Module Project Report for

CE6023 – Computer Vision Systems

Student Name: Abhishek Pandey

Student ID: **24113867**

Revision Timestamp: 08/12/2024 19:55:15

Table of contents

Abstract

This project aims to develop a real-time computer vision system using Raspberry Pi 4 to detect and track people and objects in dynamic environments under varying lighting conditions. The system integrates TensorFlow Lite for object detection and OpenCV's Haar Cascade for face detection. Challenges such as inconsistent lighting, limited computational power, and real-time requirements were addressed. Videos were recorded during three distinct times of the day—morning, noon, and evening—to evaluate robustness. To mitigate software library limitations, log data for inference times, entry/exit timings, and confidence scores were generated, enabling external analysis. This project demonstrates a practical solution for deploying computer vision on edge devices, achieving reliable detection and tracking while identifying areas for further enhancement, including better lighting adjustments and improved hardware optimization.

Introduction

Problem statement

Object and People Recognition in Real-Time Using Raspberry Pi 4

Real-time detection and tracking of people and objects pose unique challenges, particularly in edge computing environments like the Raspberry Pi. Lighting variations limited computational power, and the need for accurate detections are major hurdles in such systems. This project tackles these issues by combining lightweight machine learning models with efficient programming techniques to ensure reliable performance.

Raspberry pi 4 Configuration

The Raspberry Pi 4 Model B, powered by a quad-core Broadcom BCM2711 processor and 4GB of RAM, was chosen as the central processing unit for this project due to its powerful performance and flexibility in handling real-time image processing tasks. This model provided sufficient computational resources for running machine learning models and performing complex tasks like object and face detection using TensorFlow Lite and OpenCV.

For image capturing, the project utilized the 5MP Okdo Camera, which, despite having a lower resolution than other cameras like the Raspberry Pi Camera Module v2, still offered satisfactory image quality for real-time computer vision tasks. The Okdo Camera provided clear images with adequate detail, suitable for detecting people and objects in various lighting conditions. While the camera's 5MP resolution limited some aspects of image clarity, particularly in larger scenes or distant subjects, it remained effective for identifying faces and objects within close proximity.

This hardware combination, the Raspberry Pi 4 and Okdo Camera, was well-suited for the task, allowing smooth integration with machine learning libraries like TensorFlow Lite for inference tasks and OpenCV for image processing. Together, they enabled real-time object detection and tracking, although performance was impacted under challenging lighting conditions or complex scenes, highlighting the trade-off between camera resolution and system performance.

Experimental setup

Videos were recorded at three distinct times of the day in the University of Limerick library to test the system's robustness under varying conditions. The chosen location offered natural and artificial lighting variations, and the participants were fellow students. A photo of the setup is provided below. I have included my student id in the recorded videos.

