

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**REAL – TIME HUMAN DETECTION & COUNTING**”, is the Bonafide work of **BERNARD ALAN RAJ P (211720104027)**, **BOOBESH K (211720104033)** and **DEEPAK KUMAR P (211720104038)** who carried out the mini project work under my supervision.

SIGNATURE:

Dr. R. SARAVANAN,
M.E., Ph.D.,
HEAD OF THE DEPARTMENT,
Professor,
Dept. of Computer Science and
Engineering.,
Rajalakshmi Institute of Technology,
Kuthambakkam Post,
Chennai - 600 124

SIGNATURE:

Ms. D. SUGANTHI,
M.Tech.,
SUPERVISOR,
Asst. Professor,
Dept. of Computer Science and
Engineering.,
Rajalakshmi Institute of Technology,
Kuthambakkam Post,
Chennai - 600 124

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere gratitude to our honorable Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, M.A., M.Phil., Ph.D.**, and Chairman **Thiru. S. MEGANATHAN, B.E., F.I.E.**, for their constant encouragement to do this mini project and also during the entire course period.

We thank our Principal **Dr P. K. NAGARAJAN** and our Head of the department **Dr R. SARAVANAN, M.E., Ph.D.** for their valuable suggestions and guidance for the development and completion of this mini project.

Words fail to express our gratitude to our Mini Project coordinators **Dr T. NITHYA, M.E., Ph.D.** and **MR R. ARUNKUMAR, M.Tech.** Internal guide **MS D. SUGANTHI, M.Tech.** who took special interest in our mini project and gave their consistent support and guidance during all stages of this mini project.

We thank all the teaching and non-teaching faculty members of the **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING** of our college who helped us to complete this mini project.

Above all we thank our parents and family members for their constant support and encouragement for completing this mini project.

ABSTRACT

This project investigates and reports benchmarks for detecting and enumerating humans through real time images, videos and camera. This is very useful in various image processing and performing computer vision tasks. These schemes have been implemented in Python programming language, and using various tech-stacks like OpenCV [2], TensorFlow [3], etc.

Keywords: Computer Vision, Human Detection, Enumeration

TABLE OF CONTENT

| S.NO | CONTENTS | PAGE |
|------|--------------------------------|------|
| 1. | Introduction | 07 |
| 2. | Problem statement | 09 |
| 3. | Objective | 11 |
| 4. | Literature Survey | 12 |
| 5. | Proposed System | 14 |
| 6. | Architecture Diagram | 16 |
| 7. | Source Code | 17 |
| 8. | Implementation& Output | 18 |
| 9. | Graphical User Interface (GUI) | 21 |
| 10. | Accuracy | 24 |
| 11. | Performance | 25 |
| 12. | Further Improvement | 26 |
| 13. | Challenges | 28 |
| 14. | Result | 31 |
| 15. | Future Works | 32 |
| 16. | conclusion | 34 |
| 17. | Reference | 35 |

TABLE OF FIGURES

| FIG.NO | TITLE | PAGE |
|--------|------------------------|------|
| 1 | Architecture Diagram | 16 |
| 2 | Source Code | 17 |
| 3 | Implementation | 18 |
| 3.1 | Sample output-I | 19 |
| 3.2 | Sample output-II | 20 |
| 4 | GUI-I | 21 |
| 4.1 | GUI-II | 21 |
| 4.2 | GUI-III | 22 |
| 4.3 | GUI-IV | 22 |
| 4.4 | GUI-V | 23 |
| 5 | Further Improvement-I | 26 |
| 5.1 | Further Improvement-II | 27 |

INTRODUCTION

The rapid advancements in computer vision and artificial intelligence have led to significant developments in real-time human detection and counting systems. These systems find numerous applications in various domains, including surveillance, crowd management, and retail analytics. In this mini project, we aim to develop a real-time human detection and counting system using OpenCV, Python, and the Haar Cascade algorithm.

The objective of this project is to detect and count the number of humans present in a given video stream or real-time video feed. The project utilizes OpenCV, an open-source computer vision library, which provides powerful tools and algorithms for image and video processing. One of the key algorithms used in this project is the Haar Cascade algorithm.

The Haar Cascade algorithm is an efficient object detection technique that uses a series of trained classifiers to detect specific objects in images or video frames. It is particularly well-suited for detecting objects with distinct features, such as human faces. The algorithm works by comparing the patterns of pixel intensities in different regions of an image to pre-trained patterns of the object of interest.

For our project, we will adapt the Haar Cascade algorithm to detect human bodies instead of faces. The Haar Cascade classifiers for human detection have been trained on a large dataset of positive and negative samples, allowing them to accurately distinguish human bodies from other objects or backgrounds.

To implement the real-time human detection and counting system, we will use Python programming language along with OpenCV's Python bindings. Python provides a simple and intuitive syntax, making it ideal for rapid prototyping and development of computer vision applications. OpenCV's extensive functionality, including image processing, object detection, and video analysis, will enable us to efficiently process the video stream and perform real-time human detection.

Throughout this mini project, we will explore the steps involved in developing the system, including preprocessing the video feed, applying the Haar Cascade algorithm for human detection, tracking the detected human objects, and incrementing the count based on the detected humans. We will also focus on optimizing the system for real-time performance, ensuring smooth and accurate human detection and counting in various scenarios.

Our project aims to provide a solution that can accurately detect and count humans in real-time, enabling applications in various domains. By utilizing OpenCV and the Haar Cascade algorithm, we will develop a system that can process video streams, identify and track human presence, and generate real-time counts. The outcome of this project will contribute to the field of computer vision and enable the development of intelligent systems capable of understanding and analyzing human activity.

Throughout this project, we will face challenges related to training the Haar Cascade classifier, optimizing the algorithm for real-time performance, handling occlusion and environmental variations, and ensuring robustness and accuracy.

PROBLEM STATEMENT

Real-time Human Detection: The system should be able to detect human bodies in real-time from the video stream. It should utilize the Haar Cascade algorithm, which is specifically trained for human detection. The algorithm should be efficient and accurate in identifying human bodies, even in varying lighting conditions and different camera angles.

Accurate Counting: Once human bodies are detected, the system should increment the count for each detected human. The counting mechanism should be reliable and accurate, even in scenarios where humans may partially overlap or move in close proximity to each other.

Robust Performance: The system should be designed to handle real-time video feeds without significant latency or performance issues. It should efficiently process video frames, perform human detection, update the count, and display the results in real-time.

