SOP EYE

Group Members

Abdul Rauf (01-131182-003)

Muhammad Umair Tahir (01-131182-027)

Supervisor: Engr Joddat Fatima

A Final Year Project submitted to the Department of Software Engineering, Faculty of Engineering Sciences, Bahria University, Islamabad in the partial fulfillment for the award of degree in Bachelor of Software Engineering

July 2022

THESIS COMPLETION CERTIFICATE

Student Name: Abdul Rauf Enrolment No: 01-131182-003

Student Name: Umair Tahir Enrolment No: 01-131182-027

Programme of Study: Bachelor of Software Engineering

Project Title: SOP EYE

It is to certify that the above students' project has been completed to my satisfaction and to my belief, its standard is appropriate for submission for evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at __13% _ that is within the permissible limit set by the HEC. I have also found the thesis in a format recognized by the department.

Supervisor's Signature: _____

Date: _____ Name: _____

CERTIFICATE OF ORIGINALITY

This is certified that the intellectual contents of the project SOP EYE are the product of my/our own work except, as cited properly and accurately in the acknowledgements and references, the material taken from such sources as research journals, books, internet, etc. solely to support, elaborate, compare, extend and/or implement the earlier work. Further, this work has not been submitted by me/us previously for any degree, nor it shall be submitted by me/us in the future for obtaining any degree from this University, or any other university or institution. The incorrectness of this information, if proved at any stage, shall authorities the University to cancel my/our degree.

Name of the Student: Abdul Rauf

Signature: _____ Date: _____

Name of the Student: Umair Tahir

Signature: _____ Date: _____

ABSTRACT

The pandemic of 2019, the infamous COVID-19, hit the world and caused a global crisis. It affected everyone single person, directly and indirectly. People lost their jobs, their houses and most importantly lost two years of their life. From offices to educational institutes to out-door activities everything had been halted. The economic as well as the social damages caused by the pandemic have still not been recovered. Nearly 3.3 billion workers and employees were affected and productivity around the world had been heavily compromised. A year later, in order to regain and recover from the losses, companies started bringing back their employees with strict adherence of SOPs advised by the World Health Organization (WHO). In order to properly implement these SOPs a monitoring agent was of dire need. With advancements in technology and artificial intelligence, the problem of a monitoring agent could be solved. Hence, SOP EYE, a tool to monitor and detect SOP violations using cutting edge software technology came into existence. SOP EYE detects COVID-19 SOPs through live camera feed and alerts the administrators of the violations. With this tool there will not be any need of a physical agent since the monitoring part will be automated. SOP EYE will be a helpful tool in detecting three types of SOP violations, face-mask, six-foot distance and physical touch detection. The agent has been trained on custom objects using appropriate pre-trained models. It detects violations in near real-time using only a camera feed as the input. On detecting violations more than a set threshold, the product sounds an alarm alerting the allocated people to reduce the violations and follow the SOPs. SOP EYE can be re-trained according to many other SOPs other than those mentioned. SOP EYE was initially planned to help in monitoring COVID-19 SOPs but it is not static and can be modified if, God-forbid, any other pandemic occurs.

Keywords: Machine Learning, Object Detection, SOPs, Custom Model, Model Training, Pre-trained Models.

ACKNOWLEDGEMENT

First all, all praises and thanks to Almighty Allah (SWT) for giving me the strength and understanding required to work on this project, our parents for their love and support and their unmatched prayers and our professors. Special thanks to our supervisor, Engr. Joddat Fatima, for her perceptive remarks, guidance and continuous ideas that helped us a lot during our research and writing of this thesis. Her vast knowledge and experience helped us to complete our project in. Also Sir Aleem Ahmad to guide us in the closure of the project. Lastly to my friends for supporting us and helping us achieve our goals.

CONTENTS

List of Figures

List of Tables

CHAPTER – 1

INTRODUCTION

Chapter 1

Introduction

In the early months of 2020, the disease called the Coronavirus disease also known as COVID-19 emerged from China and spread all over the world rapidly. It affected around 210 countries with more than 67 million confirmed cases and more than 1.5 million deaths were recorded. Pakistan was also very deeply damaged by this disease, considering the density of population, health care and poverty levels, over 420,000 people were infected and about 8300 or more people lost their lives.

This was a small example of what dangers and implications this deadly virus brought to the world. COVID-19 affected every person around the world whether directly or indirectly. According to the experts, an economic loss of around 10%, i.e. 1.1 trillion PKR, was observed in the year 2021.

