GMR Institute of Technology
An Autonomous Institute Affiliated to JNTU-GV

GMRIT
Training Tomorrow's Engineers Today

ENHANCING MOBILITY TROUGH AUGMENTED REALITY FOR VISUALLY IMPAIRED

A project report submitted in partial fulfilment of the requirement

for the award of degree of

BACHELOR OF TECHNOLOGY

In

CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

Submitted by

V. NAVYA SRI	**(21341A4258)**
CH. SAI SARAYU	**(21341A4207)**
M. MUDDU KRISHNA YADAV	**(22345A4204)**
K. SRIDHAR	**(21341A4224)**

Under the esteemed guidance of

Mrs. P. Madhavi

Assistant Professor

Dept. of CSE (Artificial Intelligence & Machine Learning)

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTU-GV, Vizianagaram

GMR Nagar, Rajam – 532127

Andhra Pradesh, India

April 2025

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTU-GV

GMRIT
Training Tomorrow's Engineers Today

Department of CSE (Artificial Intelligence & Machine Learning)

CERTIFICATE

This is to certify that the thesis entitled **ENHANCING MOBILITY THROUGH AUGMENTED REALITY FOR VISUALLY IMPAIRED** submitted by **V. Navya Sri (21341A4258), Ch. Sai Sarayu (21341A4207), M. Muddu KrishnaYadav (22345A4204), K. Sridhar (21341A4224)** has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology** in **CSE (Artificial Intelligence & Machine Learning)** of **GMRIT, Rajam** affiliated to **JNTU-GV Vizianagaram** is a record of bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Signature of the Supervisor	Signature of the H.O.D
Mrs. P. Madhavi	**Dr. K Srividya**
Assistant Professor	Associate Professor and Head
Department of CSE (AI&ML)	Department of CSE (AI&ML)
GMRIT, Rajam.	GMRIT, Rajam.

The report is submitted for the viva-voce examination held on …………………

Signature of Internal Examiner	Signature of External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to my guide **Mrs. P. Madhavi,** Assistant Professor, Department of CSE- Artificial Intelligence & Machine Learning of wholehearted and invaluable guidance throughout the project work. Without her sustained and sincere effort, this project work would not have taken this shape. She encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. K. Srividya**, Associate Professor & HOD, Department of CSE-Artificial Intelligence & Machine Learning for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr. C. L. V. R. S. V. Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank all the faculty members and the non-teaching staff of the Department of CSE-Artificial Intelligence & Machine Learning for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all our friends and family members for their continuous help and encouragement.

V. Navya Sri	**21341A4258**
Ch. Sai Sarayu	**21341A4207**
M. Muddu Krishna Yadav	**22345A4204**
K. Sridhar	**21341A4224**

ABSTRACT

Navigating safely and independently remains a critical challenge for visually impaired individuals, highlighting the need for innovative solutions. By leveraging advanced technologies such as object detection, edge detection, augmented reality (AR), scene understanding, audio interaction, and path planning, the system provides real-time feedback and guidance, enabling users to traverse complex environments with confidence and precision. The project focuses on developing an AR-based navigation system designed for visually impaired individuals, combining advanced technologies to deliver real-time assistance. The system employs object detection and edge detection to identify and track obstacles and boundaries, preventing collisions. Integrated AR technology, developed using Unity and AR Foundation, overlays virtual information onto real-world environments, enhancing spatial awareness with auditory cues instead of visual markers. Scene understanding dynamically interprets the environment and adjusts navigation accordingly, while path planning recalculates efficient and safe routes in real time. To ensure accessibility, the system relies on audio interaction, providing spoken instructions and immersive sound cues tailored to the user's position. This innovative approach empowers visually impaired users with a reliable navigation tool, fostering independence and enhancing mobility across diverse environments.

Keywords: AR-based navigation, Visually impaired assistance, Object detection, Edge detection, Path planning, Audio feedback , Unity engine.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Navigating safely and independently is a major challenge for visually impaired individuals, as traditional aids like white canes and guide dogs provide limited assistance in dynamic environments. These tools often fail to detect overhead obstacles, moving hazards, or sudden environmental changes, making independent mobility difficult.This project introduces an Augmented Reality (AR)-based navigation system designed to enhance mobility through real-time object detection, scene understanding, and interactive audio guidance. Unlike conventional methods, it leverages AI and computer vision to actively interpret surroundings and assist users with step-by-step voice instructions..With deep learning-based object detection, the system recognizes and classifies obstacles such as furniture, vehicles, and staircases, allowing users to navigate safely even in complex environments. LiDAR and depth sensing create a 3D map, refining navigation precision. Unlike traditional GPS tools, this system dynamically adapts to real-world changes, analyzing spatial relationships for seamless mobility.Beyond personal navigation, the system integrates with smart city infrastructure, allowing interaction with IoT-enabled crosswalks, traffic signals, and public transit systems. By combining computer vision, augmented reality, and AI, this project offers a real-time navigation assistant, fostering independence and redefining mobility for a more inclusive and accessible future.

Navigating daily environments can be a significant challenge for visually impaired individuals, often limiting their independence and quality of life. Traditional mobility aids such as white canes or guide dogs, while useful, have limitations when it comes to detecting obstacles at a distance or providing contextual environmental awareness. In recent years, technology has offered new possibilities to bridge this accessibility gap. Among the most promising innovations is augmented reality (AR), which combines real-world inputs with digital overlays to provide enriched sensory information. This project explores how AR can be harnessed to enhance mobility for visually impaired individuals, offering a more intuitive, informative, and safe navigation experience.

The goal of this project is to design and implement an AR-based system that interprets the surrounding environment and conveys it to the user through auditory or tactile feedback. By using computer vision and real-time object detection, the system identifies obstacles, paths, and key landmarks, translating them into actionable cues. This not only supports users in avoiding hazards but also provides spatial awareness that conventional tools cannot offer. With smartphones and wearable AR devices becoming increasingly accessible, integrating AR into assistive technologies is both feasible and scalable.

This solution leverages advancements in artificial intelligence, image recognition, and AR software to create a user-centric tool tailored to the unique needs of visually impaired users. Key considerations include ease of use, accuracy of detection, latency, and the form in which information is delivered to the user. The system is designed with a modular approach, allowing for future enhancements such as GPS integration, crowd-sourced updates, and support for indoor navigation. Moreover, usability testing with visually impaired participants is a critical component of the project to ensure the technology aligns with real-world needs and expectations.

Ultimately, this project aims to contribute to the growing field of inclusive technology by providing a prototype that demonstrates how AR can empower visually impaired individuals to move confidently and independently. By addressing technical challenges and focusing on human-centered design, this work aspires to pave the way for more innovative and empathetic solutions in the assistive tech space. The long-term vision is not only to enhance mobility but to foster greater autonomy, safety, and dignity for those living with visual impairments.

1.1 Problem Statement

To develop a scalable and intelligent augmented reality-based mobility aid for visually impaired individuals that enhances spatial awareness, delivers real-time navigational cues, and automates obstacle detection to support safer, more independent, and confident navigation in diverse environments.

1.2 Objectives

- Enhance Mobility: Develop a system that enables visually impaired individuals to navigate complex environments safely and independently.
- Real-Time Assistance: Provide real-time feedback and guidance to users through auditory cues and haptic feedback, improving navigation accuracy.

- Adaptability: Provide adaptive guidance by processing data from cameras and sensors, allowing the system to respond to environmental changes for seamless navigation.
- Dynamic Scene Understanding: Implement scene understanding to adapt the navigation system in response to changes in the environment, such as shifting obstacles or reconfigured spaces.
- Obstacle Detection and Avoidance: Utilize object detection and edge detection technologies to identify and track obstacles such as walls and stairs, preventing collisions.

1.3 Existing System Drawbacks

- Limited Real-Time Feedback
- Limited Environmental Understanding
- High Cost and Limited Accessibility
- Reliance on Static Data

1.4 Proposed System Advantages

- Accurate Obstacle Detection
- Enhanced Spatial Awareness with AR
- Dynamic Scene Understanding
- Improved Autonomy and Safety
- Intuitive Audio

2. LITERATURE SURVEY

[1] Zhang, X., Yao, X., Zhu, Y., & Hu, F. (2019). An ARCore based user centric assistive navigation system for visually impaired people. Applied Sciences, 9(5), 989. The paper highlights the significant challenges faced by visually impaired individuals, emphasizing their reliance on traditional aids like white canes and guide dogs, which have limitations in various situations. The literature indicates that existing systems often utilize audio instructions for navigation, which can lead to confusion and errors due to latency and limited information. Research by Ahmetovic and Guerreiro points out the drawbacks of turn-by-turn audio instructions, suggesting that they may hinder effective navigation and increase recovery times when users deviate from their paths. The proposed system in this paper, ANSVIP, aims to address these issues by integrating ARCore for scene understanding and employing haptic feedback for real-time navigation guidance, thus providing a more intuitive user experience. The literature also emphasizes the need for multimodal feedback mechanisms to improve user interaction and navigation accuracy, which the proposed system seeks to implement.

[2] Yang, G., & Saniie, J. (2017, May). Indoor navigation for visually impaired using AR markers. In 2017 IEEE International Conference on Electro Information Technology (EIT) (pp. 1-5). IEEE. The indoor navigation system assists visually impaired individuals using AR markers and computer vision, improving accuracy where satellite signals are unavailable. It offers high positioning accuracy, surpassing GPS in indoor environments, and enhances user independence through smartphone-based guidance. Challenges include measurement errors due to occlusion and marker registration, as well as reliance on properly placed and visible AR markers. The dataset consists of video clips with 576 frames capturing AR marker positions for performance evaluation. The system uses AR markers and a camera to determine relative poses and positions within a defined coordinate system.

[3] Elgendy, M., Herperger, M., Guzsvinecz, T., & Lanyi, C. S. (2019, October). Indoor navigation for people with visual impairment using augmented reality markers. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) (pp. 425-430). IEEE. Various technologies have been explored for aiding people with visual impairment (PVI) in indoor navigation, including Wi-Fi, Bluetooth, RFID, NFC, and tags. Each solution varies in signal type, positioning method, and accuracy. The paper emphasizes the use of computer vision (CV) tags, particularly ArUco markers, which are found to be more effective than QR codes for navigation purposes. This comparison highlights the

importance of selecting the right type of marker for optimal results. The authors developed a prototype navigation system that utilizes CV tags to identify the location of PVI and guide them to their destination using the shortest path. This system is designed to be user-friendly, allowing PVI to input their destination via voice commands . The literature indicates that CV tags, such as ArUco markers, are robust and easy to use, making them suitable for indoor navigation. The ability to detect these markers from a distance is a significant advantage over QR codes .