Preprocessing and Optimization: To improve the accuracy and efficiency of the human detection process, the system may require preprocessing steps such as background subtraction, noise reduction, or frame stabilization. These preprocessing techniques should be implemented to enhance the performance of the Haar Cascade algorithm.

User Interface: The system should provide a user-friendly interface to display the real-time video feed, the count of detected humans, and any additional relevant information.

Real-Time Performance: The system should achieve real-time performance, processing video frames at a sufficient frame rate to provide instant human detection and counting. It should be optimized for efficient computation and utilize parallel processing techniques, if necessary, to ensure fast and responsive operation.

Integration and User Interface: The system should be integrated with an appropriate user interface that allows users to input video streams, view the real-time human detection results, and access the human count information. The user interface should be intuitive, user-friendly, and provide necessary controls for configuration and interaction.

By addressing these challenges and implementing an efficient and accurate real-time human detection and counting system, we aim to provide a solution that can be utilized in various domains, such as crowd monitoring, surveillance, and people flow analysis. The successful completion of this mini project will demonstrate our ability to leverage computer vision techniques, specifically the Haar Cascade algorithm, to solve real-world problems related to human detection and counting.

OBJECTIVE

The main objective of this mini project is to develop a real-time human detection and counting system using OpenCV, Python, and the Haar Cascade algorithm. The project aims to leverage computer vision techniques to accurately detect and count the number of humans present in a given video stream or real-time video feed.

Utilize the Haar Cascade algorithm to detect human bodies in video frames or a live video feed. Adapt the pre-trained Haar Cascade classifiers to accurately identify human body patterns and distinguish them from other objects or backgrounds.

Count Detected Humans: Track the detected human objects across consecutive video frames and increment the count each time a new human is detected. Implement efficient algorithms to handle occlusions, scale changes, and other challenges that may arise during real-time human tracking. By accomplishing these objectives, this mini project aims to demonstrate the feasibility and effectiveness of using the Haar Cascade algorithm in real-time human detection and counting applications.

The project will showcase the capabilities of computer vision techniques and their potential in areas such as surveillance, crowd management, and people counting systems.

LITERATURE SURVEY

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 1, pp. I-511). IEEE. This seminal paper introduced the Haar Cascade algorithm and its application in rapid object detection. It provides an in-depth explanation of the algorithm and its training process.

Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media. This book serves as a comprehensive guide to the OpenCV library and covers various computer vision techniques, including object detection. It includes practical examples and code snippets in Python, making it an excellent resource for learning and implementing computer vision projects.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 1, pp. 886-893). IEEE.

This paper presents the Histograms of Oriented Gradients (HOG) algorithm for human detection. It compares the performance of HOG with other detection methods, including Haar Cascades, and provides insights into the strengths and limitations of different algorithms.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on

Computer Vision and Pattern Recognition (CVPR) (pp. 779-788). IEEE. This paper introduces the YOLO (You Only Look Once) algorithm for real-time object detection, which offers high accuracy and speed. While not directly related to Haar Cascade, it presents an alternative approach to real-time object detection and provides valuable insights into efficient implementation techniques. OpenCV Documentation: Haar Cascade Classifier.

In Malaysia, the government has established a series of standard operating procedure (SOP) for enterprises like limiting the store's crowd size, maintaining a safe social distance between individuals, and many more.

The government has even implemented an emergency law to fine enterprises which have violated or fail to perform the SOP. Even more, the fine amount has been enormously increased to 50,000 Malaysian Ringgit for companies and corporations effective from 11 March 2021 (Bernama, 2021). This has caused countless fears to enterprises, and manually monitoring its crowd all time is difficult and requires a heavy investment in human resources.

The official documentation of OpenCV provides detailed information about the Haar Cascade classifier, including its usage, training process, and examples.

PROPOSED SYSTEM

Video Feed Acquisition: The system will acquire a video feed from a camera or a pre-recorded video file. It will use OpenCV's Video Capture module to access the video stream and process the frames in real-time.

Preprocessing: To enhance the accuracy of human detection, preprocessing techniques will be applied to the video frames. These techniques may include noise reduction, image resizing, and contrast adjustment. Preprocessing helps to improve the quality of the frames and optimize the performance of the human detection algorithm.

Haar Cascade Algorithm: Haar Cascade algorithm will serve as the core detection algorithm in the system. It will be employed to detect human bodies in the video frames. OpenCV provides pre-trained Haar Cascade classifiers specifically trained for human detection. These classifiers will be loaded and used to detect human patterns in the video frames.

Human Tracking: Once humans are detected in a frame, a tracking mechanism will be implemented to track the detected human objects across consecutive frames. Tracking will ensure that the same human is consistently identified even if they move or are occluded in certain frames. Various tracking algorithms, such as the Kalman filter or optical flow, can be explored to achieve robust and accurate tracking.

Counting and Visualization: The system will maintain a count of the number of detected humans. Each time a new human is detected, the count will be incremented. The count will be displayed in real-time on the video stream, providing a visual representation of the number of humans present. Bounding boxes or markers can be overlaid on the video frames to indicate the location of detected humans.

Evaluation and Testing: The proposed system will be evaluated and tested using various video datasets or real-world scenarios. Performance metrics, including accuracy, processing speed, and robustness, will be measured to assess the effectiveness of the system. The system can be fine-tuned and optimized based on the evaluation results.

Human Counting: The system will employ counting techniques to determine the number of humans present in the video stream. It will keep track of detected humans and update the count in real-time as new humans enter or exit the scene.

Performance Evaluation: The proposed system will be evaluated on various real-world video streams to assess its accuracy, efficiency, and robustness. Evaluation metrics such as precision, recall, and processing speed will be used to gauge the system's performance.

Optimization for Real-Time Performance: Efficient implementation techniques and optimizations will be employed to ensure real-time performance of the system.

