KLE Technological University
Creating Value, Leveraging Knowledge
Dr. M. S. Sheshgiri Campus, Belagavi

Department of
Electronics and Communication Engineering

Senior Design Project Report

on

Raspberry Pi-Based Object Detection With Speech And Text Output

By:

1. **Sakshi Bagewadi** USN:02FE21BEC080
2. **Samruddhi V S** USN:02FE21BEC083
3. **Srushti Pattanshetti** USN:02FE21BEC100
4. **Ujwala V T** USN:02FE21BEC105

Semester: VII, 2024-2025

Under the Guidance of

Prof.S. M. Hunagund

KLE Technological University,
Dr. M. S. Sheshgiri College of Engineering and Technology
BELAGAVI-590 008
2023-2024

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled **" Raspberry Pi-Based Object Detection With Speech And Text Output "** is a bonafide work carried out by the student team of **"Sakshi Bagewadi 02fe21bec080, Samruddhi V S 02fe21bec083, Srushti pattanshetti 02fe21bec100, Ujwala V T 02fe21bec105"**. The project report has been approved as it satisfies the requirements with respect to the minor project work prescribed by the university curriculum for B.E. (VI Semester) in Department of Electronics and Communication Engineering of KLE Technological University Dr. M. S. Sheshgiri CET Belagavi campus for the academic year 2023-2024.

Prof. S. M. Hunagund	**Dr. Dattaprasad A. Torse**	**Dr. S. F. Patil**
Guide	**Head of Department**	**Principal**

External Viva:

Name of Examiners **Signature with date**

1.

2.

ACKNOWLEDGMENT

I want to thank Prof. S. M. Hunagund for helping me a lot with our project on "Automated polymer bag handling systemg". He gave me great advice, supported us all the way,and worked really hard. His teaching and guidance made our project better. Prof. S. M. Hunagund was not just a teacher but also a mentor, making our learning more practical. He was patient and encouraging when things got tough, and his help was not just about the technical stuff but also about working well together as a team. I'm really thankful for his support, and our project wouldn't be as good without him. I extend my sincere gratitude to our Head of Department, Dr. Dattaprasad A. Torse, and our college Principal, Dr. S. F. Patil, for their invaluable support during our project on "Raspberry Pi-Based Object Detection With Speech And Text Output". Dr.Torse provided crucial guidance and encouragement, significantly improving our project with insightful advice. I appreciate the conducive learning environment fostered by both Dr. S. F. Patil and Dr. Dattaprasad A. Torse , which allowed us to apply theoretical knowledge to real-life situations. Their leadership has been instrumental in shaping our academic journey, and I am thankful for their unwavering support and contributions to our project's success.

-The project team

ABSTRACT

This project focuses on developing a Raspberry Pi-based object detection system with integrated speech and text output capabilities. Using a PiCam for real-time image capture, the system employs advanced pretrained YOLOv5 or YOLOv8 models to detect and identify objects in live video streams. The detected objects are displayed on a screen as text and conveyed audibly using a text-to-speech engine, offering dual feedback for accessibility and usability. The system is optimized for efficiency, ensuring smooth performance despite the Raspberry Pi's hardware limitations. Its compact and portable design makes it suitable for various real-world applications such as assistive technology, smart home automation, and surveillance systems.

This solution is tailored to enhance user interaction through seamless integration of real-time image processing and machine learning algorithms. Designed with versatility in mind, the project aims to aid visually impaired individuals by providing auditory descriptions of their surroundings. Additionally, it can be applied in security systems for detecting suspicious objects or activities. The combination of low-cost hardware and state-of-the-art algorithms ensures affordability without compromising accuracy or reliability. By bridging technology with accessibility, the project demonstrates the potential of small-scale computing devices in solving everyday challenges effectively

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The advancement of artificial intelligence and machine learning has enabled the development of powerful object detection systems capable of identifying and recognizing objects in real-time. These systems, when integrated with accessible computing platforms like the Raspberry Pi, open the door to a wide range of practical applications. This project leverages the capabilities of Raspberry Pi and pretrained YOLOv5/YOLOv8 models to create an efficient, low-cost object detection system. Equipped with a PiCam for real-time image capture, the system is designed to detect objects in its environment and provide both text-based and speech-based feedback.