[4] Ouali, I., Sassi, M. S. H., Halima, M. B., & Ali, W. A. L. I. (2020). A new architecture based ar for detection and recognition of objects and text to enhance navigation of visually impaired people. Procedia Computer Science, 176, 602-611. Introduces an augmented reality (AR) architecture to assist visually impaired individuals in detecting and recognizing objects and text, focusing on drug identification and navigation in unfamiliar environments. Enhances daily life by providing real-time information through a mobile application, enabling users to quickly identify objects and text for improved navigation and independence. Offers real-time interaction and a user-friendly interface but faces challenges such as smartphone camera positioning issues for reading pill names and the need for improved recognition accuracy. Utilizes two datasets: one for object recognition (drug images) and another for text recognition (drug names), both stored in the AR engine database for efficient retrieval. Employs the Vuforia engine for object and text detection, with future enhancements proposed through ontology-based recognition to improve accuracy across different writing styles and complex backgrounds.

[5] Lo Valvo, A., Croce, D., Garlisi, D., Giuliano, F., Giarré, L., & Tinnirello, I. (2021). A navigation and augmented reality system for visually impaired people. Sensors, 21(9), 3061. ARIANNA+ is an augmented reality system that helps visually impaired users navigate without physical markers, using object recognition and feedback mechanisms. It allows flexible navigation and provides automatic guidance through haptic, speech, and sound feedback. The system's performance may be affected by lighting conditions, and real-time processing can strain smartphone battery life. It uses the COCO dataset for training CNNs in object recognition and employs EKF and WMA for user tracking and motion estimation.

[6] Zhang, H., & Ye, C. (2016, December). An indoor navigation aid for the visually impaired. In 2016 IEEE international conference on robotics and biomimetics (ROBIO) (pp.

467-472). IEEE. Presents a 6-DOF pose estimation method and an indoor wayfinding system for visually impaired individuals using a two-step SLAM process to improve navigation accuracy and efficiency. Employed a 2-step Graph SLAM method that integrates floor plane and wall line extraction for pose estimation .Utilized RANSAC-based algorithms for plane and line extraction to improve computational efficiency. Effectively reduces cumulative pose error for more accurate navigation and achieves a faster runtime (59.4 ms per frame) compared to existing planar SLAM methods, enhancing real-time performance. The method may still struggle in environments with low inlier data, potentially affecting wall plane extraction accuracy. The experiments utilized point cloud data collected from indoor environments, specifically a hallway in the ETAS building.

[7] Joseph, S. L., Zhang, X., Dryanovski, I., Xiao, J., Yi, C., & Tian, Y. (2013, October). Semantic indoor navigation with a blind-user oriented augmented reality. In *2013 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 3585-3591). IEEE. A wearable navigation system uses augmented reality (AR) and metric localization to assist visually impaired users with haptic feedback and voice guidance for independent travel. Real-time navigation assistance improves safety, with visual odometry and landmark localization correcting pose estimation errors for better accuracy. Pose estimation errors may accumulate over time, impacting navigation precision, and reliance on semantic floor plans limits usability in environments without such maps. No specific dataset is mentioned, but a global 3D feature model is used for alignment and mapping. Visual odometry with RGB-D cameras, a Kalman Filter framework for managing uncertainty, and a human motion model predict paths while avoiding obstacles.

[8] Abidi, M. H., Siddiquee, A. N., Alkhalefah, H., & Srivastava, V. (2024). A comprehensive review of navigation systems for visually impaired individuals. *Heliyon*.. The paper reviews navigation systems for visually impaired individuals, exploring technologies like smartphone-based solutions, wearable devices, and non-visual data systems to enhance mobility and independence. Continuous monitoring through wearable systems provides reliable navigation support in diverse environments, improving accessibility and safety for users. High costs, potential gait alterations with wearable devices, and challenges in real-world conditions affect system effectiveness. Visual imagery systems may struggle in low light, while non-visual systems may lack detail. Navigation assistance solutions are categorized into Visual Imagery Systems, Non-Visual Data Systems, Map-Based Solutions,

and 3D Sound Systems, each analyzed for effectiveness and technological advancements. The study uses a comprehensive review methodology, evaluating various navigation technologies and the role of smartphones and sensors in improving accessibility for visually impaired individuals.

[9] Fathi, K., Darvishy, A., & Van de Venn, H. W. (2022, September). Augmented reality for the visually impaired: Navigation aid and scene semantics for indoor use cases. In *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)* (pp. 1-6). IEEE. Various technologies have been developed to assist the visually impaired (VI), including sonar-based systems and computer vision techniques for simultaneous localization and mapping (SLAM) . However, these methods often have high computational demands and may not be practical for real-time navigation . Some researchers have utilized Computer Aided Design (CAD) models for indoor navigation, but creating accurate models can be time-consuming and may not include dynamic obstacles like furniture. This limitation highlights the need for more adaptable solutions.The integration of game engines like Unity into assistive systems has shown promise. These engines can enhance user experience through immersive auditory feedback, which is crucial for reducing cognitive load. The proposed framework in the paper emphasizes the importance of providing semantic scene information through auditory means, which can significantly aid VI individuals in unfamiliar environments .

[10] Xie, T., & Seals, C. (2023, January). Design of Mobile Augmented Reality Assistant application via Deep Learning and LIDAR for Visually Impaired. In *2023 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 1-4). IEEE. The paper highlights that over 2.2 billion people globally suffer from vision impairment, which significantly impacts their daily lives and increases anxiety and depression levels due to challenges like finding lost items indoors . Current assistive devices often require additional purchases and involve cumbersome operations, such as inputting extensive item information before searching, which can be particularly challenging for visually impaired users . The NAAD system integrates Augmented Reality (AR), LIDAR, and Deep Learning technologies to enhance navigation and object identification for visually impaired individuals. This approach aims to provide a more flexible and user-friendly solution compared to existing devices. The design of the NAAD system focuses on user experience, ensuring that it does not require network connectivity and can be improved based on user feedback. This aspect is crucial for

enhancing the overall effectiveness and safety of the navigation process. The system has shown promising results, with an average utilization rate of over 98% and a 100% success rate in helping participants find target objects during trials.

[11] Real, S., & Araujo, A. (2019). Navigation systems for the blind and visually impaired: Past work, challenges, and open problems. *Sensors*, *19*(15), 3404. The paper provides a historical overview of navigation systems for the blind and visually impaired (BVI), tracing the evolution from early "Electronic Travel Aids" to modern systems utilizing artificial vision. It emphasizes the importance of user-centered design, addressing criticisms of past approaches and highlighting the need for designs that better meet the needs of BVI users. The survey includes a review of various technical resources that have not been fully exploited, such as remote processing techniques and simultaneous localization and mapping (SLAM) . The paper discusses the integration of smartphones and wearable technology, which are becoming increasingly prevalent in the daily lives of BVI individuals, as potential platforms for advanced navigation solutions. It also identifies challenges and open problems in the field, suggesting that many unmet needs are being addressed across different research areas, including indoor positioning and spatial cognition. Overall, the article aims to provide a multidisciplinary perspective on navigation systems, encouraging developers to leverage both classic and innovative designs for future solutions

[12] Du, P., & Bulusu, N. (2021, October). A smartphone-based mobility assistant using depth imaging for visually impaired and blind. *Applied Sciences*, *12*(6), 2802. The paper addresses the increasing number of visually impaired and blind (VIB) individuals, projected to rise to over 550 million by 2050, highlighting the urgent need for effective assistive technologies. Current assistive devices often lack portability and flexibility, making them inconvenient for users. This has spurred research into more adaptable solutions. The evolution of assistive technologies has led to the development of electronic travel aids, but many still face challenges related to usability and hardware limitations. The study emphasizes the importance of cognitive and spatial awareness for VIB individuals, which traditional devices do not adequately address. The integration of smart sensing technologies, such as depth cameras in smartphones, is proposed as a solution to enhance mobility and navigation for VIB users. Previous research has focused on various aspects of assistive technology, including user interface design and the need for inclusive features that cater to the unique challenges faced by VIB individuals .

[13] See, A. R., Sasing, B. G., & Advincula, W. D. (2022). A smartphone-based mobility assistant using depth imaging for visually impaired and blind. *Applied Sciences*, *12*(6), 2802. The paper addresses the increasing number of visually impaired and blind (VIB) individuals, projected to rise to over 550 million by 2050, highlighting the urgent need for effective assistive technologies. Current assistive devices often lack portability and flexibility, making them inconvenient for users. This has spurred research into more adaptable solutions. The evolution of assistive technologies has led to the development of electronic travel aids, but many still face challenges related to usability and hardware limitations. The study emphasizes the importance of cognitive and spatial awareness for VIB individuals, which traditional devices do not adequately address. The integration of smart sensing technologies, such as depth cameras in smartphones, is proposed as a solution to enhance mobility and navigation for VIB users. Previous research has focused on various aspects of assistive technology, including user interface design and the need for inclusive features that cater to the unique challenges faced by VIB individuals .

[14] Sokolov, A., Avrunin, O., Selivanova, K., & Shushliapina, N. (2024, October). Application of Augmented Reality Technologies for Determining Distances in Navigation System for the Blind. In *2024 IEEE 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)* (pp. 530-533). IEEE. The paper discusses how AR enhances human perception by integrating computer analysis and visualization, which is crucial for tasks like distance measurement in navigation systems for the blind . It highlights the capabilities of AR libraries such as ARCore and ARKit, which not only augment reality but also facilitate real-world interaction and analysis, essential for developing navigation aids . The study emphasizes the importance of accurately determining distances to objects, which is vital for creating effective navigation systems for visually impaired users. The paper includes experimental verification of the distance measurement methodology, comparing the AR system's measurements with traditional tape measures to assess accuracy .The findings suggest that ARCore can be tailored for low-speed mobility scenarios, making it suitable for visually impaired users, with an acceptable error margin of up to 7.5% .