ARCHITECTURE DAIGRAM

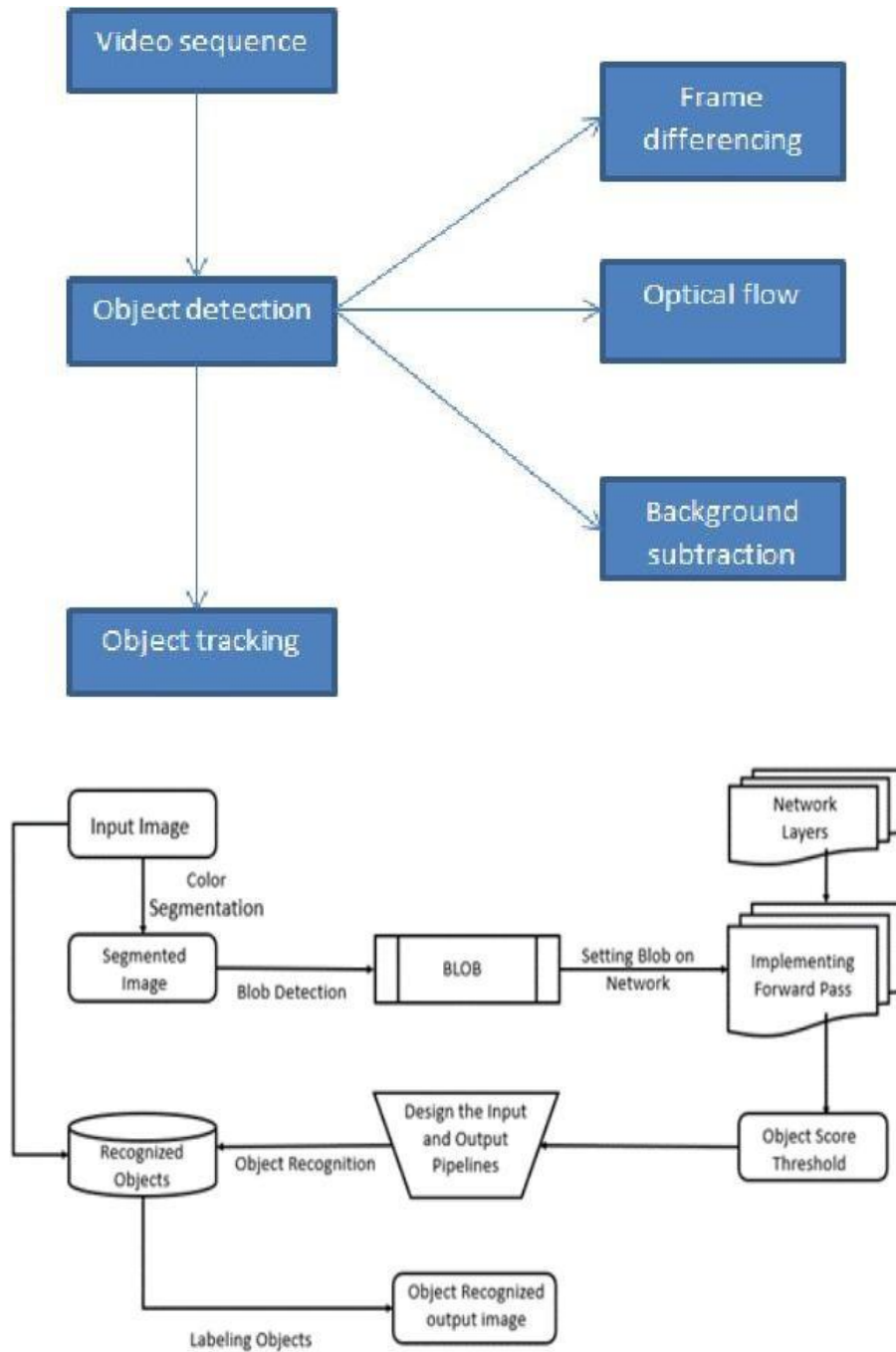


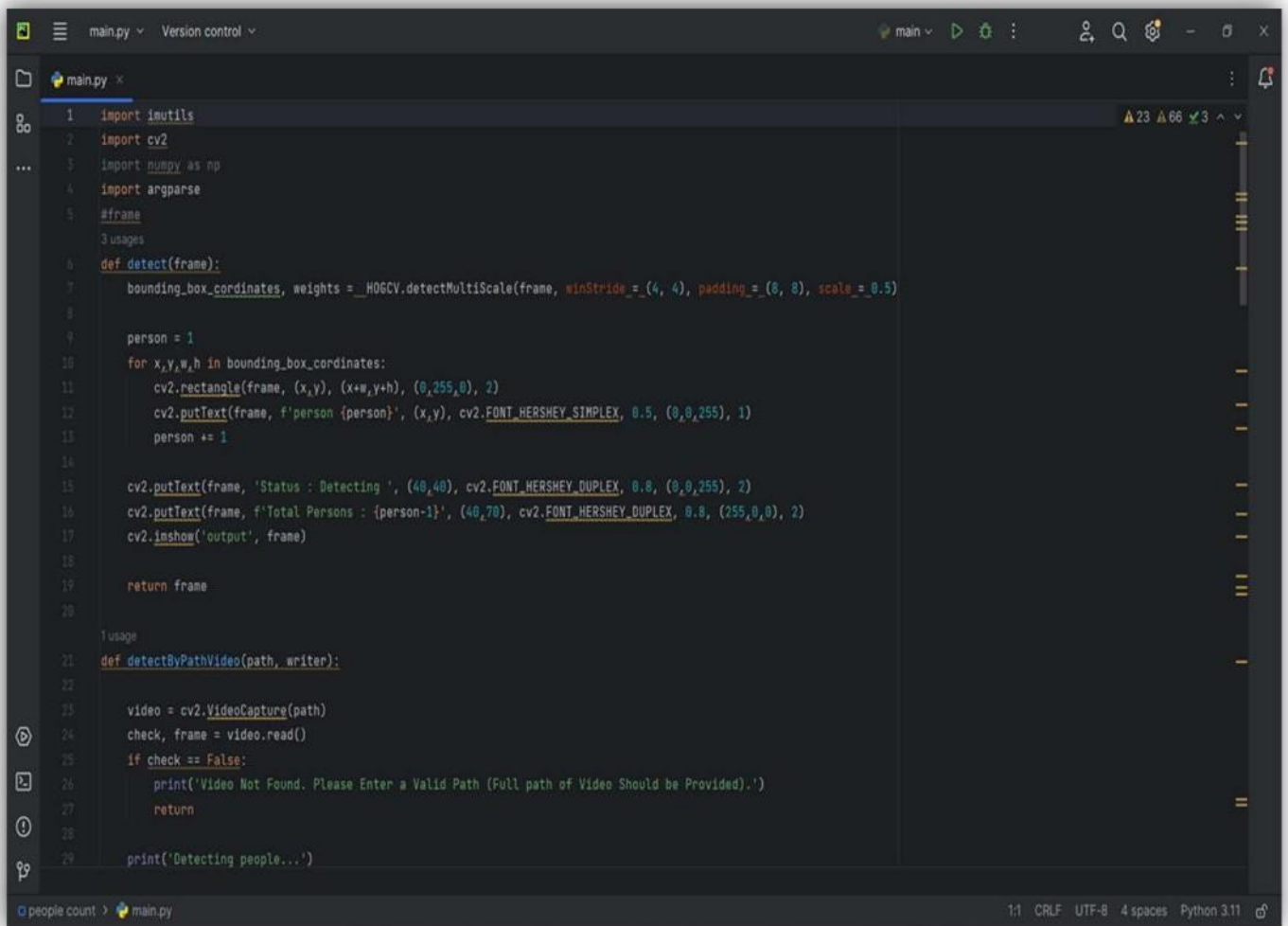
Fig 1

- This is the architecture diagram of the project.