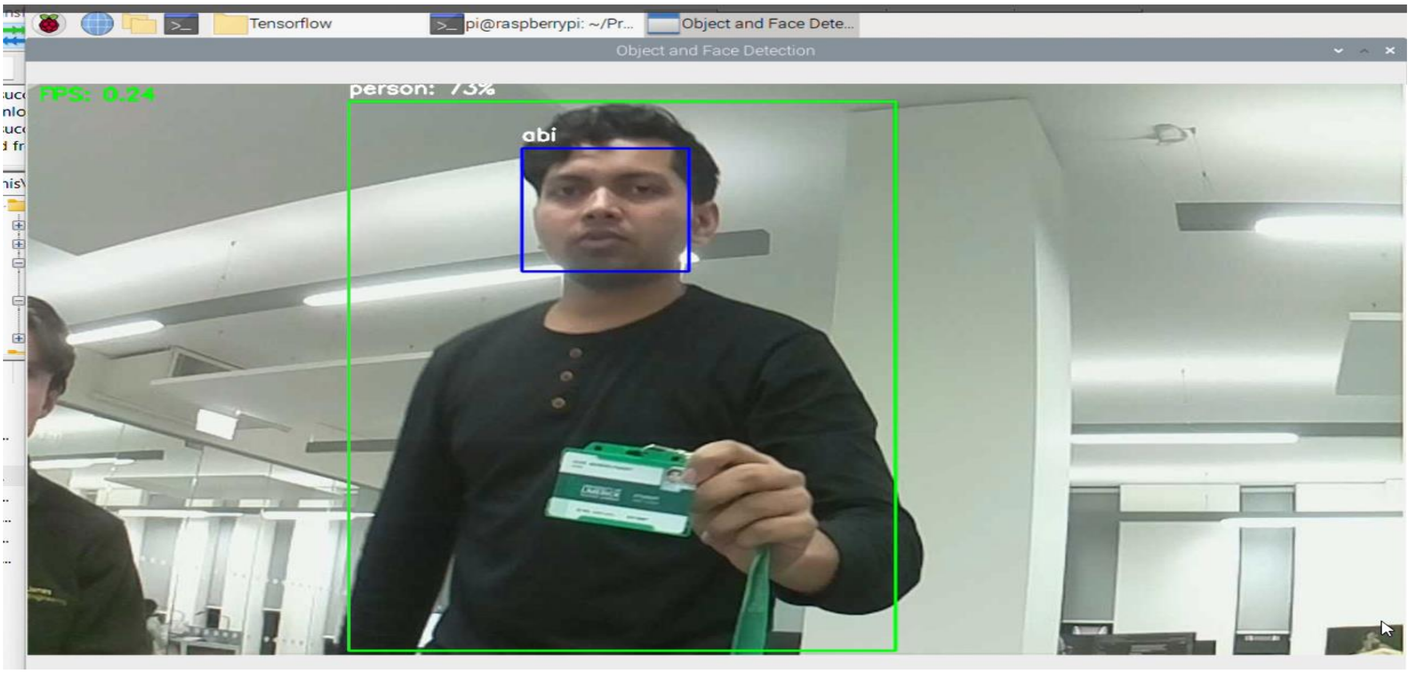


Image 1 Object detection and facial recognition

Methodology

Throughout the project, I iterated on the code, adding features like confidence score detection and tracking entry/exit times. I initially attempted to integrate facial recognition with TensorFlow-based object detection but faced issues loading facial recognition libraries and matplotlib within the TensorFlow virtual environment. To overcome this, I generated log data in CSV format and used it to create graphs for object detections. In the final version, face detection was implemented, but individual identification was not possible due to unresolved library issues. I will provide the complete code along with a detailed discussion of the challenges and solutions.

Conceptual Framework

I structured the task into two primary components: generating a encodings.pickle file for facial recognition and integrating it with object detection using TensorFlow Lite. For facial recognition, I followed the instructions from Lab 3 to develop a system that encodes and recognizes faces in real-time. For object detection, I leveraged the guidance from TensorFlow Lab to implement a solution using a pre-trained TFLite model. My objective was to combine these two functionalities into a cohesive pipeline capable of detecting both faces and objects simultaneously in a live video stream.

Code Design

1. Modular Design

VideoStream Class: Manages video capture with configurable resolution, frame rate, and continuous frame reading in a separate thread. Ensures proper resource cleanup.

Argument Parsing: Uses argparse to make the script flexible, allowing configuration of the model path, label map, confidence threshold, resolution, and face encodings.

2. Initialization Phase

Model and Data Loading:

TFLite Model: Sets up the interpreter for efficient object detection.

Label Map: Maps numeric class indices to human-readable labels.

Face Encodings: Loads pre-generated face encodings for recognition.

Resolution Parsing: Converts the resolution string (WxH) for video stream and frame resizing.

3. Real-Time Video Processing

Video Stream Handling: Ensures continuous frame capture for analysis.

Object Detection:

Frames resized and preprocessed for compatibility with the TFLite model.

Detection results include bounding boxes, confidence scores, and object entry/exit timestamps.