[15] Troncoso Aldas, N. D., Lee, S., Lee, C., Rosson, M. B., Carroll, J. M., & Narayanan, V. (2020, October). Aiguide: An augmented reality hand guidance application for people with visual impairments. In *Proceedings of the 22nd International ACM SIGACCESS Conference*

on Computers and Accessibility (pp. 1-13). Current applications assist users in various tasks like identifying products, reading documents, and recognizing faces. However, they often lack the ability to provide the relative position of objects, which is crucial for tasks requiring physical interaction, known as the "last meter problem". Some systems, like ThirdEye, utilize specialized hardware to recognize grocery items in real-time. However, these systems face limitations such as bulkiness and dependency on internet connectivity. Other systems, such as those proposed by Bigham et al., rely on crowd-sourced information for guidance, which can be inconsistent and dependent on external factors like image quality and internet access. AIGuide stands out as a self-contained smartphone application that does not require external hardware or internet connection. It provides real-time guidance using augmented reality frameworks, making it accessible and practical for users. The study highlights user preferences for feedback modes, revealing that while sound feedback was preferred by some, haptic feedback was also valued for its practicality in noisy environments or for users with hearing impairments

2.1 COMPARISION TABLE OF LITERTURE SURVEY

Table 1: comparision table on literature survey

S.No	Paper	Year	Objectives	Limitations	Advantages	Performance Metrics
1	An ARCore based user centric assistive navigation system for visually impaired people	2019	Develop intuitive navigation for visually impaired	Audio instructions lead to confusion	Real-time AR and haptic guidance	Navigation accuracy and recovery efficiency
2	Indoor navigation for visually impaired using AR markers	2017	Enable accurate indoor visual navigation	Depends on visible AR markers	Outperforms GPS in indoor accuracy	Pose accuracy using AR markers
3	Indoor navigation for people with visual impairment using augmented	2019	Develop CV-based navigation using ArUco markersand prototyping	Dependent on marker visibility	Robust detection; voice command support	Path accuracy, tag detection success

	reality markers					
4	A new architecture based ar for detection and recognition of objects and text to enhance navigation of visually impaired people	2020	Context-aware mobile computing with AR for text/object detection	Struggles in complex backgrounds; hardware dependent	Real-time object/text recognition	Accuracy, processing speed
5	A navigation and augmented reality system for visually impaired people. Sensors	2021	Markerless navigation using object recognition	Sensitive to lighting and battery	Flexible, multimodal real-time guidance	Tracking accuracy and recognition rate
6	An indoor navigation aid for the visually impaired	2016	Improve indoor navigation with SLAM	Struggles with sparse inlier data	Fast, accurate 6-DOF pose estimation	Runtime per frame and pose error
7	Semantic indoor navigation with a blind-user oriented augmented reality	2013	Assist travel using AR navigation	Needs semantic maps, accumulates errors	Accurate, real-time path prediction	Pose accuracy and obstacle avoidance
8	A comprehensive review of navigation systems for visually impaired individuals	2024	Review navigation technologies for accessibility	High costs, environmental limitations, accuracy	Reliable navigation support in environments	System effectiveness in real-world conditions
9	Augmented reality for the visually impaired: Navigation	2022	Develop adaptable navigation for visually impaired	High computational demands, static models	Enhanced experience with immersive feedback	Adaptability and user cognitive load

	aid and scene semantics for indoor use cases					
10	Design of Mobile Augmented Reality Assistant application via Deep Learning and LIDAR for Visually Impaired	2023	Enhance navigation and object identification	Cumbersome setup and additional costs	Flexible, network-free, user-centered design	Utilization rate and success rate
11	Navigation systems for the blind and visually impaired: Past work, challenges, and open problems	2019	Trace evolution of navigation systems	Unmet needs in indoor positioning	Integration of smartphones and wearables	User-centered design and effectiveness
12	An automated AR-based annotation tool for indoor navigation for visually impaired people	2021	Improve indoor navigation with AR	Labor-intensive crowd-sourced data collection	Automated, machine learning-powered annotation tool	
13	A smartphone-based mobility assistant using depth imaging for visually impaired and blind	2022	Enhance mobility through smart sensing	Lack of portability and flexibility	Adaptable solutions for improved usability	User interface and navigation effectiveness
14	Application of Augmented Reality	2024	Determining Distances in Navigation	Error margin of up to 7.5%	AR integration for real-world	

	Technologies for Determining Distances in Navigation System for the Blind		System for the Blind using AR		interaction	
15	Aiguide: An augmented reality hand guidance application for people with visual impairments	2020	Provide real-time guidance with AR	Dependency on hardware or connectivity	Self-contained, offline smartphone application	User feedback preferences and practicality

WORKFLOW

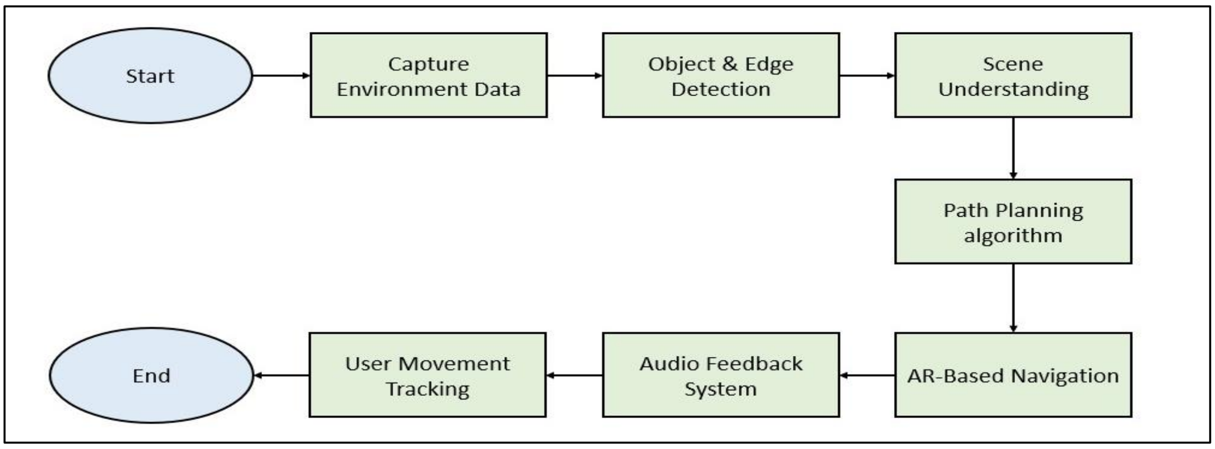


Fig 1.1: Work Flow of the Project

Fig 2.1 illustrates the process of an AR-based indoor navigation system designed for visually impaired individuals. The system begins by capturing environmental data using sensors like cameras or depth sensors. This data is then processed for object and edge detection, identifying physical elements and boundaries within the space. Following this, scene understanding is performed to interpret the spatial layout and recognize key objects and obstacles. Using this information, a path planning algorithm calculates the safest and most efficient route for navigation. Augmented Reality (AR) is then used to guide the user, often through audio instructions rather than visual cues. An audio feedback system delivers these instructions in real-time, directing the user effectively. Simultaneously, user movement is tracked to ensure they remain on the intended path, allowing the system to adapt and provide continuous guidance. The process concludes once the user reaches the destination or ends the navigation session. This integrated approach offers a reliable and supportive navigation solution tailored to the needs of visually impaired users.

3. REQUIREMENT SPECIFICATION

A requirement specification is a comprehensive document that outlines the expectations, functions, and constraints of a project before development begins. It serves as a formal agreement between stakeholders, such as clients, users, and developers, ensuring that everyone involved has a clear understanding of what the final product should deliver. This document typically includes both functional requirements—describing what the system should do—and non-functional requirements—detailing how the system should perform, such as in terms of speed, security, or usability. It may also cover user needs, system constraints, and any assumptions or dependencies relevant to the project. By clearly defining the scope and objectives, a requirement specification helps prevent misunderstandings, guides the development process, and serves as a basis for testing and validation of the final outcome.

System Requirements for Object Detection Navigation System

3.1 Hardware Requirements

Component	Minimum Specification	Recommended Specification
Processor	Quad-core 1.5GHz (ARMv7/ARM64 or x86)	Octa-core 2.0GHz+ (Snapdragon 845/Apple A12 or better)
RAM	2GB	4GB+
Storage	500MB free space	1GB+ free space
Camera	5MP rear camera	12MP+ with autofocus
GPU	OpenGL ES 3.0 support	Vulkan/NNAPI support
Sensors	Accelerometer, Gyroscope	Depth sensor (for advanced scenarios)
Audio	Standard speaker/headphone	Noise-canceling microphone

3.2 Software Requirements

Operating System: Android 9.0 (API 28) or later and iOS not fully supported

Unity Engine: 2021.3 LTS or later and Requires Barracuda package

ML Backend: Barracuda 3.0.0+ For YOLO model inference

Dependencies:

1) Android NDK 21+
2) OpenJDK 11+ For native plugin support

Permissions:

1) Camera
2) Microphone
3) Storage (optional) Required at runtime

3.3 Development Environment

Tool	Version	Purpose
Unity Hub	3.0+	Project management
Android Studio	2022.2+	SDK/NDK setup
Visual Studio	2022+	Code editor (C#)
Git	2.35+	Version control

3.4 Supported Devices

Mobile: Android smartphones (2018+ models)

3.5 Performance Considerations

Inference Speed: 15-30 FPS (depends on model size)

Memory Usage: ≤300MB RAM for detection pipeline

Thermal Constraints: Sustained usage may throttle performance

4. METHODOLOGY.

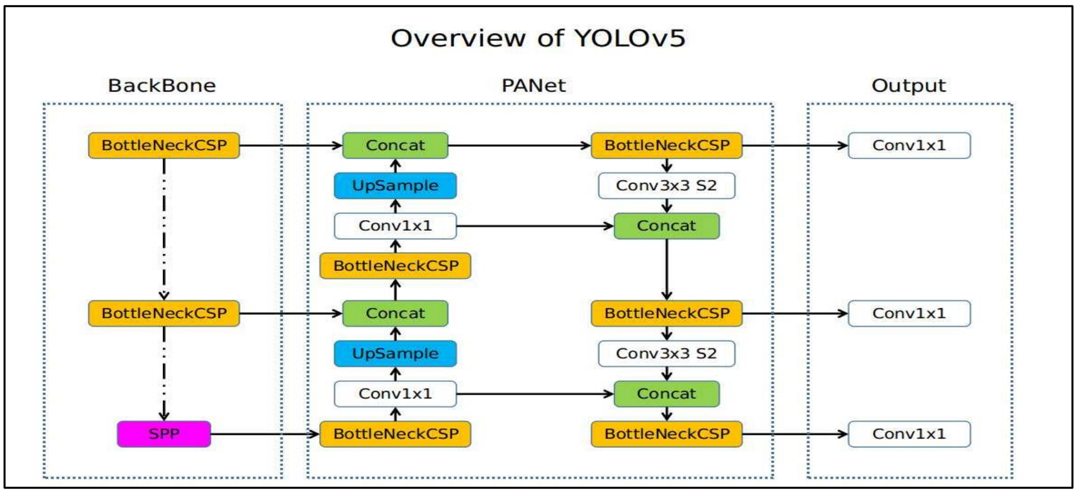
4.1 System Structure and Design

The system follows a client-side architecture tailored for mobile AR deployment, primarily built using Unity with the AR Foundation framework to ensure compatibility across Android (ARCore) and iOS (ARKit) platforms. Unity serves as the development environment for both AR scene creation and real-time interaction logic. The YOLO object detection model, trained externally in Python and exported as an ONNX model, is integrated into Unity using Barracuda, Unity’s neural network inference engine. This design allows on-device real-time object detection without needing server-side computation, ensuring responsiveness and offline usability—both essential for visually impaired users.

