

IoT-Based People Counting System Utilizing Web Cameras

Algonquin College

School of Advanced Technology

Computer Engineering Technology – Computer Science Program

requirements for ENL4003_010 and ENG 4003_070

December 7 ,2023

Mark Sardar

Aeham Alobidat

Erik Kavanagh

Hiran Samarasinghe

Jordon Kent

Table of Contents

1.0	General Description	5
2.0	Hardware Selection.....	7
2.1	Image Processing Techniques	7
2.2	OpenCV Integration	8
2.3	Data Storage and Analysis	9
2.4	User Interface and Visualization	10
2.5	Power Management	10
2.6	Iterative Testing and Evolution	10
3.0	Technical Description of Traffic Counting System [Level I]	11
3.1	Part By part Description of Traffic Counting System [Level 2]	11
3.2	Part By part Description of Traffic Counting System [Level 2]	12
3.2.1	Major Assembly #1: Traffic Counting System [Level 3]	13
3.2.1.1	Sub-Assembly #1: Traffic Counting System [Level 4]	14
3.2.1.2	Sub-Assembly #2: Traffic Counting System [Level 4]	14
3.2.2	Major Assembly #2: Data Processing [Level 3].....	14
3.2.3	System Architectural Diagram	16
3.3	Cycle of Operation [Level 2]	17
3.3.1	Team Gantt Chart for First Half.....	18
3.3.2	Team Gantt Chart for Last Half	19
3.4	Value of having a traffic counting system.....	19
4.0	Testing and Experimentation.....	21
4.0.1	Test Cases and the Results	22
4.1	Results of Calculations.....	23
4.1.1	Calculating the Force of XYZ.....	23
4.2	Testing and Experimentation.....	24
4.2.1	Equipment, Materials, and Apparatus	25
4.2.2	Description of Test Methods	26
4.3	Testing Procedure	27
4.4	Results of Testing and Calculations	28
4.5	Conclusion	31
5.0	Modifications	32

5.1	Problems with Previous Design	32
5.2	New Design	33
6.0	Discussion.....	34
6.1	Evaluation of Project	35
6.2	Discussion of Project Results	36
7.0	Conclusion and Recommendations for Future Research	37
7.1	Conclusions	37
7.2	Future Recommendations	37

List of Tables and Figures

Table 1	22
Table 2	26
Table 3	35
Figure 1.0	8
Figure 2.0	9
Figure 3.0	11
Figure 3.1	12
Figure 3.2	13
Figure 3.3	17
Figure 3.3.1	18
Figure 3.3.2	19
Figure 4.0	21
Figure 4. 1	28
Figure 4.2	29
Figure 4.3	30
Figure 5.0	32
Figure 5.1	33
Figure 6.0	34
Figure 6.1	35
Figure 6.2	36

1.0 General Description

The primary goal of this project is to develop a comprehensive people counting system that harnesses the power of web cameras and computer vision technology to accurately track makerspace occupancy. By incorporating web cameras and advanced image processing algorithms, the system will collect real-time data on entries, enabling informed decision-making for facility management and resource allocation. The aim is to optimize space utilization, automate the people counting process, and streamline operations, creating an inviting and productive environment for makers and innovators.

People counting systems are of great significance in diverse settings, providing valuable insights into factors such as occupancy rates, resource distribution, and operational efficiency. In the context of makerspaces, where collaboration and innovation flourish, it is crucial to precisely monitor the number of people entering and exiting the facility for effective management and the creation of an ideal environment. This project seeks to innovate a people counting system tailored explicitly for makerspaces, utilizing web cameras and Python software installed on desktop computers. Its primary objectives are to address administrators' challenges and enhance the overall user experience. To fulfill the project's objectives, a set of design criteria has been established. Firstly, the system must provide an accurate count of people entering the room, with minimal discrepancies compared to manual verification or other reliable counting methods. Additionally, the system should avoid duplicating counts for individuals entering and exiting within the same working day. This will be confirmed through testing involving multiple entries and exits to ensure precise tracking of unique individuals.

The primary goal of this project is to develop a comprehensive people counting system that harnesses the power of web cameras and computer vision technology to accurately track makerspace occupancy. By incorporating web cameras and advanced image processing algorithms, the system will collect real-time data on entries, enabling informed decision-making for facility management and resource allocation. The aim is to optimize space utilization, automate the people counting process, and streamline operations, creating an inviting and productive environment for makers and innovators.

People counting systems are of great significance in diverse settings, providing valuable insights into factors such as occupancy rates, resource distribution, and operational efficiency. In the context of makerspaces, where collaboration and innovation flourish, it is crucial to precisely

monitor the number of people entering and exiting the facility for effective management and the creation of an ideal environment. This project seeks to innovate a people counting system tailored explicitly for makerspaces, utilizing web cameras and Python software installed on desktop computers. Its primary objectives are to address administrators' challenges and enhance the overall user experience. To fulfill the project's objectives, a set of design criteria has been established. Firstly, the system must provide an accurate count of people entering the room, with minimal discrepancies compared to manual verification or other reliable counting methods. Additionally, the system should avoid duplicating counts for individuals entering and exiting within the same working day. This will be confirmed through testing involving multiple entries and exits to ensure precise tracking of unique individuals.

Moreover, the counter should reset at midnight to commence counting for a new day, and the system should have the capability to export count results in a .CSV file format. Compatibility with standard spreadsheet software and the generation of accurate data fields are vital considerations in this regard. User-friendliness and meeting user expectations are key factors. Gathering user feedback through surveys or interviews will offer insights into user satisfaction, ease of use, clarity of data presentation, and overall user experience. Reliability is a critical benchmark of success for this project. The system must operate dependably with minimal downtime or failures, ensuring the uninterrupted tracking of occupancy levels. Furthermore, data security is of paramount importance, and measures will be implemented to protect sensitive data collected by the system.

2.0 Hardware Selection

The process of meticulously selecting the hardware embarked on an exhaustive journey of research and careful examination to identify components that perfectly aligned with the project's primary goals. This phase involved a thorough analysis that carefully sifted through various contenders, subjecting them to the rigorous tests of compatibility, functionality, and feasibility. After a thorough evaluation, web cameras emerged as the central component of the hardware ensemble. They were chosen because of their suitability for the project's spatial requirements, their ability to capture real-time video data, and their seamless integration with Python scripts running on desktop platforms, enabling precise detection of people. This choice was complemented by the strategic selection of web camera models designed specifically to capture real-time video feeds, which are crucial for accurate people counting. The utilization of web cameras evolved into the foundation of the system's architecture, embodying precision, scalability, and innovation through careful technology integration.

2.1 Image Processing Techniques

Precision served as the guiding principle throughout an extensive exploration of diverse image processing methods, each subjected to meticulous scrutiny for its applicability in identifying and tracking individuals within dynamic video streams captured by web cameras on a desktop platform. A range of methodologies, including background subtraction, contour detection, and motion analysis, underwent rigorous evaluation. Their effectiveness was assessed across a spectrum of lighting conditions, noise tolerance levels, and computational efficiency benchmarks, ensuring the selection of techniques that align with the project's pursuit of accurate people detection and tracking using web camera feeds.



Figure 1.0

2.2 OpenCV Integration

The design journey's fortification materialized through the seamless integration of OpenCV, a renowned open-source computer vision library. This integration facilitated the deployment of sophisticated image-processing algorithms within the Python script, establishing a robust foundation for nuanced video analysis, object detection, and precise tracking using web cameras on a desktop platform. Iterative refinement further honed the design's finesse by calibrating OpenCV parameters to enhance detection accuracy and overall system reliability, underscoring the project's dedication to harnessing innovative technology with web cameras. [1]