The integration of text-to-speech technology allows the system to audibly announce detected objects, making it especially useful for visually impaired individuals. Simultaneously, the detected objects are displayed on a screen, providing a visual representation of the analysis. By combining these outputs, the project ensures inclusivity and usability for a diverse audience. Its compact, portable design makes it adaptable for various use cases, including home automation, security, and assistive devices, demonstrating the flexibility of the Raspberry Pi in addressing real-world challenges.

This project also emphasizes the efficiency of implementing advanced machine learning algorithms on a lightweight platform. Despite the computational limitations of the Raspberry Pi, the system delivers accurate and reliable object detection in real-time. The use of YOLO models ensures high performance, enabling the identification of multiple objects simultaneously. By exploring the synergy of hardware and software, this project showcases the potential of small-scale computing devices to deliver impactful, cost-effective solutions that enhance daily life.

1.1 Motivation

The motivation for this project stems from the growing need for accessible, cost-effective, and real-time object detection systems that can address diverse societal challenges. As technology advances, the gap between its capabilities and its accessibility to underserved communities continues to widen. This project aims to bridge that gap by utilizing the Raspberry Pi, an affordable yet versatile computing platform, to implement cutting-edge machine learning algorithms for object detection. The integration of speech and text outputs enhances usability, making the system particularly beneficial for visually impaired individuals who rely on auditory feedback to navigate their surroundings.

Furthermore, the demand for intelligent and adaptable systems in domains such as home automation, surveillance, and assistive technology has never been higher. Conventional solutions in these areas are often expensive or require high-end hardware, limiting their adoption. This project seeks to provide a low-cost alternative that does not compromise on performance or reliability. The portability and flexibility of the Raspberry Pi-based system make it suitable for deployment in various environments, from households to public spaces, reinforcing its practical significance.

This work is driven by the aspiration to showcase how small-scale hardware, when combined with advanced machine learning algorithms, can deliver impactful results. By exploring innovative ways to enhance the accessibility and applicability of technology, the project aims to contribute meaningfully to the growing field of embedded AI systems while addressing real-world challenges effectively.

1.2 Objectives

- Accurate Object Identification : Employ pretrained YOLOv5/YOLOv8 models to ensure reliable and efficient object detection.
- Dual Feedback Mechanism : Provide visual feedback via text displayed on a screen and auditory feedback through text-to-speech for enhanced user interaction.
- Accessibility and Inclusivity : Design the system to assist visually impaired individuals by describing detected objects audibly.
- Cost-Effective Design : Utilize affordable components to ensure the system is low-cost while maintaining high performance.
- Portability and Flexibility : Build a compact, lightweight, and adaptable system suitable for diverse applications such as assistive technology, home automation, and surveillance

1.3 Literature survey

Table 1.1: Literature Review

SI.No	Paper Title	Objective	Algorithm used	Limitations
1	[1]	Detects objects in images without requiring prior knowledge of their specific classes. It is efficient and scalable to handle a large number of object classes.	The core of the proposed model is a DCNN, which is trained to predict a set of bounding boxes and their corresponding confidence scores.	Might not be sufficient for images with a large number of objects.
2	[2]	Ensure that the object detection process is efficient and can recognize multiple objects in a timely manner.	Utilize machine learning algorithms and computer vision techniques to analyze visual data and identify objects.	The system might struggle in low-light or challenging lighting conditions.
3	[3]	Develop a text-to-speech conversion system using a Raspberry Pi for visually impaired people	Edge Detection Algorithms like Sobel, Kirsch, Canny, or Prewitt can be used to identify text edges in the image	The system can only convert captured images to speech.
4	[4]	Develop a system to aid blind and visually impaired individuals in navigating their surroundings.	Stereoscopic Sonar Algorithm to process ultrasonic sensor data and create a 3D representation of the environment.	The accuracy of obstacle detection might be affected by factors like environmental noise, object reflectivity.

1.4 Problem statement

Existing object detection systems are often expensive, require high-performance hardware, or lack accessibility features, making them unsuitable for resource-constrained environments or underserved populations. Visually impaired individuals face challenges in identifying objects due to costly or impractical devices. This project addresses these issues by developing a cost-effective, portable Raspberry Pi-based system with dual feedback through speech and text output.

