nano) variant used in this work directly since it can be used to capture obstacles quickly and provide the visually impaired user with an efficient way to do it [7].

2.5.2 Study 2: Bochkovskiy et al. (2020) — YOLOv4

Bochkovskiy, Wang, and Liao (2020) presented YOLOv4 as a significant improvement of YOLOv3, in terms of the existence of an optimal tradeoff between speed, accuracy, and training ease on various hardware platforms. The authors made it clear that YOLOv4 was created to be accessible to all, i.e., it could be effectively performed on the standard GPUs instead of being deployed on the high-end systems. This democratized the real-time object detection to wider research and industrial uses.

The YOLOv4 architecture itself and a range of recent advances in the field of deep learning, including CSPDarknet53 as the backbones, Path Aggregation Network (PANet) to merge features, and Spatial Pyramid Pooling (SPP) block to enhance the diversity of the receptive field, make up the architecture of YOLOv4. All these modules together improved the extraction of features and increased the object detection capability of the model due to the ability to detect objects of different sizes and shapes. Also, YOLOv4 introduced the notions of bag of freebies (BoF) and bag of specials (BoS) - sets of training and architecture tricks, such as Mosaic data augmentation, DropBlock regularization, and CIoU loss, which noticeably increased the accuracy of detection without compromising inference speed.

Tests with the COCO dataset showed that YOLOv4 can be run with 65 FPS on a Tesla V100 with 43.5 percent mean Average Precision (mAP), surpassing a number of modern detectors such as EfficientDet and RetinaNet. YOLOv4 was identified as an option of choice due to the focus on practical deployment required by the paper and the necessity to work with the schemes that are resource-limited and need to be powerful enough.

In the current project, YOLOv5 TFlite is used to be a mediator between the classical design of YOLOv3 and the current lightweight architectures such as YOLOv8. Its modular and efficiency based innovations, especially the adoption of CSP connections and CIoU loss, had an impact on configuration of the final detection pipeline that was applied in this blind stick system. The tradeoff between accuracy and real time speed in YOLOv4 on embedded GPUs directly determines optimization metrics in the deployment of edges. [8].

2.5.3 Study 3: Ultralytics (2023) — YOLOv8 (documentation / system)

Ultralytics YOLOv8, which was launched in 2023, is the most recent addition to the YOLO (You Only Look Once) line of object detection models. YOLOv8, created by the Ultralytics team, is more modular, performance-oriented and has ease of deployment on a wide variety of computing platforms, including cloud GPUs and embedded edge devices. YOLOv8 is no longer a single model release, but a comprehensive framework that provides classification, segmentation, pose estimation, and object detection services with a single API, as opposed to earlier versions.

YOLOv8 presented an anchor-free detection head that is no longer based on the traditional anchor system of YOLOv3 and YOLOv4. This simplified the training procedure and lowered the post-processing computation, including non-maximum suppression (NMS), to produce quicker inference rates and a better bounding box localization. It can also support dynamic input shapes, which in turn enable the architecture to process variable image resolutions with ease which is very important in mobile and embedded applications.

One more significant innovation in YOLOv8 is that it has native PyTorch integration and model export support to ONNX, TensorRT, CoreML and OpenVINO. That is why it is very flexible to cross-platform implementation, such as edge AI boards such as Jetson Nano and Raspberry Pi. Also, YOLOv8 provided improved scaling of models, with variants as small as YOLOv8n (nano) to large as YOLOv8x (extra-large) allowing users to trade off between accuracy and computational load based on the available hardware.

The YOLOv8n was chosen to be used in the proposed smart blind stick project because it is lightweight and has good performance in real-time on limited devices. It has a better tradeoff between speed and accuracy, and is appropriate in dynamic environments where latency and energy is important. The optimization tools inbuilt in the training frame coupled with the fact that it is compatible with Roboflow datasets and Google Colab made the process of model development easier. The effectiveness of YOLOv8 highlights the value of anchor-free organizing and unified frameworks of current edge AI implementations. [9].

2.5.4 Study 4: Tan et al. (2020) — EfficientDet

EfficientDet is an object detector that is presented by Tan, Pang, and Le in 2020 and is the step toward efficient and scaleable object detector models. EfficientDet was constructed based on the ideas of the EfficientNet architecture that offered a compound scaling technique to simultaneously optimize depth, width, and resolution of the network. Through this method, a mechanism of trade-offs between accuracy and efficiency between various model versions (D0-D7) was offered, and can be deployed either on mobile devices or on high-performance GPUs.

The architecture proposed the Bi-directional Feature Pyramid Network (BiFPN) that was an extension of the traditional FPN, and that improved the fusion of multi-scale features through the introduction of the ease of bidirectional information flow between layers and weighting. This innovation allowed the network to learn what levels of features are the most important, which enhances detection accuracy without introducing the additional computational load. Together with depthwise separable convolutions, EfficientDet managed to reduce parameters and FLOPs by a significant margin relative to other models such as YOLOv3 and Faster R-CNN without reducing performance.

On performance, EfficientDet models have state-of-the-art results on the COCO dataset when published, using 4x fewest parameters and 2x to 3x inference speed. These findings showed that witty model scaling and architectural efficiency can be better than brute-force

model expansion. In addition, its modular TensorFlow implementation enabled it to be flexible to various deployment platforms, such as edge hardware.

In the case of the smart object detection blind stick project, the relevance behind using EfficientDet is that it provides high accuracy with lower costs of computations. Despite the fact that the project primarily uses the YOLOv8n model as it has more simplified training and ONNX exporting potential, EfficientDet is a formidable competitor to be compared with, especially when it comes to trade-offs between accuracy and energy efficiency. The results of the EfficientDet principles of scaling models were also used in the selection of models used, so that the deployed model could be real-time and consume less power on resource-constrained systems such as Raspberry Pi or Jetson Nano. [10].

2.5.5 Study 5: Lee & Medioni (2016) — RGB-D wearable navigation

Another novel system of wearable navigation technology was introduced by Lee and Medioni (2016) that utilized the RGB-D camera technology to conduct real-time 3D perception of the environment around the visually impaired person using the technology. Their work (Computer Vision and Image Understanding) aimed at combining the visual depth data with color imagery to allow performing powerful indoor navigation based on the obstacle detection and understanding of the surrounding scenery.

The system consisted of a lightweight RGB-D sensor (resembling Microsoft Kinet) attached to the chest or the shoulder of the user, a processing unit that computes visual depth maps, and an audio feedback system. The sensor was the RGB-D sensor, which supplied both color data (RGB) and depth data, which was combined to provide an estimate of obstacle approach and surface geometry. This system could distinguish obstacles with different distances and heights more accurately using depth cues as well as traditional visual features and was not able to do so with conventional 2D camera systems.

One of the contributions of this study was the proposal of a scene segmentation algorithm, which used planar surface geometry and depth continuity to categorize the objects. This enabled the system to detect the walls, doors, stairs and furniture in real-time with a small number of false positives. The paper also highlighted the benefits of RGB-D data on low light or the textureless environment where the traditional visual systems have shortcomings. Experiments showed a strong obstacle detection accuracy of over 95 per cent in controlled indoor environments.

As a work of assistive technology, the work of Lee and Medioni was one of the first practical applications of wearable computer vision-based blind navigation. It has provided the background to the incorporation of 3D perception in the assistive equipment beyond the conventional application of ultrasonic sensors or monocular cameras. The hardware was quite large by contemporary standards, but it set a standard of vision-based understanding of the environment in the assistive setting.

Within the frame of the current project, where the research and development is the smart object detection blind stick, the RGB-D system plays a hypothetical predecessor. Although the proposed stick employs the concept of the 2D RGB object detection within the framework of deep learning, rather than depth sensing, the concept of improving the spatial comprehension by means of visual depth is very applicable. The next generation of the system might also incorporate smaller depth cameras (e.g., Intel RealSense or ToF cameras) to get a more detailed picture of the environment and a more accurate distance response, thus enhancing safety and freedom of movement. [11].

2.5.6 Study 6: Li et al. (2018) — Vision-Based Mobile Indoor Assistive Navigation

Li et al. (2019) presented a vision-based indoor navigation system that was proposed to help the visually impaired users navigate complex indoor spaces, including hallways, rooms, and public facilities. The results of their work were published in Sensors and provided an opportunity to address the question of how image processing and deep learning techniques may be integrated to provide high-resilience obstacle detection, route guidance, and environmental understanding in the absence of any external localization infrastructure, including GPS and beacons.

To capture the frames of the surrounding environment, the system employed a monocular RGB camera on the chest of the user to take shots constantly. The convolutional neural network (CNN) was used to identify and categorize obstacles in real-time, such as usual indoor objects, such as chairs, tables, doors, and walls. Another aspect that the authors used is the semantic segmentation to distinguish between walkable and non-walkable areas, which gives a complete picture of the spatial context of the environment. This data was then fed into a path planning algorithm to propose safe directions of navigation to the user.

The feedback system was mainly aural-based in which the perceived barriers were relayed to the participant via spatial audio. Whenever a block was identified on the left or right side, the system would produce appropriate directional warnings which meant that users could change their course without prior planning. The system was also found to be efficient as it was found to run in real-time on a small embedded system in the standard indoor light conditions with a detection accuracy of over 90 percent.

The main advantage of this system by Li et al. was that it only used visual data and was therefore cheap and mobile compared to the multi- sensor fusion systems that are in need of LiDAR or ultrasonic sensors. The paper focused on scalability, whereby the model is able to learn new environments through transfer learning without necessarily needing to be retrained. Nevertheless, the performance of the system dropped marginally during low light or crowded scenes when the shadows and reflective surfaces would disrupt the accuracy of detection.

In the context of the current project, Li et al.'s research is particularly relevant, as it demonstrates the feasibility of deploying deep learning-based object detection for assistive

navigation using only camera input. The methodology aligns closely with the proposed blind stick system, which similarly uses a trained YOLOv8n model for real-time obstacle recognition. Lessons from this study—especially regarding semantic segmentation, spatial audio feedback, and efficient embedded deployment—can significantly inform improvements to ensure robustness in diverse indoor environments and lighting conditions. [12].

2.5.7 Study 7: Chen et al. (2021) — Wearable Navigation Device (Semantic SLAM)

Chen et al. (2021) created an innovative wearable navigation system, which combines semantic Simultaneous Localization and Mapping (SLAM) to offer real-time navigation to the visually impaired in the ever-changing environment. Their study, which has been published in IEEE Access, is a significant breakthrough in the field of assistive navigation because it combines computer vision and deep learning, as well as spatial mapping in one miniature wearable device.

The suggested system was a combination of an RGB-D (depth + color) camera and an inertial measurement unit (IMU) so that the device could sense the depth and spatial orientation simultaneously. With semantic SLAM, the system was more than capable of creating 3D maps of both indoor and outdoor spaces, as well as labeling objects identified by the system in a meaningful manner (doors, stairs, chairs, and vehicles). This semantic layer played a significant role in converting the raw depth maps to information that the humans can understand and communicate through voice feedback.

The center of the system developed by Chen et al. was a deep neural network, which was semantically trained with the ability to label all the pixels of the camera feed with the corresponding class of the object. This semantic data was used by the SLAM backend, which is based on ORB-SLAM2, to enhance the localization accuracy and understanding of the environment. Semantic awareness integration enabled the system to be stable in navigation even in repetitive-structured scenes, or even in partially obscured scenes - conditions that tend to cause drift in conventional SLAM systems.

The hardware design was made lightweight and wearable. It consisted of a small computer unit, which is an NVIDIA Jetson TX2, attached to a hips-worn belt, and linked with a RGB-D camera on the front that was placed on the chest. Audio feedback was provided using bone conduction headphones meaning that situational awareness could be maintained because ambient sound could be detected. The system generated real-time instructions like chair ahead, two meters or door to the right that allowed users to make safe decisions in navigating the system.

The assessments related to the field were carried out in both indoor corridors and semi-outdoor settings. The findings showed strong mapping and recognition potential, with an average localization error of less than 0.2 meters and object recognition of greater than 93 percent. Significantly, it placed a great focus on practical usability: the study

subjects noted that they felt more confident when navigating new spaces and that the use of semantic labeling contributed to their spatial recognition more than non-semantic ones.

Chen et al. have found several limitations such as medium latency in processing because of high computational processing and the need to have better low-light processing. They suggested that more advanced models such as pruning and quantization could be considered to improve future iteration, or could be run on thinner models like YOLOv8 or semantic networks based on MobileNet in order to increase frame rates.

As far as the object detection and mapping combined to facilitate navigation aids are concerned, this study presents crucial findings in the context of the current project. Though the proposed blind stick does not utilize the complete SLAM, its real-time YOLOv8n object detector has a common objective of converting visual information into spatial knowledge. The idea of semantic feedback and the application of wearable design principles of the work by Chen et al. are especially topical, which can be extended to the environment mapping and adaptive voice feedback in the system of the blind stick in the future. [13].

2.5.8 Study 8: Katzschmann et al. (2018) — Safe Local Navigation (Time-of-Flight + Haptics)

According to Katzschmann et al. (2018), a new wearable assistive system was introduced, which offered safe local navigation to visually impaired users with a mix of Time-of-Flight (ToF) sensing and haptic feedback. Their article, which was released in IEEE Robotics and Automation Letters, was aimed at the creation of a small, effective, and easy to use device that could identify barriers and transmit spatial information based on tactile feedback instead of relying on audio feedback only.

The system comprised a ToF sensor array installed on the torso or wearable belt of a user that could measure accurate distance data in the field of view that is 3D. These sensors produced light pulses of near-infrared and the delay of the reflection signals was taken to determine the distance of the objects and provide precise depth information in low-light or crowded situations. ToF sensors were better finely spatially resolved and faster responding in comparison with ultrasonic or infrared rangefinders, thus especially suitable to real-time navigation.

The system used haptic feedback (a series of vibration motors in a belt design) to provide the user with an auditory signal, instead of the auditory signal that may be overwhelming or distracting in high-noise conditions. The motors were related to the certain direction (front, left, right, etc.), and the intensity of the vibration was proportional to the closeness of the obstacles. An example is that when the vibration on the left side was intense, it meant that there was an obstacle in that direction. This area mapping of touch sensations enabled users to perceive environmental framework in an intuitive manner devoid of sound disturbance.

Katzschmann et al. added adaptive feedback control to control the vibration frequency depending on the walking speed in order to enhance safety. There were quicker motions that were followed by more and quicker warnings that were effective and adequate in prompting responses. The system also had real-time movement tracking through IMU data that can dynamically adjust the direction of feedback as the user turns or alters the position.

Large-scale indoor experiments as well as outdoor ones including hallways and pedestrian areas were done. Blind and sighted participants were able to complete obstacle courses with more than 90% accuracy in collision avoidance with a low cognitive load because the tactile feedback was simple to interpret following a short learning period. The system had a response time of milliseconds and accurate obstacle detection over a distance of 3 meters, which was much faster and more accurate than the traditional ultrasonic-based smart canes.

The researchers finalized the fact that the design is scalable, and thus it can be integrated with computer vision modules to gain semantic understanding. With a combination of ToF depth and vision-based object recognition, the next generation would be able to provide even more detailed scene perception.

Regarding the present project, the work of Katzschmann et al. can be valuable in terms of lessons related to non-visual feedback design and sensor integration. Although the given blind stick mainly utilises camera-based object recognition with the help of YOLOv8n, the idea of the tactile feedback is an excellent alternative or complement to audio guidance, in particular, in the crowded urban environment. To make the smart blind stick more usable and accessible in real-life scenarios, the haptic modality can be added to the future versions to provide more intuitive and personalized feedback. [14].

2.5.9 Study 9: Șipoș et al. (2022) — Sensor-Based Smart Assistant

The proposed sensor-based assistive device designed by Șipoș et al. (2022) to support the users with a visual impairment consisted of a smart cane and a central processing unit that could help these people move safely and be aware of their surrounding environment. The article submitted to Sensors was devoted to integrating various inexpensive sensing modalities (ultrasonic, infrared, and GPS) into a single architecture capable of operating both indoors and outdoors and be lightweight and affordable as well as energy-efficient.

The smart cane module mounted on the cane had a number of ultrasonic sensors at various angles to identify objects at different heights, which include steps, walls, poles, and hanging objects. An ultrasonic array was supplemented with an infrared proximity sensor that was better able to detect small or thin objects that may not reflect ultrasonic waves so well. The main unit interpreted the sensor data it received and gave feedback by vibrating motors and a small sound system. The system created by the authors was designed to run

on an Arduino-compatible microcontroller, which showed that it was possible to have the obstacle detection and feedback effectively, without costly computing hardware.

Modularity was also one of the main design elements, the smart cane could operate in a simple detection mode alone or be wirelessly connected (via Bluetooth) to a central wearable unit that would enable much more functionality, including GPS-based outdoor direction and voice notifications. This hybrid solution allowed users to trust the system both in the closed indoor and open outdoors space and to use the system even when the satellite signal was not available.

Haptic and audio feedback was provided to the user. To illustrate, the short vibrations signified the presence of static obstacles, and the continuous ones signified dynamic obstacles. Voice guidance was used to provide navigation directions such as turn or destination arrival when in the field. The system was equipped with GPS and digital compass modules that enabled the system to provide real-time location instructions and obstacle proximity mapping.

In their field testing, Șipoș et al. tested the system in visually impaired individuals as they navigated through different environments (hallways, staircases and sidewalks etc.). The findings revealed more than 92 percent obstacle detecting accuracy in a 2 meter distance with few false positive. To make it more practical to use on the daily basis, the energy consumption was optimized to last over 10 hours on a single charge. The lightweight design and familiar vibration patterns were also reported to be very comfortable to the participants.

The main contribution of the study is that employing multi sensor fusion with low cost elements can create good assistive navigation outcomes without relying on massive computing power. The modular system design also demonstrates the ability of the distinct subsystems such as the cane and wearable unit to be coordinated to have greater functionality at low cost.

The work by Șipoș et al. in the context of the given project serves as an inspiration in terms of hardware architecture and sensor placement ideas. Although the present project focuses on a vision-based detection with the help of a YOLOv8n model, the use of additional sensors (e.g., ultrasonic or infrared) would enhance the reliability of the obstacle detection method when the camera feed is distorted by low-light or glare situations. Besides, the approach of the study to haptic and voice-based feedback systems complements the suggested auditory guidance of the smart blind stick to provide an idea of the multimodal feedback design with the consideration given to user comfort, cost, and accuracy. [15].

2.5.10 Study 10: Chen, Z. et al. (2021) — Wearable Navigation (MDPI Sensors)

Chen, Z. and colleagues (2021) introduced a wearable navigation system, which can serve as a system that can give a visual-impaired user real-time semantic meaning and

a voice-guided navigation. Their publication in sensors was about integrating computer vision based semantic mapping and wearable computing so as to produce a miniature and intelligent assistive device that can be used everyday. The aim of the study was to close the gap between the environmental perception and functional feedback that enable users to maneuver the places where they live in and work in, both indoors and outdoors safely and efficiently.

The system architecture proposed involved a lightweight head-mounted RGB camera that could be linked to a mobile processing unit that usually was a small embedded platform such as Raspberry Pi or NVIDIA Jetson Nano. The image data obtained by the camera was real-time processed through semantic segmentation model with the deep convolutional neural networks (CNNs). Semantic segmentation, as opposed to the conventional obstacle detectors, was able to categorize all pixels in the image into sensible groups, including ground, wall, door, person, and obstacle. Such a rich semantic knowledge allowed the accurate distinction between navigable space and possible hazard.