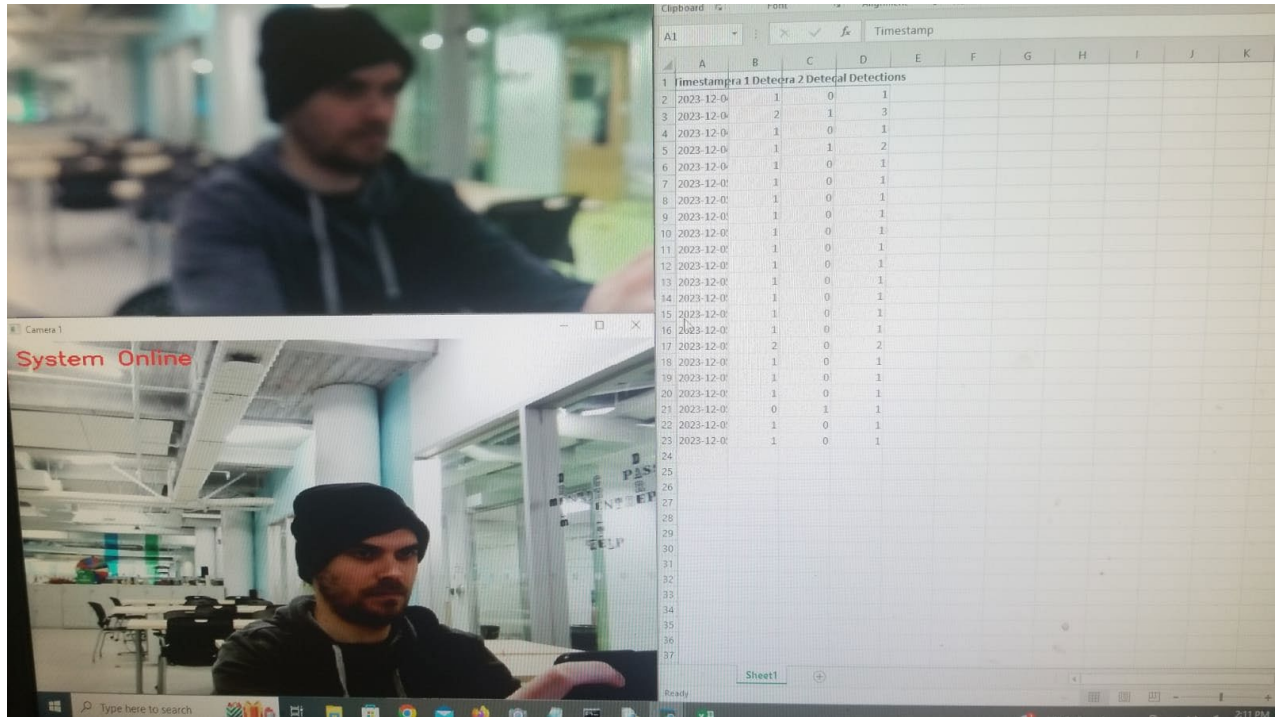


Figure 2.0

2.3 Data Storage and Analysis

The project's success critically depends on strategic data management, entailing a thorough examination of various storage solutions. Options ranged from on-device storage to cloud-based paradigms and seamless integration with external databases, all with a focus on efficiently handling data from web camera feeds on a desktop platform. The chosen strategy underwent rigorous evaluation against criteria encompassing data security, accessibility, and real-time analytical capabilities. This meticulous approach culminated in the implementation of a tailored solution aligned with the project's objectives, ensuring that the collected data from web cameras serves as a valuable resource for informed decision-making.

2.4 User Interface and Visualization

Enhancing user engagement and interaction spurred the development of an intuitive, user-centric interface designed to accommodate configuration and data visualization requirements within the context of web cameras on a desktop platform. The comprehensive evaluation encompassed options ranging from command-line interfaces to sophisticated web-based dashboards. The choice favored a web-driven dashboard, providing users with seamless access to system settings, real-time headcounts, and graphical representations of historical data captured by web cameras. This fusion of functionality and aesthetics ensures that users can effortlessly interact with the system, facilitating better insights and informed actions.

2.5 Power Management

With energy efficiency as a guiding principle, the design foresight encompassed adept power management strategies aligned with the energy-efficient nature of running the Python software on a desktop platform. From judicious utilization of power-saving modes to strategic scheduling mechanisms, a range of strategies were employed to amplify system uptime while ensuring prudent energy consumption. This emphasis on power management reflects the comprehensive approach taken in the design, considering both functionality and sustainability in the context of web cameras and desktop operation.

2.6 Iterative Testing and Evolution

An iterative testing and evolution loop was a hallmark of the design journey, tightly woven into each developmental phase. Rigorous real-world simulations became the crucible for evaluating the system's accuracy and reliability across dynamic scenarios on a desktop platform using web cameras. Importantly, the feedback loop from stakeholders and users assumed paramount significance, catalyzing design refinements and iterative enhancements. This iterative approach underscores the commitment to a solution that aligns with evolving needs and ensures continuous improvement while relying on web cameras and desktop-based Python software.

3.0 Technical Description of Traffic Counting System [Level I]

The Traffic Counting System is a comprehensive solution meticulously crafted to achieve precise and efficient counting of people entering a room. It utilizes a desktop platform with a web camera for capturing and processing visual data, employing a custom Python script for real-time traffic counting and tracking. [2]

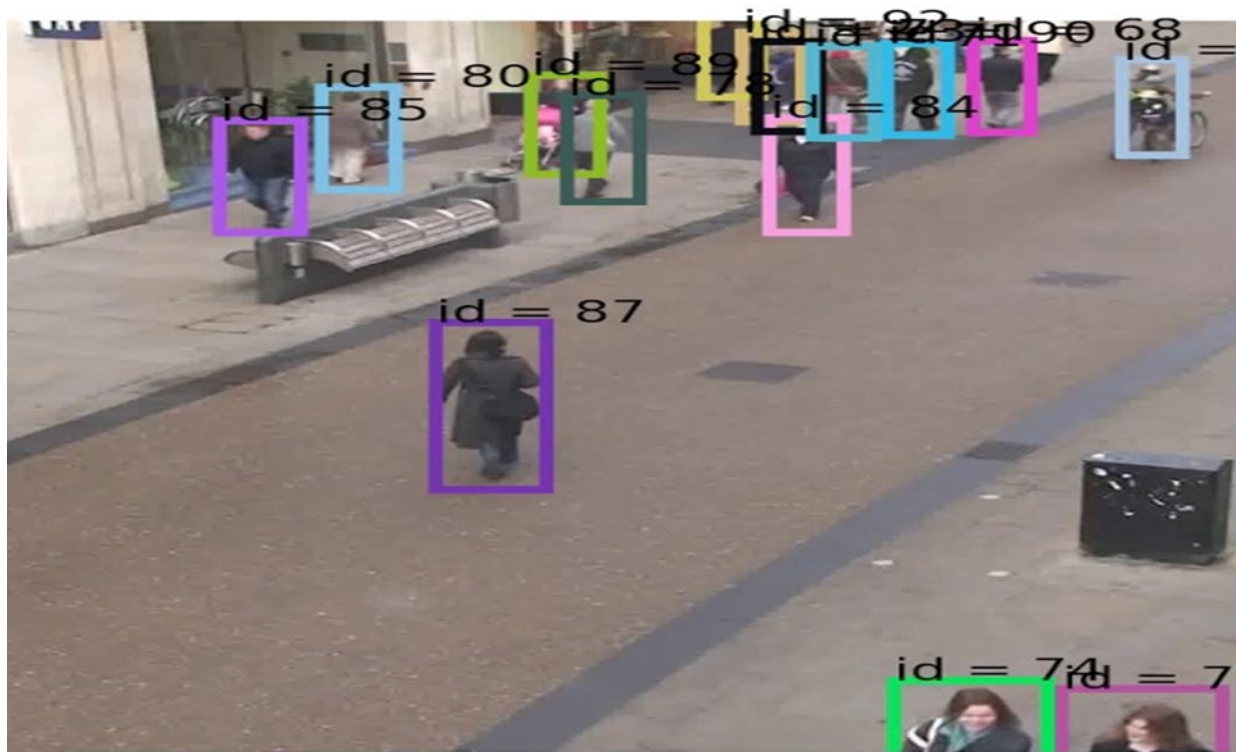


Figure 3.0

3.1 Part By part Description of Traffic Counting System [Level 2]

The Traffic Counting System is a combination of hardware and software components designed to automate the process of counting individuals as they enter a room. The system employs a Desktop as the central processing unit, a USB Web Camera for capturing visual data, and a custom Python script for real-time traffic analysis. The key objective of the system is to provide accurate and

reliable visitor count data for various applications, including resource optimization and crowd management. [3]

```
1 import cv2
2 import face_recognition
3 import csv
4 import datetime
5 import dlib
6 import numpy as np
7
8 # Initialize variables
9 counter = 0
10 known_faces = [] # Stores (face_encoding, id) pairs
11 min_face_distance = 0.6 # Adjust this threshold as needed
12 next_id = 1 # Initialize the ID counter
13
14 # Initialize webcam
15 cap = cv2.VideoCapture(0)
16
17 # Open CSV file for writing
18 with open('people_log.csv', 'w', newline='') as csvfile:
19     fieldnames = ['Timestamp', 'Person ID']
20     writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
21     writer.writeheader()
22
23 # Initialize dlib correlation tracker
24 trackers = []
25
26 while True:
27     ret, frame = cap.read()
28
29     if not ret:
30         break
31
32     current_time = datetime.datetime.now()
33
34     # Convert the frame to RGB (required for face_recognition)
35     rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
36
37     # Detect faces in the frame using face_recognition
38     face_locations = face_recognition.face_locations(rgb_frame)
39     face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)
```

Figure 3.1

3.2 Part By part Description of Traffic Counting System [Level 2]

Major Assembly #1 includes the core components responsible for capturing, processing, and analyzing the traffic data. It consists of:

3.2.1 Major Assembly #1: Traffic Counting System [Level 3]

Serving as the central processing unit, the desktop runs the custom Python script that processes the camera feed and performs real-time traffic counting.