SOURCE CODE

- Main python function for the application
- The below Image contains the Main Python Source code for Real time Human detection and counting. The Software used to implement the code is PyCharm.

Fig 2



```
1 import imutils
2 import cv2
3 import numpy as np
4 import argparse
5 #frame
6 3 usages
7 def detect(frame):
8     bounding_box_coordinates, weights = _HOGCV.detectMultiScale(frame, winStride=(4, 4), padding=(8, 8), scale=0.5)
9
10    person = 1
11    for x,y,w,h in bounding_box_coordinates:
12        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 2)
13        cv2.putText(frame, f'person {person}', (x,y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1)
14        person += 1
15
16    cv2.putText(frame, 'Status : Detecting ', (40,40), cv2.FONT_HERSHEY_DUPLEX, 0.8, (0,0,255), 2)
17    cv2.putText(frame, f'Total Persons : {person-1}', (40,70), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)
18    cv2.imshow('output', frame)
19
20    return frame
21
22 1 usage
23 def detectByPathVideo(path, writer):
24
25    video = cv2.VideoCapture(path)
26    check, frame = video.read()
27    if check == False:
28        print('Video Not Found. Please Enter a Valid Path (Full path of Video Should be Provided).')
29        return
30
31    print('Detecting people...')
```

IMPLEMENTATION & SAMPLE OUTPUT

- Selecting the input image
- This is the step to select the image for processing

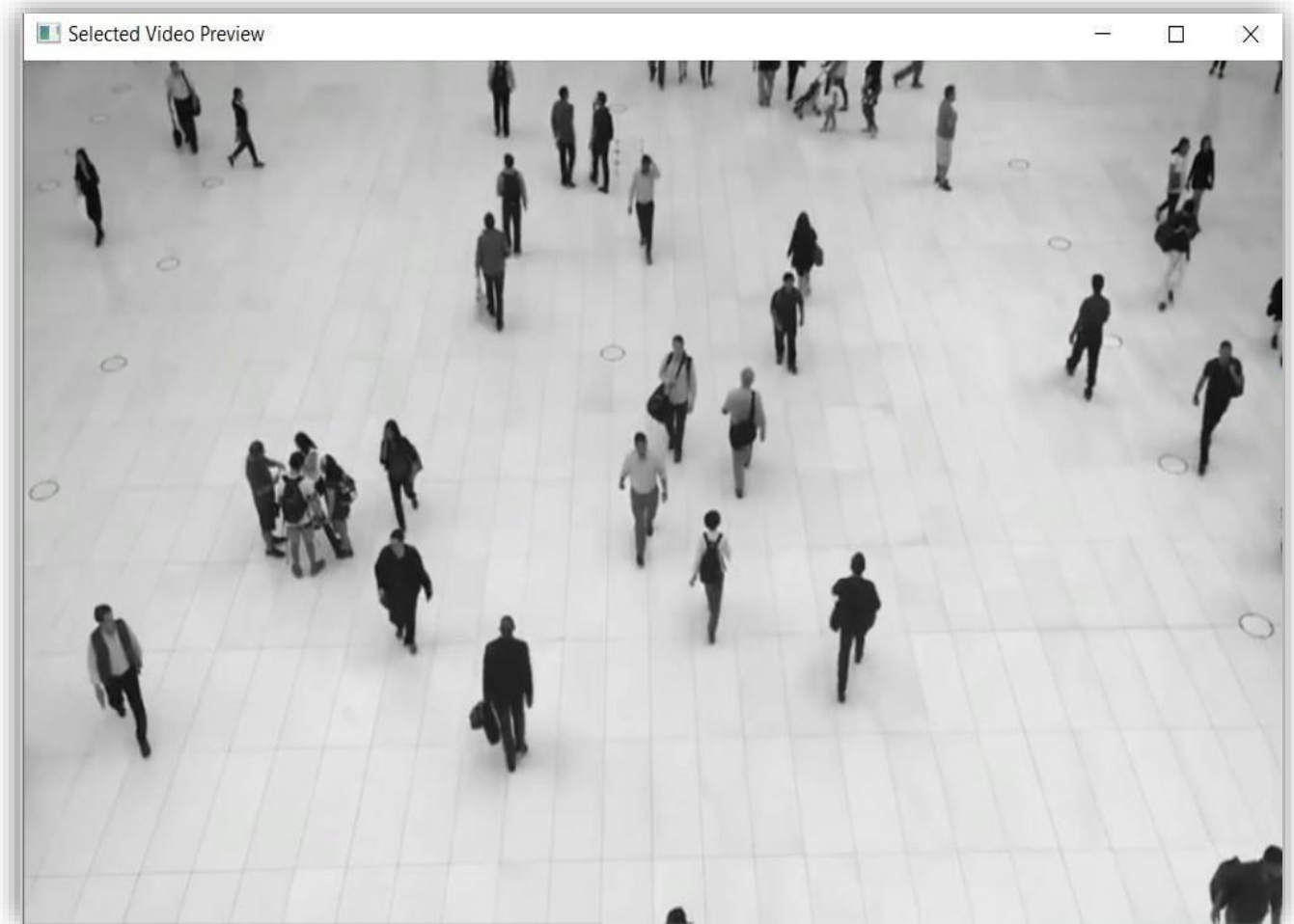


Fig 3

- In the same way when we select video from the local system, it will give the output by detecting the humans in the video.
- Like same way we can select for live camera for detecting the human



Fig 3.1

- This is the Output acquired for a given Image. The humans in the image are detected and also the total people count is noted.