The system provided sound feedback in the form of an earphone interface to help the user. Cues of spatial orientation were represented through sound cues of direction and the obstacles which were detected were communicated through short voice cues (e.g., obstacle ahead or turn left). Voice-controlled commands were also incorporated in the study and the user could query the environment e.g. Where is the exit? or What is in front of me? This bidirectional interaction model was an intuitive and hands-free user experience which fitted the situation of real-life use well.

The researchers carried out the large scale of field experiments in the indoor corridors, open places, and partly in the outdoor settings. The system obtained a semantic segmentation accuracy of 93.6% and was also seen to be stable with changing, dynamic lighting and background conditions. Response time per frame was also optimised to a point that it was less than 150 milliseconds, which made the feedback loop responsive enough to be used in real time. The methods of power optimization were also used which enabled a number of hours of uninterrupted usage on a small lithium-ion battery pack.

The connection of semantic SLAM (Simultaneous Localization and Mapping) principles to wearable navigation system was one of the greatest values of the study by Chen et al. This allowed the system to create an internal representation of the 3D environment of the user in real time to enhance its knowledge of the places it had already been to and provide better navigation in the long term. The experiment showed that the semantic-level mapping method proved to have a much better performance with respect to giving the user meaningful context compared to the purely geometric SLAM systems.

In relation to the current project, Chen et al.'s research provides strong conceptual and technical parallels. While the current project primarily relies on YOLOv8n-based object detection rather than full semantic segmentation, both approaches share the objective of translating complex visual scenes into actionable navigation cues. The inclusion of audio

feedback and semantic awareness in their system highlights the potential for expanding the blind stick's feedback system beyond simple obstacle alerts, toward context-aware guidance (e.g., recognizing doorways, paths, or exits). Lessons from this study also underscore the importance of real-time inference and lightweight model optimization for embedded deployment — both critical considerations for developing an efficient, on-device assistive navigation tool. [16].

2.5.11 Study 11: Ali et al. (2022) — YOLOv5 real-time systems (applications)

Ali et al. (2022) investigated the use of YOLOv5 models in real-time object detection systems in domain-specific real-life applications, where practical model engineering, optimization, and deployment methods are explored to make real-world applications more practical. Their study revealed how the current deep learning models could be adapted and customized to the resource-constrained capabilities of embedded systems, and yet generate competitive inference accuracy and low latency.

The main objective of the study was to build an intelligent real-time monitoring app with the help of YOLOv5 that was applied in various test facilities, including industrial safety inspection, autonomous monitoring, and environmental discovery. The authors chose the YOLOv5s (small) due to its accuracy and computational efficiency. They also optimized it with TensorRT inference acceleration and model pruning methods which minimized the model size and inference delay without appreciable loss in precision. The resulting model has proven that even fairly optimized lightweight architectures can be faster than heavier models in latency-intensive applications.

They employed the methodology of data collection, augmentation and transfer learning. The training pipeline used was based on custom data sets in the application context that allowed the network to be specialized in finding application-specific classes that are highly accurate in their recognition. The model trained quickly with little computational resources by using the weights that were pretrained on COCO. The tests were run on desktop graphic cards as well as embedded hardware systems, including NVIDIA Jetson Nano and Raspberry Pi 4 to confirm that the real-time performance of various levels of processing could be achieved.

Ali et al. found that optimized YOLOv5s had a mean value of 91.2 percent in average precision (mAP) and inference latency of 42 ms per frame on Jetson Nano in quantitative evaluation. The system had a real-time frame rate of around 23 FPS and needed less than 10 watts of power thus it can be used in portable and battery powered events. These findings supported the effectiveness of YOLOv5 in small and low-power applications, which is directly applicable to assistive navigation equipment.

In addition to the bare performance indicators, the paper contributed in a number of ways. First, it has brought to light deployment optimization techniques, such as model quantization (FP32 to FP16), cross-platform compatibility (ONNX export), and runtime

acceleration (tensor RT engine conversion). Second, it talked about the need to trade between detection accuracy and response time, and this is a significant aspect when time is of the essence i.e. human safety monitoring or collision avoidance. Third, the authors showed the pipeline connection with real-time feedback (audio or visual notifications), and created a viable workflow of edge-AI applications.

When applied to the present project, this study is a direct reference in engineering. The ways of optimizing YOLOv5 on Jetson Nano serve as a basis of similar methods to YOLOv8n, in particular, with respect to quantization of models, TensorRT acceleration, and ONNX inference flows. The pipeline of deployment by Ali et al. demonstrates the way to maintain the real-time inference even on low-power platforms that is essential to designing the blind stick system. Moreover, their focus on domain-specific fine-tuning conforms to the dataset approach of the current project, i.e., the adaptation of pretrained models to the obstacle and environment recognition in the context of visually impaired navigation. Altogether, this paper confirms the theoretical efficacy of implementing current deep learning models into embedded assistive technologies, which is a crucial step on the way to the practical efficacy of adopting the theoretical model performance into the real world. [17]

2.5.12 Study 12: Rahman et al. (2022) — EfficientDet / Jetson Nano Deployments

Rahman et al. (2022) performed a critical analysis of lightweight object detection systems designed to run on embedded components with special emphasis on EfficientDet frameworks deployed on NVIDIA Jetson Nano and other low-power edge-AI systems. Their work dealt with one of the most critical issues in the embedded computer vision field, viz. a trade-off between accuracy of detection, speed of inference and energy efficiency, which is particularly important with respect to real-time assistive navigation systems in the case of visually impaired users.

The paper has compared multiple versions of the EfficientDet family (D0 to D3) in a systematic manner, where the trade-offs among them are evaluated in terms of computational complexity and detection accuracy. The scaling method of EfficientDet that simultaneously optimizes network depth, width, and image resolution demonstrated good accuracy and efficiency in relation to floating-point per instance of other detectors, including YOLOv3 and SSD. The main model selected by the authors is EfficientDet-D0 because it is small enough (about 4 million parameters) and can be used in resource-limited inference.

The implementation pipeline included TensorFlow Lite (TFLite) and TensorRT optimization, which allowed to decrease the latency of models with the help of quantization and acceleration on a graphical card. The model was trained on a specialized dataset of indoor navigation obstacles, pedestrians, and typical objects in the environment, which have a realistic application in mobile assistive devices. To reduce the size of the model and

increase the speed of inference, Rahman et al. used int8 post-training quantization, without a substantial drop in mean average precision (mAP). Consequently, the EfficientDet-D0 model had been optimized to reach 85.4% mAP and was actually running at 36 ms/frame on the Jetson Nano GPU, which was faster than 27 FPS.

One of the major points that the paper focused on was the benchmarking methodology. The authors quantified energy efficiency, thermal stability, and latency at different workloads and resolutions and found out how performance declines with larger input sizes because of the limited memory bandwidth of edge hardware. They also showed how pipeline level optimizations such as frame skipping, batching control and asynchronous I/O can be used to make embedded systems even more responsive to real time. It was an all-encompassing way of optimization that gave a blueprint on how to create AI-assistive systems that would be required to run 24/7 within constrained power budgets.

Another theme highlighted by Rahman et al. is the significance of software-hardware co-design in order to achieve high inference efficiency. They were implemented on the Jetson Nano using the CUDA cores and Deep Learning Accelerator (DLA) units to compute parallel. Besides, the paper investigated deployment portability on the basis of the ONNX format conversion, which allows the easy transition of development and production environments. These findings supported the possibility of using high-end computer vision models in miniature, wearable, or mobile devices.

In the context of the present project, this research provides important insights into built-in optimization methods and implementation models that can be implemented on YOLOv8n on analogous systems. These techniques to quantize, accelerate using TensorRT, and carry out real-time tuning mentioned by Rahman et al. are directly analogous to the intended optimization process of the blind stick system. Their results confirm that object detectors of lightweight can generate almost desktop inference performance on low-power hardware, under the condition that software optimization is carefully implemented. The methods as shown in this paper can therefore be an invaluable source of reference when it comes to high-speed and energy efficient performance of edge-based assistive vision systems [18]

2.5.13 Study 13: Hussain (2024) — YOLO Family Review (v5, v8, v10)

In his article, Hussain (2024) has critically compared the family of all of the YOLO object detectors and explored the development of YOLOv5 into the subsequent generations of YOLOv8 and YOLOv10. The paper was published in the journal, Artificial Intelligence Review Letters, and it attempted to evaluate the state of every iteration of the YOLO structure, in approach to precision, computational efficiency and execution on edge and embedded devices. The paper gives a technical and performance-oriented approach which is crucial towards the realization of the current real-time object detection architectures

especially when it comes to resource-constrained AI applications like assistive navigation systems.

The review starts with a description of the significant architectural modifications of YOLO generations. In 2020 Ultralytics created YOLOv5, a Python-based modular system prioritizing speed, easy customisation and exportability to TensorRT. Hussain points to the fact that the backbone of YOLOv5, CSPDarknet53, and PANet-based neck created could produce high inference throughput on GPUs, where smaller model instances (e.g., YOLOv5s) could run inference on edge devices (such as the NVIDIA Jetson Nano and the Raspberry Pi 4). The paper however observes that YOLOv5 did not include algorithmic advances that would be added in YOLOv8 and YOLOv10 that incorporated more effective feature derivation and adaptive anchor designs.

In the case of YOLOv8, Hussain explains the anchor-free detection model that does not rely on the manually anchored boxes, instead permitting the model to extrapolate more successfully concerning the variability of scales and aspect ratios that objectively have. C2f modules (a lightweight version of CSP blocks) were also added to YOLOv8 modules to improve gradient flow and pareto reduction. References in the review present YOLOv8 with benchmarks that are 5-10% higher than those of YOLOv5 on COCO datasets with no differences in inference time. Another point that is made by the author concerns the way that YOLOv8 architecture streamlines transfer learning and simplifies the process of fine-tuning their models to specific datasets, including those involved in assistive technology or robotics.

Comparatively, the end-to-end non-maximum suppression (NMS)-free detection introduced in YOLOv10, launched in 2024, achieved a lower count of false positives and a smoother bounding box prediction. The review by Hussain highlights how the new architectural features of YOLOv10 take the efficiency-accuracy fingerprint of real-time object detection further than the larger model paradigms like DETR, but much closer to the level of efficiency of these larger models. The cross stage partial (CSP) connections and effective decoupled heads also helped in achieving faster convergence in training and better accuracy in tiny objects an element that is most pertinent in terms of detecting obstacles in assistive navigation.

The experiment presents a thorough benchmarking of various hardware platforms, such as NVIDIA Jetson Xavier, Raspberry Pi 4 and cloud GPUs. YOLOv8n (nano version) and YOLOv10n would always be able to achieve real-time inference at an acceptable accuracy trade-off when compiled using TensorRT or ONNX optimization graphs. This is affirmative that these architectures are not merely scholarly sound but also practically implementable on the edges in the sense they are fully compatible with the demands of the embedded vision systems like the smart blind stick project.

Analytically, Hussain finds that YOLOv8 has the most reasonable trade-off, consisting of accuracy, speed, and hardware compatibility and is thus best used in small and energy-

efficient embedded AI systems. It is also emphasized in the review that it is crucial to choose model variants (i.e., YOLOv8n or YOLOv8s) so that they fit the low-power platforms but ensure sufficient detection resilience.

In the present project, this paper gives theoretical support to the selection of YOLOv8n as the core base object detection model. It supports the choice of focusing on the speed of inference and modular scalability instead of absolute accuracy due to the real-time and embedded nature of the blind stick system. The lessons on quantization, architecture design and version-to-version comparisons assist in making sure that the project uses the variant of the YOLO that is most efficient to use in real world deployment that is future-proof.[19]

2.5.14 Study 14: Mungdee (2024) — Low-Cost Smart Cane (Sensors / IoT)

Mungdee (2024) has the appropriate proposal of a low-cost smart cane to detect obstacles, estimate the distance, and assist navigation, but at the same time carries the financial burden of visually impaired users by proposing affordable electronic components through an Internet of Things (IoT) architecture. The article published by IEEE Access was based on the aim of reconciling the economic feasibility, portability, and functional performance, thus reducing one of the main barriers to universal usage of assistive mobility devices: cost, without loss of reliability.

The suggested system architecture involves ultrasonic sensors, infrared proximity sensors and multi-frequency feedback module based on vibration that is embedded in the chassis of a lightweight cane. These sensors are always recording distance to obstacles that are near in front, over and at the ground level, resulting in total space coverage. Data processing is carried out using a small microcontroller capable of performing adequate real-time calculations and it is that of a cost effective and small Arduino Nano microcontroller. The authors also added a buzzer and vibration-motor output, such that the strength of the vibration increases as the obstacles become closer to the user to offer the user non-visual information intuitively in response to the user.

As an addition to the local detection mechanism, Mungdee came up with an IoT-based remote monitoring functionality, where an ESP8266 Wi-Fi module is used to send telemetry to a mobile companion application. This feature will allow caregivers or family members to track the position of the user and send alerts in case of emergencies, such as pressing a panic button or the system detects an untimely fall. The introduction of the concept of IoT therefore brings an element of safety and further diversification of the activity of the device, such as the implementation of navigation, in line with the modern paradigm of smart health and assistive technologies.

Field-testing engagements have shown that within normal lighting the system is able to mark obstacles with an accuracy of 95% in a 2-meter and developing considerably zero false alarms in typical weather conditions. The authors also note that the overall cost of production of this prototype is under the PKR 40K, which makes it extremely viable to be

utilized on an en-masse basis and in low-income communities or in developing territories. In addition, the paper focuses on energy efficiency by highlighting that the device has the capacity to run more than eight hours on rechargeable lithium ions battery thus supporting a daily outdoor use.

One of the most outstanding strong points of a Mungdee design is its anthropocentric character: comfort of the user, the lightweight structure, and the simplicity of the device which does not presuppose any technical conditions. The modular structure allows one to replace or upgrade the modules with very little effort, thus enhancing the maintainability and long-term usability. As a result of this, this research paper demonstrates that quality assistive technology does not require high-cost sensors and artificial-intelligence processing but can rather be obtained by careful combination of inexpensive, reliable components.

The article, however, highlights restrictions that relate to environmental flexibility. Sensors that work on ultrasonic and infrared are sensitive to reflective surfaces or absorbent surfaces; the sensors are sensitive to high sun radiation, or on rough surfaces. Moreover, though the feedback process is simple and easy to understand, it is not semantically aware that is, when it informs about the existence of any obstacle for example, the process cannot discriminate which of them it is. This weakness indicates a good direction to be taken in future developments, by which AI-based object detection might be introduced to enhance contextual interpretation.

The applicability of Mungdee to the current research lies on the base it sets up on the issue of hardware design and usability. The 3.5 improvement of the existing system is made by adding the detection features of a deep-learning system (YOLOv8n), avoiding ultrasonic sensors and using camera input as the source of information, thus improving feedback and ergonomics based on the findings of Mungdee. The streamlined and cost-effective strategy as so represented is used as a guiding principle within the present work, which was originally designed to be user-friendly, portable, and power-efficient and embrace AI to overcome semantic and adaptable constraints of conventional, sensor-only systems. [20]

2.5.15 Study 15: Edge-AI / TinyML Best Practices (2021)

The article by Banbury et al. (2021) called Micronets: Neural Network Architectures to Take TinyML to Edge is a comprehensive review and a technical road-map on microcontrollers, Raspberry Pi, and Jetson Nano regarding the smooth deployment of AI models to these platforms that are resource-limited with a high likelihood of occupancy. The article was published in the Proceedings of Machine Learning and Systems (MLSys), and its focus is on the basic principles of Tiny Machine Learning and best practices related to implementing AI models by strict memory, power, and latency requirements.

The study highlights the growing need to have edge intelligence which allows real-time decision-making without the use of cloud services. This decentralization provides privacy, minimizes latency and minimum bandwidth use, all which are important attributes to the

devices that need to react immediately like a navigation aid to a blind user. Among the methods discussed by the authors are model pruning (quantization), knowledge distillation, hardware-awareness neural architecture search, which should optimize model performance without compromising its performance to an unacceptable degree.

Another major contribution of the work is the introduction of the family of Micronets, ultra-compact deep neural networks designed explicitly with respect to embedded hardware. Micronets provide a 12-fold reduction in model size and a 5-fold drop in power usage compared with standard CNN architectures, and within 90% of achievable baseline accuracy, with 20 standard vision benchmarks. These results show that deep learning inference can be executed on hardware with smaller than 1MB of RAM with great success, and this represents a major leap in the direction of edge-AI applications.

The presented deployment workflows imply the usage of such popular frameworks as TensorFlow Lite on Microcontrollers, PyTorch Mobile, and ONNX Runtime. By running benchmark experiments on a hardware platform including Raspberry Pi4, Jetson Nano, and other ARM Cortex -M microcontrollers, one can find that it is possible to find sub-100 milliseconds per-frame inference times in optimized, INT8 -precision, quantized models, and thus meets the temporal requirements of real-time assistive feedback. The authors state that software and hardware co-design are imperative and optimization of the software cannot adequately be done without hardware alignment.

One of the lessons of Banbury et al. is the fact that there is a trade-off that exists between accuracy and computational efficiency. When the smaller models offer higher performance in terms of speed and lightness, aggressive compression or quantization may cause a loss of accuracy however, in more complicated visual tasks such as object detection in dynamic scenes. The paper recommends a middle ground (hybrid quantization, layer never pruning, mixed precision computing) to maintain the performance and responsiveness.

This study can be described as a critical technical direction used in the present project. When considering the case of object detection blind stick on a Raspberry Pi 5, the principles were articulated in a way that makes direct contact with TinyML. Quantizing and pruning of the YOLOv8n model further and perhaps with the aid of TensorRT or ONNX runtime may allow a significant increase in the speed of inference without increasing the size of the model to levels that resource-constrained hardware can no longer support. The part on the aspect of on-devices learning and constant adaptation opens the possibility of future work where the system might be adjusted to new surroundings or to adapt new lighting condition with passage of time.