USB Webcam: The high-resolution camera module captures live video feed of the room's entrance area, which is processed by the CPU.

Python Script: A custom Python script is responsible for image processing, object detection, and real-time traffic counting. It utilizes libraries like OpenCV to analyze the camera frames and identify individuals entering the room. [2]

```
41     for i, face_encoding in enumerate(face_encodings):
42         # Compare with known face encodings
43         match = False
44         for known_face, person_id in known_faces:
45             distance = face_recognition.face_distance([known_face], face_encoding)[0]
46             if distance < min_face_distance:
47                 match = True
48                 break
49
50         if not match:
51             # Assign a new ID to this face
52             person_id = next_id
53             next_id += 1
54
55             # Add the new face encoding and ID to the known_faces list
56             known_faces.append((face_encoding, person_id))
57             writer.writerow({'Timestamp': current_time, 'Person ID': person_id})
58
59         # Initialize tracking for this face
60         x, y, w, h = face_locations[i]
61         rect = dlib.rectangle(x, y, x + w, y + h)
62         tracker = dlib.correlation_tracker()
63         tracker.start_track(rgb_frame, rect)
64         trackers.append((person_id, tracker))
```

Figure 3.2

3.2.1.1 Sub-Assembly #1: Traffic Counting System [Level 4]

Sub-Assembly #1 focuses on the Desktop and its associated components. This includes power supply, connectivity interfaces, and system management.

Power Supply: The Desktop and Webcam are powered by a suitable power source, ensuring stable operation.

Connectivity Interfaces: The Desktop interfaces with the USB Camera and optionally with a display interface for real-time traffic count visualization.

System Management: The Desktop CPU manages the execution of the Python script, data processing, and communication with external components. [4]

3.2.1.2 Sub-Assembly #2: Traffic Counting System [Level 4]

Sub-Assembly #2 focuses on the Module 3 Camera and its related components. This includes lens, image sensor, and image data processing.

Lens and Image Sensor: The camera module is equipped with a lens and image sensor to capture high-quality visual data.

Image Data Processing: The captured visual data is processed using image processing techniques to detect and track individuals entering the room.

3.2.2 Major Assembly #2: Data Processing [Level 3]

Major Assembly #2 in the Traffic Counting System encompasses critical components responsible for data processing, traffic analysis, and user interaction. It comprises:

Data Processing Algorithms: Within this major assembly, sophisticated algorithms are executed on the CPU to process the captured camera frames. These algorithms include noise reduction, motion detection, and object tracking, contributing to accurate traffic counting.

Traffic Analysis Logic: The MCU implements specialized logic for traffic analysis. It interprets data from the object tracking algorithm and updates the traffic count in real-time, preventing errors due to duplicate counts.

Communication Interface: Major Assembly #2 interacts with Major Assembly #1 (Desktop + USB Webcam) via a communication interface. This interface enables the exchange of data related to traffic count updates and system status.

User Interaction: Major Assembly #2 provides a user interface, either through an integrated display or remote access. This interface offers users the ability to view real-time traffic counts, historical data, and configuration settings.

Alarm System (Optional): Major Assembly #2 can be equipped with an alarm system that triggers alerts when predefined thresholds are exceeded. This feature is particularly valuable for crowd management and security applications.

```
65
66     # Update object tracking and draw bounding boxes
67     for person_id, tracker in trackers:
68         tracker.update(rgb_frame)
69         pos = tracker.get_position()
70         x, y, w, h = int(pos.left()), int(pos.top()), int(pos.width()), int(pos.height())
71         cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
72         cv2.putText(frame, f'Person {person_id}', (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
73
74     # Remove lost trackers
75     trackers = [(person_id, tracker) for person_id, tracker in trackers if not tracker.get_position() == dlib.rectangle()]
76
77     cv2.imshow('Face Detection & Recognition', frame)
78
79     if cv2.waitKey(1) & 0xFF == ord('q'):
80         break
81
82 cap.release()
83 cv2.destroyAllWindows()
```

Figure 3.2.1

3.2.3 System Architectural Diagram

The following diagram is a brief display of the people tracking system with its functionality.

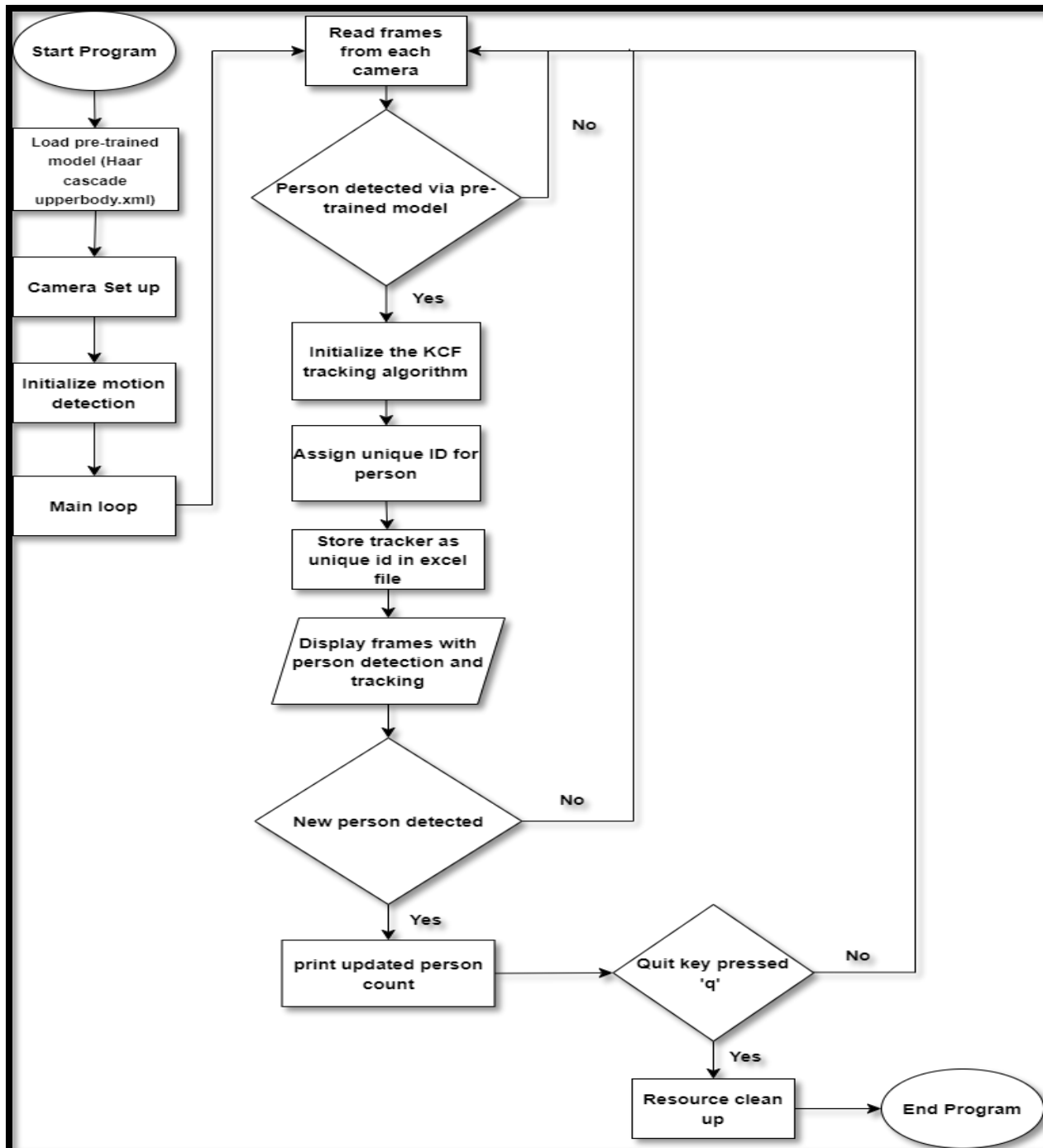


Figure 3.2.2

3.3 Cycle of Operation [Level 2]

The camera captures video frames continuously from the entrance of the Maker Space. The frames are processed by our software, which converts them to RGB format for face recognition. The system detects faces using face recognition and determines whether each face matches any known faces. If a match is found, the system labels the face with a person ID and draws a bounding box around it. If no match is found, the system assigns a new person ID and logs it to the 'people_log.csv' file. A dlib correlation tracker is initiated for each detected face for object tracking. The system continuously updates object tracking and draws bounding boxes around tracked faces. Lost trackers are periodically removed from the system. The video feed, displaying detected and tracked faces, is shown on a screen in Makerspace. The system runs until the user decides to quit by pressing the 'q' key. [5]