Companies around the world suffered financially as well as socially a lot due to COVID-19. Small businesses had to shut down because there was no profit. Schools and educational institutes were also closed due to the pandemic and were shifted online which was insufficient as compared to education on the premises. Hope was running out and crisis starting to mount. But then, WHO announced some Standard Operating Practices (SOPs) which would reduce the chances of getting the disease drastically. Since the disease spread due to people coming in contact which each other physically, the SOPs were mostly against people touching each other, wearing face masks, avoid gathering in public and keeping a distance of 6ft in between themselves.

Following SOPs was a hope for the world to recover. In order to apply these SOPs a monitoring body was required, a system which could detect violations of such SOPs easily. Now, a human eye can only focus on somethings at a time that too for a short period of time but as compared to machines and Artificial Intelligence, this case is for too convenient.

With the world evolving in every aspect of technology and innovation, Artificial Intelligence and Machine Learning allows us to make things which could easily do things a human find very difficult to do. Using this technology, we can automate the process of monitoring and improve it to a great extent. The main idea of this project was to monitor violation of some of the COVID-19 SOPs through live camera feed.

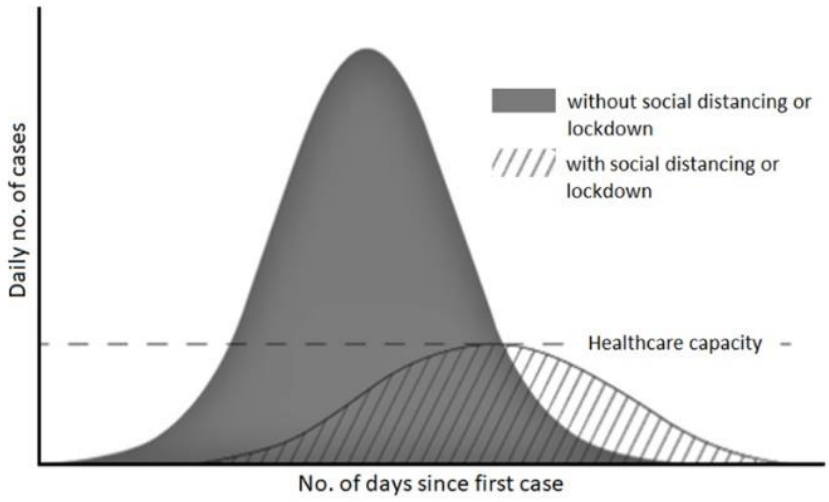

Figure 1-1 An outcome of social distancing as the reduced peak of the epidemic and matching with available healthcare capacity.

SOP EYE is a tool which can help authorities in Schools, Hospitals and Detainment centers to implement SOPs. SOP EYE uses a live camera feed and performs and shows the detections of SOP violations in real time.

Figure 1-1 shows graphical analysis of how the cases reduced with increase in social distancing

SOP EYE will be a helpful tool in detecting the following SOP violations:

- Face Mask Detection
- 6ft Distance Detection
- Physical Touch Detection

The agent has been trained on custom objects using appropriate pre-trained models which would be discussed further. The product detects objects and distances in real-time using only a camera feed as the input and performs detections and calculations.

Another feature in this monitoring system is that it triggers alarm once number of violations gets higher than the set threshold. The alarm will be used to alert the allocated people to reduce the violations and follow the SOPs.

SOP EYE is extensible and it can be re-trained according to many other SOPs other than those mentioned. SOP EYE was initially planned to help in monitoring COVID-19 SOPs but it is not static and can be modified if, God-forbid, any other pandemic occurs.

1.1. Motivation

The motivation behind making this project was mostly due to the troubles we faced when everything got shutdown due to COVID-19. From malls to work environments and educational institutes. Students had to attend online classes which had a lot of micro problems leading to difficulty in understanding and learning things. Hence the idea to make something which could help us overcome the pandemic came into existence. Along with that, we had also made a couple of projects related to tracking COVID-19 cases.

1.2. Problem statement

In order to control the rapid spread of COVID-19, an automated monitoring agent which could detect violations and alert authorities of COVID-19 SOPs in workplaces like classes and offices was required.

1.3. Objectives

To prevent the further spread of COVID-19 in offices and educational institutes by implementing SOPs. SOP EYE will detect violations and alert the authorities to take the appropriate actions.