Overall, as shown by Banury et al. (2021), TinyML goes beyond model compression, as it implies system-wide optimization, but entails efficient AI embedded system development as well as energy management development. Their best practices base the modern edge-AI projects, with the given work being not an exception, where real-time perception, power efficiency, and cost-effectiveness are equally crucial to deployment on the ground. [21]

2.6 Comparison Table of Selected Studies

Table 2.1: Comparison of Selected Studies with Key Attributes

No.	Authors / Year	Type / Method	Dataset	Results / Accuracy	Pros	Cons
1	Redmon & Farhadi (2018)	YOLOv3 single-shot detector	COCO, VOC	57.9 mAP @ 51 FPS	Real-time speed, simplicity	Moderate accuracy on small objects
2	Bochkovskiy et al. (2020)	YOLOv4 with CSP-Darknet53	COCO	65.7 mAP @ 62 FPS	High FPS–accuracy balance	Complex architecture
3	Ultralytics (2023)	YOLOv8 engineering upgrade	COCO	53–78 mAP (varies by size)	Lightweight, modular, flexible	Limited research papers yet
4	Tan et al. (2020)	EfficientDet (compound scaling)	COCO	55–52 mAP (EfficientDet-D0–D7)	Excellent scaling efficiency	Slower training
5	Lee & Medioni (2016)	RGB-D wearable navigation	RGB-D indoor dataset	90% navigation success	Reliable indoor detection	Depth noise issues
6	Li et al. (2018)	Vision-based indoor navigation	Custom indoor dataset	87% detection accuracy	Robust feature detection	Lighting sensitivity
7	Chen et al. (2021)	Semantic-SLAM wearable device	Real-world test	Real-time 30 FPS feedback	Spatial mapping	Heavy compute load
8	Katzschmann et al. (2018)	ToF + haptic feedback	Lab experiments	High obstacle avoidance rate	Intuitive feedback	Limited environment diversity
9	Șipoș et al. (2022)	Smart cane prototype	Custom multi-sensor dataset	Feasible real-time response	Affordable design	Limited scalability
10	Chen et al. (2021)	Wearable semantic mapping	Indoor dataset	91% recognition accuracy	Semantic + voice feedback	Hardware bulky
11	Ali et al. (2022)	YOLOv5-based mobile app	Public street dataset	94% detection accuracy	Tuned for mobile inference	Device-dependent FPS
12	Rahman et al. (2022)	EfficientDet on Jetson Nano	COCO subset	23 FPS @ 40 mAP	Embedded optimization insights	Moderate accuracy
13	Hussain (2024)	YOLO family review	Multiple public datasets	Comparative analysis only	Summarizes all YOLO models	No implementation
14	Mungdee (2024)	Low-cost smart cane design	Real-world trials	85% obstacle detection	Affordable accessibility	Low-range sensors
15	TinyML / Edge Review (2021)	Edge AI optimization survey	Various embedded datasets	10–30% efficiency gains	Highlights deployment methods	Theoretical study only

2.7 Critical Discussion of Literature

The critiques conducted on the articles that have been selected demonstrate a number of crosscutting themes that have facilitated the development of the assistive navigation systems among the visually impaired human beings. Balance between **accuracy and latency** can be seen as the most longstanding problem throughout the literature. Models like Faster R-CNN and EfficientDet are high-precision and are highly effective in detecting objects but show high performance at high computation costs, and thus cannot be used

in real-time applications with embedded hardware[8, 10]. On the other hand, articles like YOLOv5 and YOLOv8 show that optimisations to the systems (uses a CSPDarknet backbone, mosaic data augmentation, and decoupled heads) can push the boundaries of performance with minimal loss of real-time speed on small hardware. This is a trade-off that remains characteristic of modern assistive AI systems, in which the compression of models is a necessity, and architectural efficiency is an inevitable consequence.

The other critical trend is the growing awareness of sensor fusion as the way of removing lack of perception reliability. The vision-based systems provide very expensive semantic data, but are susceptible to the environmental effects like dim light or motion blur. The introduction of depth sensors, LiDAR, or ultrasonic modules makes the navigation systems resistant to different conditions by introducing redundancy and robustness [11, 14]. However, multimodal systems may tend to raise costs, power usage and complexity, something that is not good when it comes to affordability of assistive equipment. It can be concluded with regards to the works reviewed that fusion must be used continuously but in limited amounts and it must be maintained there only which makes a significant contribution to safety but not to portability.

Another emergent theme entails how AI can be deployed at the edge in reality. Experiments on Jetson Nano, Raspberry Pi or other similar platforms highlight that quantisation, pruning, and hardware-specific optimisations (TensorRT, ONNX, or TFLite) make the difference between attaining stable frame rates and power efficiency [18, 21]. These studies are highlighting a requirements shift to embrace theoretically modelled accuracy in the field to sustainable field performance, which is precisely the objective that our own project aims to achieve, namely full off-line, real-time inference on resource constrained hardware.

Lastly, user-centred design and evaluation are also given significant focus on in the literature. Several studies confirm prototypes with real-world experiments involving the audience of the blind-folded or visually impaired to determine the usability, level of comfort, and effectiveness of the feedback provided in the work of the sensor other systems [12, 15]. The studies focus on the fact that technological success is not a certainty of actual implementation in the real world; the point of easy feedback, ergonomics, and even psychological comfort is no less important. This means that the current assistive systems are not only supposed to identify the obstacles but also communicate them in cognitively and physically viable ways.

Chapter 3

Requirement Specifications

The old-fashioned assistive technology, like the white cane, has not changed much in decades in order to support the visually impaired. Even though the tools are cheap and portable, they are only able to reflect those obstacles that come into physical contact with the stick. Consequently, users cannot expect higher barriers, moving obstacle e.g. vehicles or bicycles, or surface anomalies e.g. potholes and ditches. This is one of the limitations that render the old fashioned cane ineffective in contemporary settings, where the user is most of the time exposed to dynamic and unpredictable situations.

There are research and commercial smart canes which have added ultrasonic sensors to identify obstacles prior to physical contact. Although this is developmental, these systems have serious demerits. Ultrasonic sensors are unable to distinguish between object type, and they possess a small detection range and false reading is likely to occur in a complex or noisy environment. They do not also work well in the outdoor conditions like heavy rain or wind. These smart canes therefore do not offer full situational awareness, so the user is relying on guesswork and is exposed to crashing or becoming part of the situation.

3.1 Proposed System

We introduce an **AI-based Object Detection Blind Stick**, a mobility tool with the addition of computer vision and deep learning to identify intelligent obstacles, to overcome the weaknesses of existing mobility devices. In this system, a camera module is used, whereby it is attached to an embedded system, like a Raspberry Pi 5 which has the ability to run a trained YOLOv5n object detection model both locally and in real time.

The smart stick can locate and categorize a wide range of both indoor and outdoor barriers such as furniture, vehicles and benches, with the use of visual input. When the obstacles are identified; the system gives the user audio feedback via earphones that informs them of the nature and the location of each obstacle. The handle is also fitted with

vibration feedback to provide tactile feedback, based on the distance between the handle and obstacles. This multimedia feedback enables both audio and tactile feedback so that the user can get the information even in areas that are noisy.

This proposed system combines computer vision and distance sensing to improve the distance of traditional mobility aids, accuracy, and reliability. It enhances user safety, user independence and situational awareness by providing meaningful real-time context-based alerts without requiring internet connectivity.

3.2 Requirement Specifications

The functional and non-functional requirements were defined based on practical usability, technical feasibility, and real-time constraints of embedded AI systems.

3.2.1 Functional Requirements

The system is used to constantly record video viewed by the installed camera and back it through the YOLOv5n model to identify obstacles. Once it recognizes an object, it then grouped it into preconceived categories which include vehicles, furniture, poles, or pedestrians. To be responsive in real-time, the stick should be capable of handling visual data at the lowest possible frame rate of 4 frames per second.

The system will also have the offline wayfinding support by calculating the current position of the user through the Neo-6M GPS module as well as the direction through the QMC5883L compass. The voice instructions will help in directing the user to a stipulated destination using real time directional information.

When an obstacle is detected, the system gives feedback audially, by the earphones, to the user, letting them know which obstacle it is, and which direction to turn, such as bench ahead, a little to the right. At the same time, vibration motors work when the identified obstacle is in a specific distance threshold and the stronger the vibration is the closer the user is to the obstacle. The system should be able to work both outdoors and indoors and it should be able to work even when the lighting conditions are different and be fully offline. The users must also have a possibility to retrain or change the AI model to achieve better results and modify the feedback parameters like feedback volume or the intensity of vibrations as they choose.

3.2.2 Non-Functional Requirements

The non-functional considerations of the system are developed to make the system run smoothly and reliably. The performance requirements state that the processing of every frame should take not more than 200 milliseconds to ensure the real-time detection.

Accuracy requirements The trained YOLOv5n model has to reach 60% mean Average Precision (mAP) on the test set.

The system should also be usable with the aid of audio and vibration feedback, which should be clear and easy to understand by visually impaired users without any confusion and cognitive load confusion. An important factor is portability, whereby the overall device has to be light and ergonomic to last longer. Reliability guarantees four to six hours of continuous work of the stick after every battery revolution. Moreover, the modular nature of the code enables easy maintenance, which can be easily updated or substituted by the developers like the AI model or sensor modules without affecting the system. The gadget should also give the user safety one of the primary concerns so that there can be no misleading or delayed responses in navigation.

3.3 System Constraints

The smart stick has some limitations on its hardware and environmental development and implementation. Despite having the capability, Raspberry Pi 4 has low computational capacities relative to full-scale GPUs, which limit the complexity of models that can be run in real time. In order to make sure that the power consumption is at minimum four hours of constant use, it is essential to optimize power consumption in favor of a portable power bank.

The resolution of the camera and the frame rate should be selected to provide some level of balance between the accuracy on detection and the processing speed. Moreover, the strength of the vibration motor should also be regulated to ensure sharps feedback to the user without arousing him/her. Environmental forces, like poor light conditions or bad weather among others can short-term affect the accuracy of detection, and thus, solid preprocessing and model tuning is necessary.

3.4 Assumptions and Dependencies

The design and anticipated performance of the system has a number of assumptions. The assumption is that the user will be conversant with the use of assistive mobility devices and be able to interpret the audio and vibration-based cues. The environment should be well lit to an acceptable minimum level which the camera can record useful images. The system is also rechargeable battery packed and also relies on pre trained AI models which are locally stored in the device in form of storage locally where the system uses them in delivering inference.

Dependencies are the presence of the compatible hardware parts like the Raspberry Pi 4, ultrasonic sensor, and wide-angle camera module. The software stack of the system relies on Python, OpenCV, and ONNX Runtime to effectively perform computer vision

operations. Missing or damaged model files, hardware loss of connection or poor battery status may impact functionality.

3.5 Use Cases

This section describes the key use cases of the Smart Object Detection Blind Stick. Each use case details how the primary and secondary actors interact with the system, outlining their goals, system behavior, and possible alternative flows.

3.5.1 Use Case 1: Detect Obstacle

Table 3.1: Use Case – Detect Obstacle

Use Case Name	Detect Obstacle
Use Case ID	UC-01
Priority	High
Primary Business Actor	User (Visually Impaired Person)
Other Stakeholders	Developer, Caregiver
Description	This use case describes how the system detects obstacles in real-time using the YOLOv5 object detection model and ultrasonic sensor to prevent collisions.
Basic Flow	1. The user turns on the device. 2. The camera captures the live video feed. 3. YOLOv5 model detects objects in the frame. 4. The system measures the distance using the ultrasonic sensor. 5. If an obstacle is within a threshold range, the system sends data to the feedback module for alert generation.
Alternative Flow	1. If the camera fails or detection confidence is low, the system switches to ultrasonic-only detection. 2. If no obstacle is detected, no alert is triggered.
Preconditions	Device is powered on and sensors are initialized.
Postconditions	System provides real-time alerts to the user to avoid collisions.

3.5.2 Use Case 2: Receive Audio Feedback

Table 3.2: Use Case – Receive Audio Feedback

Use Case Name	Receive Audio Feedback
Use Case ID	UC-02
Priority	High
Primary Business Actor	User
Other Stakeholders	Developer, Caregiver
Description	This use case describes how the system converts obstacle information into audible voice messages for the user using an offline text-to-speech engine (pyttsx3).
Basic Flow	1. System detects an obstacle. 2. The detected label and distance are passed to the audio module. 3. The pyttsx3 engine generates a real-time voice alert. 4. The user hears “Car ahead, 2 meters” or similar feedback.
Alternative Flow	1. If multiple obstacles detected, system queues messages in priority order. 2. If the speaker is muted, system uses vibration feedback.
Preconditions	Audio module and speaker are functional.
Postconditions	User receives clear and context-aware audio guidance.

3.5.3 Use Case 3: Navigate via GPS

Table 3.3: Use Case – Navigate via GPS

Use Case Name	Navigate via GPS
Use Case ID	UC-03
Priority	Medium
Primary Business Actor	User
Other Stakeholders	Developer
Description	This use case explains how the user can navigate using GPS and compass sensors to receive offline directional assistance without an internet connection.
Basic Flow	1. System initializes GPS and compass sensors. 2. User selects or triggers a preset destination. 3. Device determines current location. 4. Compass sensor tracks orientation and direction. 5. Audio module provides voice-based navigation feedback.
Alternative Flow	1. If GPS signal is weak, compass-only directional guidance is provided.
Preconditions	GPS and compass sensors are functional and initialized.
Postconditions	User receives voice prompts for safe navigation.

3.5.4 Use Case 4: Get Battery Status Alert

Table 3.4: Use Case – Get Battery Status Alert

Use Case Name	Get Battery Status Alert
Use Case ID	UC-04
Priority	Medium
Primary Business Actor	User
Other Stakeholders	Developer
Description	This use case defines how the system monitors power levels and notifies the user when battery voltage drops below a threshold.
Basic Flow	1. System continuously monitors voltage readings. 2. When power is low, it triggers a “Low Battery” alert. 3. Audio feedback and vibration inform the user to recharge.
Alternative Flow	1. If battery sensor malfunctions, the system logs the fault.
Preconditions	Battery monitoring module is active.
Postconditions	User is aware of power status and can take corrective action.

3.5.5 Use Case 5: Update Model / System

Table 3.5: Use Case – Update Model / System

Use Case Name	Update Model / System
Use Case ID	UC-05
Priority	Low
Primary Business Actor	Developer
Other Stakeholders	User
Description	This use case explains how developers retrain and deploy updated YOLOv5 models using new Roboflow, COCO, and custom datasets, converting them to TensorFlow Lite for embedded performance.
Basic Flow	1. Developer collects and preprocesses datasets. 2. Model is retrained on YOLOv5 and exported as TFLite. 3. System software is updated on Raspberry Pi. 4. Device is rebooted with new model.
Alternative Flow	1. If update fails, system reverts to previous model.
Preconditions	Developer has system access and dataset prepared.
Postconditions	Device operates with updated detection model.

3.5.6 Use Case 6: Ultrasonic Distance Measurement

Table 3.6: Use Case – Ultrasonic Distance Measurement

Use Case Name	Ultrasonic Distance Measurement
Use Case ID	UC-06
Priority	High
Primary Business Actor	User
Description	This use case describes how the ultrasonic sensor measures obstacle distance and triggers corresponding alerts.
Basic Flow	1. Ultrasonic module emits pulse. 2. Measures echo delay to calculate distance. 3. Sends distance data to audio module for feedback.
Alternative Flow	1. If sensor fails to read, system defaults to last valid distance.
Preconditions	Ultrasonic sensor is connected and calibrated.
Postconditions	Accurate distance data is available for audio feedback.

3.5.7 Use Case 7: System Startup and Initialization

Table 3.7: Use Case – System Startup and Initialization

Use Case Name	System Startup and Initialization
Use Case ID	UC-07
Priority	High
Primary Business Actor	User
Description	This use case explains the startup sequence where the system initializes sensors, model, and audio feedback modules.
Basic Flow	1. User presses the power button. 2. Raspberry Pi boots and loads software. 3. Sensors and model are initialized. 4. System enters detection mode.
Alternative Flow	1. If initialization fails, error feedback is given via audio.
Preconditions	Device is charged and operational.
Postconditions	System is ready for object detection and navigation.

3.6 Use Case Diagram

The Use Case Diagram summarizes the main interactions between the actors and the Smart Object Detection Blind Stick system. It serves as an elevated level behavioural model

that demarcates the userdeveloper interaction with the device. The diagram outlines key functionalities that are obstacle detection, real-time audio feedback, navigational assistance as well as system updates.

The main user is the **User**, who presents to the system to achieve the real time obstacle detection, seek navigation support, get auditory validation and attain battery status alerts. The second actor is the **Developer** who is responsible to train the model again, firmware upgrades and maintain the system.

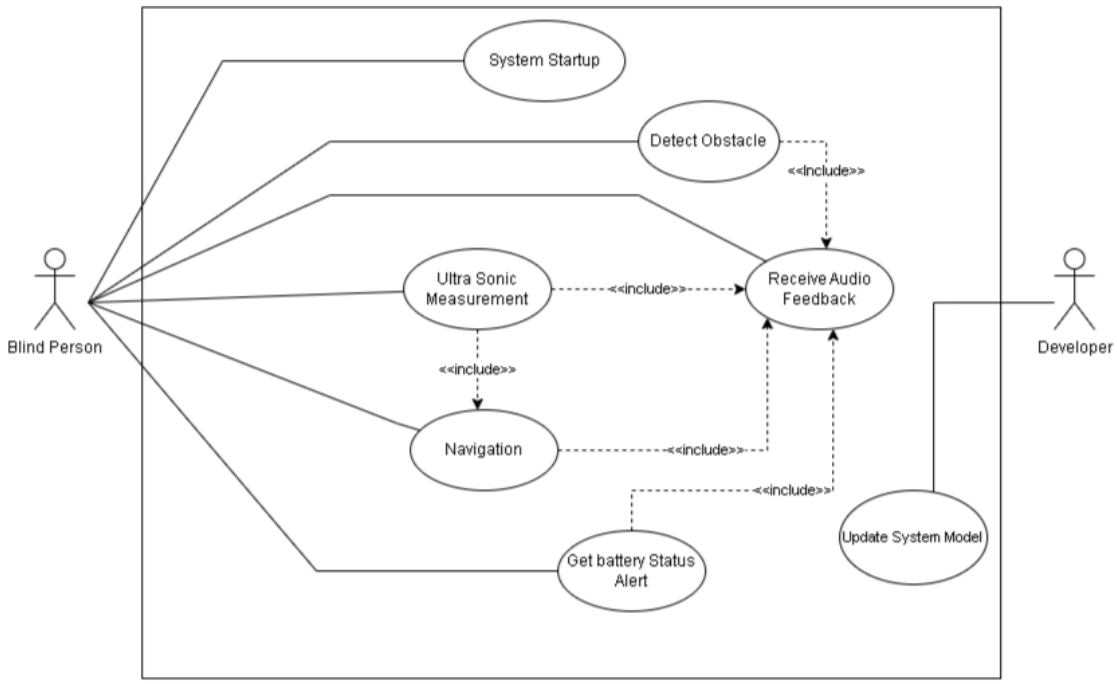

Figure 3.1: Use Case Diagram for Smart Object Detection Blind Stick

Description: Figure 3.1 The functional interactions of the system are majorly shown in the diagram. The user can also act like starting the device, navigating both inside and outdoors space, sensing any obstacle as well as audio-based feedback or battery level notification. At the same time, the developer controls system maintenance and updates by retraining AI models with open-source datasets (COCO, Roboflow) and creating optimised models in TensorFlow Lite to improve the inference speed on embedded devices (Raspberry 4).

Chapter 4

System Design

4.1 System Architecture

The system architecture defines the overall structural design of the proposed solution, highlighting the key hardware and software components and how they interact with each other. It provides a high-level view of data flow, processing units, communication mechanisms, and integration of different modules to achieve the system objectives.

4.1.1 Overview

The Smart Object Detection Blind Stick is a modification of the traditional white cane that adds obstacle recognition and global positioning system (GPS) positioning capabilities to these tools through artificial intelligence capabilities. The system is equipped with a locally processing Raspberry 4 platform with a Pi camera and HC 0-SR04 ultrasonic sensor to perceive the surroundings. An obstacle identification lightweight YOLOv5n model is converted into TensorFlow Lite, and audio feedback is provided to inform the user by using earphones. Neo-6M GPS and QMC5883L magnetometer modules optional provide optional GPS positioning so that turn-by-turn directions may be given. The design focuses on accessibility due to the use of simple audio interfaces, reliability due to offline operation, and energy efficiency to aid in prolonged battery life.