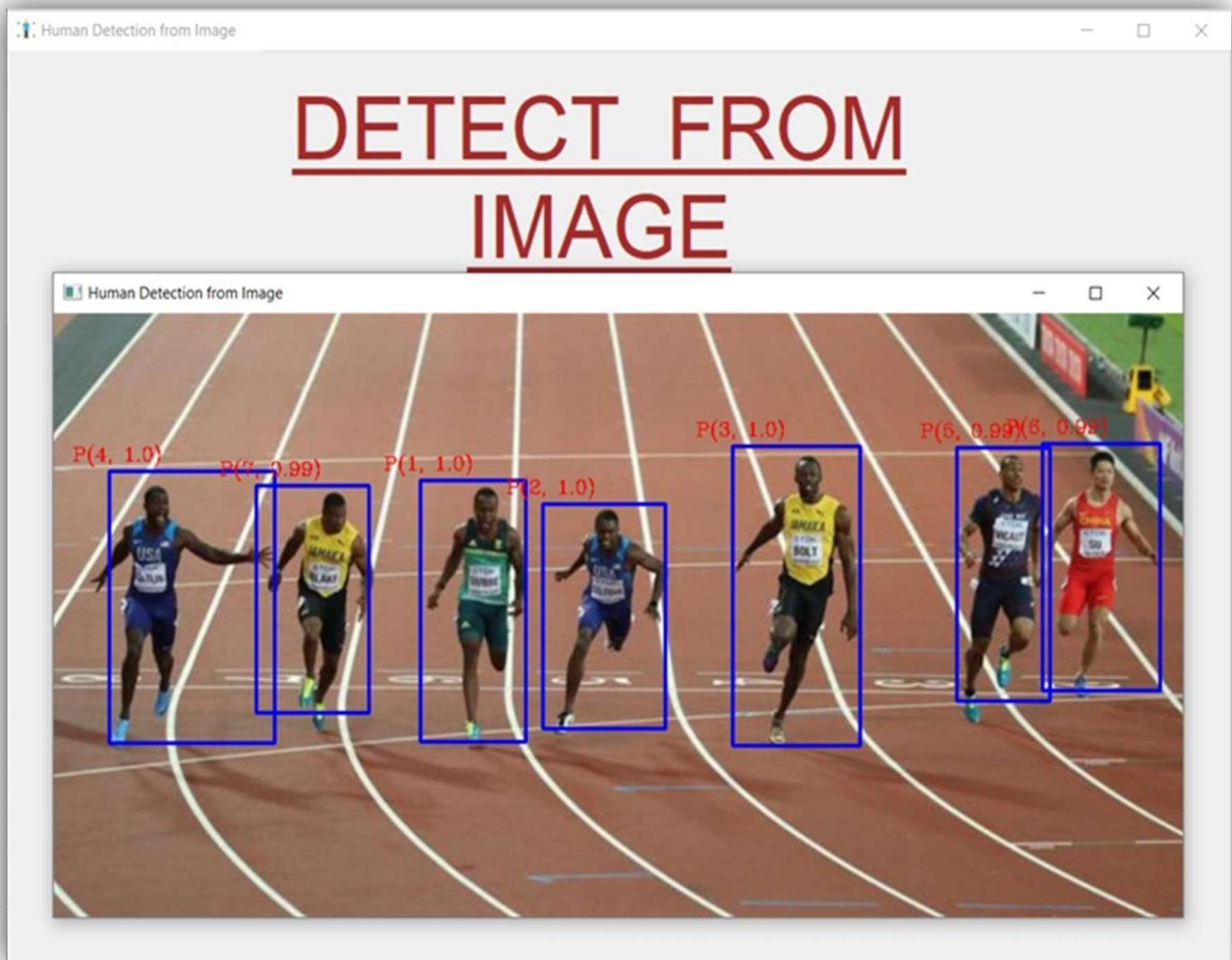


Fig 3.2

GRAPHICAL USER INTERFACE (GUI)

- When we click on the start button the application will start and we get different option to detect the human or person from them.

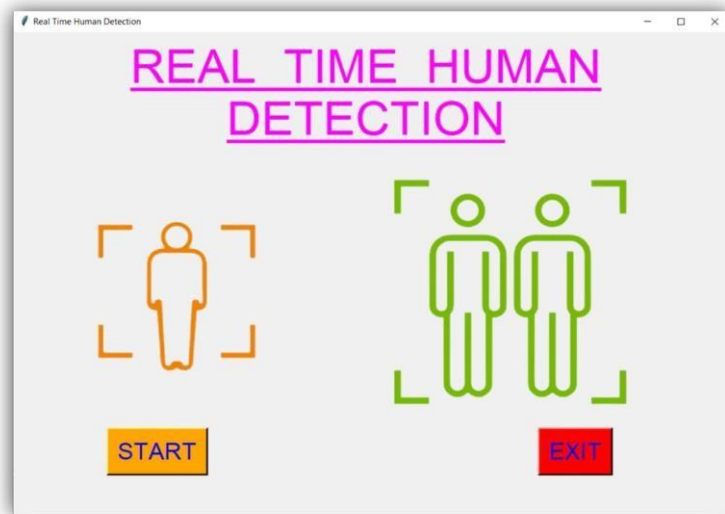


Fig 4

- The Software provides three different options, either to detect from a Local image or Video file or to detect from the live camera feed of the device.

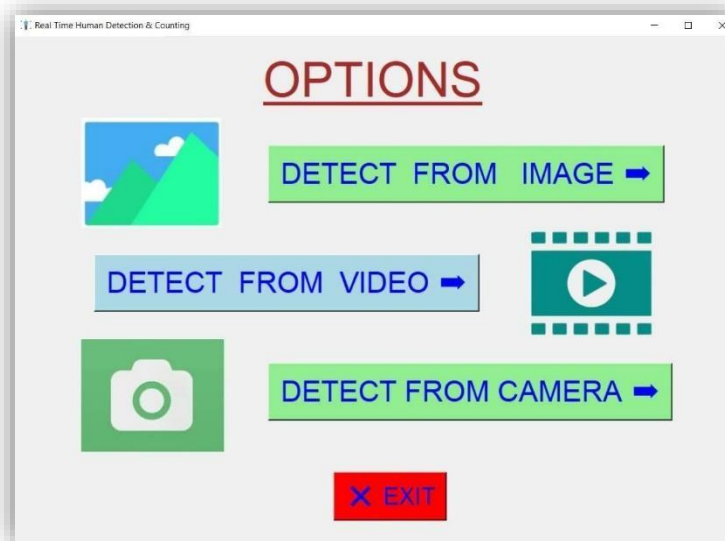


Fig 4.1

- When we click on select button on image option, we get option to select image from the local system and when click on the detect button, an output image is shown which detected human and their count