The main objectives of SOP EYE are as follows:

1. *Face-Mask Detection*
2. *Social Distance Detection (6ft)*
3. *Touch Detection*

1.4. Main contributions

1.4.1. What is new, better and significant?

The new thing introduced in SOP EYE is firstly, the integration of three modules that too in near real time: Face mask Detection, 6ft Distance and touch detection. This has

not been performed anywhere as of yet. Lastly, the use of YoloV5, A cutting edge Object Detection model by DarkNet known to be one of the best pre-trained models.

1.4.2. How will it help?

SOP EYE would make it very convenient for authorities to implement SOPs in their work space. They will not have to worry about allotting people to monitor and maintain SOPs. This process will be automated.

1.5. Constraints and Scope

The project is very sensitive to the constraints since it involves real-time application of AI and ML techniques. The constraints of the SOP EYE are as follows:

- Lighting: The lighting in the video feed should be not to bright and not too dark. This would cause true false detections.
- GPU with CUDA computation capability of > 3.0 for faster processing of video.
- Processing power more than 2.5 GHz.
- Quality of the camera: The camera should be able to capture video in at least 30 frames per second (fps) and should be clear.

1.6. Report organisation

Chapter 1 of the report illustrates the intro and basic functionality of SOP EYE. In this chapter we've discussed main introduction, problem statement, motivation and objectives related to SOP EYE and our contributions.

Chapter 2 focuses on detailed background study and literature review. It illustrates all the work we've gone through and the past work contributed to this field. We've studied and discussed different research papers related to our project. Previously used techniques and models were discussed and compared with out piece of work.

Chapter 3 discusses system requirements including functional and non-functional requirements. We tried to explain system through Use-case diagram and different use cases.

Chapter 4 focuses on the design approaches, strategies and constraints necessary to carry out the project. Different UML diagrams such as Class diagram, activity diagram, sequence and component diagrams for SOP EYE are displayed.

Chapter 5 focuses on different methods, tools and technologies used. Different models were discussed. All the issues faced during implementation and their solution was discussed in this chapter.

Chapter 6 illustrates the test strategy, techniques and test cases. Unit, Integration and System testing techniques are discussed in this chapter.

Chapter 7: Lastly, in this chapter, we have concluded the whole thesis that depicts the work that has been done already and the future updates that we can look forward to adding upon this Web Application.

CHAPTER – 2

LITERATURE REVIEW

Chapter 2

Background Study/Literature Review

2.1. Literature Review and Background study

Computer vision and artificial intelligence are fields with a lot of potential of research. There was not a lot of research done on the application of computer vision in the field of managing COVID-19 SOPs until the late 2020's. We found some research papers which were very helpful in our work. Some of them are discussed below:

In [1] the authors discuss about the importance of having SOPs in order to control the spread of diseases. The identification of COVID-19 was being performed in the following methods (using computer vision): X-ray scans of lungs and detecting anomalies [2], Predicting the spread of COVID-19 using existing cellular wireless networks [3]. The paper [1] discusses about the two basic SOPs to maintain: 6ft distance and face masks since they can be detected through CCTV cameras.

2.1.1. Classical techniques to detecting humans and objects.

Zhang et al [4] proposed the use of Histogram of Gradients with SVM Classifier for detecting persons. They improved the approach by using multi-scale HOG features narrowed down by AdaBoost algorithm. This approach was able to detect multiple people but it had a problem. The high dimensionality of the feature vectors increased the computation cost of SVM classification. The authors solve this problem by using HOG as basic features and then creating reduced features using the AdaBoost algorithm. There results can be seen in **Error! Reference source not found.**:

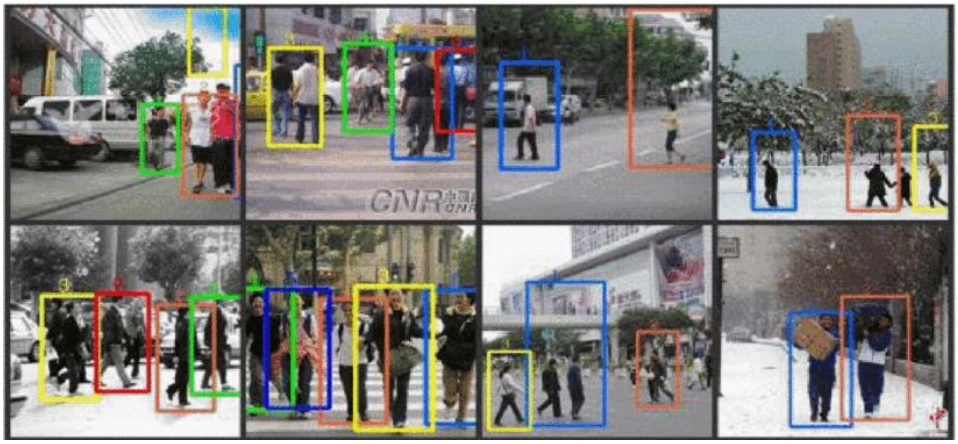

Figure 2-1 Experimental results of HOG

Figure 2-1 shows the results of the methodology proposed by Zhang.'s implementation of HOG with SVM Classifier