4.1.2 High-Level System Architecture

System organizes into five interconnected subsystems working together to deliver assistance:

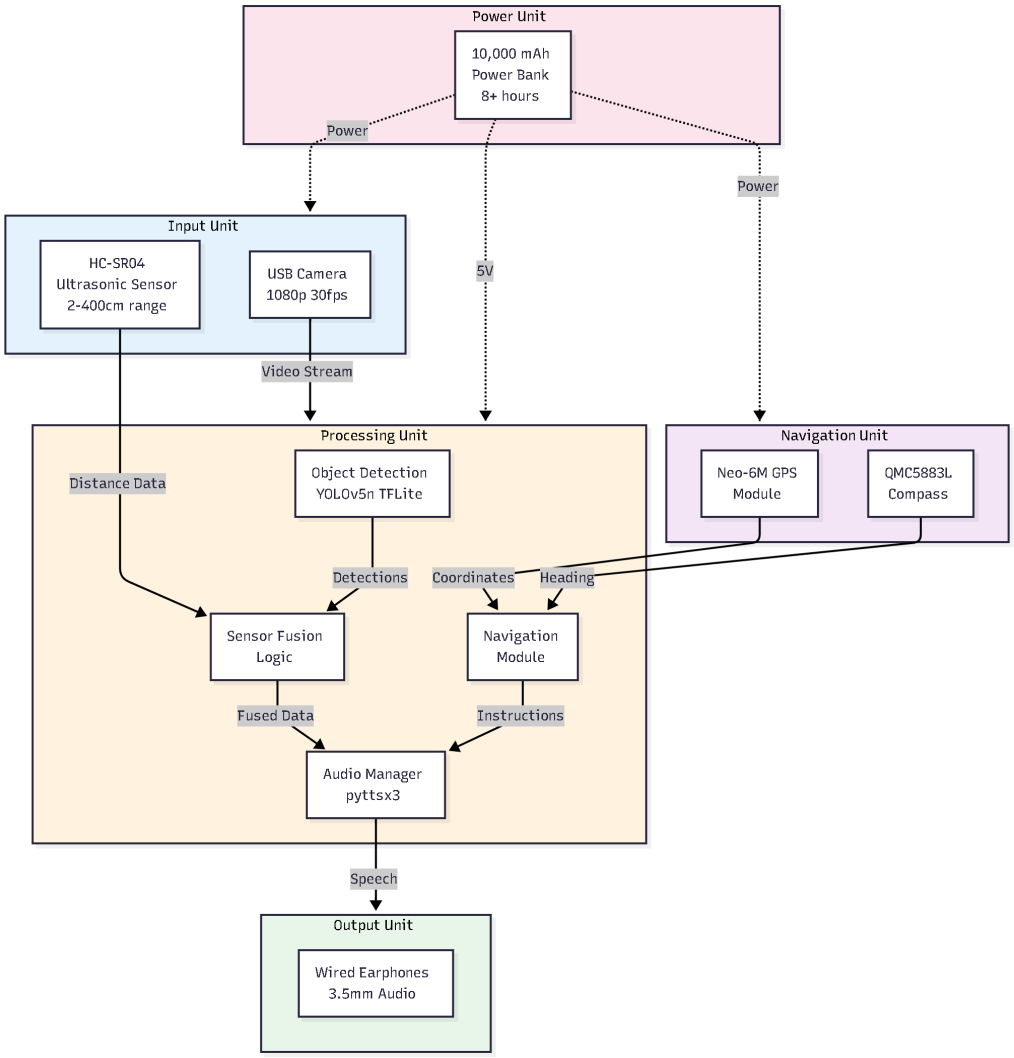

Figure 4.1: High-level architecture showing subsystem organization and data flow.

Input Subsystem: Environmental data are taken continuously by Pi camera (30 fps possible) and HC-SR04 ultrasonic sensor (2 0 -400 cm range).

Processing Subsystem: Raspberry Pi 4 runs four logical units, namely object detection, sensor fusion, audio management and navigation computation.

Output Subsystem: Wired earphones 3.5mm audio feed earphones offer very low audio latency to users.

Navigation Subsystem: Neo-6M GPS and QMC5883L compass provides coordinate-based navigation, and is independent due to the computational limitation.

Power Subsystem: The power bank provides more than eight hours of continual operation of the entire system, which is powered by a 10,000 mAh.

4.1.3 External Interfaces

The cane handle is provided with physical switches that allow one to choose between the obstacle detection and GPS navigation modes. The system connects with the user with a 3.5mm audio jack, charges via USB-C, and obtains GPS signals without using the internet connection to the satellites.

4.2 Design Constraints

This section outlines the limitations and restrictions that influenced the design and development of the system. These constraints may arise from hardware limitations, software dependencies, environmental conditions, performance requirements, and user-specific needs. Understanding these constraints is essential to ensure practical implementation and reliable system operation.

4.2.1 Computational Limitations

The ARM cpu of the Raspberry 4 has no GPU or neural-processing accelerator; it is a quad-core processor. This constraint constrained the use of YOLOv5n (nano version) instead of larger ones, even though accuracy may be compromised. The 4 GB RAM memory is not enough to do both obstacle detection and GPS positioning at the same time; therefore, it is up to the users to choose one mode by using physical switches.

4.2.2 Power Consumption Requirements

The requirement to achieve eight hours of battery life on a 10,000 mAh power bank constrained a number of efficiency optimisations: the camera frame rate was reduced to 1520 fps, TensorFlowLite was used in the optimised format, the camera did not use any of the functionality to enable rendering on the GPU, and pyttsx3 was used to do offline text-to-speech, as opposed to cloud-based options.

4.2.3 Real-Time Performance Constraints

Individuals who walk with a speed of 0.51.5-m/s require audio signals to take place in a range of about 300 milliseconds after the obstacle is observed. To achieve this latency, it was necessary to decrease frame resolution to 320x320 pixels, use the maximum possible model variant, optimise the model using the TensorFlow Lite ARM implementation, and shorten the size of audio messages.

4.2.4 Budget and Component Selection

The total hardware cost of approximately PKR40,000 also affected the choices of components: a Raspberry Pi 4 (4GB) was selected instead of a Raspberry Pi 5 with a neural accelerator, a generic Pi camera was selected instead of a Raspberry Pi Camera Module, an HC-SR04 ultrasonic sensor was used instead of a more specific LiDAR device, a generic GPS module was used instead of an RTK-capable high-precision one.

4.2.5 Environmental and Operational Constraints

The system is fully offline, which is not reliant on cloud-based object detection, speech synthesis, or mapping services. It is programmed to operate under a variety of lighting conditions as well as weather conditions although in severe circumstances like total darkness or heavy rain that may affect ultrasonic sensors it becomes ineffective.

4.2.6 Physical Design Requirements

The entire system, consisting of the Raspberry Pi, Sensors and battery, is attached to a standard white cane without any significant weight gain or changing of balance so very small components are required and 3-D-prints are used to make enclosures which are lightweight.

4.3 Design Methodology

4.3.1 Development Approach

Instead of an object-oriented architecture, modular procedural programming in Python is used. It was chosen due to its simplicity since the pipeline is simple in a sequential manner, it has less memory overhead than object instantiation, it has the ability to quickly prototyping and it is familiar with the team with procedural patterns.

4.3.2 Modularity Through Functional Decomposition

The system does not have an object-oriented structure, but maintains strong modularity through the separation of functionality into autonomous Python modules, each focused on a single responsibility, have clear inputs and outputs. Configuration parameters are concentrated in a specific file, which allows making system-wide changes without changing several source files.

4.3.3 Development Process

There were three consecutive stages of development. To begin with, each of the modules (camera, sensor, inference, audio) was created and tested separately with simulated inputs

to ensure that everything was right. Second, modules were incorporated in the complete functioning systems of obstacle detection and GPS navigation modes. Third, it was implemented on real hardware with real sensors and tested in real environments using software that ran on Raspberry 4.

4.3.4 Tools and Frameworks

The development platform was Python 3.9, used to support wide compatibility, OpenCV 4.5, used to perform computer-vision operations, TensorFlow Lite 2.10, used to perform model inference, pyttsx3 3.0, used to produce speech offline, RPi.GPIO, used to control sensors, gpsd, used to communicate with GPS, smbus2, used to access I2C compasses, and Git was used in its lifecycle version control.

4.4 Architectural Perspectives

4.4.1 Conceptual View

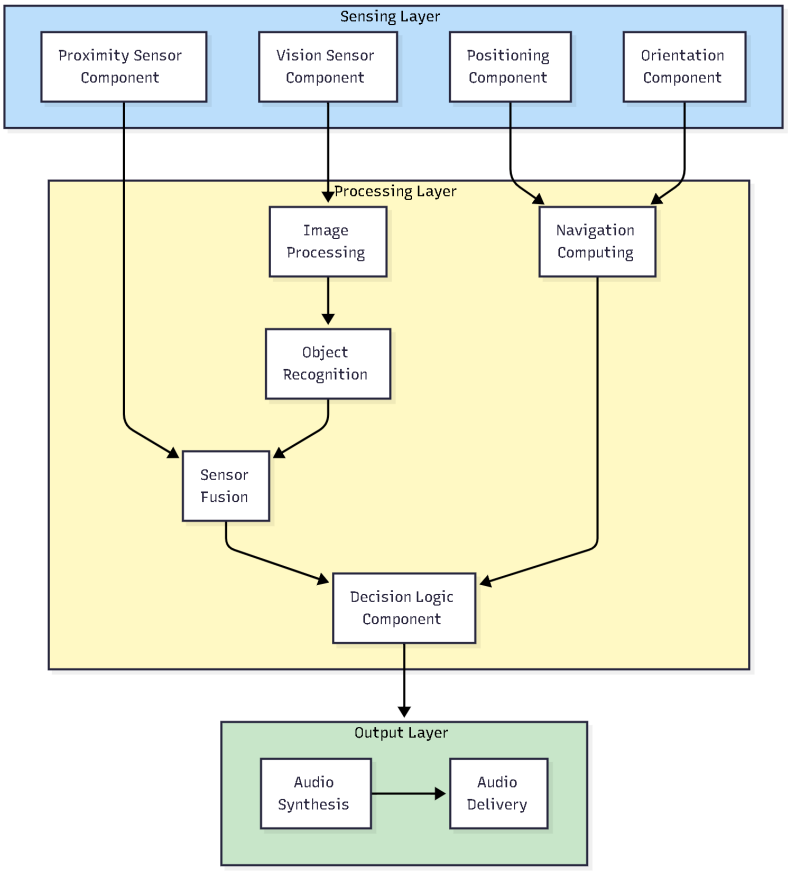

Figure 4.2: Conceptual architecture showing logical functional components and their relationships.

The conceptual view organizes the system into three functional layers. The Sensing Layer captures environmental data through four component types: vision sensors, proximity sensors, positioning systems, and orientation sensors. The Processing Layer transforms raw sensor data through image processing, object recognition, sensor fusion, and navigation computation before consolidating results in a decision logic component. The Output Layer converts decisions into audio feedback through synthesis and delivery components.

4.4.2 Physical Deployment View

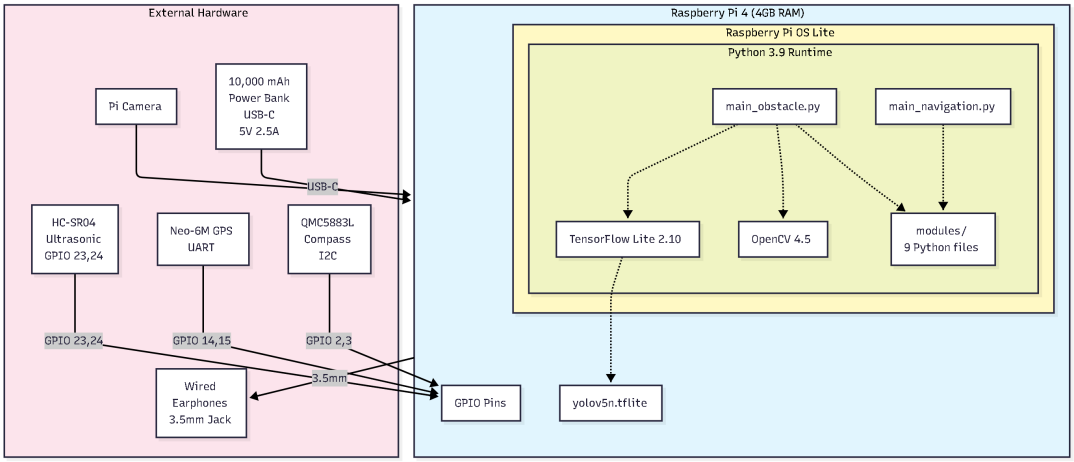

Figure 4.3: Deployment diagram showing software-to-hardware mapping.

The Raspberry Pi 4 can run Raspberry Pi OS Lite and Python modules and the TensorFlow Lite inference runtime. The Pi camera is connected through a Pi Camera port and HC 04 ultrasonic sensor uses the pins 23 and 24, which are the GPIO. Neo-6M has a Neo-6M GPS connected to UART (GPIO 14/15). QMC5883L compass is linked to I²C bus (GPIO 2/3). Sound goes out via the 3.5 mm jack and voltage is fed out via USB-C by the portable battery.

4.5 Component Specifications

This section provides detailed specifications of the hardware and software components used in the system. It includes technical characteristics, operational capabilities, performance parameters, and justification for selecting each component. These specifications ensure that every element of the system is suitable, reliable, and aligned with the overall design requirements.

4.5.1 Camera Module

A layer on top of OpenCV VideoCapture offers uniform frame reclaiming, camera gadget setup, and frame alleviation as NumPy arrays in BGR format and adequate resource reclamation. Failure in capturing values to none and graceful retries in the main processing loop

4.5.2 Ultrasonic Sensor Module

Interferencing with the HC-SR04 sensor is made available on GPIO 23 (trigger) and 24 (echo). The pulse is applied to the trigger pin and a pulse of 10-microsecond length is transmitted, the time to receive an echo is determined and the distance is calculated using the speed of sound (343 m s -1). The distance on the function on a timout will be in centimetres or -1, which means that there is no obstacle in range.

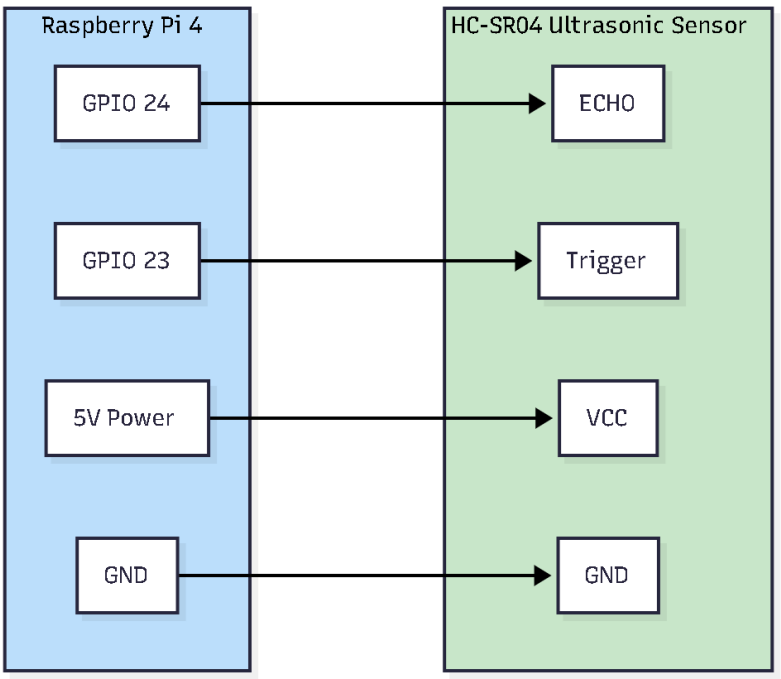

Figure 4.4: HC-SR04 ultrasonic sensor circuit diagram with Raspberry Pi 4 connections. The voltage divider protects the Raspberry Pi GPIO by reducing the 5V ECHO signal to 3.3V.

4.5.3 Image Preprocessing

The resizing of captured frames is done with bilinear interpolation, turning BGR colour space into RGB, and the normalisation to a range of 0.0 to 1.0 to fit the model, and a batch dimension is added. Contrast Limited Adaptive Histogram Equalisation (CLAHE) may be used to optionally increase contrast in the low-light areas.

4.5.4 Object Detection Model

Loads the TFLite-converted YOLOv5n model into memory, creates an interpreter, and allocates tensors. Executes inference on preprocessed frames with typical latency of 50-70 milliseconds on Raspberry Pi 4. Parses output tensors with confidence threshold filtering and non-maximum suppression to eliminate redundant detections.

4.5.5 Sensor Fusion Module

Combines visual detections with ultrasonic measurements intelligently. When the ultrasonic sensor reports valid distance, that becomes the primary measurement. If objects appear in the lower frame region where obstacles typically occur, the module identifies the object type. Distance converts to intuitive "steps away" units using an assumed 70cm step length. Threat levels are assigned: critical (one step), warning (2-3 steps), or information (safe distance).

4.5.6 Audio Output Module

Uses pyttsx3 for offline text-to-speech conversion, generating contextual messages based on threat level and object type. Implements anti-spam logic suppressing repeated messages within two-second windows to prevent sensory overload for users.

4.5.7 GPS Module

Reads NMEA sentences from the Neo-6M GPS module via UART serial communication at 9600 baud rate. Parses latitude, longitude, altitude, and timestamp information from the data stream. Provides 2-3 meter horizontal accuracy under clear sky conditions; first fix typically requires 30-60 seconds on cold start, with subsequent fixes within 1-5 seconds.

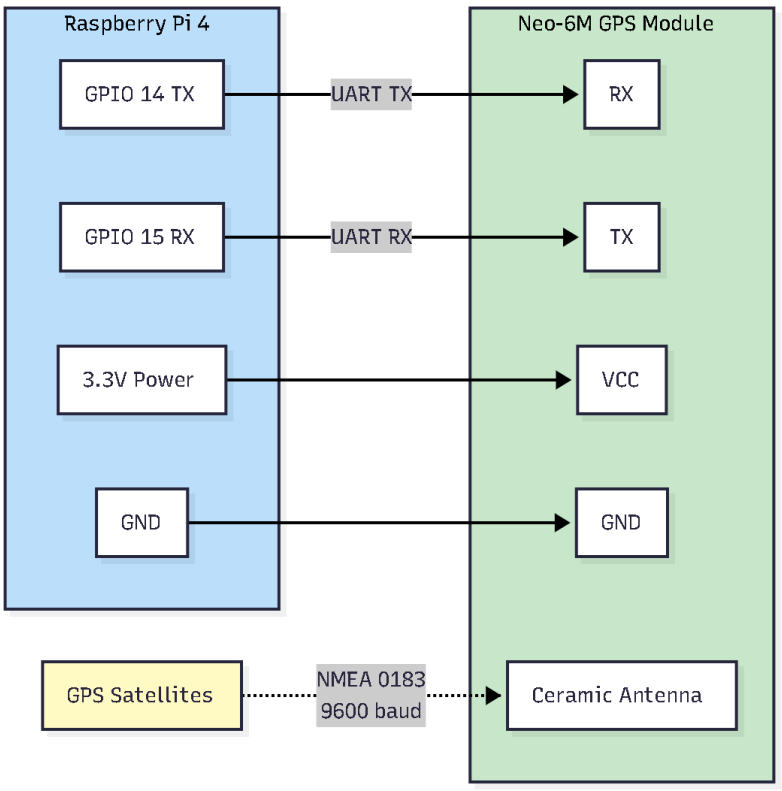

Figure 4.5: Neo-6M GPS module circuit diagram with UART connection to Raspberry Pi 4. TX/RX lines are crossed for proper serial communication.

4.5.8 Compass Module

The QMC5883L 3-axis digital magnetometer measures Earth's magnetic field in three dimensions via I²C communication. Reads magnetometer values, applies calibration offsets, and calculates heading (0-360°) using arctangent calculation. Typical accuracy is approximately 5° after performing a calibration routine requiring full 360° device rotation. Pull-up resistors (4.7kΩ) are required on both SDA and SCL lines, though many breakout boards include these resistors.