1.5 Application in Societal Context

- Assistive Technology for the Visually Impaired : Provides auditory descriptions of objects, enabling visually impaired individuals to navigate their surroundings with greater independence.
- Affordable Accessibility Solutions : Offers a low-cost alternative to expensive assistive devices, making advanced technology accessible to underserved communities.
- Home Automation and Security : Enhances home safety by identifying objects or potential hazards and notifying users in real-time.
- Educational Tools : Serves as an interactive learning aid for students to understand AI, machine learning, and object recognition technologies.
- Community Safety and Monitoring : Helps in identifying and alerting about suspicious objects in public spaces, contributing to community safety.
- Inclusivity in Technology : Demonstrates the integration of affordable technology in solving real-world challenges, promoting inclusivity and social equity.

1.6 Project Planning and bill of materials

- Project initiation -

  The initiation phase involves defining the project's objective to develop a Raspberry Pi-based real-time object detection system with dual feedback via speech and text output. Feasibility analysis is conducted to evaluate the technical and financial viability of running YOLO models on the Raspberry Pi, considering hardware constraints and cost estimations. Necessary resources, including the Raspberry Pi, PiCam, speakers, and software libraries like OpenCV and PyTorch, are identified and procured. Stakeholders, including end-users, educators, and technical advisors, are engaged to align the project with real-world needs, incorporating their feedback throughout development. A detailed timeline with milestones for initiation, planning, testing, and deployment is established, ensuring progress remains on track while accommodating potential delays

- Planning phase –

  Currently focusing on

1 System Design: Develop a system architecture, specifying the hardware setup, image processing workflow, and feedback mechanisms.

2 Software Selection: Choose appropriate libraries and frameworks (e.g., OpenCV, PyTorch) for implementing object detection and text-to-speech functionalities.

3 Risk Assessment: Identify potential risks, such as hardware limitations or software compatibility issues, and plan mitigation strategies.

4 Test Plan Creation: Design test cases for validating object detection accuracy, speech output clarity, and system performance under different conditions.

5 Team Coordination: Assign roles and responsibilities to team members to ensure smooth execution of tasks.

- Simulation testing -

1 Model Testing: Test YOLO models on sample datasets to evaluate detection accuracy and optimize performance for Raspberry Pi.

2 System Integration: Integrate the PiCam, object detection software, and text-to-speech engine into a cohesive system.

3 Simulation Setup: Simulate real-world scenarios to test object detection in various environments (e.g., lighting, object density).

4 Performance Evaluation: Measure system latency, detection accuracy, and feedback response times, and refine as needed.

5 User Testing: Conduct trials with potential end-users (e.g., visually impaired individuals) to gather feedback on usability and effectiveness.

6 Final Optimization: Address identified issues, optimize code, and ensure the system meets all performance and usability goals.

  Gather feedback and make necessary adjustments

1.7 Organization of the report

- Introduction: - Brief overview and rationale for the project.
- Literature Review: - Summary of existing automated bag handling algorithms and applications.
- Objectives: - Clear definition of project goals and objectives.
- Methodology: - Explanation of the algorithm design process, and optimization strategies.
- System Architecture: - Overview of the overall system architecture.
- Algorithm Methodology: - Detailed explanation of algorithmic design principles, emphasizing optimization considerations.
- Simulation Testing: - Overview of testing procedures and presentation of simulation results.
- Ethical Considerations: - Discussion of ethical considerations related to data security.
- Results and Discussion: - Summary of key findings, comparative analysis, and optimization impact.

Chapter 2

System design

In this Chapter, we list out the interfaces. Modules:

- User Interface (UI): it doesn't have a dedicated UI, but you can integrate one for user interaction (e.g., buttons to trigger capture).
- OpenCV Library:Handles camera access, frame reading, image/video processing, and saving.
- Operating System (OS):Provides access to the webcam hardware and filesystem for saving captures.

Data Flow:

- Capture Function Selection:The user (or script) chooses between capture image and capture video functions.
- Camera Access:The chosen function opens the default webcam using cv2.VideoCapture
- Image Capture (capture image):A single frame is read using cap.read(). The frame is saved as a JPEG image using cv2.imwrite.
- Video Capture (capture video):Frames are continuously read from the camera using cap.read() in a loop.Each frame is written to the video file using out.write().The captured frames are displayed in a window named "Recording" using cv2.imshow.
- Resource Release:In both functions, the camera (cap) and video writer (out) objects are released using cap.release() and out.release(), respectively.OpenCV windows are closed using cv2.destroyAllWindows().
- User Feedback:The functions print messages indicating success or failure of capture operations and the saved file path.