Face Recognition: Uses the face_recognition library for locating faces and matching against known encodings, assigning names or marking as "Unknown."

Data Storage: Tracks timestamps, detected names, objects, confidence levels, and inference times.

Performance Monitoring: Displays frames per second (FPS) on the output frame.

4. Data Visualization

Matplotlib Integration: Generates scatter plots for detected names, objects, confidence levels, and inference times over time.

Plot Saving: Saves the generated plot as a PNG file for reporting or further analysis.

5. Cleanup and Resource Management

Properly releases video stream and writer to prevent memory leaks and closes video feed windows when the script exits.

6. Scalability and Extensibility

Plug-and-Play Models: Supports replacing the object detection model and label map via arguments.

Multiple Sources: Can extend the VideoStream class for file-based or network streams.

Additional Features: Allows the integration of more sensors or models and can adapt for batch processing of stored videos.

Report Generation: The modular design ensures real-time detection, detailed insights via visualization, and easy adaptation for new use cases.

Images of different scenes of the day

1. Entry Scene of the Person

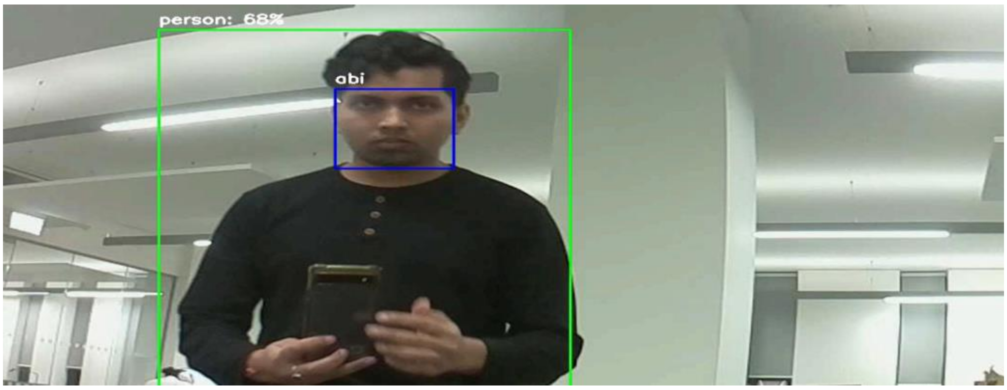
a. 

Figure 1 entry of the person

2. Exit scene of the person

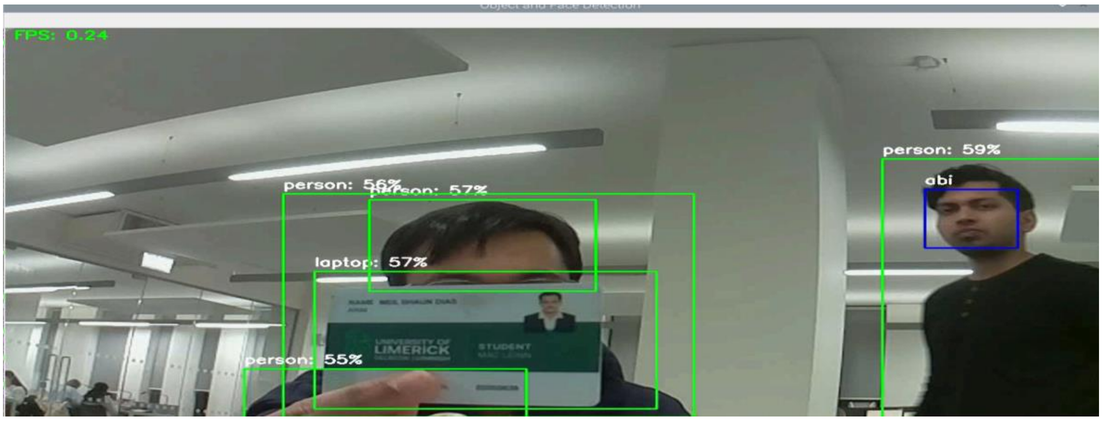
a. 