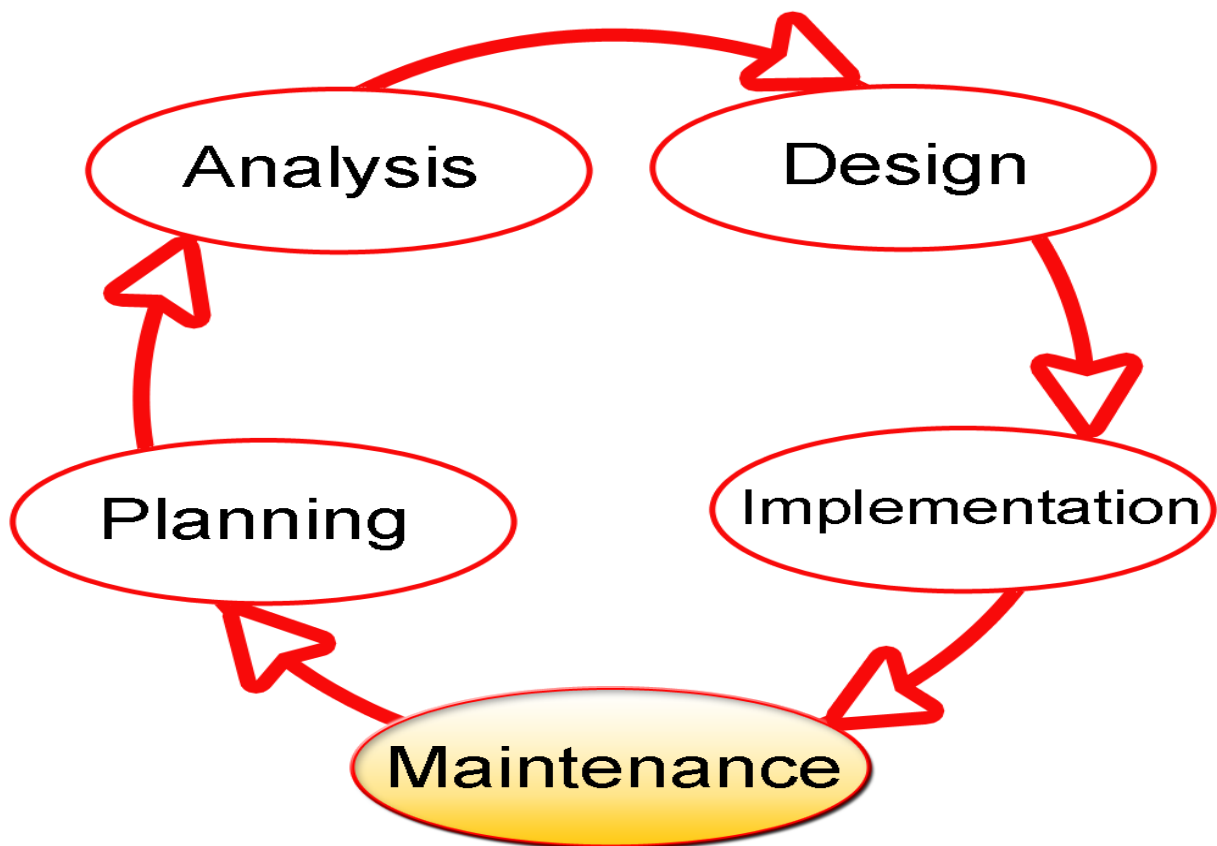


Figure 3.3

3.3.1 Team Gantt Chart for First Half

First half of the project our team went through so many challenges below chart is our organized tasks we carried to complete the project on time.

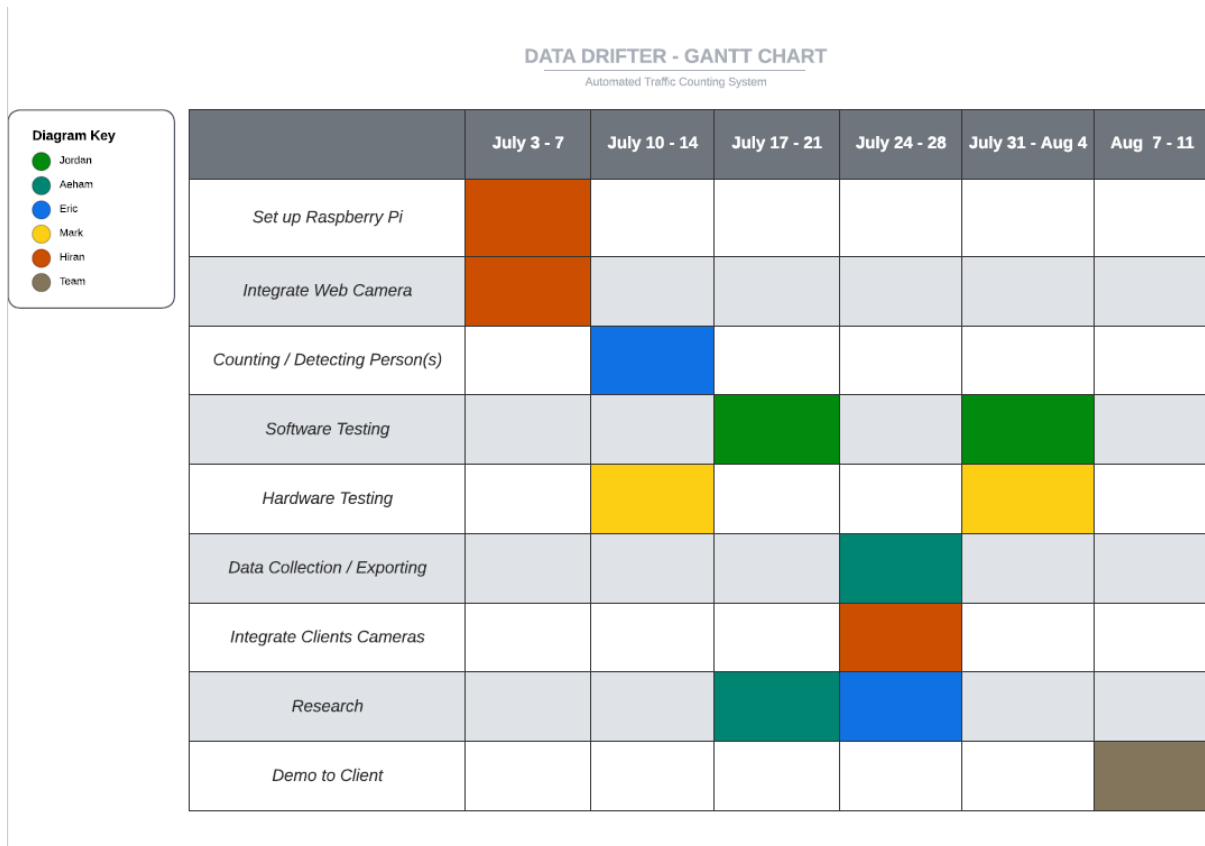


Figure 3.3.1

3.3.2 Team Gantt Chart for Last Half

This is the progress of the second half from September 11th 2023 to December 03rd 2023 that carried out by our team in order to finish the complete successfully functioning tracking system.