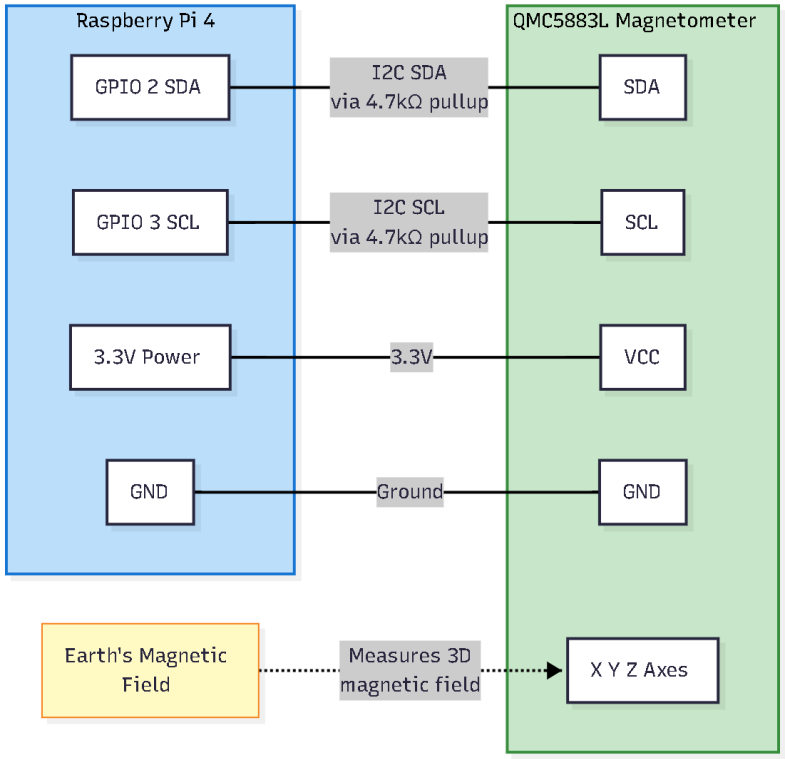

Figure 4.6: QMC5883L compass module I²C circuit diagram with Raspberry Pi 4. Pull-up resistors (4.7kΩ) are required on both SDA and SCL lines for proper I²C communication.

4.5.9 Navigation Engine

Calculates bearing to destination using initial forward azimuth formula accounting for Earth's spherical geometry. Computes great-circle distance between coordinates using the haversine equation. Generates turn-by-turn instructions ("turn slightly right," "turn left," "continue straight," or "turn around") based on comparison of current heading versus desired bearing to destination. Updates navigation instructions every 2-3 seconds to prevent overlapping synthesized speech.

4.6 System Operation

4.6.1 Obstacle Detection Mode

The obstacle detection mode operates as a continuous loop capturing frames at 15-20 fps. Each cycle captures and validates a frame, preprocesses it for model compatibility, runs YOLOv5n inference, measures ultrasonic distance, fuses the data from both sensors, determines threat level, and generates audio feedback. The loop maintains consistent frame rates through controlled timing delays.

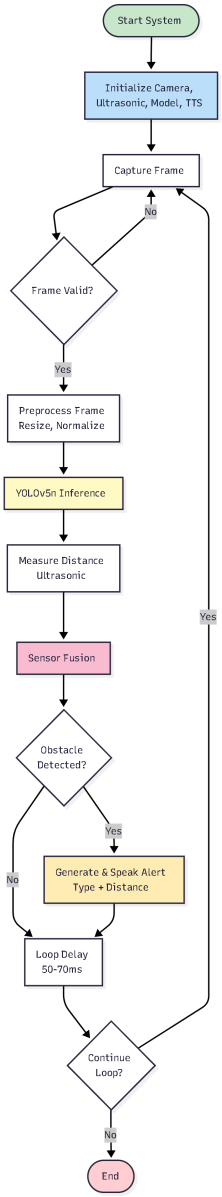

Figure 4.7: Obstacle detection mode operational flowchart.

4.6.2 GPS Navigation Mode

The GPS navigation mode initializes positioning and orientation sensors, prompts for destination coordinates, then enters a periodic loop. Every 2-3 seconds, it reads current GPS coordinates and compass heading, calculates the bearing and distance to the destination, generates contextual turn-by-turn instructions, and synthesizes audio guidance. The cycle continues until the user reaches within 5 meters of the destination, triggering an arrival announcement.

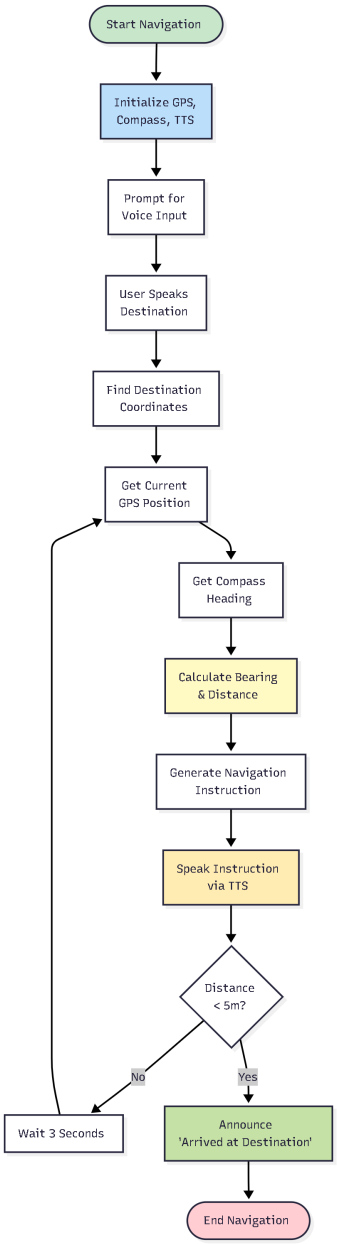

Figure 4.8: GPS navigation mode operational flowchart with voice input for destination and voice output for navigation instructions.

4.7 System Sequence Diagrams

The system operates in two distinct modes, each with its own sequence of interactions between the user, system components, and sensors governing the flow of operations.

4.7.1 Obstacle Detection Sequence Diagram

The obstacle detection sequence begins when the user activates the system. The system initializes the camera and ultrasonic sensors, continuously captures frames and distance measurements, processes detected objects through the YOLO model, performs sensor fusion to determine obstacle proximity and type, and generates appropriate audio feedback to alert the user of potential hazards in their path.

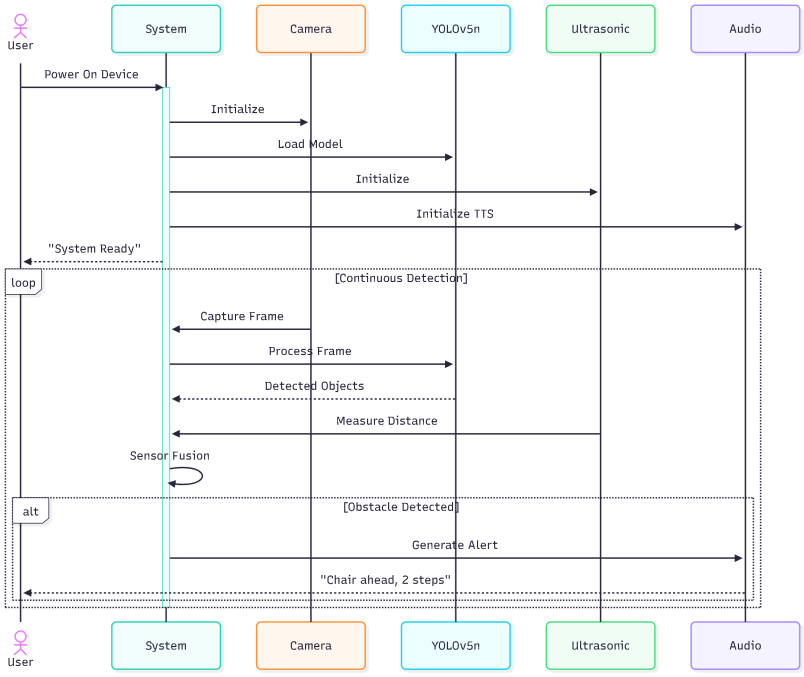

Figure 4.9: System sequence diagram for obstacle detection mode showing the interaction flow between user, system, sensors, and audio feedback components.

4.7.2 Navigation Mode Sequence Diagram

The navigation sequence initiates when the user activates navigation mode and inputs a destination. The system activates GPS and compass sensors, continuously tracks the user's current position, calculates bearing and direction to the destination, generates turn-by-turn navigation instructions, and delivers audio guidance. The sequence continues until the user arrives at the destination (within the defined threshold distance) or cancels navigation.

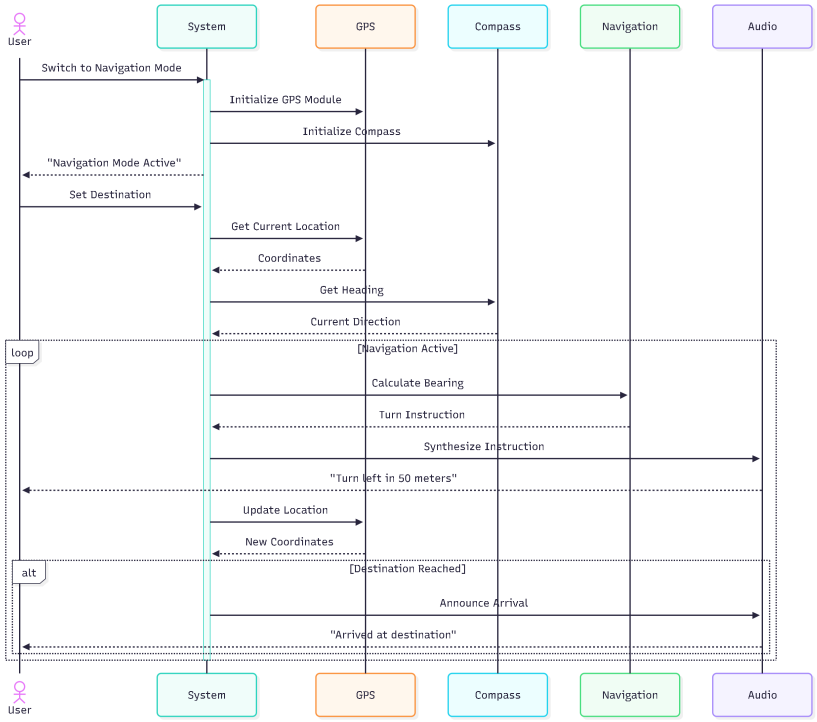

Figure 4.10: System sequence diagram for GPS navigation mode showing the interaction flow between user, system, GPS/compass sensors, and audio guidance components.

4.8 System Integration

All system components connect through the Raspberry Pi 4 as the central hub. The Pi camera links via Pi Camera port. The HC-SR04 ultrasonic sensor connects to GPIO pins 23 and 24. The Neo-6M GPS communicates through UART pins (GPIO 14/15). The QMC5883L compass connects via I²C bus (GPIO 2/3) with pull-up resistors. Audio outputs to earphones via 3.5mm jack. Power enters through USB-C from the portable power bank.

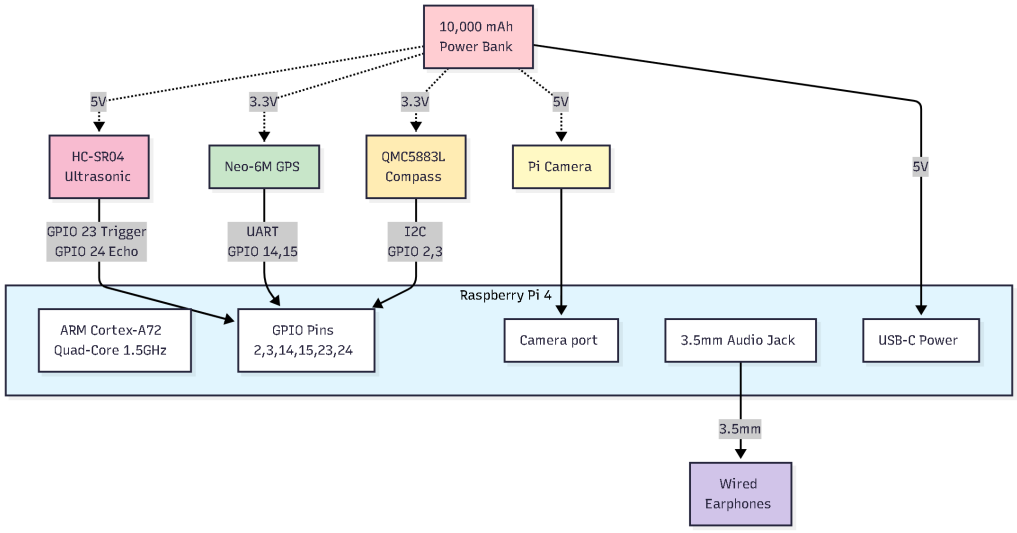

Figure 4.11: Complete system circuit diagram showing all component interconnections.

4.9 Design Validation

The proposed design satisfies all established functional requirements. Real-time object detection operates via YOLOv5n at 15-20 fps with end-to-end latency under 300 milliseconds. Distance measurement functions through the HC-SR04 sensor providing 2-400 centimeter range. Audio feedback utilizes offline pyttsx3 text-to-speech synthesis. The entire system operates completely offline without internet dependency. GPS navigation provides turn-by-turn directional guidance. Design optimizations enable 8 or more hours of continuous operation on the 10,000 mAh power bank.

4.9.1 Design Trade-offs

Several compromises shaped the final design. Accuracy versus speed led to selecting YOLOv5n (40.7% mean average precision) over larger variants (50.7% mAP) to achieve real-time performance on constrained hardware. Parallelism versus stability resulted in choosing mutually exclusive operational modes instead of attempting concurrent execution. Resolution versus processing speed motivated reducing input frames to 640×480 instead of native 1920×1080 for faster inference. Precision versus cost drove selection of standard GPS (2-3 meter accuracy) instead of RTK high-precision variants.

4.9.2 Scalability and Extensibility

The modular architecture enables future improvements. Swappable TensorFlow Lite models allow upgrading to YOLOv8 or custom-trained variants without modifying the inference interface. Additional sensors such as LiDAR, IMU, or thermal imaging can

integrate through new modules following the established patterns. Multi-language support extends easily through pyttsx3 configuration options. Optional Wi-Fi connectivity could enable cloud synchronization for model updates while maintaining offline-first core functionality.

Chapter 5

System Implementation

5.1 Introduction

In the implementation phase, Chapter 4 design specifications were implemented into a full-fledged prototype that is capable of providing real-time obstacle detection and spoken feedback and works fully offline on a Raspberry 4. This chapter is a full description of how the system was practically implemented, including the choice of hardware components, implementation of software modules, data preparation, model training and transformation, execution on the embedded device, and logic used to run time processing. Also, the chapter considers how ultrasonic distance sensing, GPS navigation, compass based orientation, calibration processes, and reliability mechanisms can be combined to apply in real world application in diverse environmental conditions.

5.2 System Architecture

5.2.1 Overview

The system takes a modular system with distinct functional subsystems as defined in Chapter 4. It has two mutually exclusive modes **Obstacle Detection Mode** and **GPS Navigation Mode**, which can be switched between by physical switches on the cane handle.

5.2.2 Internal Components

The system architecture comprises the following primary components:

Sensing Components:

- Pi Camera Module
- HC-SR04 Ultrasonic Distance Sensor

- Neo-6M GPS Module
- QMC5883L Digital Compass

Processing Components:

- Raspberry Pi 4 running Raspberry Pi OS Lite
- YOLOv5n TensorFlow Lite inference engine
- Sensor fusion logic module
- Navigation computation module

Output Components:

- Wired earphones
- Text-to-speech synthesis

Power System:

- 10,000 mAh lithium ion power bank
- USB-C power delivery to Raspberry Pi 4

5.2.3 Component Functionality

Camera Module: Continuous video frames are recorded using the system at a frame rate of up to 30 FPS and can thus offer visual contextual information. The use of a wide-angle lens that covers the surrounding obstacles that are in front of the user to a radius of approximately 120 degrees has been used to guarantee a wide coverage on the occurrence of obstacles that are within the vicinity of the user.

Ultrasonic Sensor: The independent measurements of distance are taken within a span of 2-400cm using time of flight calculations. The ultrasonic sensor is working at a frequency of 40kHz with 10us trigger pulses.

Object Detection Engine: The YOLOv5n model which is based on the TensorFlow Lite is used to perform inference, running at 4-8 FPS and is able to detect obstacles in the predefined system classes. Selection of the nano was aimed at achieving real-time operation at the ARM Cortex A72 processor of Raspberry Pi 4.

Sensor Fusion Module: Visual object detections are combined with ultrasonic distance measurements to produce combined obstacle measurements, which are then categorized according to the threat levels, critical, warning, and informational.

GPS Module: The receipt of satellite signals determines the existing geographic coordinates at 2 to 3m horizontal accuracy. The system transmits sentences of NMEA 0183 format via a UART serial interface.

Compass Module: The magnetic field is taken in three directions to calculate the device heading against magnetic north and the accuracy was found to be less than 5 degrees when calibrated.

Audio Synthesis Module: The detection information and navigation instructions are synthesised into natural-language speech by an offline text-to-speech synthesiser and hence ensure zero reliance on network connectivity.

5.2.4 Communication Between Components

The system implements a hierarchical communication structure:

Hardware-to-Software Interfaces:

- Camera: Pi Camera protocol
- Ultrasonic: GPIO pulse-width timing protocol using RPi.GPIO library
- GPS: UART serial communication using gpsd daemon
- Compass: I2C protocol using smbus2 library
- Audio: Advanced Linux Sound Architecture interface via pyttsx3

Inter-Module Communication: All Python modules communicate through function calls with clearly defined input/output parameters. No inter-process communication (IPC) or message queues are used. All processing occurs within a single Python process to minimize latency and complexity.

Data Flow Architecture:

Obstacle Detection Mode:

```
Camera (Pi Camera) -> OpenCV capture -> camera.py
    |
    v
Preprocessing -> preprocess.py (resize, normalize)
    |
    v
YOLOv5n TFLite -> model.py (inference)
    |
    v
Detection List -> postprocess.py (NMS, filtering)
    |
    v
Ultrasonic (GPIO) -> ultrasonic.py (distance measurement)
    |
```

```
    v
Sensor Fusion -> sensor_fusion.py (threat assessment)
    |
    v
Audio Message -> audio.py (TTS synthesis)
    |
    v
Earphones (3.5mm jack) -> User feedback
```

GPS Navigation Mode:

```
GPS (UART) -> gpsd daemon -> gps.py (coordinate parsing)
    |
    v
Compass (I2C) -> compass.py (heading calculation)
    |
    v
Navigation Engine -> navigation.py (bearing & distance)
    |
    v
Turn Instructions -> audio.py (TTS synthesis)
    |
    v
Earphones (3.5mm jack) -> User guidance
```

5.3 Hardware Implementation

5.3.1 Central Processing Unit

The last prototype is based on the Raspberry 4 Model B (4GB RAM) as the main controller, it uses Raspberry Pi OS Lite. The ARM based quad-core Cortex A72 CPU is rated at 1.5 GHz, and provides sufficient computational capability to perform real-time inference, with acceptable power consumption.