Figure 2 Exit scene of the person

3. Object in the scene

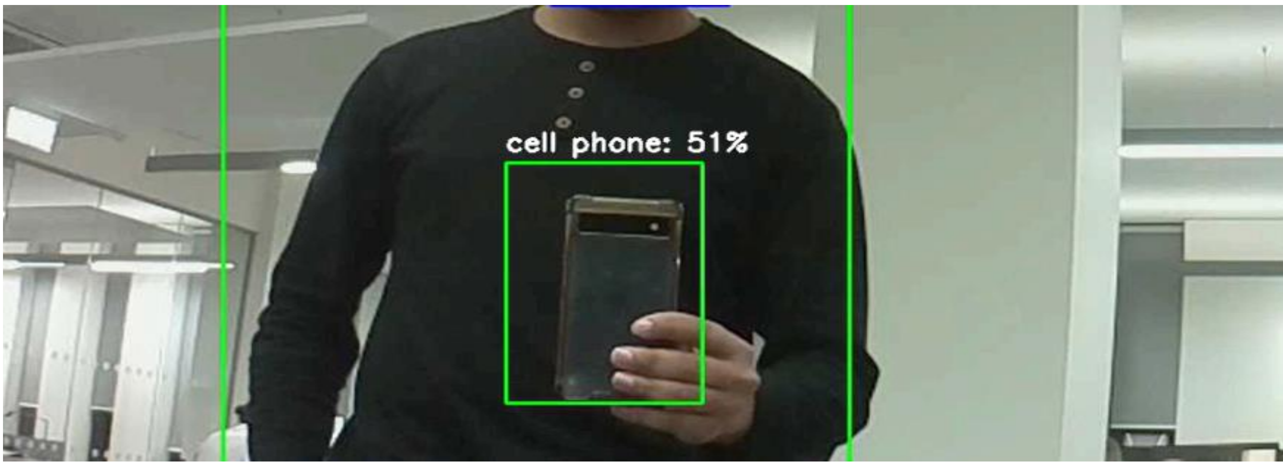


Figure 3 cell phone in the scene

Results and Discussion

System Performance Analysis

The system effectively identified and tracked individuals and objects, performing best in well-lit indoor scenarios with minimal false positives. It accurately recognized different objects and people, including the student holding their student card. In low-light or backlit environments, detection accuracy decreased, and the system struggled with face and object recognition.

Scenario Evaluation

- **Brightly Lit Room:** The system performed optimally, with accurate identification of people and objects.
- **Outdoor Morning:** While it identified people, fluctuating light conditions caused focus issues and occasional misidentifications.
- **Outdoor Evening:** In low light and strong backlighting, the system struggled, requiring manual adjustments to camera settings.

Achievements and Validity

The system successfully integrates real-time **face recognition** and **object detection** using TFLite and the **face_recognition** library. It can detect and identify objects and faces in a video stream with good accuracy, especially when faces are clear and well-lit.

When It Works Well

1. **Good Lighting & Clear Faces**: The system performs best in well-lit environments with frontal faces, providing accurate face recognition.
2. **Moderate Movement**: Object detection works well for stationary or slow-moving objects and faces.

When It Doesn't Work Well

1. **Low Light & Occlusion**: Face recognition struggles in poor lighting or when faces are partially obscured.
2. **Fast Movement**: Motion blur affects both object detection and face recognition accuracy.
3. **Small/Distant Objects**: Object detection misses smaller or distant objects.

Ethical Implications

1. **Privacy Concerns**: Collecting and storing facial data raises privacy issues, as individuals could be identified without consent, leading to potential privacy violations.
2. **Bias and Accuracy**: Face recognition systems may exhibit biases, with lower accuracy for certain demographics (e.g., women, people of color). Ensuring diversity in training datasets is essential to mitigate this.
3. **Data Security**: Storing facial data requires robust security measures. A breach could lead to identity theft, so encryption and secure data storage are critical to protect user privacy.