4.2 Model Integration and Inference

Scene Understanding Using YOLO: The **YOLOv5s** model was chosen for its ideal trade-off between precision and real-time performance, making it well-suited for navigation assistance. The training process involved fine-tuning the model on a custom dataset using transfer learning to enhance detection performance on navigation-critical objects. Hyperparameter optimization was conducted to refine parameters such as learning rate, batch size, and confidence thresholds, ensuring improved model convergence and detection reliability. The model's performance was validated using mean Average Precision (mAP) and confusion matrices, which provided comprehensive insights into detection accuracy across various object categories.

4.2.1 Architecture of Yolo v5



4.3 Real-Time Object Detection and Feedback

Once inference is performed, bounding boxes, class labels, and confidence scores are extracted from the YOLO output tensor. A post-processing pipeline, including non-maximum suppression (NMS), refines the detections. The detected objects are then used to generate audio cues that describe the environment to the user. Audio feedback is provided using Unity's text-to-speech (TTS) plugins or native platform APIs (such as Android's TextToSpeech or iOS's AVSpeechSynthesizer), ensuring prompt and clear communication of surroundings to visually impaired users.

4.3.1 Object detection mechanism

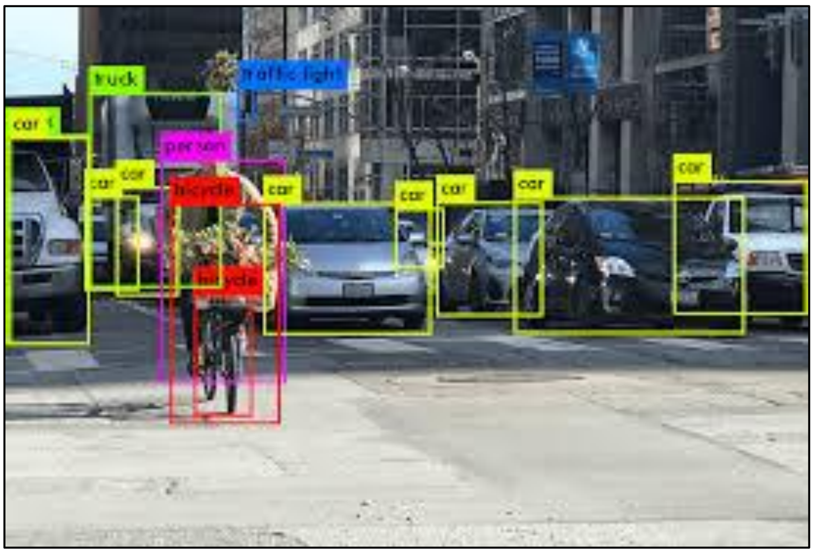


Fig 4.2:

4.4 Data preprocessing

Since this study utilizes a pretrained YOLOv5s model for real-time object detection, no separate dataset was collected. Instead, preprocessing techniques were applied to optimize input data for inference. Images Adjusted in size and standardized to align with the model's requirements while ensuring minimal computational overhead. Additionally, data augmentation techniques such as brightness adjustment and contrast enhancement were used during preprocessing to improve detection performance under varying environmental conditions. These steps ensure that the model effectively detects obstacles and navigation-critical elements in real-world scenarios.

4.5 System Optimization and Model Performance

To ensure a smooth experience on mobile devices, the system uses YOLOv5n or YOLOv8n (nano models) which are optimized for speed and size. Frame processing is throttled to avoid overloading the device's CPU/GPU—typically, inference is run every few frames rather than every frame. Unity’s multithreading capabilities and GPU-accelerated Barracuda backend ensure minimal lag during operation. Tests are conducted to validate performance on various devices, ensuring low latency and real-time feedback.

4.6 Notifications and Contextual Alerts

In addition to real-time audio feedback, the platform can deliver context-aware alerts such as "Stairs ahead", "Obstacle to the left", or "Exit 2 meters ahead". These alerts are prioritized based on object proximity and relevance to navigation. A vibration feedback mechanism can also be implemented as a secondary channel of alert, especially useful in noisy environments or for users with partial hearing impairments.

4.7 Testing and Quality Assurance

The platform undergoes rigorous usability and accessibility testing with feedback from visually impaired individuals to refine voice prompts, alert timing, and object identification reliability. Unit testing is applied to core components like frame preprocessing, model inference, and audio generation. Field testing is performed in varied indoor and outdoor environments to ensure the system performs reliably under different lighting and clutter conditions. Furthermore, load testing ensures that memory and CPU usage stay within acceptable limits during prolonged usage.

5. RESULTS

The performance evaluation of YOLOv5s and YOLOv5n for AR-based navigation in assisting visually impaired users was conducted based on three key metrics: mean Average Precision (mAP), inference speed (frames per second, FPS), and latency. The results indicate that YOLOv5s achieved a higher mAP of 75%, demonstrating superior object detection accuracy compared to YOLOv5n, which attained 68%. While YOLOv5n exhibited a slightly higher inference speed of 50 FPS compared to YOLOv5s at 45 FPS, it suffered from increased latency, requiring 40 milliseconds for processing, whereas YOLOv5s achieved a lower latency of 30 milliseconds. These findings highlight that YOLOv5s provides a better balance between accuracy and real-time performance, making it more suitable for AR-based navigation applications where precise and timely object detection is critical. The trade-off observed between speed and accuracy suggests that model selection should prioritize detection reliability, ensuring a safer and more efficient navigation experience For individuals with visual impairments.

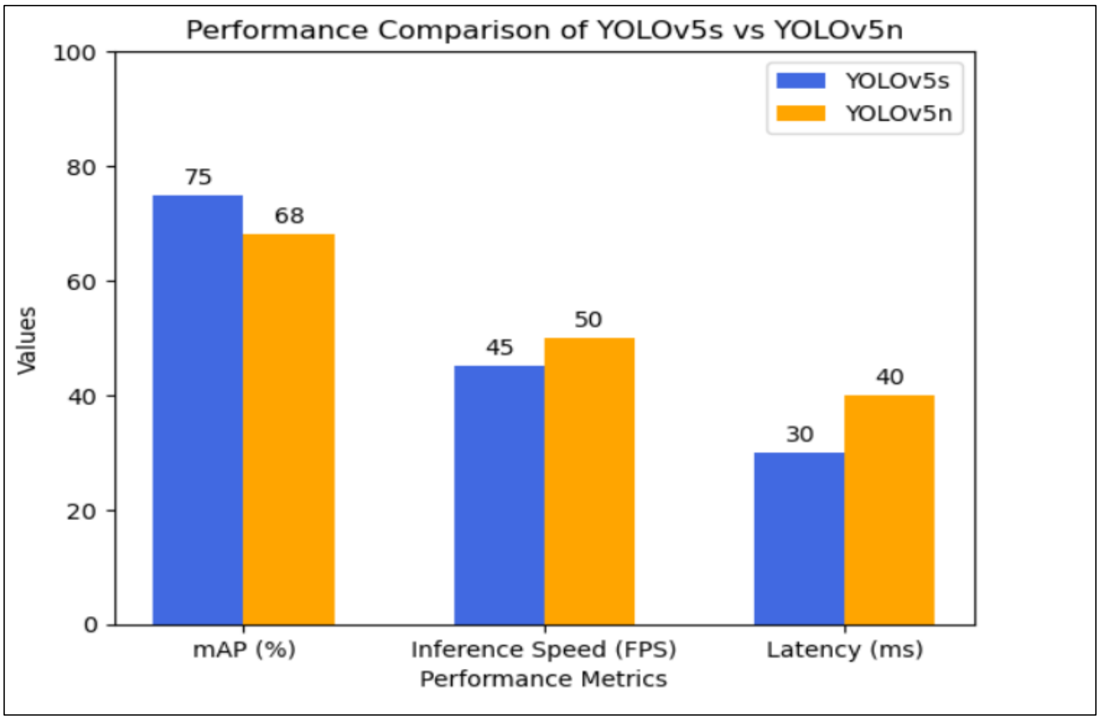


Fig 5.1: Performance comparison of Yolov5s and Yolov5n

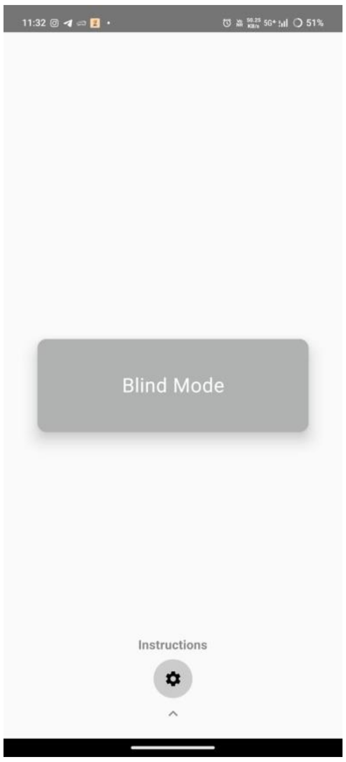


fig 5.2: Home page

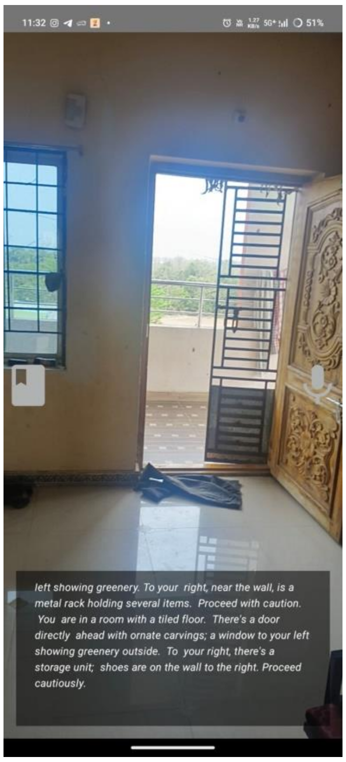
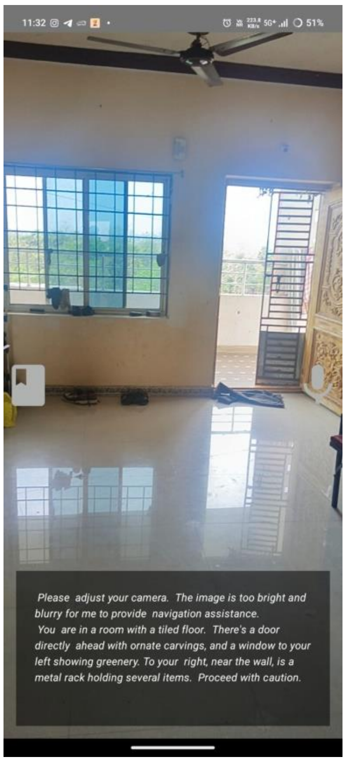
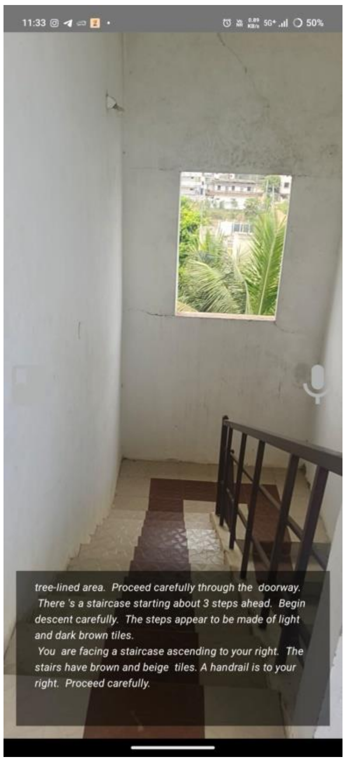

left showing greenery. To your right, near the wall, is a metal rack holding several items. Proceed with caution.
You are in a room with a tiled floor. There's a door directly ahead with ornate carvings; a window to your left showing greenery outside. To your right, there's a storage unit; shoes are on the wall to the right. Proceed cautiously.