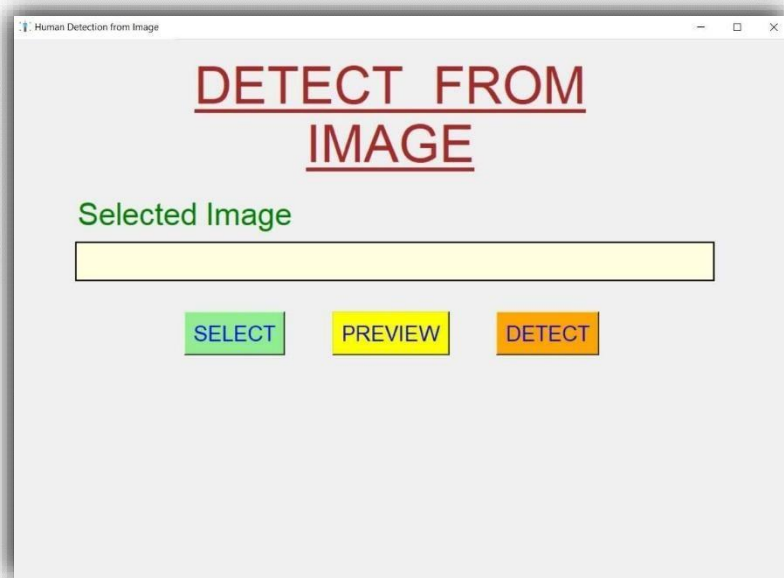


Fig 4.2

- The required image has to be selected in order to process it for Real time detection.

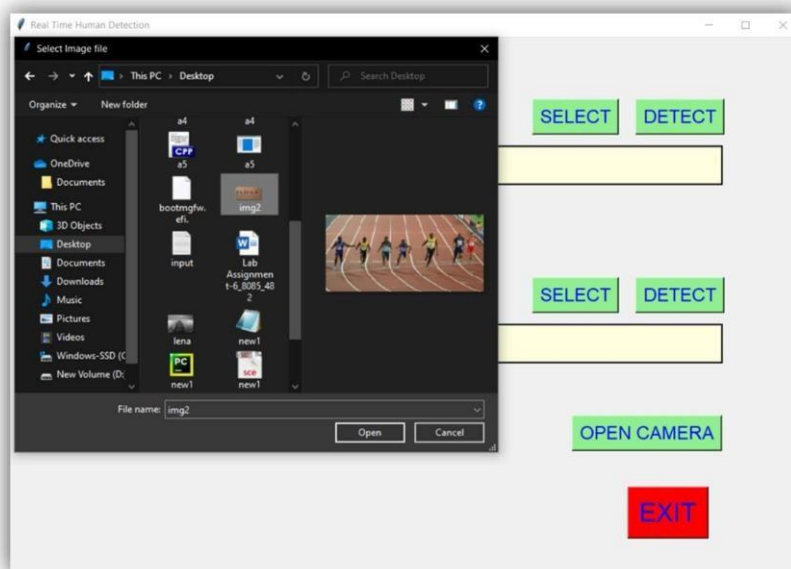
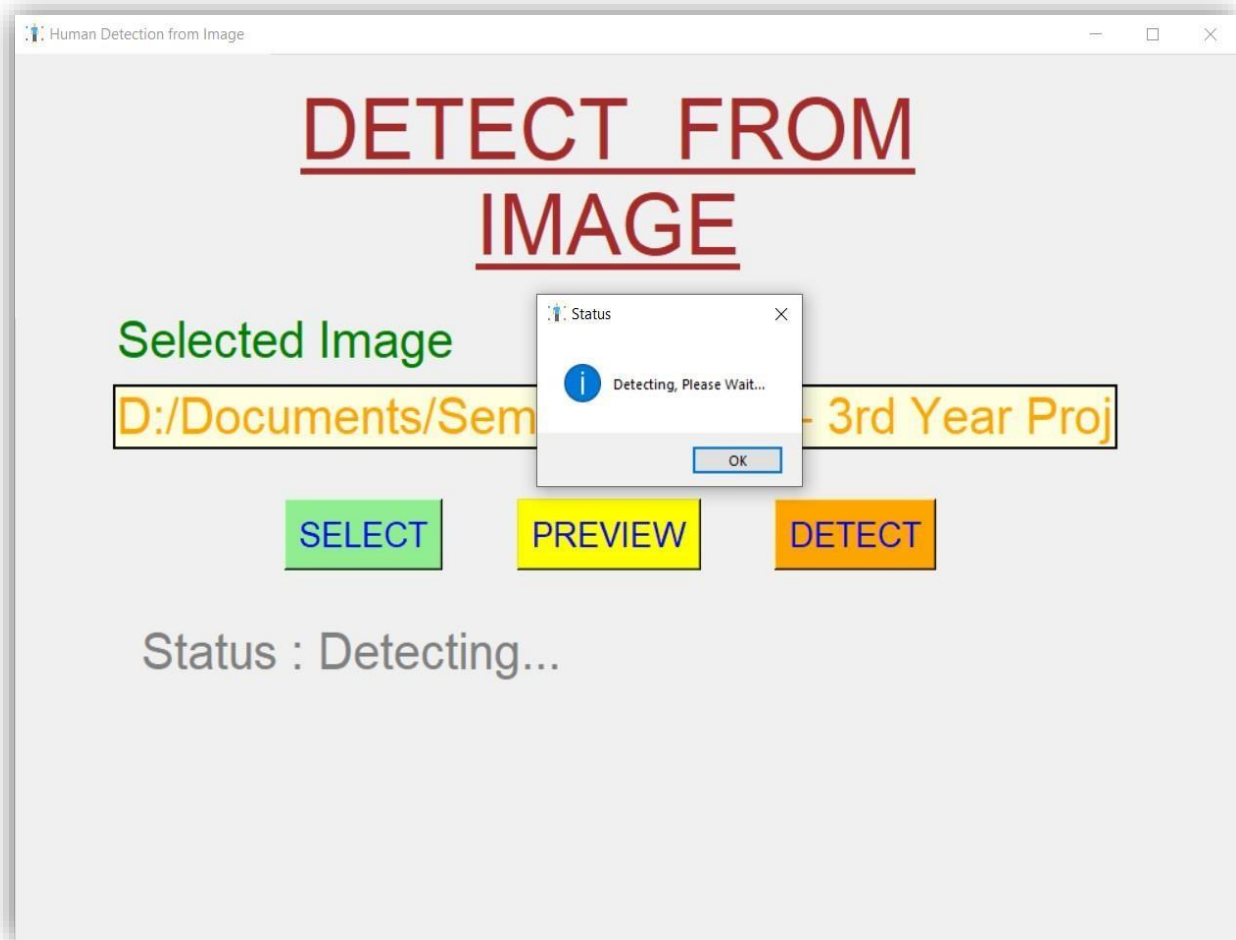


Fig 4.3

- The Path of the Image is given and the DETECT option is clicked.
- The status of the execution can be seen in the interface.
- The Output will be displayed in a new window.