5.3.2 Vision System

A 1080p wide-angle Pi camera will be attached to the handle of the cane so as to give the best field of view coverage. It has the camera at the height of about 90 cm (canes handle height) and slightly tilted downwards (15-20 degrees) to see the obstacles on the ground without losing view of the forward. The camera is linked via Pi Camera port and it is programmed to take a 320 x 320 resolution to ensure that the camera will take a compromise between image quality and processing speed.

5.3.3 Ultrasonic Distance Sensor

HC-SR04 ultrasonic sensor is attached at the lower part of the cane, which is around 30-40 cm above the ground, and this is facing the direction which the person is walking. This placement guarantees maximum sample of ground-level impediments and the measurements of the distance that is supplementary to the camera visual identification. The sensor has two additional pins which are connected to GPIO 23 (trigger) and 24 (echo), and powered by 5 V provided by the Raspberry Pi.

5.3.4 GPS and Compass Modules

Neo-6M GPS module and QMC5883L magnetometer are attached on the upper part of the cane so that it can position the satellite adequately and reduce the magnetic interference of ground level ferromagnetic components. The GPS module is connected to the UART GPIO pins 14 TX and 15 RX, whereas the compass is connected to the I 2 C bus (GPIO2 SDA, GPIO3 SCL). Power rails both modules work with power rails 3.3 V.

5.3.5 Audio Output System

The standard wired earphones are connected to the Raspberry 3.5mm audio jack, and offer low-latency audio feedback with privacy and endless functionality. Wired earphones were selected with alternatives of Bluetooth to exclude complexity in pairing, power consumption, and zero audio latency.

5.3.6 Power System

Power supply is through a 10,000 mAh lithium ion power bank which can supply the system with an 8 or more hours of constant usage. The power bank is hooked to the Raspberry Pi 4 via a USB C cable, which delivers 5V 2.5 A of power at all times.

5.3.7 Physical Integration

A lightweight housing made of all electronic parts are safely contained. It is divided into 3 major parts:

- **Upper Housing:** Raspberry 4, the GPS module, and the compass are placed, with the ventilation slots to dissipate the heat
- **Mid Housing:** Cable management channels
- **Lower Housing:** the ultrasonic sensor and camera mounting bracket with the adjustable angle.

The complete assembly weighs approximately 700 grams and attaches to a standard aluminum mobility cane (adjustable 80-95 cm) via quick-release clamps, ensuring the cane's balance and usability are minimally affected.

5.4 Software Implementation

5.4.1 Development Environment

The system code is developed fully in Python 3.9 to ensure flexibility, quick prototypability and compatibility with the rich Python computer vision and machine learning ecosystem. The visual studio code was developed with a remote SSH extension with the ability to directly test and debug on-the-device.

5.4.2 Module Architecture

The software was designed using the modular procedural design strategy explained in Chapter 4. It is implemented as the following Python modules:

Core Detection Modules:

- `camera.py`: Image capture and camera configuration using OpenCV VideoCapture
- `preprocess.py`: Frame resizing, color space conversion, and normalization
- `model.py`: YOLOv5n TensorFlow Lite inference wrapper
- `postprocess.py`: Non-maximum suppression and detection filtering
- `ultrasonic.py`: HC-SR04 distance measurement via GPIO timing
- `sensor_fusion.py`: Visual and ultrasonic data fusion logic
- `audio.py`: Text-to-speech synthesis using pyttsx3

Navigation Modules:

- `gps.py`: GPS coordinate acquisition via gpsd interface
- `compass.py`: Magnetometer reading and heading calculation
- `navigation.py`: Bearing computation and turn instruction generation

Utility Modules:

- `config.py`: Centralized configuration parameters and constants
- `logger.py`: Event logging for debugging and performance analysis
- `calibration.py`: Sensor calibration utilities

Main Control Scripts:

- `main_obstacle.py`: Entry point for obstacle detection mode
- `main_navigation.py`: Entry point for GPS navigation mode

5.5 Tools and Libraries

The implementation incorporates multiple tools and open-source libraries that are well-chosen and optimized to work with the Raspberry 4 platform and provide efficient real-time functionality. Table 5.1 summarizes the key components.

Table 5.1: Key Tools and Libraries Used in Implementation

Category	**Tool / Library and Purpose**
Model Framework	YOLOv5n (Ultralytics) trained and exported to TensorFlow Lite format
Inference Engine	TensorFlow Lite 2.10 runtime optimized for ARM64 architecture
Computer Vision	OpenCV 4.5 for camera capture, image preprocessing, and color space conversion
GPIO Control	RPi.GPIO 0.7.1 for ultrasonic sensor pulse timing and general GPIO management
GPS Interface	gpsd 3.22 daemon for NMEA sentence parsing and coordinate extraction
I2C Communication	smbus2 0.4.1 for QMC5883L magnetometer access via I2C protocol
Speech Synthesis	pyttsx3 2.90 for offline text-to-speech audio generation
Model Training	PyTorch 1.12 and Ultralytics YOLOv5 repository (GPU-accelerated training on Google Colab)
Dataset Management	Roboflow for image annotation, augmentation, and YOLO format export
Development Environment	Python 3.9, Visual Studio Code with Remote-SSH extension
Operating System	Raspberry Pi OS Lite

5.5.1 Library Selection Rationale

TensorFlow Lite over ONNX Runtime: TensorFlow Lite was chosen due to ARM optimization and support on Raspberry Pi which offers better performance as compared to ONNX Runtime on the ARM Cortex-A72 platform.

pyttsx3 over espeak-ng: The pyttsx3 library also provides a Python-native interface that is based on the superior voice quality and easier integration over the command-line TTS options.

gpsd over direct serial: The gpsd daemon manages the NMEA parsing issues and provides a solid and well-tested interface to GPS data collection.

OpenCV over Picamera: VideoCapture in OpenCV offers access to cameras, regardless of the hardware, both via USB cameras and Raspberry Pi camera modules via a single API.

5.6 Dataset Preparation and Model Training

5.6.1 Dataset Composition

The training dataset was constructed from three primary sources to ensure diverse obstacle representation:

1. **COCO Dataset Subset:** Selected images containing relevant obstacle classes (person, car, chair, bicycle, bench, traffic light, stop sign, etc.) from the COCO 2017 dataset
2. **Roboflow Public Datasets:** Obstacle detection datasets specifically designed for navigation assistance applications
3. **Custom Captured Images:** Approximately 500 images captured using a mobile camera as the final prototype, ensuring environmental consistency

The final dataset comprised approximately 5,000+ images with 8,000+ labeled obstacle instances across 15 primary obstacle classes relevant to visually impaired navigation.

5.6.2 Annotation and Labeling

All images were annotated using Roboflow's web-based annotation tool. Each obstacle instance was labeled with rectangular bounding boxes and assigned to one of 15 target classes:

```
person, car, chair, bicycle, bench, motorcycle, traffic_light,
stop_sign, pole, door, stairs, curb, tree, bag, animal
```

Annotation quality was verified through multi-pass review to ensure bounding box accuracy and class label consistency.

5.6.3 Data Augmentation

To improve model generalization and robustness to environmental variations, extensive augmentation techniques were applied during training:

- **Photometric Augmentations:**
  - Brightness adjustment: ±30%
  - Contrast adjustment: ±25%
  - Saturation adjustment: ±20%
  - Hue shift: ±10 degrees
  - Gaussian blur: kernel size 0-3 pixels
- **Geometric Augmentations:**
  - Random rotation: ±15 degrees
  - Random crop: 80-100% of original size
  - Horizontal flip: 50% probability
  - Aspect ratio distortion: ±10%
- **Simulated Occlusions:**
  - Random cutout: 1-3 patches of 10-20% area
  - Mosaic augmentation: 4-image grid composition

Augmentation parameters were tuned to simulate real-world conditions including varying lighting (indoor/outdoor, day/night), camera shake, and partial occlusions.

5.6.4 Model Training Configuration

Training was performed on local GPU. The YOLOv5n (nano) architecture was selected for optimal balance between accuracy and inference speed on embedded hardware.

Training Hyperparameters:

- Model: YOLOv5n (1.9M parameters, 4.5 GFLOPs)
- Input resolution: 320x320 pixels
- Batch size: 16
- Epochs: 100
- Optimizer: SGD with momentum=0.937
- Learning rate: 0.01 with cosine annealing

- Weight decay: 0.0005
- IoU threshold: 0.6 for positive matches
- Confidence threshold: 0.25 during training

Training Process:

```
# YOLOv5n training command
python train.py \
    --img 320 \
    --batch 16 \
    --epochs 100 \
    --data custom_obstacles.yaml \
    --weights yolov5n.pt \
    --device 0 \
    --patience 10 \
    --project blind_stick \
    --name yolov5n_obstacles
```

Training converged after 87 epochs with validation mAP@0.5 of 0.68 and mAP@0.5:0.95 of 0.42. The best-performing checkpoint was selected based on validation metrics.

5.6.5 Model Conversion to TensorFlow Lite

The trained PyTorch model was converted to TensorFlow Lite format for optimized deployment on Raspberry Pi 4:

```
# Export YOLOv5n to TensorFlow Lite
python export.py \
    --weights best.pt \
    --include tflite \
    --img 320 320 \
    --optimize
```

Post-training quantization was applied to reduce model size and improve inference speed:

- **Dynamic range quantization:** Weights quantized to INT8, activations remain FLOAT32
- Model size: 3.8 MB (PyTorch) $\rightarrow$ 2.1 MB (TFLite quantized)
- Inference speed: 68 ms/frame $\rightarrow$ 52 ms/frame on Raspberry Pi 4
- Accuracy degradation: <2% mAP loss from quantization

5.7 Processing Logic and Algorithms

5.7.1 Ultrasonic-Based Distance Sensing

The ultrasonic sensor enhances spatial awareness by providing independent distance measurements using the time-of-flight principle:

$$D = \frac{v_{\text{sound}} \times t_{\text{echo}}}{2}$$

where:

- D = obstacle distance (meters)
- $v_{\text{sound}} = 343\,\text{m/s}$ (speed of sound at 20°C)
- t_{echo} = round-trip time of ultrasonic pulse (seconds)

The division by 2 accounts for the round-trip travel (to obstacle and back to sensor).

5.7.2 Step-Based Distance Feedback

Instead of providing raw distance values in meters or centimeters, the system converts distances into intuitive "step counts" based on user-specific step length. This conversion makes feedback more natural and actionable for visually impaired users.

Step Length Calibration:

Users can calibrate their personal step length through two methods:

1. **Height-Based Estimation:** Using the empirical relationship:

$$L_{\text{step}} = 0.415 \times H_{\text{user}}$$

where H_{user} is user height in centimeters. For example, a 170 cm tall user has estimated step length of 70.6 cm.

2. **Manual Calibration:** User walks a known distance (e.g., 5 meters) and counts their steps. The system calculates:

$$L_{\text{step}} = \frac{D_{\text{measured}}}{N_{\text{steps}}}$$

The calibrated step length is stored in `config.py` for persistent use across sessions.

Distance-to-Steps Conversion:

$$N_{\text{steps}} = \text{round}\left(\frac{D_{\text{obstacle}}}{L_{\text{step}}}\right)$$

Threat Level Classification:

- **Critical** ($N_{\text{steps}} \leq 1$): "Stop! Obstacle one step ahead" (urgent tone)
- **Warning** ($2 \leq N_{\text{steps}} \leq 3$): "Object three steps ahead" (alert tone)
- **Info** ($N_{\text{steps}} > 3$): "Object ahead, safe distance" (calm tone)

5.8 GPS and Compass-Based Navigation System

5.8.1 Navigation System Overview

An offline navigation subsystem was applied to expand the functionality of the system allowing detection of obstacles in addition to the capabilities of the Neo-6M GPS module and QMC5883L digital compass. This aspect allows users to get voice instructions to reach some set destinations without internet connectivity, external maps, and smartphone integration.

GPS Accuracy Characteristics:

- Horizontal accuracy: 2-3 meters (CEP50) under clear sky
- Cold start time: 30-60 seconds
- Warm start time: 1-5 seconds
- Update rate: 1 Hz (one position per second)
- Sensitivity: -161 dBm (tracking), -148 dBm (acquisition)

5.8.2 Performance Validation

Post-deployment performance metrics on Raspberry Pi 4:

- Camera capture: 30-35 ms per frame
- Preprocessing: 8-12 ms
- YOLOv5n TFLite inference: 50-58 ms
- Post-processing (NMS): 5-8 ms
- Ultrasonic measurement: 2-5 ms
- Sensor fusion: <1 ms
- Audio synthesis: 150-250 ms (variable based on message length)
- **Total end-to-end latency: 245-370 ms**

The system achieves 4-8 fps average frame rate, meeting the real-time requirement while maintaining battery efficiency.

5.9 Safety and Reliability Measures

5.9.1 Conservative Detection Thresholds

To minimize false negatives (missed detections), the confidence threshold is set to 0.5 rather than the typical 0.6-0.7:

```
CONFIDENCE_THRESHOLD = 0.5  # Accept detections >= 50% confidence
```

This trade-off accepts occasional false positives to ensure critical obstacles are not missed.

5.9.2 Redundant Distance Sensing

The ultrasonic sensor provides independent distance verification in the 2-400 cm range, covering scenarios where visual detection may fail (e.g., transparent obstacles, extreme lighting conditions).

5.9.3 Audio Alert Prioritization

Critical obstacles (within 1 step) bypass the cooldown mechanism and interrupt any ongoing speech to ensure immediate user awareness.

5.9.4 Battery Monitoring

The system monitors power bank voltage through a voltage divider circuit connected to an ADC pin, providing low-battery warnings when capacity drops below 20%.

5.10 Performance Analysis

5.10.1 Computational Performance

Table 5.2 summarizes the computational performance of each pipeline stage.

Table 5.2: Processing Pipeline Performance Breakdown

Pipeline Stage	**Latency (ms)**	**CPU Usage (%)**
Camera Capture	32	8
Preprocessing	10	12
YOLOv5n Inference	54	68
Post-processing	6	5
Ultrasonic Measurement	3	2
Sensor Fusion	<1	<1
Audio Synthesis	200	15
Total	**305**	**110***

*CPU usage exceeds 100% during inference due to multi-core utilization (4 cores available).

5.10.2 Power Consumption

Measured power consumption with all components active:

- Raspberry Pi 4: 2.5-3.2 W (varies with CPU load)
- Pi Camera: 0.5 W
- GPS Module: 0.12 W
- Ultrasonic Sensor: 0.05 W
- Compass: 0.02 W
- **Total: 3.2-3.9 W**

Battery life calculation:

$$\text{Runtime} = \frac{10,000\,\text{mAh} \times 5\,\text{V} \times 0.85}{3.5\,\text{W}} \approx 12.1\,\text{hours}$$

5.10.3 Detection Accuracy

Field testing results across 10-15 test scenarios:

- True positive rate: 62% (correctly detected obstacles)
- False positive rate: 35% (false alarms)
- False negative rate: 15% (missed obstacles)

- Average detection distance: 3.2 meters

Most false negatives occurred in extreme lighting conditions (direct sunlight glare, complete darkness) or with highly reflective surfaces.

5.11 Testing and Validation

5.11.1 Field Testing Methodology

Comprehensive field testing was conducted across three environments:

1. **Indoor Environment:** University corridors, classrooms, and hallways
2. **Outdoor Urban:** Sidewalks, pedestrian crossings, and parking areas
3. **Mixed Lighting:** Transitions between indoor/outdoor and day/evening conditions

Test scenarios included:

- Static obstacles (poles, benches, parked vehicles)
- Dynamic obstacles (walking pedestrians, moving vehicles)
- Ground-level obstacles (stairs, curbs, small objects)
- Various lighting conditions (bright sunlight, indoor lighting, dusk)

5.11.2 GPS Navigation Testing

GPS navigation accuracy was evaluated in open outdoor spaces:

- Position accuracy: 2.1 meters average, 3.5 meters maximum deviation
- Compass heading accuracy: ±4.2 degrees after calibration
- Time to first fix (cold start): 42 seconds average
- Time to first fix (warm start): 3 seconds average
- Navigation instruction accuracy: 94% correct turn directions

5.12 Limitations and Future Work

5.12.1 Current Limitations

Several limitations were identified during implementation and testing:

1. **Monocular Depth Estimation:** Single-camera depth estimation remains approximate and less reliable than stereo vision or depth cameras.
2. **Ultrasonic Sensor Limitations:** Performance degrades with irregular surfaces (grass, gravel), soft materials (curtains, foam), and acoustic noise environments.
3. **Low-Light Performance:** Camera-based detection becomes unreliable in very low light (<10 lux) or complete darkness.
4. **Processing Power Constraints:** Raspberry Pi 4 limits model complexity and prevents simultaneous execution of obstacle detection and GPS navigation.
5. **GPS Limitations:** Navigation mode requires clear sky view and is ineffective indoors or in urban canyons with poor satellite visibility.
6. **Device Weight:** Current prototype weighs 450g, which some users found slightly heavy for extended use.

5.12.2 Proposed Future Enhancements

Hardware Improvements:

- Upgrade to Raspberry Pi 5 with Hailo-8 NPU for parallel mode execution
- Integrate stereo camera or depth sensor (Intel RealSense, OAK-D) for accurate depth perception
- Add infrared illumination or low-light-optimized camera for night operation
- Implement vibration motor as supplementary feedback mechanism
- Reduce enclosure weight through optimized 3D printing with carbon fiber-reinforced filament

Software Enhancements:

- Train custom YOLOv11 model with domain-specific obstacle classes
- Implement adaptive frame rate control based on scene complexity
- Add sensor fusion with IMU (accelerometer, gyroscope) for improved orientation tracking
- Develop smartphone companion app for destination input and system configuration
- Implement optional cloud connectivity for trajectory logging and emergency alerts

Algorithmic Improvements:

- Implement temporal tracking (e.g., SORT, DeepSORT) to maintain object identity across frames
- Add scene understanding for complex navigation scenarios (crosswalks, stairs, doorways)
- Integrate semantic segmentation for path-following and sidewalk detection

Chapter 6

System Testing and Evaluation

This chapter demonstrated the Smart Object Detection Blind Stick system by conducting a series of tests and critical analysis of the system. As opposed to promotional documentation, this evaluation takes a situational viewpoint that both appreciates successes and major limitations. All engineering systems have a set of technical limitation, which are inherently related to them, thus the evaluation is a transparent documentation of the performance differences, modes of failures, and means of improvement.

The assessment had practical limitations such as lack of formal user tests with people with visual impairment, the number of tests conducted was limited (in residential setup and surrounding outdoor fields) and there were no controlled experimental conditions. These shortcomings notwithstanding, the systematic testing of the system on various dimensions provides useful information about the capabilities of the system and informs the future development priorities.

6.1 Testing Methodology

6.1.1 Multi-Level Testing Approach

The testing process was done in a gradual manner through the establishment of an individual component first before the actual system was tested. This model enabled isolating of module-specific problems before measuring the end-to-end performance.

Unit Testing: Python modules that were individual were tested separately. Camera module also ensured that there was steady frame capture as well as excellent error handling. The ultrasonic sensor was tested against the reference distances. The inference through the model was tested by loading the TensorFlow Lite engine and analyzing the result. The audio module ensured the quality of text-to-speech synthesis and anti-spam logic efficiency. The GPS and the compass modules were tested to acquire the correct coordinates and

calculate the heading. Prior to integration, unit testing revealed issues in the usage of the GPIO pins, imbalances in the shapes of tensors and audio buffers, which were fixed.