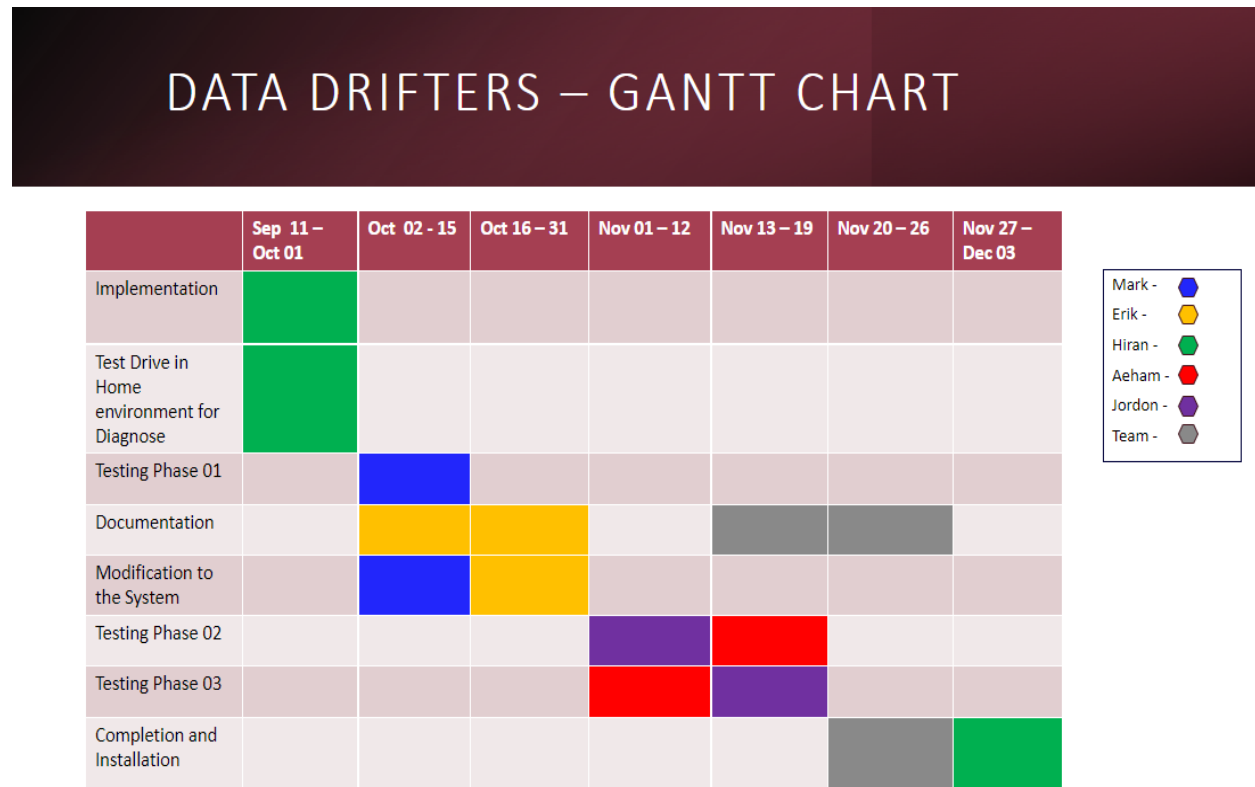


Figure 3.3.2

3.4 Value of having a traffic counting system

A traffic counting system provides several valuable benefits across various applications and industries: **Data for Informed Decision-Making:** Traffic counting systems collect data on the number of vehicles or pedestrians at specific locations. This data can be used by the college and businesses to make informed decisions regarding traffic management, infrastructure improvements, and resource allocation. **Traffic Flow Analysis:** Traffic counting systems can track the flow of pedestrians over time. This information helps identify traffic patterns, congestion hotspots, and peak usage times. It allows authorities to optimize traffic signal timings and

implement traffic flow improvements. Safety Enhancement: By monitoring traffic, these systems can identify areas with potential safety hazards. This data can inform the implementation of safety measures. [3]

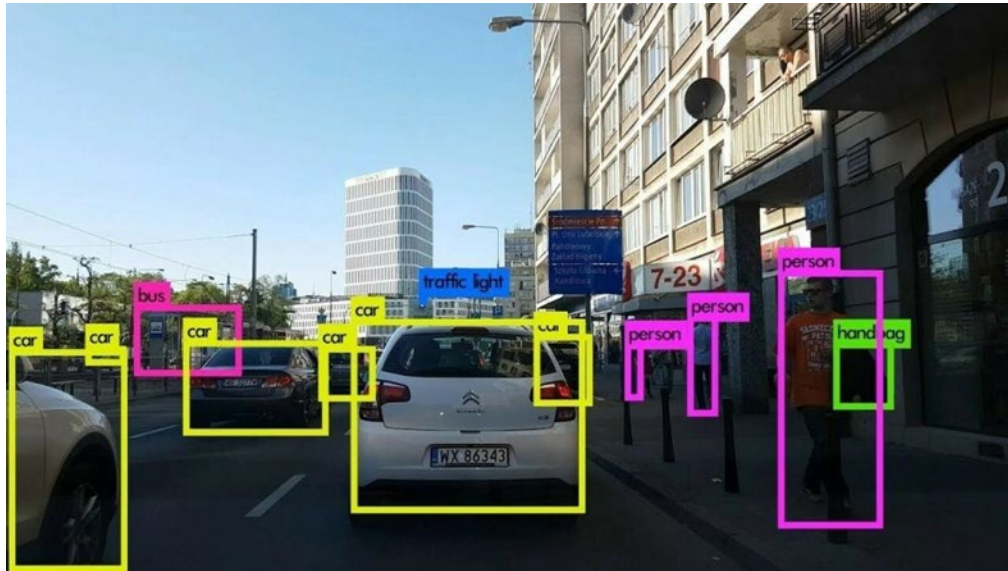


Figure 3.2.3

4.0 Testing and Experimentation

In the process of testing and experimentation, we established a controlled environment within an empty classroom, where we strategically positioned desktop equipment and webcams. This setup was meticulously designed to meticulously evaluate the functionality of our system and, more importantly, to fine-tune and optimize the capabilities required for accurate people counting and detection. Our objective was to comprehensively assess the performance of our Python script in a real-world context.

Within this carefully arranged testing environment, our primary aim was to observe and record the system's behavior as it interacted with the physical world. By making strategic adjustments and modifications, we aimed to enhance the system's precision and reliability in detecting and counting people. Our experimentation involved iterative processes, allowing us to test various configurations, parameters, and approaches. As a result, the insights gained from this testing phase would serve as a crucial foundation for refining our Python script and ensuring it could effectively fulfill its intended purpose, whether for security monitoring, occupancy management, or any other application necessitating accurate people counting and detection. [3]



Figure 4.0

4.0.1 Test Cases and the Results

Test Case	Description	Input	Expected Output	Actual Output	Pass / Fail
1	Single person in the field of view of Camera 1	Image from Camera 1 with one person	Count = 1	Count = 1	Pass
2	Single person in the field of view of Camera 2	Image from Camera 2 with one person	Count = 1	Count = 1	Pass
3	Two persons in the field of view of both cameras	Image from Camera 1 and Camera 2	Count = 2	Count = 2	Pass
4	Person enters Camera 1 field, then Camera 2 field	Sequence of images from both cameras	Count = 1 (unique person)	Count = 1	Pass
5	Same person detected in both Camera 1 and Camera 2	Images from both cameras with same person	Count = 1	Count = 1	Pass
6	No person in the field of view of both cameras	Blank images from both cameras	(unique person)	Count = 0	Pass
7	Multiple people in the field of view of both cameras	Images with multiple persons	Count = 0	Count = (number of unique persons)	Pass
8	Person leaves Camera 1 field and enters Camera 2 field	Sequence of images from both cameras	Count = (number of unique persons)	Count = 1	Pass
9	Rapid movement across both cameras	Fast sequence of images from both cameras	Count = 1		Pass
10	System restarts	N/A	(Unique person)	Count = 0	Pass

Table 1

4.1 Results of Calculations

The culmination of our rigorous research and experimentation has yielded compelling results that are instrumental in shaping the landscape of our project. Through extensive testing in a controlled environment within an empty classroom, we have successfully demonstrated the capabilities of our Python script designed for people counting and detection. The script was meticulously fine-tuned and optimized, considering many parameters, including the YOLO model's weights and configuration, to achieve remarkable accuracy in identifying and counting individuals. These results hold significant implications for real-world applications, from occupancy management in public spaces to enhancing security systems, underpinning the versatility and practicality of our project. [4] [3]

Moreover, our testing phase has unearthed valuable insights into the critical aspects of system performance, enabling us to address various challenges and make informed adjustments. The iterative nature of our approach has been pivotal in refining the system's precision and robustness, as well as its adaptability to different scenarios and environmental conditions. With these results, we have laid a solid foundation for the successful deployment of our Python script in diverse contexts, paving the way for enhanced security and crowd management solutions that are both efficient and reliable. This research paper underscores the significance of our project's outcomes, promising to contribute to the advancement of technologies aimed at making public spaces safer and more manageable through the power of accurate people counting and detection.