2.1 Functional block diagram

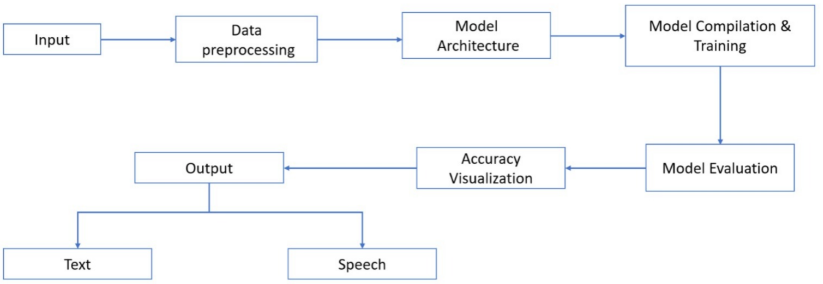

Figure 2.1: Functional block diagram

2.2 Design alternatives

- Hardware Design Alternative

1 Different Raspberry Pi Models: Consider using a more powerful Raspberry Pi model, such as the Raspberry Pi 4 or Raspberry Pi Compute Module 4, for improved performance and faster processing.

- Benefits: Portable, low-power, and compatible with headless systems (no monitor required). Challenge: Lower frame rates compared to USB webcams.

2 Alternative Cameras: Explore using other camera options, such as USB webcams or higher-resolution cameras, to suit specific needs and budgets. * Additional Sensors: Integrate additional sensors, like distance sensors or infrared sensors, to enhance the system's capabilities and enable more complex applications.

- Software Alternatives:

1 Different Object Detection Models: Experiment with other object detection models, such as EfficientDet or SSD MobileNet, to compare performance and accuracy.

2 Custom Model Training: Train a custom object detection model on a specific dataset to improve accuracy for particular objects or scenarios.

3 Different Text-to-Speech Engines: Use alternative text-to-speech engines, such as Google Text-to-Speech or Amazon Polly, to explore different voices and languages.

- Application Alternatives:

1 Home Automation: Expand the system's capabilities to control smart home devices, such as lights, thermostats, or security systems, based on object detection.

2 Security Surveillance: Utilize the system for security purposes, such as detecting intruders or monitoring specific areas.

3 Assistive Technology: Develop specialized applications for visually impaired individuals, such as object recognition for navigation or product identification.

4 Educational Tools: Create interactive learning tools for children, such as object recognition games or educational videos.

- Additional Considerations:

1 Power Efficiency: Optimize the system's power consumption to extend battery life and reduce energy usage.

2 Robustness: Implement error handling and recovery mechanisms to ensure the system's reliability.

3 User Experience: Design an intuitive and user-friendly interface that is easy to navigate and use.

4 Ethical Considerations: Address potential privacy and security concerns, especially when dealing with sensitive data or surveillance applications.

2.3 Final design

We select one of the optimal solutions based on its working and ease of the implementation.

- Final Design:

1 Hardware Components

Raspberry Pi 4 : Acts as the central processing unit for running machine learning algorithms, handling image processing, and managing system control. Its quad-core CPU and sufficient RAM make it ideal for real-time applications.

PiCam (Raspberry Pi Camera): Captures live video frames for object detection. It offers high-resolution image capture, essential for accurate object recognition.

Bluetooth Module: Enables wireless communication, allowing data exchange with external devices (like smartphones or tablets) for control, notifications, or system updates.

Power Supply: Provides stable power to ensure uninterrupted system operation.

2 System Workflow

Image Capture: PiCam captures video frames in real time.

Object Detection: The frames are processed using YOLOv5/YOLOv8 models on the Raspberry Pi 4. The system identifies and classifies objects in each frame.

Output Generation:

- Text Output: The system displays the detected object's name on a connected display or a console.
- Speech Output: The system uses a TTS engine to vocalize the detected object's name or classification.
- Bluetooth Integration: The detected information is shared with external devices via Bluetooth for notifications or control.