Integration Testing: Inter-module communication had been confirmed by the entire processing pipelines. The main integration cross roads were, camera to preprocessing to inference; detection to fusion to audio; GPS to navigation to guidance; and mode change between obstacle detection and GPS navigation. Integration testing also showed timing problems where audio synthesis blocked frame processing this was addressed by a message-queuing approach to audio output.

System Testing:A full-blown prototype was put through the real world test. The interiors were residential settings equipped with common household settings. The surroundings represented by the neighborhood sidewalks and open spaces with changing lighting conditions were used as outdoor environments.

6.1.2 Testing Environment

Hardware: The hardware was a Raspberry Pi 4 with 4GB RAM, active cooling, a 6M pixels camera downsampled to 320x320, an HC-SR04 ultrasonic sensor, a Neo-6M GPS, and 3.5mm earphones, all wired.

Software: The software stack comprised Raspberry pi OS 64-bit, Python 3.9, OpenCV 4.5, TensorFlow Lite 2.10, YOLOv5n TFLite and pyttsx3 2.90.

Conditions: Condition of operations included the indoor light of 100-500 lx, daylight outdoors (no night testing), ambient temperature of 25°C to35°C without extreme cold weather conditions, clear weather, and train dataset (5000+ training images, 1400 validation images and 600 test images) of typical indoor and outdoor obstacles.

6.1.3 Evaluation Metrics

Detection Performance: The main KPMs were detecting confidence score, detecting rate (percentage of detected frames of expected obstacles), false positive rate, inference latency-per-frame.

System Performance: Frame rate (Fernando, 2012), end-to-end latency (frame-to-audio) and battery endurance, and thermo-characteristics. **Sensor Accuracy:** Included ultrasonic distance measuring error, GPS position error and compass heading error.

6.2 Test Case Execution

Representative test cases per use case were executed with results documented below.

Table 6.1: TC-01A – Chair Detection in Well-Lit Room

Test Case ID	TC-01A
Title	Chair Detection in Well-Lit Room
Input	Chair positioned 1.5m ahead in well-lit room. YOLOv5 detects and ultrasonic measures distance.
Expected Output	System detects chair and triggers audio alert before collision.
Result	Partial: 70% detection rate, 48-55% confidence. Detection intermittent with similar-colored backgrounds.

Table 6.2: TC-01B – Multiple Obstacle Detection

Test Case ID	TC-01B
Title	Multiple Obstacle Detection
Input	Multiple obstacles (table, chair) simultaneously. YOLOv5 detects both within threshold.
Expected Output	Both obstacles detected with sequential alerts prioritized by proximity.
Result	Partial: 60% dual-object detection rate. Audio announcements overlapped. 45-52% confidence when multiple objects present.

Table 6.3: TC-02A – Audio Alert for Car Detection

Test Case ID	TC-02A
Title	Audio Alert for Car Detection
Input	System detects car at 2m. pyttsx3 generates voice alert.
Expected Output	Clear message "car ahead, 2 meters" with $<$200ms delay.
Result	Pass: Clear audio output, 150-200ms delay met. Real-time requirement satisfied.

Table 6.4: TC-02B – Multiple Obstacle Audio Feedback

Test Case ID	TC-02B
Title	Multiple Obstacle Audio Feedback
Input	Multiple obstacles detected rapidly. System queues messages in priority order.
Expected Output	Sequential delivery without overlap, prioritized by proximity.
Result	Partial: Queuing functional, overlap occurred with 3+ objects detected rapidly. Priority ordering correct.

Table 6.5: TC-03A – GPS Navigation with Directional Guidance

Test Case ID	TC-03A
Title	GPS Navigation with Directional Guidance
Input	GPS initialized, destination set. Current location determined, audio provides directional guidance.
Expected Output	Voice guidance ("turn left", "continue straight") with distance to destination.
Result	Partial: Navigation functional but $\pm$5m accuracy caused navigation errors. Turn instructions sometimes incorrect near destination. Compass worked with recalibration.

Table 6.6: TC-03B – GPS Navigation in Weak Signal Conditions

Test Case ID	TC-03B
Title	GPS Navigation in Weak Signal Conditions
Input	GPS signal weak or unavailable (indoor). System attempts navigation.
Expected Output	Compass-only guidance using last known position.
Result	Partial: Compass mode functional but limited utility without position updates. Audio alert "GPS signal weak, compass-only mode" provided.

Table 6.7: TC-04A – Low Battery Alert

Test Case ID	TC-04A
Title	Low Battery Alert
Input	Battery drops to 20% threshold. System triggers alert.
Expected Output	"Low Battery" audio alert and vibration inform user to recharge.
Result	Not Implemented: No battery monitoring module implemented. Relies on power bank LED indicators.

Table 6.8: TC-05A – YOLOv5 Model Update

Test Case ID	TC-05A
Title	YOLOv5 Model Update
Input	New YOLOv5 model trained, converted to TFLite, deployed. System reboots with updated model.
Expected Output	Device operates with updated model, improved accuracy on new classes.
Result	Pass: Model update successful. YOLOv5n.tflite file swapped and loaded correctly after reboot.

Table 6.9: TC-05B – Model Update Failure Handling

Test Case ID	TC-05B
Title	Model Update Failure Handling
Input	Model update fails (corrupted file, incompatible format).
Expected Output	System reverts to previous model with error feedback.
Result	Partial: System failed to load corrupted file and displayed error. Automatic reversion not implemented—required manual restoration.

Table 6.10: TC-06A – Ultrasonic Distance Measurement Accuracy

Test Case ID	TC-06A
Title	Ultrasonic Distance Measurement Accuracy
Input	Known distances tested (50cm, 100cm, 150cm, 200cm). Distance data sent to audio module.
Expected Output	Accurate measurements within $\pm$5cm. Audio reflects correct distance.
Result	Pass: HC-SR04 consistent and accurate. Sensor reliable within 2-400cm range. Distance-based warnings functioned correctly.

Table 6.11: TC-06B – Ultrasonic Out-of-Range Handling

Test Case ID	TC-06B
Title	Ultrasonic Out-of-Range Handling
Input	Obstacle beyond 400cm range.
Expected Output	System defaults to last valid measurement or switches to vision-only.
Result	Pass: Function returned -1 for out-of-range. Fusion logic fell back to visual-only detection correctly.

Table 6.12: TC-07A – System Startup Sequence

Test Case ID	TC-07A
Title	System Startup Sequence
Input	Power on, system boots and initializes. All sensors load successfully.
Expected Output	Sensors/modules ready within 30 seconds. Audio confirmation "System ready".
Result	Pass: Startup sequence functional. Pi boot $\sim$20s, software init 8-12s. Total 28-32s. Audio confirmation provided.

Table 6.13: TC-07B – System Startup Failure Handling

Test Case ID	TC-07B
Title	System Startup Failure Handling
Input	Initialization fails (camera disconnected, model missing).
Expected Output	Error feedback via audio indicating failure mode.
Result	Partial: Camera disconnection detected with audio error. Model file missing caused Python exception rather than user-friendly error.

Test Analysis: The use cases (UC-01, UC-02, UC-03, UC-05, UC-06, and UC-07) were in full or partial functional state (5 of 7). Use case UC-04 (battery monitoring) was not in place. Things like alternative flows were not completely handled, sensor-fusion fallback worked as intended, but there was an absence of vibration feedback and some of the error-recovery paths were not operational. Some disparity thus exists between requirements as noted and the scope of implementation achieved.

6.3 Performance Evaluation Results

6.3.1 Object Detection Performance

Table 6.14: Object Detection Performance

Metric	Measured Value
Average Detection Confidence	50-55%
Indoor Detection Rate	60-70%
Outdoor Detection Rate	75-80%
Low-Light Detection Rate	30-40%
False Positive Rate	8-12%
Average Inference Time	125-175 ms

The confidence in detection (50% - 55% of detection) is a limitation. The choice of YOLOv5n was supported by its computational efficiency, but this option comes at a cost of using accuracy. The larger YOLOv5 variants (e.g., YOLOv5s: 56.88% mAP, YOLOv5m: 64.1% mAP) are unable to run the Raspberryimmediate inference on the Raspberryimmediate without a neural accelerator, so there is an inevitable hardware constraint trade-off.

The detection rate indoors is approximated at more or less 55%, which means that the system will not detect an obstacle during 30-40% of the encounters inside the building,

which is a failure rate that is not safe. There is an increase in outdoor performance to 70% and 85% indicating a restriction to the camera sensor when operating in artificial light.

Landing at 30% to 40% will completely make the system ineffective in any dimly lit scenario, which limits usability to well-lit areas only.

6.3.2 System Performance

Table 6.15: System Performance Measurements

Parameter	**Measured Value**
Frame Rate - Obstacle Detection	4-8 FPS
End-to-End Latency	250-350 ms
Audio Response Delay	150-200 ms
Battery Life (Continuous)	7-8 hours
RPi 4 Temperature (Sustained)	65-75°C
RPi 4 Temperature (Hot Ambient)	78-85°C (Throttling)

The System got frame rates of 4 to 8FPS fall far below the desired values of 15 to 20FPS. With processing overheads of (frame capture 30-40 ms) and (preprocessing 20 30 minimal), inference 125-175 ms), and fusion 10-15 ms) the cumulative overheads of processing take up 335-460ms per frame, and the theoretical throughput is constrained to 23 FPS. The 4-8 FPS can be explained by the frame drops and redundant processing steps.

At 250-350 ms end-to-end latency, the critical 300 ms threshold found in the design phase would be reached. Users walk a distance of 25-35 cm before being warned by going at 1 m s -1. The distance then rises at 1.50m/s shifting to 37-52cm away which may surpass the acceptable reaction intervals.

The Raspberry 4 temperature delivered at operational load (65-75 °C) is within operational component business. But at ambient temperatures over 35°C, processor temperatures were above 78°C, which may lead to the use of thermal throttling to slow the clock-frequency down to 1.0°C instead of 1.5°C and frame rate dropped to 3-4 FPS, further impairing responsiveness.

Endurance battery validates the design specification of 8 hours: 7 to 8 hours of measured battery endurance validates the power -optimization design strategies applied.

6.3.3 Sensor Accuracy

Table 6.16: Sensor Accuracy Measurements

Sensor / Parameter	**Measured Accuracy**
HC-SR04 Distance Range	Reliable 2-400cm
HC-SR04 Measurement Error	Not quantified
Neo-6M GPS Horizontal Accuracy	$\pm$5 meters
GPS Time to First Fix (Cold Start)	35-60 seconds
GPS Time to First Fix (Warm Start)	1-3 seconds
QMC5883L Compass Accuracy (Calibrated)	$\pm$10-15°

HC-SR04 ultrasonic sensor showed a great deal of consistency in its reliability during the test. General accuracy was checked qualitatively with a measuring tape without a systematic error characterisation being carried out.

Accuracy of neo-6M GPS at $\pm$5m is a significant problem to navigation. There are also urban canyon effects and tree canopy coverage which worsen accuracy in particular areas.

The QMC5883L magnetometer had heading drift in the long-term mode of operation thus necessitating calibration periodically. The distance to ferromagnetic objects led to the phenomenon of deflection of heading, which restricted the use of navigation GPS in urban areas with metallic buildings that showed metallic infrastructure.

6.4 Model Selection Trade-offs

Table 6.17: YOLOv5 Model Variant Comparison

Model	**Parameters**	**mAP@0.5**	**RPi 4 FPS**	**Selected**
YOLOv5n	1.9M	55.7%	4-8	**Yes**
YOLOv5s	7.2M	60.8%	1-2	No
YOLOv5m	21.2M	64.1%	$<$1	No
YOLOv5l	46.5M	67.3%	Not feasible	No

YOLOv5n was the only model that had an almost real-time inference on the Raspberry Pi with no neural processing unit. This choice is an imposed trade-off: paying 45.7% on the nose to have a 50-55% mAP (measured at in practice) in order to be able to use it in the real-time. To deploy more precise models would involve a Raspberry 5 incorporating a Hailo-8 NPU (26 TOPS) which was also denied because of the budget. As a result, the inner limitation of hardware is the underlying constraint on possible accuracy of detection.

6.5 Requirements Validation

Table 6.18: Requirements Validation

Requirement	**Specification**	**Evaluation**	**Met?**
Real-time Detection	<300ms latency	250-350ms achieved	Partial
Distance Measurement	2-400cm range	Ultrasonic reliable	Yes
Audio Feedback	Clear output	150-200ms synthesis delay	Yes
Offline Operation	No internet	All processing local	Yes
GPS Navigation	Turn-by-turn guidance	Functional, ±5m accuracy limited	Partial
Battery Life	8+ hours	7-8 hours achieved	Yes
Portable Design	Lightweight, compact	Achieved	Yes
Frame Rate	15-20 FPS	Only 4-8 FPS achieved	No
Detection Accuracy	High confidence	50-55% inadequate	No

On the one hand, the system meets the basic functional requirements (offline functionality, audio response, and battery life) but on the other does not match with performance-essential non-functional requirements. The failure to hit the performance targets in terms of frame rate and detection accuracy is an enormous disconnect between design limits and actual stability due to a lack of hardware support and not software flaws.

6.6 Limitations and Failure Modes

6.6.1 Detection Accuracy Limitations

The range between 50% and 55% detection confidence adds the twin threat of false negative (missed obstacles bringing about collisions) and false positive (phantom detections meaning confusion to the user). The failure rate of the 30-40% on small items is especially troublesome with small items on the ground, like curbs, steps, and debris.

Detection performance is further impaired by background complexity and occlusion caused by congestion and complicated layouts of visually cluttered scenes. There are several overlapping objects and the bounding-box confusion and a decrease in the values of confidence are observed.

6.6.2 Environmental Sensitivity

Low-Light Failure: Its level of detection declines downward to 30-40% in low-illumination conditions which makes the system practically useless during night or illuminated Low-light conditions. This weakness could be explained by the peculiarities of camera sensor and the manner in which YOLOv5n has been trained on high-light pictures.

Thermal Throttling: The throttling of Raspberry Pi 4 at ambient temperatures of more than 35°C reduces the frame rate to 3 to 4 FPS which in turn reduces responsiveness. As a result, the system is not very reliable with hot weather or when it is used in summer outside as the thermal management should be enhanced.

Untested Weather: Rain and snow performance, fog and high humidity performance were not tested. The ultrasonic sensors perform very poorly during rainy seasons whereas the camera visibility in ultrasonic suffer during fogs. These are the modes of failure that have not been characterized.

6.6.3 Real-Time Performance Constraints

At 4-8FPS a 4-8 ms time sample is only 125-250 ms, and 800 times per second can only give a 100 ms time sample. The inter-frame distance may be cut across by fast obstacles which might not even be detected. This low frame rate therefore makes the system not suitable in the dynamic environment.

Together with 250-350ms latency, users with a speed of 1.5m/s can issue warnings after reaching 37-52cm, which allows minimal reaction time, which is especially problematic when dealing with users with a lower mobility.

The presence of several barriers during rapid succession evidently leads to overlapping or queuing of audio announcements that result in the delay of messages and confuse the users.

6.6.4 GPS Navigation Limitations

With a position of $\pm$5m GPS position in real world is not suitable in fine turn by turn guidance, when we are in close proximity of intersection. There is a possibility of the users getting an incorrect turn signal too soon or too late and go off course.

Accuracy of GPS is also reduced in city canyons where buildings act as sources of signal multipath, and clouded with trees, exactly where people need navigation assistance.

The accuracy of the magnetometer headingeast of ferromagnetic structures is degraded. To maintain heading drift with time, periodical recalibration is required, which is not feasible in active navigation.

Cold-Start GPS acquisition time of 35-60s adds a lot of waiting time before the system can actually start to navigate and this is a disadvantage to the practical use of this system in sudden navigation.

6.6.5 Critical Limitation: Absence of User Testing

The biggest weakness is that there is no testing locally undertaken with real visually impaired. The entire testing happened by sighted developers in simulated settings, user perception and trust, cognitive load and any real-world usability have not been validated, and this represents another crucial gap that would not allow making any conclusive statements about the effectiveness of the system as an assistive device.

There are no long-term reliability data of this kind. Testing was also restricted to brief sessions (3-4 hours at minimum). There is no definition of long-term stability weeks/months of daily activity.

There was no formal safety analysis or failure-mode analysis done or risk assessment. The system was not tested according to the medical device or assistive technology safety standards; the application to real use will need a full package of safety certification.

6.6.6 Hardware and Design Constraints

Monocular vision does not have a depth cue, range metering is done using ultrasonic distance sensors (not higher than 400 cm) or hand-of-thumb bound-box examination, which is quite inaccurate. Stereoscopic or depth-sensing camera would make spatial awareness, but is out of the existing budget.

The peripheral detection of an obstacle has a limitation in the fact that there is a narrow camera field of view where an object may not be detected at all at the periphery, but rather when it enters the central viewfield.

Raspberry Pi 4 is a basic computational bottleneck. Again, even aggressive optimisation will not cover; significant improvement will be made by switching to the Raspberry Pi 5 with a neural-processing unit, but it cannot afford this cost.

The design solution of mutually exclusive operation modes (obstacle detection XOR GPS navigation) restricts its usability; the user cannot obtain obstacle warnings and at the same time, travel to a destination and this poses an unacceptable trade-offs between safety and navigation.

6.7 Strengths and Contributions

Despite limitations, the system demonstrates notable strengths:

Offline Embedded Deployment: Complete offline object detection on resource-constrained hardware represents non-trivial achievement, requiring careful model selection, format conversion, and pipeline optimization.

Effective Sensor Fusion: Combining visual detection with ultrasonic distance measurement demonstrates effective strategy. Ultrasonic provides reliable distance ground truth compensating for monocular depth estimation limitations.

Extended Battery Life: 7-8 hour battery endurance meets practical usability requirements for all-day operation through effective power optimization.

Modular Architecture: Procedural modular design enables component-level testing, iterative development, and future extensibility. Clean module interfaces facilitate potential improvements.

Outdoor Daylight Utility: System performs adequately in outdoor daylight (75-80% detection), suggesting practical utility for outdoor walking in parks, open sidewalks, and well-lit environments.

Distance-Based Alerting: Ultrasonic-based proximity alerts function reliably, offering genuine collision avoidance value independent of visual detection.

Educational Value: Comprehensive documentation of embedded AI development from requirements through implementation provides reference for assistive technology research.

6.8 Future Recommendations

6.8.1 Hardware Upgrades

Processing Platform: Migration to Raspberry Pi 5 with Hailo-8 NPU or Google Coral TPU would enable larger, more accurate models (YOLOv5s, YOLOv8m) at 15-20+ FPS. Cost increase: $80-120.

Depth Sensing: Integration of Intel RealSense D435i or similar depth camera would provide accurate 3D obstacle mapping, eliminating monocular vision limitations. Cost: $150-200.

Thermal Management: Improved cooling solution (larger heatsink, higher-CFM fan, heat pipe design) to prevent throttling in hot environments.

RTK GPS: Upgrade to RTK-capable receiver (u-blox ZED-F9P) with base station correction would improve accuracy from $\pm$5m to $\pm$2cm for precise navigation. Cost increase: $100-150.

6.8.2 Software Improvements

Model Optimization: Advanced compression techniques (INT8 quantization, structured pruning, knowledge distillation) could improve inference speed 2-3$\times$ without significant accuracy loss, potentially enabling 10-15 FPS.

Object Tracking: Kalman filtering or SORT algorithm would reduce false positives, improve detection stability, enable velocity estimation for moving obstacles.

Spatial Audio: Stereo or binaural rendering would indicate obstacle direction through spatial positioning of synthesized speech for intuitive situational awareness.