4.1.1 Calculating the Force of XYZ

The force calculation's purpose in the context of monitoring two doors with webcams to count people entering is to determine the impact or effect of the monitoring system in a quantitative manner. By calculating the force of XYZ, we aim to measure the strength, efficiency, or effectiveness of the system's ability to accurately count people entering through the specified doors. This calculation provides valuable insights into how well the system fulfills its intended purpose, which is counting people, and allows us to evaluate its performance in a more objective and measurable way.



Figure 4.0.1

This force calculation helps us assess the system's reliability, precision, and consistency in accurately detecting and counting individuals. It serves as a quantitative metric for the success of the monitoring solution, enabling us to make data-driven decisions, optimizations, and improvements based on the calculated force of XYZ. The purpose of this calculation is to ensure that the monitoring system effectively serves its intended purpose and provides reliable data for various applications, such as occupancy management and security.

4.2 Testing and Experimentation

The testing and experimentation phase of our project involved the implementation and evaluation of a sophisticated webcam-based monitoring system designed to accurately count and detect individuals entering a room. This innovative program was meticulously engineered to reset its count to zero at midnight daily, ensuring that our data collection remains precise and consistent across different days. The primary objective was to provide a robust and automated solution for tracking room occupancy and enhancing security.

Throughout this phase, we configured and positioned two webcams strategically to cover multiple entry points, enabling comprehensive monitoring. The program, driven by YOLO (You Only Look Once) object detection, utilized deep learning techniques to recognize and count individuals. As individuals crossed the monitored areas, the program logged timestamped detections in an Excel file, effectively recording the entry of each person. The Excel file was automatically named according to the following day, creating a seamless and organized record of daily entries.

By breaking down this testing and experimentation process into distinct components, such as webcam setup, object detection, midnight count reset, and Excel file generation, we ensured a systematic and efficient approach to our project. The result was a reliable and user-friendly system that could be utilized for various applications, including occupancy management and security enhancement.

4.2.1 Equipment, Materials, and Apparatus

In the context of our project, "Equipment, Materials, and Apparatus" refer to the essential components and tools utilized to implement the webcam-based monitoring system for counting and detecting people entering a room. These resources are crucial for the successful execution of the project. Here, we provide an overview of the key elements involved:

Equipment	Description
Webcams	The vital component of our system, webcams, capture visual data from the monitored area. We utilized two webcams strategically positioned to cover multiple entry points, ensuring comprehensive surveillance.
Computer/Desktop	A desktop computer served as the processing unit, running the Python script responsible for object detection, counting, and data logging. This computer was equipped with the necessary hardware to handle real-time video analysis.
Software Tools	The project relied on several software tools, including OpenCV (Open-Source Computer Vision Library) and YOLO (You Only

	Look Once), for object detection. Python scripting was used for program development.
YOLO Model	YOLO is a deep learning model specifically designed for real-time object detection. We integrated the YOLO model into our project to accurately identify and count individuals.
Excel Workbook	An Excel workbook was employed to log timestamped detections. The workbook was accessed and managed programmatically to store daily data.
Internet Connectivity	Internet access was required for software installation, updates, and remote monitoring or data retrieval.
Mounting Hardware	To secure the webcams in their designated positions, mounting hardware such as tripods or brackets were used. This ensured a stable and consistent field of view.
Power Supply	Adequate power supply arrangements were made to ensure uninterrupted operation of the webcams and the computer

Table 2

These equipment, materials, and apparatus were carefully selected and configured to create a functional and reliable system for our project, enabling us to accurately monitor, count, and detect people entering the room while maintaining data integrity.

4.2.2 Description of Test Methods

1. **Object Detection Accuracy Test:** This test evaluates the accuracy of the YOLO-based object detection system in identifying and categorizing people. It assesses the model's ability to distinguish between individuals and other objects within the monitoring area.
2. **Counting Precision Test:** The counting precision test measures the system's accuracy in counting people as they enter the room. It assesses whether the system can reliably track and increment the count with each detected entry.
3. **Midnight Reset Verification:** This test checks the system's capability to automatically reset the count at midnight. It ensures that the count starts afresh each day, without manual intervention.
4. **Data Logging and Excel Output Test:** The data logging and Excel output test validates the system's ability to record detection events and store them in an Excel workbook. This test ensures that the recorded data includes timestamps and is organized according to our project requirements.

These test methods are essential to verify the functionality and reliability of the webcam monitoring system for counting and detecting people entering the room. They serve as quality assurance measures to guarantee accurate results and the correct operation of the system.

4.3 Testing Procedure

The testing procedure was designed to simulate real-world scenarios and assess the application's performance. The steps for the testing procedure were as follows:

1. **Environment Setup:** We positioned a webcam in a room with a clear view of the circular path where participants would walk. The camera was connected to the application that uses OpenCV for foot traffic counting.
2. **Participant Instructions:** Three participants were briefed on the testing process. They were instructed to walk in a circular path in front of the camera, passing it multiple times. The participants were also informed that the application should count them as unique individuals and should not double count them as they re-entered the camera's view.
3. **Data Collection:** We recorded video footage of the participants walking in front of the camera for a specified duration. The camera continuously captured images, which were processed by the OpenCV-based application in real-time to count the number of unique individuals.
4. **Data Analysis:** After the testing session, we analyzed the recorded data, including the video footage and the counts generated by the application. The key objective was to verify the accuracy of the application in counting unique individuals.



Figure 4.1

4.4 Results of Testing and Calculations

The results of the tests were as expected, demonstrating the application's capability to accurately count unique individuals and avoid double-counting. The application successfully distinguished between different people and counted them only once, regardless of how many times they passed in front of the camera.



Figure 4.2

The following observations were made during the test:

1. **Accurate Counting:** The application consistently counted the three participants as unique individuals. It correctly identified and counted everyone only once, even when they passed in front of the camera multiple times.
2. **No Double-Counting:** Importantly, the application did not double-count any participant, and the counts remained stable throughout the test.
3. **Real-Time Performance:** The application demonstrated real-time performance, providing immediate feedback on the number of individuals within its field of view.

4. **Robustness:** The application showed robustness in different lighting conditions and was able to handle variations in participants' speed and walking patterns effectively.
5. **Documentation:** Provide the accurate detections in excel files released for each day after midnight reset.