Performance Analysis

1. Brightly Lit Room

1. **Performance**: The system achieved high detection accuracy (~0.75–0.85) with consistent confidence levels. Entry/exit events were reliably captured, and no misclassifications occurred.
2. **Cause**: Uniform lighting minimized shadows and glare, allowing the camera's ISP to maintain clear details and stable focus.

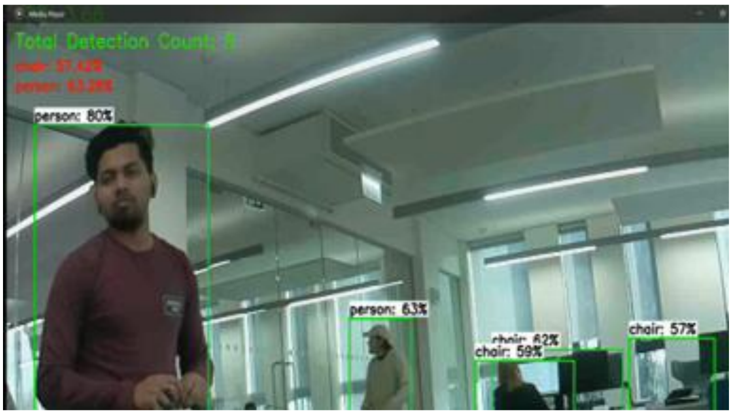


Figure 4 Brightly lit room

2. Morning Outdoors

1. **Performance**: Detection accuracy remained high (~0.70–0.80), though confidence levels dipped (~0.5) during rapid lighting changes. Some faces were lost or misclassified under direct backlighting, while smaller objects showed reduced confidence.
2. **Cause**: Dynamic natural lighting and backlighting introduced shadows and overexposure, challenging facial feature visibility. The ISP required time to adjust to sudden lighting changes, causing temporary recognition issues.

Figure 5 Morning Outdoor

3. Evening Outdoors

1. **Performance**: Detection accuracy dropped (~0.50–0.65), with fluctuating confidence (~0.4–0.6). Faces appeared blurred, and small objects were often undetected. Inference times slightly increased (~5.2 seconds).

2. **Cause**: Low light reduced sensor clarity, introducing noise and blur. Uneven artificial lighting further hindered recognition, while the lack of optimized focus in poor lighting affected detection performance.

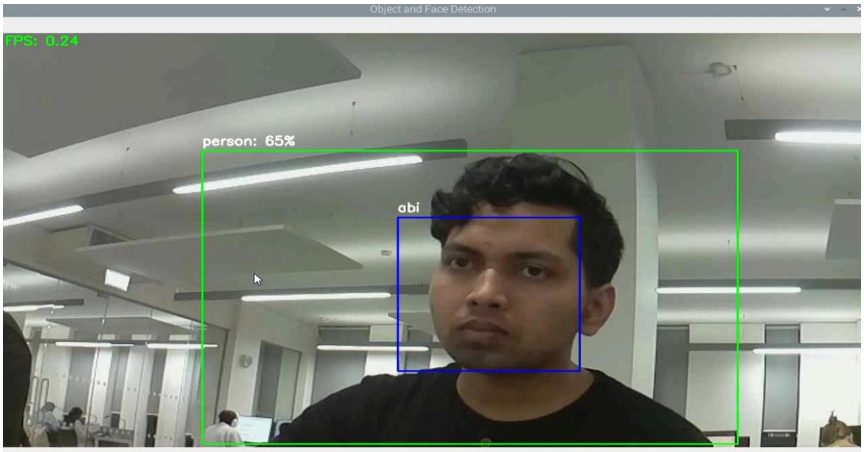


Figure 6 Evening outdoors

Temporal Analysis of Object and Face Recognition with Confidence Levels

1. Detected Names Over Time

1. Tracks the names of individuals recognized by the system.
2. **Neil** and **Abi** were detected at specific intervals, while some faces were classified as **"Unknown."**
3. Gaps in detection indicate moments when faces were not identified or were temporarily out of the scene.