3 Features and Benefits

Portability: Small, lightweight, and standalone, making it easy to deploy.

Real-Time Detection: YOLO models ensure fast and accurate detection.

Multi-Modal Output : Visual (text) and auditory (speech) output increase accessibility and user interaction.

Wireless Communication : Bluetooth enables remote monitoring and updates.

Chapter 3

Implementation details

3.1 Specifications and final system architecture

Specifications:

- Hardware Requirements

1 Raspberry Pi (preferably 4B for better processing power)

2 Pi Camera (or USB webcam)

3 Power Supply (5V, 3A)

4 MicroSD Card (16GB or higher)

- Software Setup

1 OS Installation:Flash Raspberry Pi OS (Lite or Desktop) to the SD card using Raspberry Pi Imager.

2 Environment Setup:

3 Install YOLOv5 or YOLOv8

4 Camera Integration

3.1.1 Model specifications

- Hardware Specifications

  Raspberry Pi 4

- Processor: Quad-core ARM Cortex-A72 (1.5 GHz)
- RAM: 2GB, 4GB, or 8GB (depends on system requirements; 4GB or 8GB is recommended for YOLO models)
- Storage: MicroSD card (minimum 32GB for OS, libraries, and model files)
- Ports: USB 3.0, HDMI, GPIO pins (for hardware integration if required)
- Operating System: Raspberry Pi OS (Linux-based)

  PiCam (Raspberry Pi Camera)

- Resolution: 8MP or 12.3MP (depending on the PiCam version)
- Frame Rate: Up to 30 fps (for real-time object detection)
- Interface: CSI (Camera Serial Interface)

  Bluetooth Module

- Version: Bluetooth 4.2 (built into Raspberry Pi 4) or optional external Bluetooth module if extended range is required.
- Range: 10 meters (depending on obstacles)
- Power Supply
- Input: 5V/3A USB-C power adapter for Raspberry Pi 4.
- Software Specifications

  Operating System: Raspberry Pi OS (Debian-based)

  Programming Language: Python (for YOLO, machine learning scripts, and TTS)

  Libraries and Packages :

- YOLOv5/YOLOv8: Pre-trained object detection models for real-time image recognition.
- OpenCV: For image processing and interfacing with PiCam.
- PyTorch: Required to run YOLO models for object detection.
- NumPy: For mathematical operations related to image processing.
- Text-to-Speech (TTS) Engine: eSpeak, pyttsx3, or Google Text-to-Speech for speech output.
- Bluetooth Libraries: PyBluez or Bleak for Bluetooth communication.