In the article [5] the authors mentioned in their research that a major problem in detecting human is the change of lighting, pose, backgrounds, occlusions and clothing. To tackle this issue, they proposed the use of Curvlet feature extraction. This allowed the transformation of an image into a combination of frequency bands. Helping in lighting issues in CCTV footage etc.

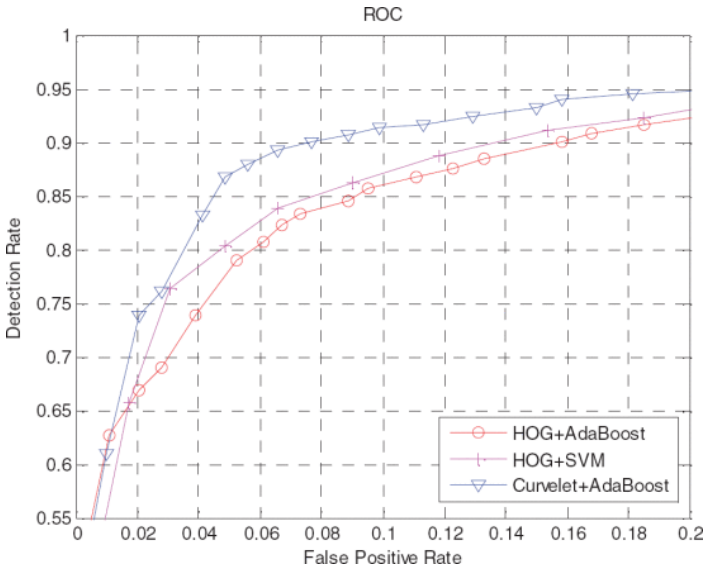

Figure 2-2 The ROC curves of HOG and curvelet feature extraction method

A graph representing the comparison of HOG with SVM and curvlet+AdaBoost is shown in Figure 2-2

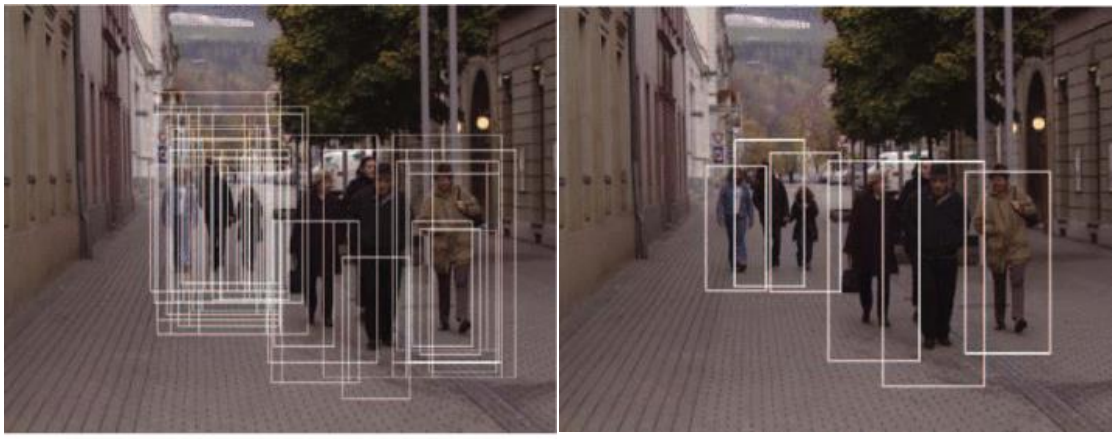

Figure 2-3 Detection result of proposed method

This method did prove to be better than HOG with SVM but it had a lot of parameters to manage and was really hectic. The results are shown in Figure 2-3

2.1.2. Face mask Detection Techniques.

Figure 2-4 Cascaded CNN structure results

Deore et al [6] suggested the use of HOG along with the Viola-Jones algorithm, a combination of Haar feature selection, integral image creation, Adaboost training and cascading classifiers. It initially classifies the parts of the face. If a person's mouth is not detected it is assumed that he or she is wearing a mask. Another author and his colleagues proposed in the paper [7] a CNN based approach for face mask detection.