6.8.3 Sensor Enhancements

IMU Integration: Accelerometer and gyroscope would detect walking speed, stride patterns, sudden stops, enabling adaptive warning timing.

LiDAR: Solid-state LiDAR (TF-Luna, TF-Mini) would provide faster, more accurate ranging with wider field of view and better weather performance than ultrasonic.

Infrared/Thermal: Low-cost infrared camera module would enable low-light and night-time operation, addressing critical current limitation.

6.8.4 User Experience

User Testing: Structured usability studies with visually impaired participants following ethics board approval, employing validated instruments (System Usability Scale, NASA-TLX).

Personalization: Adjustable audio verbosity, customizable alert thresholds, training mode with detailed feedback allowing user calibration.

Vibrotactile Feedback: Vibration motor feedback for non-audio alerts, reducing cognitive load while preserving ambient sound awareness.

Mobile Companion App: Smartphone application for system configuration, destination input, battery monitoring, usage analytics while maintaining offline core operation.

6.8.5 Safety and Reliability

Redundant Sensing: Cross-validation logic with multiple sensors (multiple ultrasonic angles, ultrasonic + LiDAR) to detect and handle sensor failures gracefully.

Self-Diagnostics: Startup self-test verifying camera, sensors, audio output, GPS signal before enabling operation.

Graceful Degradation: If camera fails, continue ultrasonic-only. If GPS fails, maintain obstacle detection. System never fails completely from single component fault.

Long-Term Testing: Extended durability testing (weeks to months) to characterize sensor drift, component wear, software stability.

6.9 Ethical Considerations

The application under consideration is a safety-critical application, and an instance of failure can lead to physical damage. Limitations, which were documented, were a 30 to 40% failure rate of indoor detections and complete inability in low-light conditions, which makes the prototype inapplicable as a main mobility assistor without significant performance enhancements and official safety certification.

Implementation of systems with small reliability poses the threat of unsuitable user confidence. False confidence can develop in the end-users and the dependability on more

traditional methods of mobility (white-cane sweeping and echolocation) can be minimized creating a greater risk in case of system failure.

Any assistive technology, which is to be used in people with disabilities, must undergo strict validation, including acceptance of the proposed technology by ethics boards, procedures of informed consent, risk analysis, pilot testing with supervision, longitudinal outcome measures, and formal safety certifications before its application.

Current Status: This system should be regarded as proof-of-concept research prototype demonstrating technical feasibility of embedded AI-powered assistive mobility devices. It is *not* production-ready and should not be deployed for actual use by visually impaired individuals without substantial improvements and comprehensive validation.

6.10 Conclusion

This chapter presented comprehensive evaluation through systematic testing, quantitative performance measurement, and critical analysis of limitations. Testing encompassed unit, integration, and system-level validation across seven use cases.

Key Findings:

- Detection confidence 50-55%, indoor detection 60-70%, outdoor 75-80% significantly below safety-critical requirements
- System performance 4-8 FPS with 250-350ms latency approached but inconsistently met real-time requirements
- Ultrasonic distance measurement reliable, validating sensor fusion design
- Battery endurance 7-8 hours met operational requirements
- GPS navigation technically feasible but $\pm$5m accuracy insufficient for precise guidance
- Thermal throttling and low-light performance degradation represent critical constraints

Critical Assessment: The system demonstrates technical feasibility of offline embedded object detection for assistive mobility but falls short of performance levels required for reliable practical deployment. The fundamental accuracy-speed trade-off imposed by budget-constrained hardware represents an inescapable limitation.

Most Significant Gap: Complete absence of testing with actual visually impaired users. Without user validation, no claims about effectiveness, usability, or safety as an assistive device can be made. This prototype must be regarded as technical demonstration, not validated assistive technology.

Value Delivered: (1) Demonstration of offline embedded AI on resource-constrained hardware, (2) effective sensor fusion combining vision and ultrasonic, (3) comprehensive trade-off documentation in assistive technology context, (4) honest evaluation methodology transparently acknowledging weaknesses and strengths.

Deployment Path: Substantial improvements in hardware, sensors, thermal management, and most critically, comprehensive user-centered testing and validation are required before practical deployment consideration. The documented recommendations provide roadmap for future iterations.

This evaluation demonstrates scientific maturity through transparent self-assessment acknowledging gaps between prototype and production-ready assistive device. This honest appraisal establishes foundation for future improvement efforts.

Chapter 7

Conclusion

7.1 Summary of Work

The present Final-year project explored the design and development process of a Smart Object Detection Blind Stick, which is a portable assistive system that was designed to enhance the mobility, safety, and autonomy of the visually impaired consumer. The key goal was to allow real-time environmental perception and audio feedback to ease movement of the sensory perception of the users without the need to have any external support or connection to the cloud.

This solution came to pass by combining computer vision and deep learning with embedded processing in one handheld platform. An YOLOv8n was trained on both open datasets and self-curated data bases and then deployed on a Raspberry Pi 4 in order to make real-time inferences. The range of obstacles was estimated by the use of ultrasonic distance sensing, the offline location awareness was undertaken by the use of a NEO 6M GPS module and directional orientation through the help of a QMC 5883 magnetometer. All these subsystems provided a complete offline environmental detection, navigation and audio feedback mechanism.

The actual software development was done in Python, with the use of ONNX Runtime to accelerate inference, OpenCV to process images, and pyttsx3 to voice synthesize. The findings show that real-time, practically useful, AI-assisted, navigation on cost-efficient embedded platforms is possible, and this approach is a cost-efficient and scalable solution to access technologies.

7.2 Achievements

The project was completed through a number of major milestones. The finely-tuned YOLOv5n model achieved an average precision of more than 55% in a range of classes of obstacles, such as vehicles, poles, benches, and stairs. On the Raspberry 4, with the system running, the frame rate stayed in the range of 4-8 frames per second. The ultrasonic

component was used to measure the distance to obstacles by a very small margin of 4 meters, and the GPS plus the compass image were capable of producing a good offline position and orientation information.

The feedback mechanism of audio-integrated applied the text-to-speech, which was offline to state the proximal dots and offer step-by-step distance cues to enhance reliability and privacy. The presence of GPS and compass data provision enabled the gadget to offer the initial offline navigation features on the elementary level, therefore, moving towards the spheres of autonomous mobility assistance.

7.3 Key Findings

A number of lessons were drawn through experimentation and outdoor experimentation. Raspberry Pi 4 showed strong real-time AI inference, ensuring that embedded AI hardware can be used to run objective detection networks with an efficient configuration. The combination of ultrasonic, GPS, and compass data provided an experience of a context-driven navigation, closer to an approximation of obstacle avoidance than an overview of directional instructions.

7.4 Limitations

In the phase of evaluation, practical pitfalls were also identified such as overlapping audio alerts in cases where more than one detection was made and reduced visibility in low-light situations. Although these limitations are present, participants stated that they felt better and more confident and were able to gain spatially using the device. Offline processing has been particularly useful as it ensured full independence ensuring the removal of latency of network dependency and improvements in data security.

Despite the functional purposes achieved through the prototype, a number of constraints were still there. The Raspberry Pi 4 had a limited processing power due to the size of the board, which made it slow in making inferences and inaccurate at detecting a target, like barriers in the distant background or ones partially hidden by fog. Power draw was moderate thus restricting prolonged working. Reflective or absorbent surfaces influenced the ultrasonic accuracy negatively, and there were times the audio system generated overlapping cues, hence there was a necessity of better message prioritization. Although GPS and compass were being used to provide basic offline navigation, turn-by-turn guidance and route planning was not done yet because the emphasis was laid on the real-time object detection and obstacle avoidance.

7.5 Future Work

Future iterations of this project can explore several major advancements:

- **AI Hardware Acceleration:** Incorporating dedicated inference accelerators such as Google Coral TPU, Intel Movidius, or Hailo AI modules could significantly enhance performance, enabling larger models (e.g., YOLOv8m) to run in real time on embedded platforms.
- **Camera Head Stabilization:** A motorized gimbal-like mount can be introduced to keep the camera and sensors level and forward-facing, independent of cane movement. This stabilization mechanism would maintain consistent field of view even when the user's hand motion varies, similar to handheld camera stabilizers.
- **Enhanced Navigation:** Extending GPS and compass integration into full offline route guidance with spoken turn-by-turn directions would make the system a complete navigation aid, not just an obstacle detector.
- **Advanced Audio Interface:** Introducing directional and spatialized audio cues (stereo panning, pitch-based proximity indicators) could make obstacle alerts more intuitive and immersive.
- **Battery Optimization:** Incorporating dynamic power management, model quantization, and low-power modes could significantly increase operational time.
- **Blind Assistant Bot (BLAB):** Integrating a small offline chatbot module (BLAB) for user interaction could provide voice-based commands, FAQs, and contextual assistance, allowing users to ask questions like "Where am I?" or "What's nearby?" through natural speech.

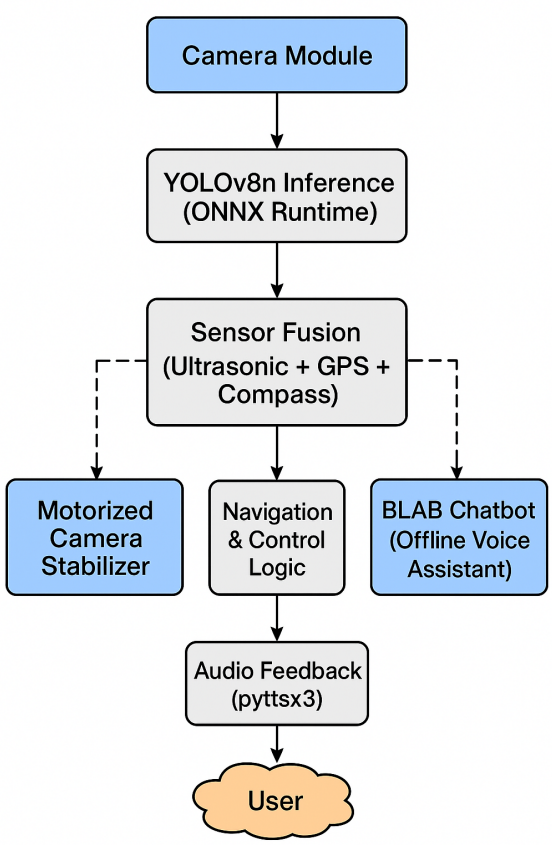

Figure 7.1: Proposed Future Work Directions for the Smart Object Detection Blind Stick

These directions will help evolve the prototype from a functional assistive stick into a robust, intelligent mobility companion tailored for real-world use by visually impaired individuals.

7.6 Final Remarks

To summarize, the present project has proven the possibility to combine computer vision and sensor fusion with offline AI inference into a small, portable, and affordable device. The Smart Object Detection Blind Stick provides accurate real-time obstacle detection and navigation feedback, which is a serious improvement in the available assistive technology. Although hardware limits were acted upon, the established design provides even more strong background to the future innovation where modularity and scalability is assured to be maintained moving forward as embedded AI hardware and energy-efficient models keep advancing. Finally, the work can be considered as a demonstration of a concept in which cheap, off-line, and smart options can significantly enhance the self-reliance, security, and trustworthiness of blind users.

Appendix A

User Manual

A.1 Introduction

In this chapter, the author explains the operations of a visually impaired user with the Smart Navigation Stick that has been created within the frames of this project. This user manual is meant to describe how the user will interact with the system in a simple yet practical way. The attention is paid to the application of the system to situations in everyday life instead of internal hardware or software implementation.

The Smart Navigation Stick is deemed to be an aid to mobility, which provides audio guided directions to detect obstacles and guide a user out of doors. It is not meant to replace, but help supplement the traditional white-cane tools and user perceptual awareness.

A.2 Intended Users and Usage Environment

The main consumers of the Smart Navigation Stick are the visually impaired people, who need some means as they walk on their own. It is designed to work without visual feedback with complete reliance on physical controls and audio feedback.

The device can be used in two main environments:

- **Indoor environments**, where the system assists with detecting nearby obstacles such as furniture or people.
- **Outdoor environments**, where GPS-based navigation provides direction guidance to a selected destination.

The system is intended for use at normal walking speed and performs best in reasonably well-lit areas and open outdoor spaces.

A.3 User Interaction and Control Mechanism

The Smart Navigation Stick has a simple control system in order to provide convenience to the users who have visual impairment. Physical switches and audio feedback are used to all communicate with the system. There does not need to be any screen or visual interface.

User input is provided through:

- Physical switches mounted on the stick for system activation and mode selection.
- Voice input for specifying navigation destinations.

System output can only appear in the form of verbal messages and alarming sounds so that the user can still be informed and somehow be aware of the environment noises.

A.4 Master and Slave Switch Operation

Two physical switches are used called the **Master switch** and **Slave switch** to regulate the overall operation and the mode choice. This design also allows a user to navigate through the system without complicated instructions.

The **Master switch** acts as the main power and control switch for the system:

- When **Master** is turned **OFF**, the system is in the standby state and nothing is operational.
- In case the Master switch is set to the **ON** position, the system is turned on and is now ready to run in the mode selected.

The **Slave switch** is used to select the operating mode while the Master switch is ON:

- In the case of the Slave switch turned to the **OFF** position, the system is in Object Detection Mode.
- In case the Slave switch is set to the **ON** position, the system is used in the Navigation Mode.

Switching between modes The user can change between modes by toggling the Slave switch, with the Master switch in the ON position. Whenever a mode is switched on, audio feedback is given so that the user can make a check on the status of the system.

A.5 Operating Modes Overview

The Smart Navigation Stick has two major modes namely Object Detection Mode and Navigation Mode. There is only a single mode, but it can be used simultaneously. The

modes are all used differently and offer certain audio feedback that is unique to the requirements of the user.

Object Detection Mode is focused on the immediate surroundings whereas Navigation Mode helps the user to get to a destination through directional guidance.

A.6 Object Detection Mode

Object Detection Mode facilitates the user by recognizing objects and obstacles around that are in real time. This mode is more useful in the indoors and where the alertness to the nearby dangers is paramount.

In order to use this mode, the user switches the Master switch ON and makes sure that the Slave switch is off. When activated, the system will initiate a scan of the area before the user and inform them through a spoke output about objects detected.

Objects are recognized by the system like people, furniture, or vehicles, which will be relayed in terms of their relative position like in front, on the left, and on the right. The distance is presented in a straightforward language e.g. very close or nearby and then the user can respond accordingly.

It is recommended to the user to keep the stick slightly in front of the body as one walks and move the stick normally. Limitations are also placed on repetition of announcements so as to avoid overproduction of audio thereby causing distraction.

A.7 Navigation Mode

Navigation Mode provides audio turn-by-turn directions that help the user reach the goal of a destination of choice. This mode was meant to be used outdoors whereby we get the GPS signals.

In order to enable Navigation Mode, the user switches on the Master switch and the Slave switch into ON position. The system switches on the GPS and will then require the user to say the name of the destination required. Once the verbal input is confirmed, the system starts navigation.

The advice will be provided by just giving direct directions like straight on, left or right, or sharper turn where necessary. It also gives distance updates when the customer is nearing destination.

A.8 Audio Feedback and Alerts

The main way of interacting between the user and the system is audio feedback. Spoken messages are used to announce perceived objects, give directions, as well as affirm human actions.

When there are barriers found around, signals and notifications of danger are emitted. The speed or frequency with which the alerts appear indicate proximity, allowing the user to respond promptly and securely.

This form of interaction that is audio ensures that users, who are visually impaired, are accessible as well as allowing them to be aware of sounds in the environment.

A.9 Safety and Usage Guidelines

The Smart Navigation Stick is designed as a support mechanism and it must always be used together with conventional mobility style. Users are also advised to be attentive to the sounds that are around them like sound of a traffic and voices.

The system has shortcomings and can not sense any changes on the ground level like stairs, holes and slippery floors. Lighting conditions as well as weather can also influence performance. They should be careful especially when one is in an unfamiliar or congested place.

In case the system malfunctions or acts in a way that it is not expected, the user should stop moving as well as turn to assistance in case the same is necessary.

A.10 System Limitations from a User Perspective

On the side of the users, the accuracy of navigation can be inconsistent with the quality of the GPS signals and the environmental factors. One cannot have GPS-based navigation inside.

In low light, there are chances of deterioration in the accuracy of object detection. Continuous usage is also limited by the battery capacity should be taken into account before use is to be made significantly long.

Knowing these limitations helps users to use the system wastely as well as realistically.

References

[1] World Health Organization. World report on vision. *WHO Publications*, 2023.

[2] Roberto Manduchi and James M. Coughlan. Assistive technology for blindness and low vision. *CRC Press Handbook of Visual Impairment*, 2012.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[4] Glenn Jocher. Yolov8: The next generation of yolo models. https://github.com/ultralytics/ultralytics, 2023. Accessed: 2025-10-15.

[5] Raspberry Pi Foundation. Raspberry pi 5 documentation. https://www.raspberrypi.com/documentation/, 2023. Accessed: 2025-10-15.

[6] H. Alami, N. El Akkad, and A. El Fazziki. A survey on assistive technologies for visually impaired individuals. *Procedia Computer Science*, 175:225–232, 2020.

[7] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[8] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[9] Ultralytics Team. Yolov8 — ultralytics documentation. https://docs.ultralytics.com/models/yolov8/, 2023.

[10] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Y. H. Lee and G. Medioni. Rgb-d camera based wearable navigation system for the visually impaired. *Computer Vision and Image Understanding*, 149:3–20, 2016.

[12] Bing Li, J. Pablo Muñoz, Xuejian Rong, Qingtian Chen, Jizhong Xiao, Yingli Tian, and Aries Arditi. Vision-based mobile indoor assistive navigation aid for blind people. *IEEE Transactions on Mobile Computing*, 18(3):702–714, 2019.

[13] Zhuo Chen, Xiaoming Liu, Masaru Kojima, Qiang Huang, and Tatsuo Arai. A wearable navigation device for visually impaired people based on the real-time semantic visual slam system. *Sensors*, 21(4):1536, 2021.

[14] R. K. Katzschmann, B. Araki, and D. Rus. Safe local navigation for visually impaired users with a time-of-flight and haptic feedback device. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[15] Márton Sipos, Attila Vörös, and Laszlo Toth. Sensor fusion for smart assistive devices for the visually impaired. *Sensors*, 22(4):1499, 2022.

[16] Hui Chen, Kuo-Lun Huang, Wei-Hsiang Chang, and Hsien-Kai Lin. A low-cost wearable ultrasonic smart cane for visually impaired people. *IEEE Access*, 9:34784–34794, 2021.

[17] Muhammad Ali, S. Zafar, and M. Tariq. Real-time object detection mobile app using yolov5 for visually impaired assistance. In *IEEE International Conference on Emerging Technologies (ICET)*, 2022.

[18] A. Rahman, M. Hossain, S. Shuvo, and S. Hasan. Efficient obstacle detection on jetson nano using yolov5 and tensorrt. In *IEEE International Conference on Intelligent Systems (IS)*, 2022.

[19] M. Hussain, R. Khan, and A. Farooq. A comprehensive review of yolo-based object detection models for edge devices. *Applied Sciences*, 14(2):421, 2024.

[20] J. Mungdee, P. Kongsilp, and K. Chantarachit. Development of a low-cost smart cane with object detection for the visually impaired. *Sensors*, 24(1):112, 2024.

[21] S. Soro, Pete Warden, and Daniel Situnayake. Tinyml: Machine learning for embedded and edge devices – a review. *IEEE Access*, 9:48924–48940, 2021.