Fig 4.4



ACCURACY

- The above table represents the accuracy rate of the Human detection software. It can be used to determine whether the software is efficient compared to others.

| No. | Camera orientation | Average performance |
|-----|--------------------|---------------------|
| 1 | Front view | 30 FPS |
| 2 | Front view | 25 FPS |
| 3 | Overhead view | 34 FPS |
| 4 | Overhead view | 34 FPS |
| 5 | Overhead view | 37 FPS |

| No. | Camera orientation | Average performance |
|-----|--------------------|---------------------|
| 1 | Front view | 7 FPS |
| 2 | Front view | 5 FPS |
| 3 | Overhead view | 7 FPS |
| 4 | Overhead view | 7 FPS |
| 5 | Overhead view | 7 FPS |

PERFORMANCE

- The below table shows the performance analysis of the Human detection software.

| No. | Camera orientation | Expected up count | Expected down count | Actual up count | Actual down count |
|-----|--------------------|-------------------|---------------------|-----------------|-------------------|
| 1 | Front view | 2 | 5 | 3 | 4 |
| 2 | Front view | 7 | 4 | 4 | 2 |
| 3 | Overhead view | 12 | 12 | 11 | 10 |
| 4 | Overhead View | 3 | 7 | 3 | 7 |
| 5 | Overhead view | 3 | 5 | 1 | 6 |

| No. | Camera orientation | Expected up count | Expected down count | Actual up count | Actual down count |
|-----|--------------------|-------------------|---------------------|-----------------|-------------------|
| 1 | Front view | 2 | 5 | 2 | 5 |
| 2 | Front view | 7 | 4 | 8 | 4 |
| 3 | Overhead view | 12 | 12 | 12 | 13 |
| 4 | Overhead view | 3 | 7 | 3 | 6 |
| 5 | Overhead view | 3 | 5 | 2 | 4 |

FURTHER IMPROVEMENT

- Executed the program and got the output as below with more accuracy on numbers detected and got the labelling also.



Fig 5

- In the above image the Software is not able to detect each and every person. This is due to the camera not being able to capture the details accurately. It is very important to have a camera with decent picture quality to implement live feed detection.

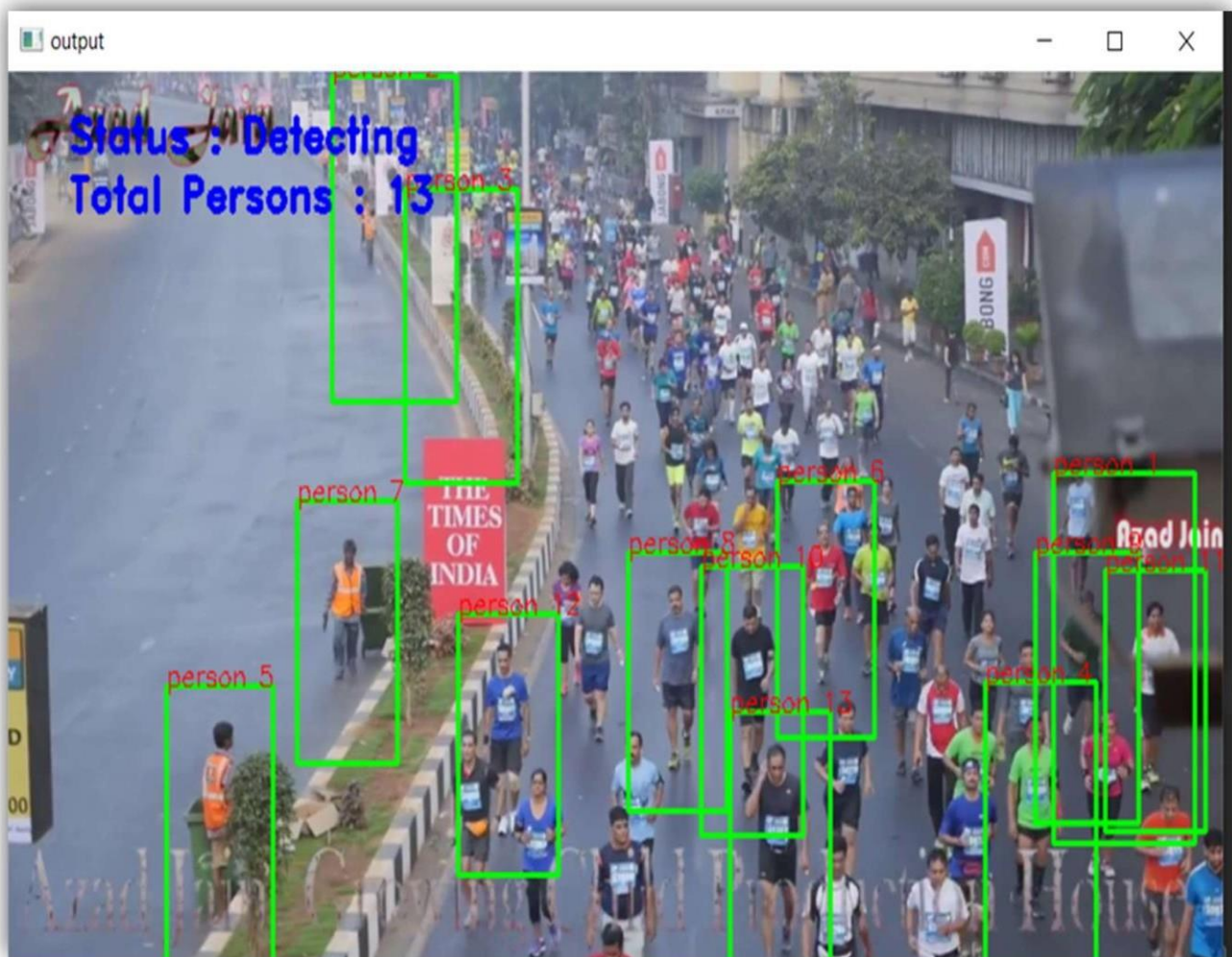


Fig 5.1

CHALLENGES

- The first major complication of object detection is its added goal: not only do we want to classify image objects but also to determine the objects' positions, generally referred to as the object localization task.
- Object detection algorithms need to not only accurately classify and localize important objects, they also need to be incredibly fast at prediction time to meet the real-time demands of video processing.
- For many applications of object detection, items of interest may appear in a wide range of sizes and aspect ratios. Practitioners leverage several techniques to ensure detection algorithms are able to capture objects at multiple scales and views.
- The limited amount of annotated data currently available for object detection proves to be another substantial hurdle. Object detection datasets typically contain ground truth examples for about a dozen to a hundred classes of objects, while image classification datasets can include upwards of 100,000 classes
- Ensuring the system's robustness to handle various real-world scenarios, different camera angles, scale variations, and different human appearances