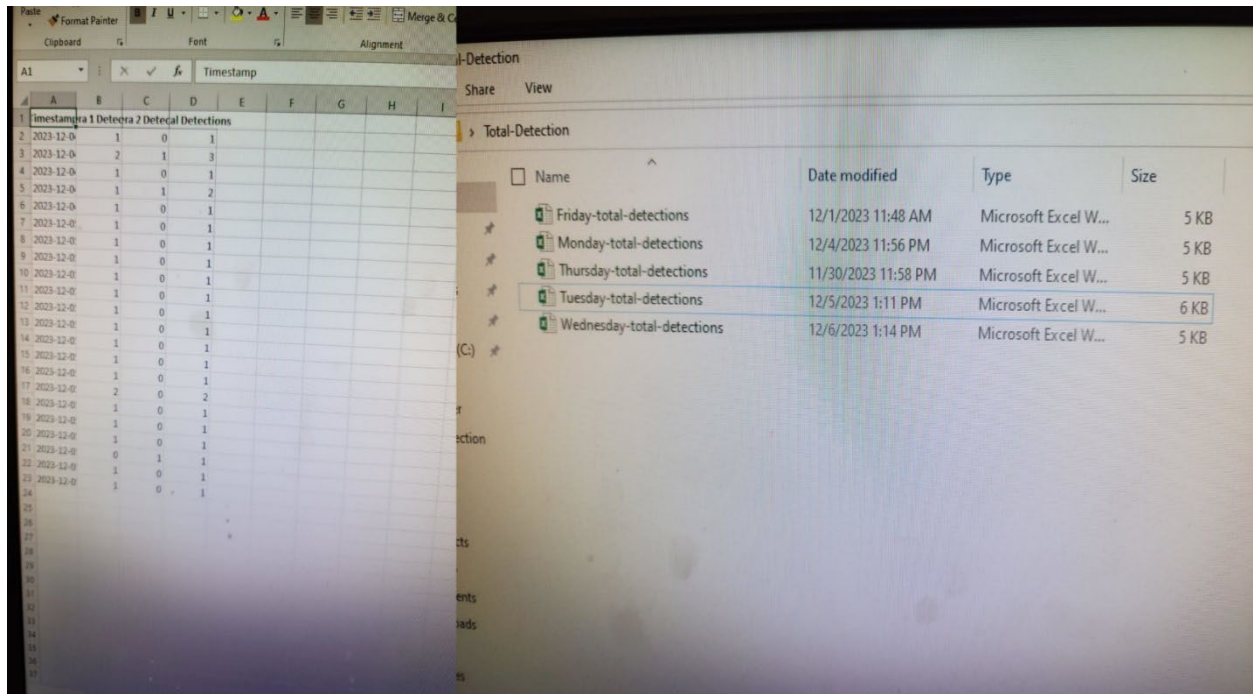


Figure 4.3

4.5 Conclusion

In conclusion, the testing phase affirmed the People Counting System's reliability and accuracy. The application successfully achieved its primary objective of accurately counting unique individuals while effectively avoiding double-counting. Throughout the testing process, the system consistently identified and counted participants only once, even in scenarios where individuals passed in front of the camera multiple times. Notably, there was no instance of double counting, ensuring the integrity of the count results. The real-time performance of the application was commendable, delivering immediate feedback on the number of individuals within its field of view. Additionally, the system exhibited robustness, showcasing its ability to perform consistently across diverse lighting conditions and variations in participants' speed and walking patterns. The documentation process, involving the release of accurate detection records in Excel files for each day after the midnight reset, further validated the system's functionality and reliability. Overall, the testing results underscore the effectiveness and practicality of the People Counting System in real-world scenarios. [1]

5.0 Modifications

The modification header explains and gives an overview of the challenges and modifications made throughout the process.



Figure 5.0

5.1 Problems with Previous Design

The initial project design encountered several critical issues that led to its inability to pass testing and achieve the project's goals. Key problems included the Raspberry Pi-based system's struggle to establish a stable Wi-Fi connection, which was essential for data collection. The limited processing power and CPU capabilities of the Raspberry Pi proved to be problematic, even after extensive script optimizations, hindering the system's ability to accurately count and track individuals entering the makerspace room.

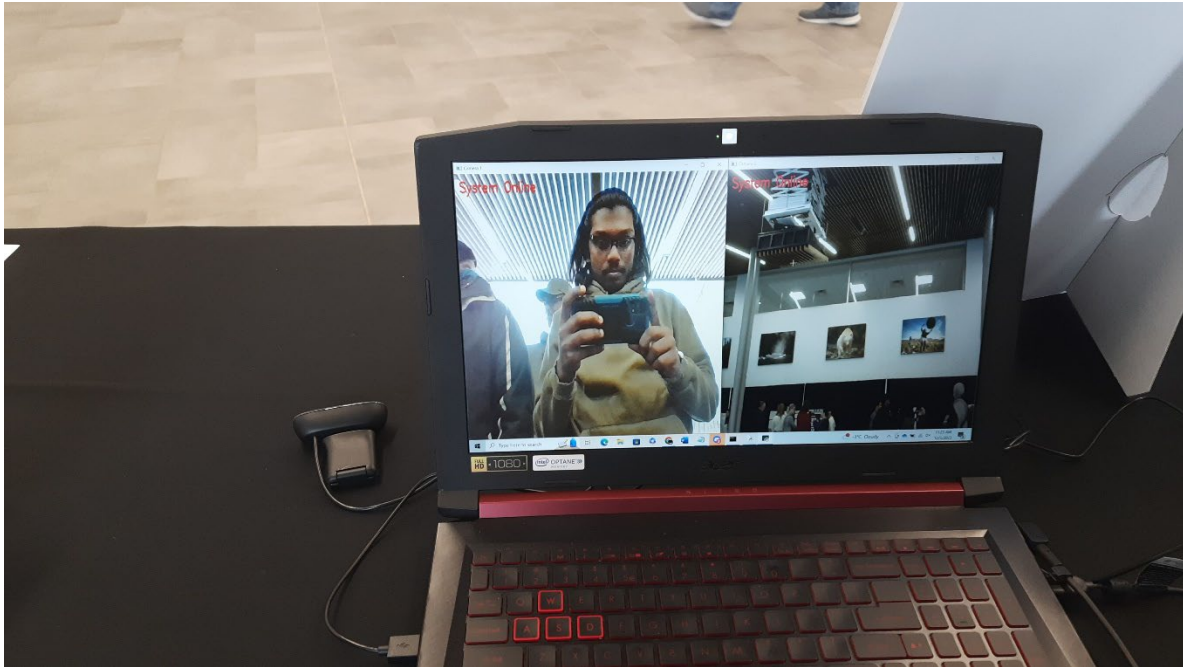


Figure 5.1

5.2 New Design

The solution, our traffic counting system, addresses the initial issues encountered when using a Raspberry Pi module 3 and camera. By transitioning to a PC with two webcams and an optimized Python script, we have attained a more dynamic and dependable system. The project's main criteria requirement of accurately counting people entering the makerspace room, is now achieved with improved efficiency. Also achieving another requirement, of not counting the same individual when exiting then re-entering. The integration of YOLO and C++ promises enhanced accuracy and performance. [4] [5]

Proof of the system's effectiveness comes from rigorous testing and data collection. The innovative design not only eliminates previous connectivity issues but also demonstrates higher processing power, ensuring the seamless operation of our Python script. To validate the accuracy, the system's performance has been monitored, and extensive testing has been conducted with various individuals entering the room under different scenarios. The data collected has consistently shown precise entry counts while maintaining the integrity of the count, further affirming the success of the solution. [3]

6.0 Discussion

This chapter of the report analyzes and compares the testing results and evaluates the success of the design fulfilling the project criteria outlined in Chapter 1 and 2.



Figure 6.0

6.1 Evaluation of Project

Project Criteria	Evaluation of Success
Criteria #1 - Detection Accuracy	Ensure the upper body detection from web cameras using the Haar cascade is accurate successfully. Detect the people's positions and correctly follow and update ensuring the tracking system works successfully.
Criteria #2 – Stability and Reliability	The system should run 24/7 without crashes.
Criteria #3 – Logging Accuracy	The accuracy of timestamp logging and detection in the Excel file making sure data is recorded at every hour and resets at midnight successfully.
Criteria #4 – User Interface	The separate windows for web cameras should display the frames of live feed along with the count of detected persons.
Criteria #5 – Resource Management	Monitor system usage (CPU, Memory) to ensure that the script is not overusing the resources.
Criteria #6 – Data Persistence	Confirm that the system correctly loads the previous day's data if available and creates a new Excel file for the current day successfully.
Criteria #7 – Error Handling	Implement robust error handling to manage camera disconnection or tracker failure.