3.1.2 Flow chart

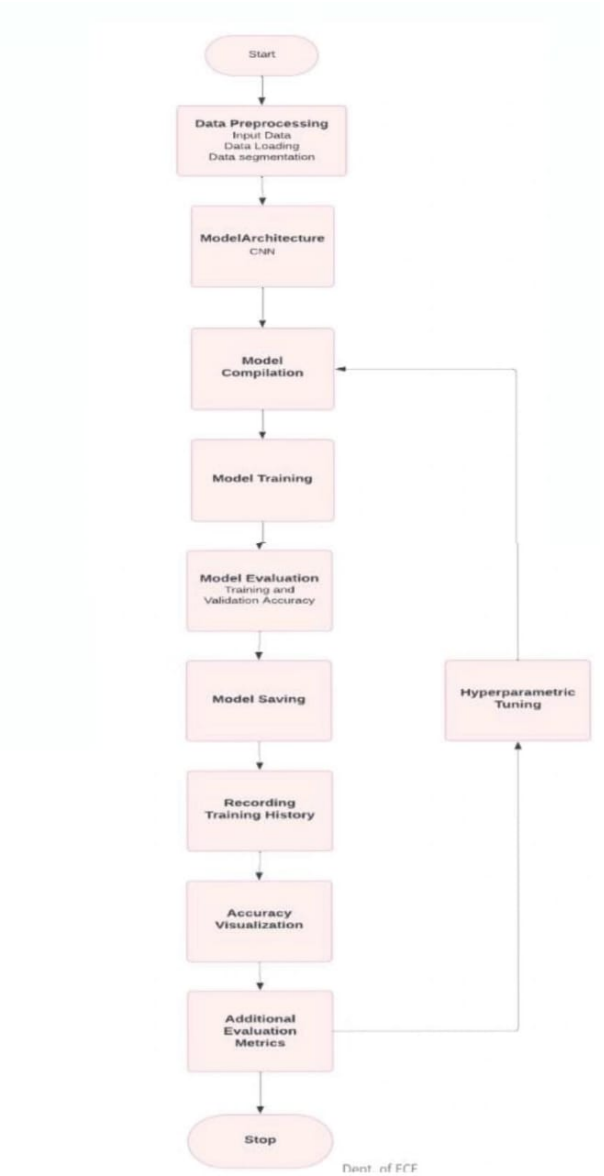

Figure 3.1: Flow chart

Chapter 4

Optimization

4.1 Introduction to optimization

Object detection on a Raspberry Pi faces unique challenges due to its limited computational power, memory constraints, and real-time processing needs. Optimization in this context refers to improving the system's speed, accuracy, and efficiency to achieve real-time object detection with minimal hardware overhead.

4.2 Types of Optimization

1 Model Optimization

Use a smaller model: Use YOLOv5n (Nano) or YOLOv8n.

Quantization: Convert the model to FP16 or INT8 with TensorRT or ONNX.

Prune the model: Remove unimportant weights to reduce computation.

2 Hardware Acceleration

Use a Coral USB Accelerator: Leverage TPU for faster inference.

Enable GPU support: If using Raspberry Pi 4, use OpenCV with OpenCL.

Overclock the Pi: Boost CPU/GPU frequency (with caution to avoid overheating).

3 System Optimizations

Reduce background services: Stop unnecessary Raspberry Pi services.

Increase swap memory: Useful if the Pi is memory constrained.

Use lightweight OS: Raspberry Pi OS Lite has less overhead.

4.3 Model Optimization

Since the Raspberry Pi has limited computational power, model optimization focuses on reducing the size, complexity, and computational demand of object detection models while maintaining acceptable accuracy. Key techniques include using smaller models like YOLOv5n or YOLOv8n, which have fewer parameters and require less computation. Quantization converts weights from FP32 to FP16 or INT8, significantly reducing model size and inference time. Pruning removes unimportant weights, neurons, or channels, reducing model complexity while preserving performance. These methods can be combined for maximum efficiency, enabling faster inference, lower memory usage, and real-time detection on resource-constrained devices like Raspberry Pi.

4.4 Hardware Acceleration

To optimize object detection on a Raspberry Pi, use a Google Coral TPU for 10x faster inference or enable OpenCL GPU support for parallel processing, though it's limited. Overclocking the CPU/GPU boosts FPS but requires proper cooling. Raspberry Pi 4+ is recommended for better CPU, GPU, and RAM. Use USB 3.0 for faster connections with Coral TPU and PiCam. These techniques together improve speed, efficiency, and system performance.

4.5 System Optimizations

To improve the performance of object detection on Raspberry Pi, system optimizations aim to free up resources and reduce overhead. One key approach is to reduce background services by stopping unnecessary system processes and daemons (like Bluetooth, printing, or desktop GUI), which frees up CPU, RAM, and I/O bandwidth for the object detection task. Increasing swap memory is another useful technique, especially for models that require more RAM than the Pi can provide. By increasing the swap file size, the system can handle larger models, though this may slightly slow down performance due to SD card read/write speeds. Lastly, using a lightweight OS like Raspberry Pi OS Lite (headless, without a desktop environment) minimizes system overhead, providing more CPU, RAM, and I/O resources for object detection. Together, these optimizations ensure smoother, faster, and more reliable detection performance on Raspberry Pi.