fig 5.3: Instructions


Please adjust your camera. The image is too bright and blurry for me to provide navigation assistance.
You are in a room with a tiled floor. There's a door directly ahead with ornate carvings, and a window to your left showing greenery. To your right, near the wall, is a metal rack holding several items. Proceed with caution.


tree-lined area. Proceed carefully through the doorway.
There 's a staircase starting about 3 steps ahead. Begin descent carefully. The steps appear to be made of light and dark brown tiles.
You are facing a staircase ascending to your right. The stairs have brown and beige tiles. A handrail is to your right. Proceed carefully.

6. CONCLUSION

The AR-based navigation system navigation system for individuals with visual impairments showcases the potential of integrating deep learning-based object detection with augmented reality to enhance spatial awareness and safety. By leveraging YOLO for real-time object detection and Unity's AR Foundation for interactive visualization, the system effectively identifies and overlays critical navigation elements such as obstacles, staircases, and pedestrian crossings within the user's field of view. The inclusion of adaptive audio feedback further improves accessibility by providing intuitive and non-intrusive guidance. Performance evaluation highlights that YOLOv5s strikes an optimal balance between accuracy and real-time efficiency, making it well-suited for AR applications in assistive navigation. The findings indicate that accurate detection with low latency is essential for providing a smooth user experience. Future improvements may focus on enhancing depth estimation, optimizing processing speed, and incorporating user feedback to refine system usability. Overall, the proposed AR-based navigation system represents a significant advancement in enabling visually impaired individuals to navigate their environment with increased independence and confidence.

7. APPENDIX

AndroidObjectDetector.cs

```
using UnityEngine;

using UnityEngine.UI;

using System.Collections;

using System.Linq;

#if UNITY_ANDROID

using UnityEngine.Android;

#endif

[RequireComponent(typeof(OnGUICanvasRelativeDrawer))]

public class AndroidObjectDetector : MonoBehaviour

{

    [Header("Camera Settings")]

    public bool useFrontCamera = false;

    public int targetFPS = 30;

    public Vector2Int resolution = new Vector2Int(640, 480);

    [Header("UI References")]

    [SerializeField] private RawImage _cameraDisplay;

    private WebCamTexture _webcamTexture;

    private OnGUICanvasRelativeDrawer _drawer;

    IEnumerator Start()

    {
        if (_cameraDisplay == null)
        {
            Debug.LogError("Camera display RawImage not assigned!");
            yield break;
        }
        _drawer = GetComponent<OnGUICanvasRelativeDrawer>();
#if UNITY_ANDROID
        if (!Permission.HasUserAuthorizedPermission(Permission.Camera))
        {
            Permission.RequestUserPermission(Permission.Camera);
            yield return new WaitForSeconds(0.1f);
            if (!Permission.HasUserAuthorizedPermission(Permission.Camera))
            {
                Debug.LogError("Camera permission denied");
                yield break;
            }
        }
#endif
        yield return StartCoroutine(InitializeCamera());
    }
    IEnumerator InitializeCamera()
    {
```

```
WebCamDevice[] devices = WebCamTexture.devices;
if (devices.Length == 0)
{
    Debug.LogError("No cameras found");
    yield break;
}
string deviceName = devices.FirstOrDefault(
    d => d.isFrontFacing == useFrontCamera).name ?? devices[0].name;
_webcamTexture = new WebCamTexture(
    deviceName,
    resolution.x,
    resolution.y,
    targetFPS);

_cameraDisplay.texture = _webcamTexture;
_webcamTexture.Play();
int maxWait = 100;
while (_webcamTexture.width <= 16 && maxWait > 0)
{
    yield return null;
    maxWait--;
}
```

```
    if (_webcamTexture.width <= 16)
    {
        Debug.LogError("Camera failed to initialize");
        yield break;
    }
    Debug.Log($"Camera started: {_webcamTexture.deviceName}");
}
void Update()
{
    if (_webcamTexture != null && _webcamTexture.isPlaying)
    {
        float aspect = (float)_webcamTexture.width / _webcamTexture.height;
        _cameraDisplay.rectTransform.sizeDelta = new Vector2(
            _cameraDisplay.rectTransform.sizeDelta.y * aspect,
            _cameraDisplay.rectTransform.sizeDelta.y);
    }
}
void OnDisable()
{
    if (_webcamTexture != null)
    {
        _webcamTexture.Stop();
```

```
            Destroy(_webcamTexture);
        }
    }
}
```

Functionality:
This script handles Android camera initialization and display. It:

1. Requests camera permissions on Android
2. Initializes either front or rear camera based on settings
3. Displays the camera feed on a RawImage UI element
4. Automatically adjusts the display aspect ratio
5. Handles proper cleanup when disabled

WebCamTextureProvider.cs

```
using UnityEngine;
using System;
namespace Assets.Scripts
{
    public class WebCamTextureProvider
    {
        private Texture2D resultTexture;
        private WebCamTexture webCamTexture;
        public bool IsPlaying => webCamTexture != null && webCamTexture.isPlaying;
        public WebCamTextureProvider(int width, int height, int targetFPS = 30, bool
useFrontCamera = false)
        {
            resultTexture = new Texture2D(width, height, TextureFormat.RGB24, false);
            string deviceName = GetCameraDevice(useFrontCamera);
            webCamTexture = new WebCamTexture(deviceName, width, height, targetFPS);
        }
        private string GetCameraDevice(bool useFrontCamera)
        {
            if (WebCamTexture.devices.Length == 0)
                throw new Exception("No cameras available");

            foreach (var cam in WebCamTexture.devices)
            {
                if (cam.isFrontFacing == useFrontCamera)
                    return cam.name;
            }
            return WebCamTexture.devices[0].name;
        }
        public void Start()
        {
            if (webCamTexture != null && !webCamTexture.isPlaying)
```

```
            webCamTexture.Play();
        }
        public void Stop()
        {
            if (webCamTexture != null && webCamTexture.isPlaying)
                webCamTexture.Stop();
        }


        public Texture2D GetTexture()
        {
            if (webCamTexture == null || !webCamTexture.isPlaying ||
!webCamTexture.didUpdateThisFrame)
                return resultTexture;
            return TextureTools.ResizeAndCropToCenter(webCamTexture, ref resultTexture,
                resultTexture.width, resultTexture.height);
        }
    }
}
```

Functionality:

A utility class that:

1. Wraps WebCamTexture functionality
2. Provides resized and cropped Texture2D output
3. Handles camera device selection (front/back)

4. Manages camera start/stop operations
5. Maintains a consistent output texture size regardless of input

YOLOHandler.cs

```
using System;
using System.Collections.Generic;
using Unity.Barracuda;
using UnityEngine;
namespace NN
{
   [RequireComponent(typeof(NNHandler))]
   public class YOLOHandler : MonoBehaviour, IDisposable
   {
      [Header("Configuration")]
      public WorkerFactory.Type workerType = WorkerFactory.Type.ComputePrecompiled;
      [Header("Debug")]
      [SerializeField] private bool verboseLogging = false;
      private NNHandler nnHandler;
      private IOps ops;
      private Tensor premulTensor;
      private bool isInitialized = false;
      void Awake()
      {
```

```
    nnHandler = GetComponent<NNHandler>();
    if (nnHandler == null)
    {
        Debug.LogError("NNHandler component missing!", this);
        enabled = false;
        return;
    }
}
void Start()
{
    try
    {
        ops = BarracudaUtils.CreateOps(workerType);
        premulTensor = new Tensor(1, 1, new float[] { 255 });
        isInitialized = true;
        if (verboseLogging)
            Debug.Log("YOLOHandler initialized successfully", this);
    }
    catch (Exception e)
    {
        Debug.LogError($"Initialization failed: {e.Message}", this);
        enabled = false;
    }
}
public List<ResultBox> Run(Texture2D inputTexture)
{
    if (!isInitialized || inputTexture == null)
    {
        if (verboseLogging)
            Debug.LogWarning("Skipping detection - not initialized or null input");
        return new List<ResultBox>();
    }
    try
    {
        using (Tensor input = new Tensor(inputTexture))
        {
            Tensor preprocessed = Preprocess(input);
            ExecuteNetwork(preprocessed);
            using (Tensor output = GetNetworkOutput())
            {
                return Postprocess(output);
            }
        }
    }
    catch (Exception e)
    {
        Debug.LogError($"Detection failed: {e.Message}");
        return new List<ResultBox>();
    }
}
public void SetConfidenceThreshold(float threshold)
{
    YOLOv2Postprocessor.DiscardThreshold = Mathf.Clamp01(threshold);
}
private Tensor Preprocess(Tensor x)
{
    if (ops == null) throw new InvalidOperationException("Ops not initialized");
    return ops.Mul(new[] { x, premulTensor });
}
private void ExecuteNetwork(Tensor input)
{
    nnHandler.worker.Execute(input);
    nnHandler.worker.FlushSchedule(true);
}
private Tensor GetNetworkOutput()
{
    return nnHandler.worker.PeekOutput();
}
private List<ResultBox> Postprocess(Tensor output)
{
    return DuplicatesSupressor.RemoveDuplicats(
        YOLOv2Postprocessor.DecodeNNOut(output)
    );
}
public void Dispose()
{
    if (!isInitialized) return;
    premulTensor?.Dispose();
    isInitialized = false;

    if (verboseLogging)
        Debug.Log("YOLOHandler disposed", this);
}

void OnDestroy()
{
    Dispose();
}
```

```
    }
}
```

Functionality:

The core YOLO (You Only Look Once) object detection handler that:

1. Interfaces with Barracuda neural network engine
2. Preprocesses input textures for neural network
3. Executes YOLO model inference
4. Postprocesses raw network output into usable detection results
5. Handles confidence threshold configuration
6. Manages GPU/CPU resources efficiently

YOLOv2Postprocessor.cs

```
using System;
using System.Collections.Generic;
using Unity.Barracuda;
using UnityEngine;
namespace NN
{
    static public class YOLOv2Postprocessor
    {
        public static float DiscardThreshold { get; set; } = 0.1f;
        public const int ClassesNum = 20;
        const int BoxesPerCell = 5;
        static readonly float[] Anchors = new[] { 1.08f, 1.19f, 3.42f, 4.41f, 6.63f, 11.38f, 9.42f,
5.11f, 16.62f, 10.52f };

        static IOps cpuOps;

        static YOLOv2Postprocessor()

        {

            cpuOps = BarracudaUtils.CreateOps(WorkerFactory.Type.CSharp);

        }

        static public List<ResultBox> DecodeNNOut(Tensor output)

        {

            List<ResultBox> boxes = new();

            float[,,] array = ReadOutputToArray(output);

            int widht = array.GetLength(0);

            int height = array.GetLength(1);

            for (int y_cell = 0; y_cell < height; y_cell++)

            {

                for (int x_cell = 0; x_cell < widht; x_cell++)

                {

                    var cell_boxes = DecodeCell(array, x_cell, y_cell);

                    boxes.AddRange(cell_boxes);

                }

            }

            return boxes;
        }
        private static float[,,,] ReadOutputToArray(Tensor output)
        {
            const int boxSize = 25;
            var reshapedOutput = output.Reshape(new[] { output.height, output.width,
BoxesPerCell, boxSize });
            var array = TensorToArray4D(reshapedOutput);
            reshapedOutput.Dispose();
            return array;
        }
        private static IEnumerable<ResultBox> DecodeCell(float[,,,] array, int x_cell, int y_cell)
        {
            int boxes = array.GetLength(2);
            for (int box_index = 0; box_index < boxes; box_index++)
            {
                var box = DecodeBox(array, x_cell, y_cell, box_index);
                if (box != null)
                    yield return box;
            }
        }
        static private ResultBox DecodeBox(float[,,,] array, int x_cell, int y_cell, int box)
        {
            (int bestClassIndex, float bestClassScore) = DecodeBestBoxIndexAndScore(array,
x_cell, y_cell, box);
            if (bestClassScore < DiscardThreshold)
                return null;
            Rect box_rect = DecodeBoxRectangle(array, x_cell, y_cell, box);
            var result = new ResultBox
            {
                rect = box_rect,
                score = bestClassScore,
                bestClassIndex = bestClassIndex,
            };
            return result;
        }
        static private (int, float) DecodeBestBoxIndexAndScore(float[,,,] array, int x_cell, int
y_cell, int box)
        {
            float[] classesScore = DecodeBoxClasses(array, x_cell, y_cell, box);
            int highestClassIndex = 0;
            float highestScore = 0;
            for (int i = 0; i < ClassesNum; i++)
            {
                float currentClassScore = classesScore[i];
                if (currentClassScore > highestScore)
                {
                    highestScore = currentClassScore;
                    highestClassIndex = i;
                }
            }
            float boxScore = DecodeBoxScore(array, x_cell, y_cell, box);
            highestScore *= boxScore;
            return (highestClassIndex, highestScore);
        }
        static private float DecodeBoxScore(float[,,,] array, int x_cell, int y_cell, int box)
        {
            const int boxScoreIndex = 4;
            return Sigmoid(array[y_cell, x_cell, box, boxScoreIndex]);
        }
        static private float[] DecodeBoxClasses(float[,,,] array, int x_cell, int y_cell, int box)
        {
            const int classesOffset = 5;

            float[] boxClasses = new float[ClassesNum];
            for (int i = 0; i < ClassesNum; i++)
            {
                boxClasses[i] = array[y_cell, x_cell, box, classesOffset + i];
```

```
            }
            boxClasses = Softmax(boxClasses);
            return boxClasses;
        }
        static private Rect DecodeBoxRectangle(float[,,,] data, int x_cell, int y_cell, int box)
        {
            const float downscaleRatio = 32;
            const int boxCenterXIndex = 0;
            const int boxCenterYIndex = 1;
            const int boxWidthIndex = 2;
            const int boxHeightIndex = 3;
            float boxCenterX = (x_cell + Sigmoid(data[y_cell, x_cell, box, boxCenterXIndex])) *
downscaleRatio;
            float boxCenterY = (y_cell + Sigmoid(data[y_cell, x_cell, box, boxCenterYIndex])) *
downscaleRatio;
            float boxWidth = Mathf.Exp(data[y_cell, x_cell, box, boxWidthIndex]) * Anchors[2 *
box] * downscaleRatio;
            float boxHeight = Mathf.Exp(data[y_cell, x_cell, box, boxHeightIndex]) * Anchors[2
* box + 1] * downscaleRatio;
            float box_x = boxCenterX - boxWidth / 2;
            float box_y = boxCenterY - boxHeight / 2;
            return new Rect(box_x, box_y, boxWidth, boxHeight);
        }
        static private float Sigmoid(float value)
        {
            return 1f / (1f + Mathf.Exp(-value));
        }
        static private float[] Softmax(float[] values)
        {
            Tensor inputTensor = new(1, values.Length, values);
            float[] output = cpuOps.Softmax(inputTensor, axis: -1).AsFloats();
            inputTensor.Dispose();
            return output;
        }
        private static float[,,,] TensorToArray4D(this Tensor tensor)
        {
            float[,,,] output = new float[tensor.batch, tensor.height, tensor.width, tensor.channels];
            var data = tensor.AsFloats();
            int bytes = Buffer.ByteLength(data);
            Buffer.BlockCopy(data, 0, output, 0, bytes);
            return output;
        }
    }
}
```

Functionality:

Specialized YOLOv2 output processor that:

1. Decodes raw neural network output tensors
2. Applies YOLOv2-specific transformations
3. Calculates bounding boxes using anchor boxes
4. Converts cell-relative coordinates to absolute coordinates
5. Applies sigmoid and softmax activations
6. Handles class probability calculations

PerformanceCounter.cs

```
using System.Collections.Generic;

using System.Diagnostics;

using System.Text;

using UnityEngine;

using UnityEngine.UI;

public class PerformanceCounter : MonoBehaviour

{

    public class Counter

    {

        protected string name;

        protected float value;

        public float Value { get => value; set => this.value = value; }

        public string Name { get => name; set => name = value; }

        public Counter(string name, float value)
        {
            this.Name = name;
            this.Value = value;
        }
    }

    public class StopwatchCounter : Counter
    {
        public new float Value { get => value; }
        Stopwatch stopwatch;
        public StopwatchCounter(string name) : base(name, 0)
        {
            this.name = name;
            stopwatch = new Stopwatch();
        }
        public void Start()
        { stopwatch.Restart(); }
        public void Stop()
        {
            stopwatch.Stop();
            value = stopwatch.ElapsedMilliseconds;
        }
```

```
}

static PerformanceCounter firstInstance;

public Text textField;

public bool showFPS = true;

[Range(1, 250)]

public float FPSSmothness = 33;

List<Counter> counters = new List<Counter>();

float previousFPS;

StringBuilder builder = new StringBuilder();

void Start()

{

    if (firstInstance == null)

        firstInstance = this;

}

void Update()

{

    if (showFPS)

    {

        builder.Append(GetSmoothFPS());

        builder.Append(" FPS");

        builder.AppendLine();

    }
    foreach (var counter in counters)

    {

        builder.Append(counter.Name);

        builder.Append(": ");

        builder.Append(counter.Value);

        builder.AppendLine();

    }

    textField.text = builder.ToString();

    builder.Clear();

}

public static PerformanceCounter GetInstance()

{

    return firstInstance;

}

public Counter AddCounter(string name)

{

    var c = new Counter(name, 0);

    counters.Add(c);

    return c;

}

public void AddCounter(Counter counter)

{
        counters.Add(counter);
    }
    private int GetSmoothFPS()
    {
        float fps = 1f / Time.unscaledDeltaTime;
        previousFPS = (previousFPS + fps * (1 / FPSSmothness)) / (1 + (1 / FPSSmothness));
        return (int)previousFPS;
    }
}
```

Functionality:

A performance monitoring utility that:

1. Tracks and displays FPS with smoothing
2. Allows adding custom performance counters
3. Provides stopwatch functionality for timing
4. Displays all metrics in a UI Text element
5. Supports both immediate and accumulated measurements
6. Implements singleton pattern for easy access

OnGUICanvasRelativeDrawer.cs

```
using System.Collections.Generic;
using UnityEngine;
public class OnGUICanvasRelativeDrawer : MonoBehaviour
{
```

```
struct Label

{

    public string text;

    public Rect rect;

}

[HideInInspector]

public RectTransform relativeObject { set { relative = value; rootCanvas = 
GetRootCanvas(value); } }

RectTransform relative;

RectTransform rootCanvas;

List<Label> labels = new List<Label>();

GUIStyle style;

public void DrawLabel(string text, Vector2 position)

{

    Vector2 anchor = GetAnchorPosition();

    Rect rect = new Rect(anchor + position * relative.rect.size, new Vector2(150, 100));

    labels.Add(new Label { text = text, rect = rect });

}

public void Clear()

{

    labels.Clear();

}
```

```
    private GUIStyle GetStyle()
    {
        return new GUIStyle { fontSize = 20, normal = new GUIStyleState { textColor =
Color.white } };
    }
    private void OnGUI()
    {
        style = style != null ? style : GetStyle();
        foreach (var label in labels)
        {
            GUI.Label(label.rect, label.text, style);
        }
    }
    private Vector2 GetAnchorPosition()
    {
        return new Vector2(relative.localPosition.x, -relative.localPosition.y) +
rootCanvas.rect.size / 2 - relative.rect.size / 2;
    }
    private RectTransform GetRootCanvas(RectTransform rectTransform)
    {
        Transform parent = rectTransform.transform;
        while (true)
        {
            if (parent.parent != null && parent.parent.GetComponent<RectTransform>() != null)
                parent = parent.parent;
            else
                return parent.GetComponent<RectTransform>();
        }
    }
}
```

Functionality:

A GUI drawing utility that:

1. Draws labels relative to a UI element
2. Maintains proper positioning across resolutions
3. Automatically finds root canvas
4. Provides simple API for temporary on-screen text
5. Handles coordinate conversion between screen and canvas space
6. Supports multiple simultaneous labels with clean styling

Extensions.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
static class Extensions
{
    public static string ArrayToString(this IEnumerable enumerable)
```