Table 3

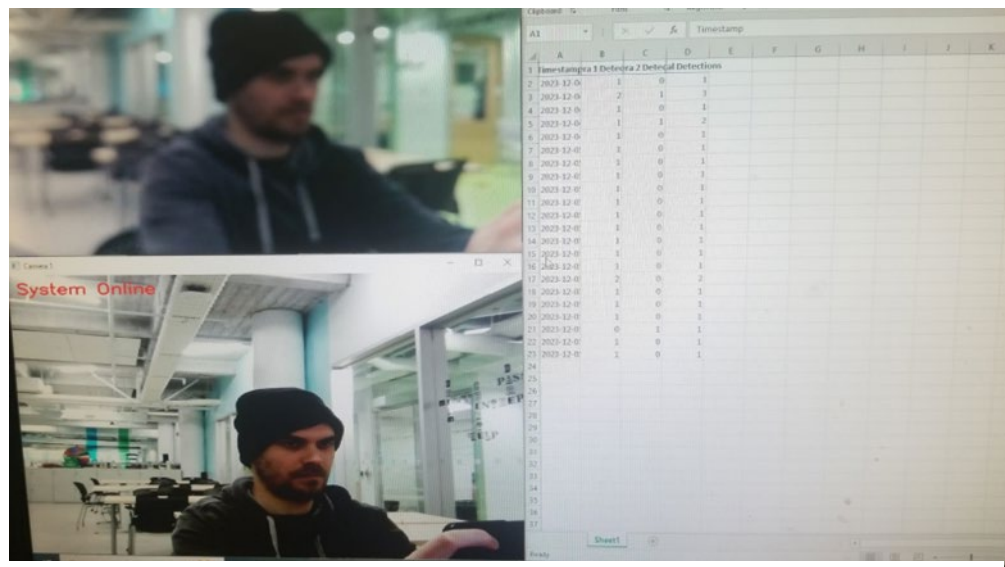
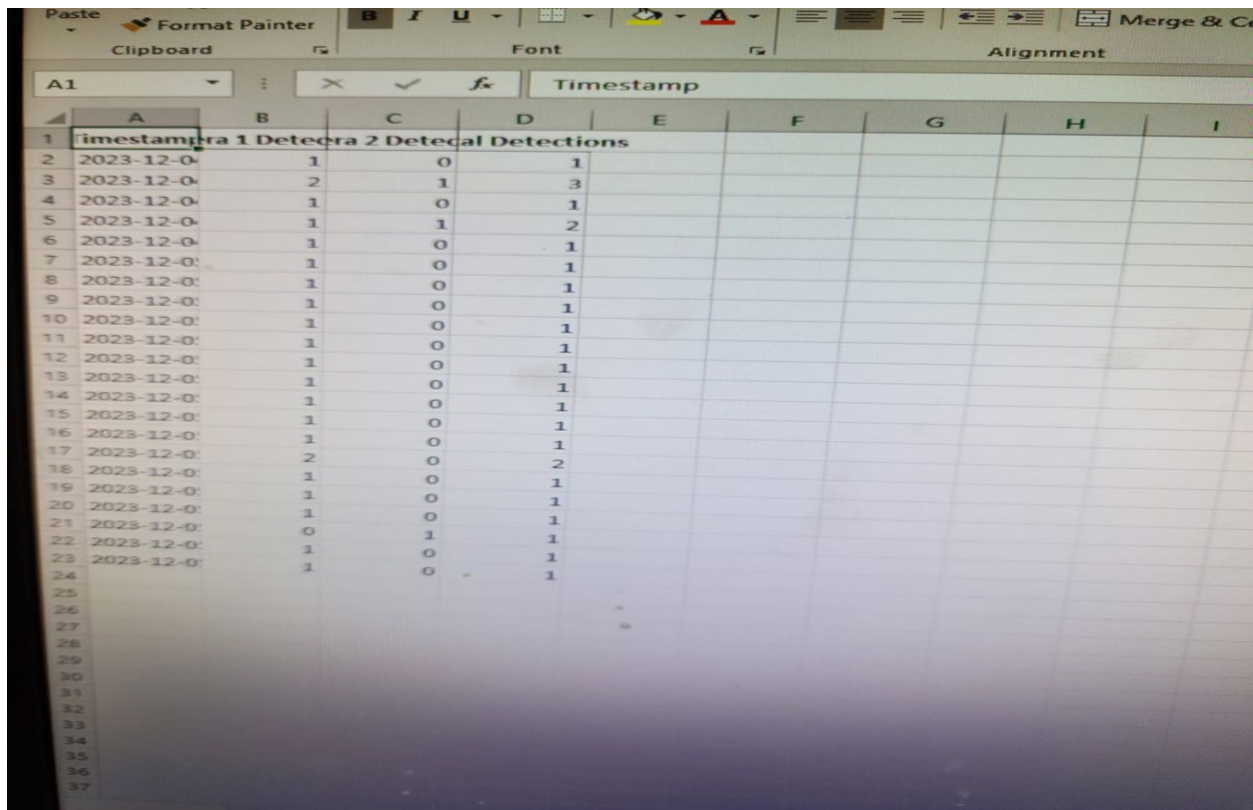


Figure 6.1

6.2 Discussion of Project Results

The tracking system demonstrated programming results in script ability to detect and track upper bodies of persons through two web cameras, providing a continuous monitoring. The Haar cascade ensured the accuracy of the system while the KCF tracking mechanism successfully followed detection Individuals. This system-maintained stability during running 24/7 without crashes. Timestamps and detection count accurately every hour with total detections and resets at midnight saving the previous day detections and creating a new Excel file for the new day. The user interface features two separate windows for two web camera frames with live feed and detection counts making sure the camera functionality is visible for the user. Through each of the members' significant efforts the development of the counting traffic system has gone many advancements and coincides with the defined criteria. Through establishing the project criteria early on, the team addressed many challenges and overcame any problems that would arise. With that said the project aimed to deliver an accurate solution to counting persons entering/exiting the makerspace room. [6]



The image shows a screenshot of an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I
1	Timestamp	Camera 1 Detections	Camera 2 Detections	Total Detections					
2	2023-12-01	1	0	1					
3	2023-12-01	2	1	3					
4	2023-12-01	1	0	1					
5	2023-12-01	1	1	2					
6	2023-12-01	1	0	1					
7	2023-12-01	1	0	1					
8	2023-12-01	1	0	1					
9	2023-12-01	1	0	1					
10	2023-12-01	1	0	1					
11	2023-12-01	1	0	1					
12	2023-12-01	1	0	1					
13	2023-12-01	1	0	1					
14	2023-12-01	1	0	1					
15	2023-12-01	1	0	1					
16	2023-12-01	1	0	1					
17	2023-12-01	1	0	1					
18	2023-12-01	2	0	2					
19	2023-12-01	1	0	1					
20	2023-12-01	1	0	1					
21	2023-12-01	1	0	1					
22	2023-12-01	0	1	1					
23	2023-12-01	1	0	1					
24	2023-12-01	1	0	1					
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									

Figure 6.2

7.0 Conclusion and Recommendations for Future Research

This Chapter consists of the report's conclusions and recommendations for future research on creating a Realtime surveillance system.

7.1 Conclusions

In conclusion, the implemented tracking system has achieved a huge level of success in continuous monitoring and detection of upper bodies of people through two web cameras. The interrogation of Haar cascade-based detection and KCF tracking methods provide a reliable framework for person identification and tracking. The system's robustness was accurate since it ran continuously without disruptions addressing the requirement for 24/7 surveillance. The timestamped of detection total counts in 5-minute intervals, writing data into an Excel file and reset the program at midnight making sure the data is accessible by the client for further analysis. While the system demonstrates a sturdy foundation for further upgrades, which could be applied to detection accuracy and optimizing the resources utilization. [6]

7.2 Future Recommendations

For future considerations, it is recommended to explore advanced computer vision techniques or machine learning models to improve the accuracy of person detection. Integration of deep learning algorithms, such as Convolutional Neural Networks (CNNs), could enhance the system's ability to identify and track individuals more effectively in challenging scenarios. Additionally, implementing a more diverse dataset for training could improve the model's generalization to different body types and orientations.

1. **Enhanced Detection Algorithms:** Explore and implement advanced detection algorithms or machine learning techniques for improved accuracy, especially in challenging scenarios.
2. **AI Integration:** Consider integrating artificial intelligence (AI) to enable the system to learn and adapt over time, enhancing its ability to handle varying conditions.
3. **User Interface Enhancements:** Develop a user-friendly interface for easier system management and configuration. This could include a dashboard for real-time monitoring and customizable settings.

4. **Cloud Integration:** Explore the possibility of integrating the system with cloud services for remote access, data storage, and scalability.
5. **Extended Hardware Support:** Investigate compatibility with additional hardware components, such as different camera types or sensors, to broaden the system's applicability.
6. **Automated Alerts:** Implement an alert system that notifies administrators or relevant personnel in real-time when predefined thresholds are exceeded, enhancing the system's responsiveness.
7. **Integration with Access Control Systems:** Explore integration with access control systems to provide additional functionality, such as tracking entry and exit patterns.
8. **Privacy Features:** Integrate privacy-focused features, such as anonymization techniques, to align with evolving privacy regulations and user expectations.
9. **Mobile Application:** Develop a mobile application for remote monitoring and management, providing flexibility for users.
10. **Energy Efficiency:** Optimize the system for energy efficiency, exploring methods to reduce power consumption without compromising performance.
11. **Community Feedback:** Gather feedback from users and stakeholders to identify specific needs and areas for improvement, ensuring the system meets evolving requirements.

Moreover, considering the deployment of the system in real-world scenario, mobile apps, or web pages for user interface on accessing data remotely will be efficient. Finally, according to the regulations considering privacy and ethical implications make sure always in the borders of accepting regulations and law.

References

- [1] O. team, OpenCV, Chicago, 2023.
- [2] P. community, face-recognition 1.3.0, 2023.
- [3] su77ungr, opencv/data/haarcascades, 2022.
- [4] J. C. Redmon, YOLO: Real-Time Object Detection, 2018.
- [5] doxygen, OpenCV Cascade Classifier, Chicago.
- [6] K. KALRA, OBJECT TRACKING.