2. Detected Objects Over Time

1. Highlights the objects identified, including **person, laptop, cell phone, chair,** and **TV.**
2. Detection of **persons** is consistent throughout, while other objects like **laptops** and **chairs** appear sporadically, likely due to scene dynamics or occlusion.

3. Confidence Levels Over Time

1. Shows the confidence score of each detection, ranging from 0 to 1.
2. Most scores are between **0.6 and 0.8**, indicating reliable recognition.
3. Occasional dips (~0.4) suggest challenges in detection accuracy due to environmental factors.

4. Inference Times and Entry/Exit Events

1. **Inference Time**: Remains stable between **3.5–4 seconds**, indicating consistent processing speed.
2. **Entry/Exit Events**: Tracks when people and objects enter or leave the scene, providing timestamps for each occurrence.

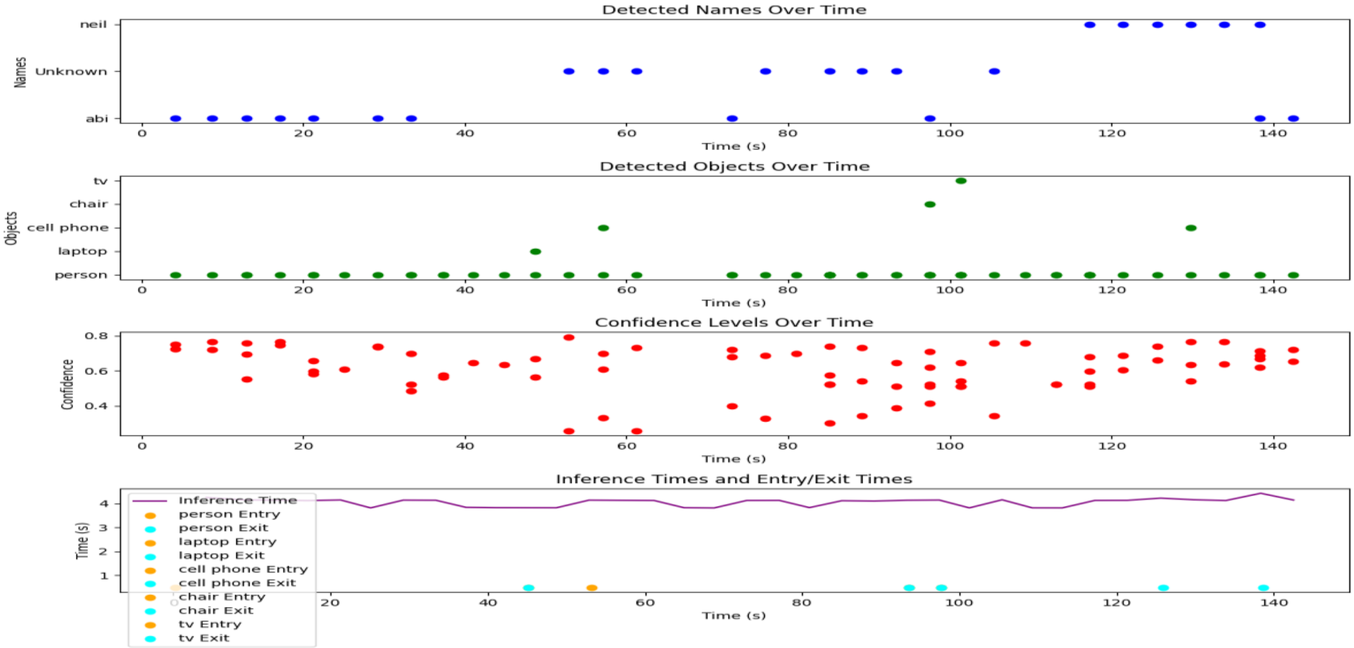


Figure 7 Temporal analysis of confidence levels and inference times

Limitations

Limitations of the System

1. **Slow Processing Speed**:
    - The system is significantly slow in running and identifying objects. Although the code is designed for a 20-second runtime, it takes approximately 1 minute to complete in real-time. This indicates a bottleneck in processing efficiency, likely due to:
        - Limited computational power of the hardware (e.g., Raspberry Pi 4).
        - Inefficiencies in the code or model architecture, leading to prolonged inference times.
2. **Delayed Object Identification**:
    - The slow processing affects the system's ability to capture and identify objects promptly, resulting in potential misalignment between real-world events and system responses.

3. **Misidentification of the objects:**