- Changes in lighting conditions, shadows, or variations in the environment (e.g., indoor vs. outdoor) can affect the accuracy of human detection. Adapting the detection algorithm to handle different lighting conditions and environmental variations is a challenge to ensure robust performance in real-world scenarios.
- Haar Cascade-based human detection algorithms can suffer from false positives (incorrectly identifying non-human objects as humans) and false negatives (failing to detect actual humans). Balancing the detection thresholds, fine-tuning the classifier parameters, and incorporating additional techniques (e.g., post-processing, machine learning) can help mitigate these issues.
- Integrating the human detection and counting system with real-time video input and ensuring compatibility with specific hardware (e.g., cameras, processing units) can be challenging. Optimizing the implementation for the target hardware and addressing hardware limitations (processing power, memory) are essential considerations. (e.g., different clothing, hairstyles) is a challenge. The algorithm should be able to generalize well across different environments and conditions

- Assessing the accuracy, reliability, and performance of the human detection and counting system requires appropriate evaluation metrics and validation techniques. Gathering ground truth data for evaluation and comparing the system's output against the ground truth can be challenging.
- Training an accurate Haar Cascade classifier for human detection can be time-consuming and requires a diverse dataset of positive and negative samples. Collecting and labeling a comprehensive dataset that adequately represents different human poses, clothing variations, and environmental conditions can be a challenge.
- Real-time human detection and counting require efficient algorithms and optimizations to achieve fast processing speeds. Optimizing the performance of your code, leveraging parallel computing techniques (e.g., multi-threading), and implementing strategies like region of interest (ROI) detection can be challenging to achieve real-time performance.

RESULT

In this Project, we implemented a software for Real Time Human detection and counting. We used OpenCV module in Python along with Harr Cascade algorithm in this project. We ran the software with a various set of images and videos and found it to be accurate. The software was also able to detect through live camera feed. The Performance and Accuracy of the software was noted.

FUTURE WORKS

1. **Multi-camera Support:** Extend the system to support multiple camera inputs simultaneously. This would involve developing algorithms to handle the synchronization fusion of multiple video streams for improved human detection and counting accuracy in complex scenarios.
2. **Integration of Deep Learning Techniques:** Investigate the integration of deep learning-based approaches, such as convolutional neural networks (CNNs) and object detection frameworks like YOLO or SSD, to enhance the performance and accuracy of human detection. Explore the use of pre-trained models or train custom models on large-scale human detection datasets.
3. **Crowd Analysis and Behavior Understanding:** Expand the scope of the project to include crowd analysis and behavior understanding. Explore techniques to classify different types of crowd behavior, detect anomalies, and track individual movement patterns. This can have applications in crowd management, security, and public safety.

4. Occlusion Handling: Improve the system's capability to handle occlusions where humans may be partially or fully blocked by other objects or individuals. Investigate advanced techniques, such as pose estimation or part-based models, to handle occlusions and improve the accuracy of human detection and tracking.
5. Real-time Performance Optimization: Further optimize the system's performance to achieve even faster real-time processing. Explore techniques like model quantization, efficient hardware utilization, or algorithmic optimizations to reduce computational requirements and improve overall system performance.
6. Integration with Higher-Level Systems: Integrate the real-time human detection and counting system with higher-level systems, such as smart surveillance or intelligent video analytics platforms. This could involve feeding the human count data into a central monitoring system or integrating with other modules for advanced analytics and decision.
7. Pose Estimation: Incorporate pose estimation techniques to extract additional information about the detected humans, such as body orientation, gestures, or activity recognition. The system's capabilities and enable more advanced analysis of human behavior.

CONCLUSION

Using OpenCV and Python, we built a people counter. It is possible to incorporate a model that calculates the distance between the bounding boxes and thus improves the precision of the violation. The performance of object detection in image processing is required for a growing number of real time applications, and we can detect any type of object with this application. We will use various types of extraction process techniques in the future for various purposes, and this technique can be used in airports, shopping malls, businesses, parks.

REFERENCES

- W. Liu, M. Salzmann, and P. Fua, “Context-aware crowd counting,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 5099–5108.
- Object Detection with Deep Learning” Zhao, Z.-Q.; Zheng, P.; Xu, S.-T.; Wu, X(2019).
- V. A. Sindagi and V. M. Patel, “HA-CCN: Hierarchical attention-based crowd.
- counting network,” IEEE Trans. Image Process., vol. 29, pp. 323–335, 2020.
- M. Li, Z. Zhang, K. Huang, and T. Tan, “Estimating the number of people incrowded scenes by MID based foreground segmentation and head shoulder detection,” in Proc. 19th Int. Conf. Pattern Recognit., Dec. 2008.

- Programming Computer Vision with Python, 1st Edition, Jan Eric Solem, 2012, O' Reily.
- Learning OpenCV, Adrian Kaehler and Gary Rost Bradski, 2008, O' Reily.
- Deep Learning with TensorFlow, Giancarlo Zaccone, Md. Rezaul Karim, Ahmed Menshawy, 2017.
- Python GUI Programming with Tkinter, Alan D. Moore, 2018.
- Python Standard Library, Fredrik Lundh, 2001, O' Reily.
- Piciarelli, C., & Foresti, G. L. (2011). Surveillance-oriented event detection in video streams. In Video event detection (pp. 32–41).

- Raghavachari, C., Aparna, V., Chithira, S., & Balasubramanian, V.(2015). A comparative study of vision based human detection techniques in people counting applications. In Second international symposium on computer vision and the internet (pp. 461–469).
- Jalled, F., & Voronkov, I. (2016). Object detection using image processing. In Computer vision and pattern recognition (pp. 1–6).
- javaTpoint. (2021, August 18). Advantage and Disadvantage of TensorFlow. Retrieved from javatpoint: <https://www.javatpoint.com/advantage-and-disadvantage-of-tensorflow>
- Li, B., Zhang, J., Zhang, Z., & Xu, Y. (2014). A people counting method based on head detection and tracking. In 2014 International conference on smart computing (pp. 136–141).