```
{
    string str = "[";
    foreach (var e in enumerable)
    {
        str += e;
    }
    str += "]";
    return str;
}
public static T[] GetRange<T>(this ICollection<T> collection, int start = 0, int end = -1)
{
    end = end < 0 ? end = collection.Count + end + 1 : end;
    var arr = Array.CreateInstance(typeof(T), end - start);
    for (int i = start, j = 0; i < end; i++, j++)
    {
        var v = collection.ElementAt(i);
        arr.SetValue(v, j);
    }
    return (T[])arr;
}
public static int MaxIdx(this ICollection<float> collection)
{
    int idx = 0;
    float max = collection.ElementAt(0);
    for (int i = 1; i < collection.Count; i++)
    {
        if (collection.ElementAt(i) > max)
        {
            idx = i;
            max = collection.ElementAt(i);
        }
    }
    return idx;
}
public static void ForEach<T>(this IEnumerable<T> enumerable, Action<T> func)
{
    foreach (var e in enumerable)
        func(e);
}
public static void ForEach<T>(this ICollection<T> collection, Action<T, int> func)
{
    for (int i = 0; i < collection.Count; i++)
        func(collection.ElementAt(i), i);
}
```

```
    public static IList<T> Update<T>(this IList<T> collection, Func<T, T> func)
    {
        for (int i = 0; i < collection.Count; i++)
            collection[i] = func(collection[i]);
        return collection;
    }
    public static IList<T> Update<T>(this IList<T> collection, Func<T, int, T> func)
    {
        for (int i = 0; i < collection.Count; i++)
            collection[i] = func(collection[i], i);
        return collection;
    }
}
```

Functionality:
A collection of useful extension methods for:

1. Converting collections to strings
2. Getting array ranges with negative indices
3. Finding maximum value indices
4. Functional-style collection operations (ForEach, Update)
5. Working with ICollection and IEnumerable interfaces
6. Providing LINQ-like operations for non-generic collections

BarracudaUtils.cs

```
using Unity.Barracuda;
```

```
public class BarracudaUtils

{

    public static IOps CreateOps(WorkerFactory.Type type, bool verbose = false)

    {

        WorkerFactory.ValidateType(type);

        switch (type)

        {

            case WorkerFactory.Type.ComputePrecompiled:

                return new PrecompiledComputeOps(verbose: verbose);

            case WorkerFactory.Type.Compute:

                return new ComputeOps(verbose: verbose);

            case WorkerFactory.Type.ComputeRef:

                return new ReferenceComputeOps();

            case WorkerFactory.Type.CSharp:

                return new UnsafeArrayCPUOps();

            default:

                return new ReferenceCPUOps();

        }

    }

}
```

Functionality:

Barracuda neural network utility that:

1. Creates optimized IOps instances
2. Supports multiple worker types (CPU/GPU)
3. Provides a unified interface for tensor operations
4. Handles backend-specific optimizations
5. Simplifies worker type creation
6. Validates worker types before creation

DuplicatesSupressor.cs

```
using NN;
using System;
using System.Collections.Generic;
using System.Linq;
using UnityEngine.Profiling;
public static class DuplicatesSupressor
{
    const float OverlapThreshold = 0.3f;
    const int ClassesNum = 20;
    static public List<ResultBox> RemoveDuplicats(List<ResultBox> boxes)
    {
        Profiler.BeginSample("DuplicatesSupressor.RemoveDuplicats");
        if (boxes.Count == 0)
            return boxes;
        List<ResultBox> result_boxes = new();
        for (int classIndex = 0; classIndex < ClassesNum; classIndex++)

        {

            List<ResultBox> classBoxes = boxes.Where(box => box.bestClassIndex ==
classIndex).ToList();

            RemoveDuplicatesForClass(classBoxes);

            IEnumerable<ResultBox> filteredClassBoxes = classBoxes.Where(box => box.score
> 0);

            result_boxes.AddRange(classBoxes);

        }

        Profiler.EndSample();

        return result_boxes;

    }

    private static void RemoveDuplicatesForClass(List<ResultBox> boxes)

    {

        SortBoxesByScore(boxes);

        for (int i = 0; i < boxes.Count; i++)

        {

            ResultBox i_box = boxes[i];

            if (i_box.score == 0)

                continue;

            for (int j = i + 1; j < boxes.Count; j++)

            {

                ResultBox j_box = boxes[j];
                float iou = IntersectionOverUnion.CalculateIOU(i_box.rect, j_box.rect);
                if (iou >= OverlapThreshold && i_box.score > j_box.score)
                {
                    j_box.score = 0;
                }
            }
        }
    }
    private static List<ResultBox> SortBoxesByScore(List<ResultBox> boxes)
    {
        Comparison<ResultBox> boxClassValueComparer =
            (box1, box2) => box2.score.CompareTo(box1.score);
        boxes.Sort(boxClassValueComparer);
        return boxes;
    }
}
```

Functionality:

Post-processing for object detection that:

1. Removes duplicate detections using non-maximum suppression
2. Uses Intersection-over-Union (IOU) for overlap detection
3. Filters by class and confidence
4. Maintains only the highest confidence detections
5. Includes profiling markers for performance analysis

6. Handles empty detection cases gracefully

IntersectionOverUnion.cs

```
using UnityEngine;

public static class IntersectionOverUnion

{

    public static float CalculateIOU(Rect box1, Rect box2)

    {

        float intersect_w = IntervalOverlap(box1.xMin, box1.xMax, box2.xMin, box2.xMax);

        float intersect_h = IntervalOverlap(box1.yMin, box1.yMax, box2.yMin, box2.yMax);

        float intersect = intersect_w * intersect_h;

        float union = box1.width * box1.height + box2.width * box2.height - intersect;

        return intersect / union;

    }

    static float IntervalOverlap(float box1_min, float box1_max, float box2_min, float
box2_max)

    {

        if (box2_min < box1_min)

        {

            if (box2_max < box1_min)

                return 0;

            else

                return Mathf.Min(box1_max, box2_max) - box1_min;
        }
        else
        {
            if (box1_max < box2_min)
                return 0;
            else
                return Mathf.Min(box1_max, box2_max) - box2_min;
        }
    }
}
```

Functionality:

A mathematical utility that:

1. Calculates Intersection over Union (IOU) between bounding boxes
2. Handles axis-aligned rectangle intersections
3. Provides precise overlap calculations
4. Used for duplicate detection removal in object detection
5. Contains helper method for interval overlap calculation
6. Returns values between 0 (no overlap) and 1 (complete overlap)

NNHandler.cs

```
using Unity.Barracuda;
using UnityEngine;
public class NNHandler : MonoBehaviour, System.IDisposable
{
    public Model model;

    public IWorker worker;

    public NNHandler(NNModel nnmodel)

    {

        model = ModelLoader.Load(nnmodel);

        worker = WorkerFactory.CreateWorker(model);

    }

    public void Dispose()

    {

        worker.Dispose();

    }

}
```

Functionality:

Neural network wrapper that:

1. Loads Barracuda models from NNModel assets
2. Creates inference workers for model execution
3. Manages model lifecycle
4. Handles resource disposal properly
5. Serves as base class for model-specific handlers
6. Implements IDisposable for clean resource management

ResultBox.cs

```
using Unity.Barracuda;

using UnityEngine;
```

```
namespace NN

{

    public class ResultBox

    {

        public Rect rect;

        public float score;

        public int bestClassIndex;

    }

}
```

Functionality:
A data structure that:

1. Stores object detection results
2. Contains bounding box coordinates (Rect)
3. Holds confidence score (0-1)
4. Tracks detected class index
5. Used as standardized output format for detectors
6. Lightweight container for detection results

UniversalTTS.cs

```
using UnityEngine;

using System.Collections;

#if UNITY_ANDROID && !UNITY_EDITOR

using UnityEngine.Android;
```

```
#endif

public class UniversalTTS : MonoBehaviour

{

    public float speechCooldown = 2f;

    private float lastSpeechTime;

#if UNITY_ANDROID && !UNITY_EDITOR

    private AndroidJavaObject tts;

    private bool isInitialized = false;

#endif



    IEnumerator Start()

    {

#if UNITY_ANDROID && !UNITY_EDITOR

        if (!Permission.HasUserAuthorizedPermission(Permission.Microphone))

        {

            Permission.RequestUserPermission(Permission.Microphone);

            yield return new WaitForSeconds(0.1f);

        }

        InitializeAndroidTTS();

#else

        Debug.Log("TTS ready (Editor mode)");

#endif
```

```
        yield return null;
    }
#if UNITY_ANDROID && !UNITY_EDITOR
    private void InitializeAndroidTTS()
    {
        try
        {
            AndroidJavaClass unityPlayer = new
AndroidJavaClass("com.unity3d.player.UnityPlayer");
            AndroidJavaObject activity =
unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");
            tts = new AndroidJavaObject("android.speech.tts.TextToSpeech", activity, new
AndroidTTSInitListener(this));
        }
        catch (System.Exception e)
        {
            Debug.LogError("TTS Init Error: " + e.Message);
        }
    }

    private class AndroidTTSInitListener : AndroidJavaProxy
    {
        private UniversalTTS parent;
        public AndroidTTSInitListener(UniversalTTS tts) :
base("android.speech.tts.TextToSpeech$OnInitListener")

        {

            parent = tts;

        }

        void onInit(int status)

        {

            if (status == 0)

            {

                parent.isInitialized = true;

                AndroidJavaObject locale = new AndroidJavaObject("java.util.Locale", "en",
"US");

                parent.tts.Call("setLanguage", locale);

            }

        }

    }

#endif

    public void Speak(string text)

    {

        if (Time.time - lastSpeechTime < speechCooldown) return;

#if UNITY_ANDROID && !UNITY_EDITOR

        if (!isInitialized) return;

        tts.Call("speak", text, 0, null, "tts1");
```

```
#else
        Debug.Log("[TTS]: " + text);
#endif
        lastSpeechTime = Time.time;
    }
    void OnDestroy()
    {
#if UNITY_ANDROID && !UNITY_EDITOR
        if (tts != null)
        {
            tts.Call("shutdown");
            tts.Dispose();
        }
#endif
    }
}
```

Functionality:

Text-to-speech system that:

1. Works on Android and in Editor (with debug fallback)
2. Handles Android TTS initialization
3. Manages speech cooldown to prevent spam
4. Provides simple Speak() interface
5. Handles platform-specific implementation details

6. Properly cleans up resources on destruction
7. Includes initialization callback handler

UniversalWebCam.cs

```
using UnityEngine;

using System.Collections;

public class UniversalWebCam : MonoBehaviour

{

    public int requestedWidth = 640;

    public int requestedHeight = 480;

    public int requestedFPS = 30;

    public bool useFrontCamera = false;

    private WebCamTexture webCamTexture;

    private Texture2D outputTexture;

    public Texture2D OutputTexture => outputTexture;