Chapter 5

Results and discussions

5.1 Result Analysis

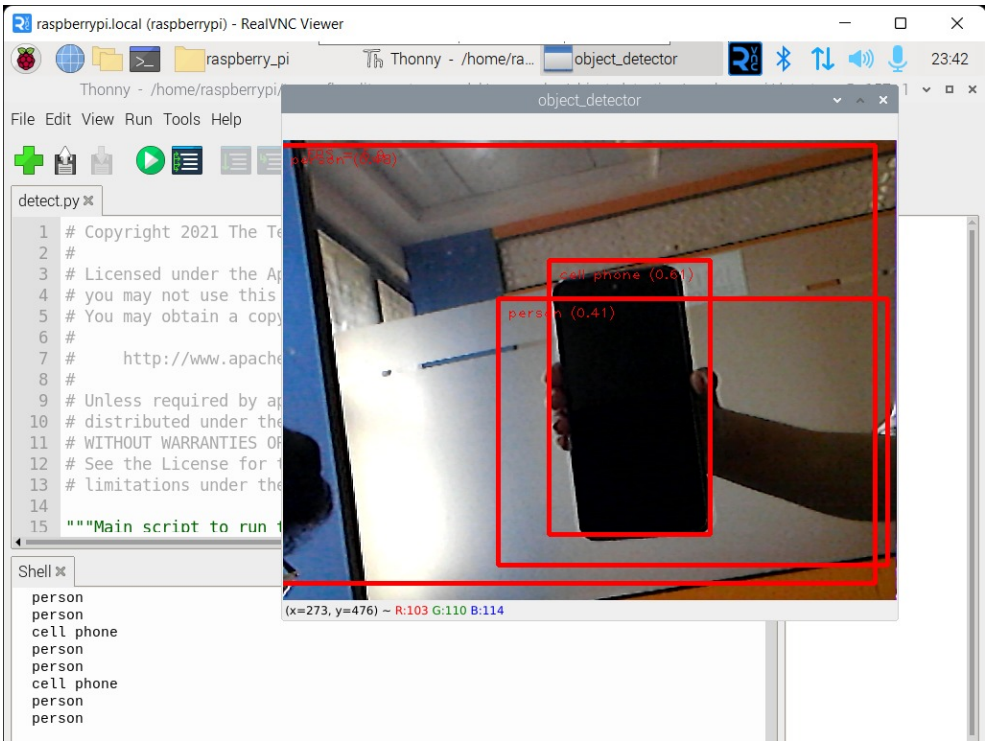

Figure 5.1: Result

Chapter 6

Conclusions and future scope

6.1 Conclusion

In this project, we successfully implemented object detection on a Raspberry Pi by leveraging a combination of hardware, model, and system optimizations to achieve efficient, real-time performance on a low-power device. On the model side, we used smaller models like YOLOv5n or YOLOv8n, applied quantization to reduce precision and employed pruning to reduce model complexity. System-level improvements, such as stopping unnecessary background services, increasing swap memory, and using a lightweight OS (like Raspberry Pi OS Lite), further enhanced the Pi's ability to handle object detection efficiently.

By integrating these techniques, we achieved significant improvements in inference speed, memory usage, and energy efficiency, making real-time object detection feasible on Raspberry Pi. This project demonstrates how low-cost edge devices can support AI-driven applications, making them suitable for IoT, surveillance, robotics, and automation. Future work could explore custom training of YOLO models for specific use cases and incorporating edge accelerators like the Google Coral Edge TPU for even faster performance.

6.2 Future scope

- Custom Object Detection:

  Train the system on custom datasets to identify specific objects relevant to visually impaired users (like public signs, personal items, or vehicles), making the system more context-aware and user-friendly.

- Improved Feedback Systems:

  Enhance feedback options with vibration alerts, haptic feedback, and customized voice commands to make it more accessible for visually impaired users. Real-time guidance (like directional cues) can be added to assist users in locating objects.

- Commercialization and Affordability:

  Efforts could be made to commercialize the system as an affordable assistive device for visually impaired people. Partnerships with nonprofits or government agencies could facilitate large-scale distribution to underserved populations.

- Portability and Power Efficiency:

  Develop a more compact, battery-powered version for increased portability. This could involve integrating a low-power display or touchscreen for settings and status updates

Bibliography

1 Erhan, D., Szegedy, C., Toshev, A. and Anguelov, D., 2014. Scalable object detection using deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2147-2154).

2 Mandhala, V.N., Bhattacharyya, D., Vamsi, B. and Thirupathi Rao, N., 2020. Object detection using machine learning for visually impaired people. International Journal of Current Research and Review, 12(20), pp.157-167.

3 Lakshmi, K., 2016. Design and Implementation of Text to Speech conversion using Raspberry pi. Dimensions, 85, p.x56mm.

4 Hayashi, T. and Watanabe, S., 2020. Discretalk: Text-to-speech as a machine translation problem. arXiv preprint arXiv:2005.05525.