    - The system's slow performance may be attributed to hardware limitations of the Raspberry Pi 4 and camera, algorithmic inefficiencies in optimizing the detection model for the hardware, the overhead of additional tasks like confidence tracking, entry/exit logging, and plotting, as well as misidentification of persons, leading to

delays in accurate recognition.

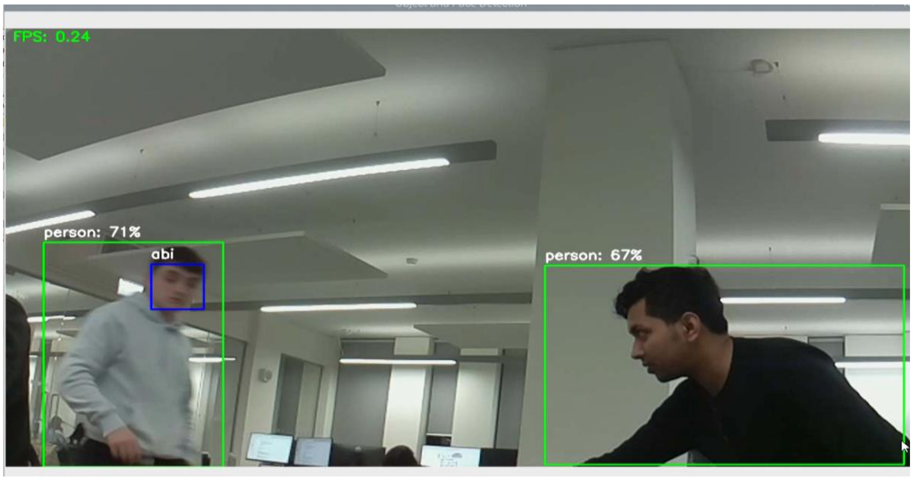


Figure 8 Misidentifcation of the Person

Acknowledgement

Acknowledgement

I would like to extend my heartfelt gratitude to my classmates **Neil Shaun Das**, **Robert Frost**, **Tom Brett**, and **Harison Shivendran**, for their invaluable support and collaboration throughout this project.

A special thanks to **Terence Coffey** for suggesting the creation of a new virtual environment, which successfully resolved the library installation issues I encountered with the facial recognition system.

References

1. Denny, P. (2024, December 5): *Object Detection content*. Retrieved from CE6023 -Object Detection: https://learn.ul.ie/d2l/lms/dropbox/user/folder_submit_files.d2l?db=30175&grpid=0&isprv=0&bp=0&ou=45572
2. Denny, P. (2024, December 5)*: Face Recognition Project*. Retrieved from CE6023 -Face detection content: https://learn.ul.ie/d2l/lms/dropbox/user/folder_submit_files.d2l?db=29381&grpid=0&isprv=0&bp=0&ou=45572
3. Denny, P. (2024). *Wk 10 Lighting*. Retrieved from https://learn.ul.ie/content/enforced/45572-CE6023_SEM1_2024_5/Lectures/Wk%2010%20Release/Lighting.pdf
4. Denny, P. (2024). *Physical Computer Vision System*. Retrieved from https://learn.ul.ie/content/enforced/45572-CE6023_SEM1_2024_5/Lectures/Wk%2010%20Release/Physical%20Computer%20Vision%20System.pdf