    IEnumerator Start()

    {

        yield return Application.RequestUserAuthorization(UserAuthorization.WebCam);

        if (!Application.HasUserAuthorization(UserAuthorization.WebCam))

        {

            Debug.LogError("WebCam permission not granted!");

            yield break;
        }
        InitWebCam();
    }
    private void InitWebCam()
    {
        WebCamDevice[] devices = WebCamTexture.devices;
        if (devices.Length == 0)
        {
            Debug.LogError("No cameras found!");
            return;
        }
        string cameraName = null;
        foreach (var device in devices)
        {
            if (useFrontCamera && device.isFrontFacing)
            {
                cameraName = device.name;
                break;
            }
            else if (!useFrontCamera && !device.isFrontFacing)
            {
                cameraName = device.name;
```

```
                break;
            }
        }
        if (string.IsNullOrEmpty(cameraName))
            cameraName = devices[0].name;
        webCamTexture = new WebCamTexture(cameraName, requestedWidth,
requestedHeight, requestedFPS);
        outputTexture = new Texture2D(requestedWidth, requestedHeight,
TextureFormat.RGB24, false);
        webCamTexture.Play();
    }
    public Texture2D GetLatestFrame()
    {
        if (webCamTexture == null || !webCamTexture.isPlaying ||
!webCamTexture.didUpdateThisFrame)
            return outputTexture;
        Color32[] pixels = webCamTexture.GetPixels32();
        outputTexture.SetPixels32(pixels);
        outputTexture.Apply();
        return outputTexture;
    }
    void OnDestroy()
    {
        if (webCamTexture != null)
        {
            webCamTexture.Stop();
            Destroy(outputTexture);
        }
    }
}
```

Functionality:

Camera system that:

1. Handles webcam permission requests
2. Manages camera device selection
3. Provides consistent texture output
4. Handles front/back camera selection
5. Maintains proper resource cleanup
6. Offers frame-by-frame texture access
7. Includes fallback for when camera isn't ready

BlindNavigationSystem.cs

```
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
[RequireComponent(typeof(UniversalTTS))]
[RequireComponent(typeof(UniversalWebCam))]
```

```
[RequireComponent(typeof(YOLOHandler))]

public class BlindNavigationSystem : MonoBehaviour

{

    [Header("Navigation Settings")]

    public float safeDistance = 1.5f;

    public float warningDistance = 0.75f;

    [Range(0.01f, 0.5f)]

    public float minObjectWidth = 0.2f;

    [Header("Component References")]

    [SerializeField] private YOLOHandler yoloHandler;

    private UniversalTTS tts;

    private UniversalWebCam webCam;

    private string lastSpokenCommand = "";

    private float lastCommandTime;

    private float lastFrameProcessTime;

    void Start()

    {

        tts = GetComponent<UniversalTTS>();

        webCam = GetComponent<UniversalWebCam>();

        yoloHandler = GetComponent<YOLOHandler>();

        if (yoloHandler == null)

        {
            Debug.LogError("YOLOHandler component missing!", this);
            enabled = false;
            return;
        }
        Debug.Log("Navigation system initialized");
    }
    void Update()
    {
        if (Time.time - lastFrameProcessTime < 0.066f) return;
        lastFrameProcessTime = Time.time;
        Texture2D frame = webCam.GetLatestFrame();
        if (frame == null) return;
        List<ResultBox> detections = yoloHandler.Run(frame);
        AnalyzeNavigation(detections);
    }
    private void AnalyzeNavigation(List<ResultBox> detections)
    {
        if (detections == null || detections.Count == 0)
        {
            SpeakCommand("Path is clear, continue straight ahead");
            return;
        }
        ResultBox mainObstacle = detections
            .Where(o => o.rect.width > minObjectWidth * Screen.width)
            .OrderByDescending(o => o.rect.width * o.rect.height)
            .FirstOrDefault();
        if (mainObstacle == null)
        {
            SpeakCommand("Path is clear");
            return;
        }
        Vector2 screenCenter = new Vector2(Screen.width / 2f, Screen.height / 2f);
        float horizontalOffset = (mainObstacle.rect.center.x - screenCenter.x) / screenCenter.x;
        float distanceEstimate = 1f / (mainObstacle.rect.width / Screen.width);
        string direction = GetDirectionFromOffset(horizontalOffset);
        string distance = GetDistanceDescription(distanceEstimate);
        string objectName = "object";
        SpeakCommand($"{distance} {objectName} detected, {direction}");
        if (distanceEstimate < warningDistance)
        {
#if UNITY_ANDROID && !UNITY_EDITOR
            Handheld.Vibrate();
#endif
        }
```

```
}

private string GetDirectionFromOffset(float offset)

{

    if (Mathf.Abs(offset) < 0.3f) return "move slightly left or right";

    return offset > 0 ? "move left" : "move right";

}

private string GetDistanceDescription(float distance)

{

    if (distance < warningDistance) return "Very close";

    if (distance < safeDistance) return "Close";

    return "Ahead";

}

private void SpeakCommand(string command)

{

    if (string.IsNullOrEmpty(command)) return;



    if (command != lastSpokenCommand || Time.time - lastCommandTime > 5f)

    {

        tts.Speak(command);

        lastSpokenCommand = command;

        lastCommandTime = Time.time;

    }
    }

    void OnDisable()

    {

        lastSpokenCommand = "";

    }

}
```

Functionality:

The main navigation system that:

1. Integrates camera, TTS and YOLO detection
2. Analyzes detected objects for navigation hazards
3. Generates voice commands based on obstacles
4. Provides distance estimation
5. Handles vibration warnings for close objects
6. Implements command throttling to avoid spam
7. Processes frames at controlled rate (~15 FPS)
8. Handles clear path scenarios gracefully

YOLOWrapper.cs

```
using UnityEngine;

using System.Collections.Generic;

using NN;

[RequireComponent(typeof(YOLOHandler))]

public class YOLOWrapper : MonoBehaviour
```

```
{
   private YOLOHandler yoloHandler;
   void Awake()
   {
      yoloHandler = GetComponent<YOLOHandler>();
      if (yoloHandler == null)
      {
         Debug.LogError("YOLOHandler missing! Disabling YOLOWrapper.", this);
         enabled = false;
      }
   }
   public List<ResultBox> RunDetection(Texture2D inputTexture)
   {
      if (yoloHandler == null) return new List<ResultBox>();
      return yoloHandler.Run(inputTexture);
   }}
```

Functionality:

A simple wrapper that:

1. Provides clean interface to YOLOHandler
2. Ensures proper component dependencies
3. Offers fail-safe operation
4. Simplifies detection calls for other components
5. Handles null cases gracefully

6. Serves as abstraction layer for detection system

8. REFERENCES

[1] Zhang, X., Yao, X., Zhu, Y., & Hu, F. (2019). An ARCore based user centric assistive navigation system for visually impaired people. Applied Sciences, 9(5), 989.

[2] Yang, G., & Saniie, J. (2017, May). Indoor navigation for visually impaired using AR markers. In 2017 IEEE International Conference on Electro Information Technology (EIT) (pp. 1-5). IEEE.

[3] Elgendy, M., Herperger, M., Guzsvinecz, T., & Lanyi, C. S. (2019, October). Indoor navigation for people with visual impairment using augmented reality markers. In 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) (pp. 425-430). IEEE.

[4] Ouali, I., Sassi, M. S. H., Halima, M. B., & Ali, W. A. L. I. (2020). A new architecture based ar for detection and recognition of objects and text to enhance navigation of visually impaired people. Procedia Computer Science, 176, 602-611.

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

[6] Zhang, H., & Ye, C. (2016, December). An indoor navigation aid for the visually impaired. In 2016 IEEE international conference on robotics and biomimetics (ROBIO) (pp. 467-472). IEEE.

[7] Joseph, S. L., Zhang, X., Dryanovski, I., Xiao, J., Yi, C., & Tian, Y. (2013, October). Semantic indoor navigation with a blind-user oriented augmented reality. In *2013 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 3585-3591). IEEE.

[8] Abidi, M. H., Siddiquee, A. N., Alkhalefah, H., & Srivastava, V. (2024). A comprehensive review of navigation systems for visually impaired individuals. *Heliyon*..

[9] Fathi, K., Darvishy, A., & Van de Venn, H. W. (2022, September). Augmented reality for the visually impaired: Navigation aid and scene semantics for indoor use cases. In *2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin)* (pp. 1-6). IEEE.

[10] Xie, T., & Seals, C. (2023, January). Design of Mobile Augmented Reality Assistant application via Deep Learning and LIDAR for Visually Impaired. In *2023 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 1-4). IEEE.

[11] Real, S., & Araujo, A. (2019). Navigation systems for the blind and visually impaired: Past work, challenges, and open problems. *Sensors*, *19*(15), 3404.

[12] Du, P., & Bulusu, N. (2021, October). An automated AR-based annotation tool for indoor navigation for visually impaired people. In *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility* (pp. 1-4).

[13] See, A. R., Sasing, B. G., & Advincula, W. D. (2022). A smartphone-based mobility assistant using depth imaging for visually impaired and blind. *Applied Sciences*, *12*(6), 2802.

[14] Sokolov, A., Avrunin, O., Selivanova, K., & Shushliapina, N. (2024, October). Application of Augmented Reality Technologies for Determining Distances in Navigation System for the Blind. In *2024 IEEE 17th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)* (pp. 530-533). IEEE.

[15] Troncoso Aldas, N. D., Lee, S., Lee, C., Rosson, M. B., Carroll, J. M., & Narayanan, V. (2020, October). Aiguide: An augmented reality hand guidance application for people with visual impairments. In *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility* (pp. 1-13).

9. CO- PO MAPPING

Department of CSE-Artificial Intelligence& Machine Learning, GMRIT | Syllabi | Academic Regulation 2021

21PWX01Project Work

0 0 16 8

Course Outcomes

At the end of the project work the students will be able to

1. Identify a contemporary engineering application to serve the society at large
2. Use engineering concepts and computational tools to get the desired solution
3. Justify the assembled/fabricated/developed products intended.
4. Organize documents and present the project report articulating the applications of the concepts and ideas coherently
5. Demonstrate ethical and professional attributes during the project implementation.
6. Execute the project in a collaborative environment.

CO–PO Mapping

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2	-	-	-	3	2	-	-	-	-	-
CO2	3	3	-	-	3	-	-	-	-	-	-	-
CO3	3	3	3	2	-	-	-	-	-	-	2	-
CO4	-	-	-	-	-	-	-	-	-	3		2
CO5	-	-	-	-	-	-	-	3	-	-	-	-
CO6	-	-	-	-	-	-	-		3	-	-	-

3–Strongly linked | 2–Moderately linked | 1–Weakly linked