The paper mentioned three approaches to detect face masks on people. The models had very good accuracy but they were not suitable for near real-time detections. The results can be seen in Figure 2-4

N. S. Punn et al [8] suggested a technique in which they used Yolo V3 for detecting distances. In order to track and calculate distances among people, the authors used YoloV3 along with Deepsort for tracking people and then the pairwise L2-norm is calculated with the aid of the bounding boxes. The author also mentions that according to another approach where the framework uses a deep CNN. Region of interest (ROI) is employed to abstain from crowding by changing the inflow. Both YOLO v4 and faster R-CNN are incorporated for pedestrian detection. **Error! Reference source not found.** shows the performance measure of the proposed model.

Table 2.1 Performance comparison of the object detection models

Model	TT (in sec.)	NoI	mAP	TL	FPS
Faster RCNN	9651	12135	0.969	0.02	3
SSD	2124	1200	0.691	0.22	10
YOLO v3	**5659**	**7560**	**0.846**	**0.87**	**23**

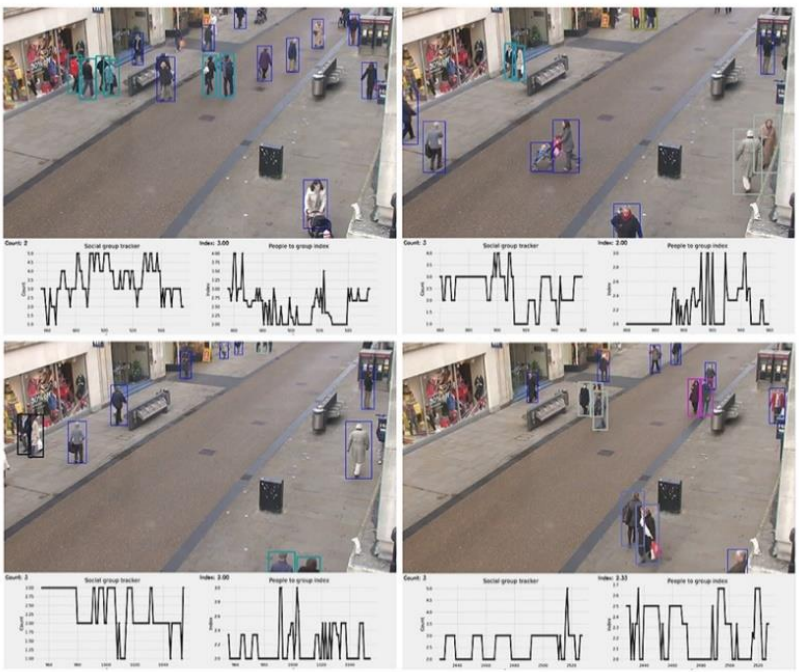

Figure 2-5 Sample output of the proposed framework for

Figure 2-5 shows the result of the proposed approach of the author of [8] article. Where they used Yolov3 to detect distances.

2.2. How previous work being described relates to my own.

We have opted the proposed research methodology of N.S. Punn [8]. Since we have used YoloV5 and they have used YoloV3, they have similar properties and methods. We have used its pre-trained model of Object Detection. It comes with a person detector by default and then we have used the bounding box coordinates to find the distance and Bounding Box Overlap for touch detection.

CHAPTER – 3

SYSTEM REQUIREMENTS

Chapter 3

System Requirements

COVID-19 broke out in 2020 which resulted in pandemic and lockdown across the World. It's rapid spread and deadly virus resulted in great increase in death toll and large drop in states economy. We as a student's confronted a massive disaster in our studies. However, to keep things going critical sectors such as hospitals, industries, educational institutions and government divisions must not be shut down. So, the WHO suggested some SOPs necessary to reduce the spread of COVID-19 virus. There was requirement and need of someone to monitor the regularity of those SOPs. With the world evolving, machine learning techniques are applied to automate the real-world problems.

So, this system was to automate the detection of COVID-19 SOPs violation using machine learning algorithms. The system has following features.

- Detect Face Mask
- Detect Social Distance (6 ft)
- Detect Touches.
- Alarm Tigger for several violations.

Basically, system is only to monitor SOPs violations so there is no such focus on User Interface.

3.1. Use Case Diagram

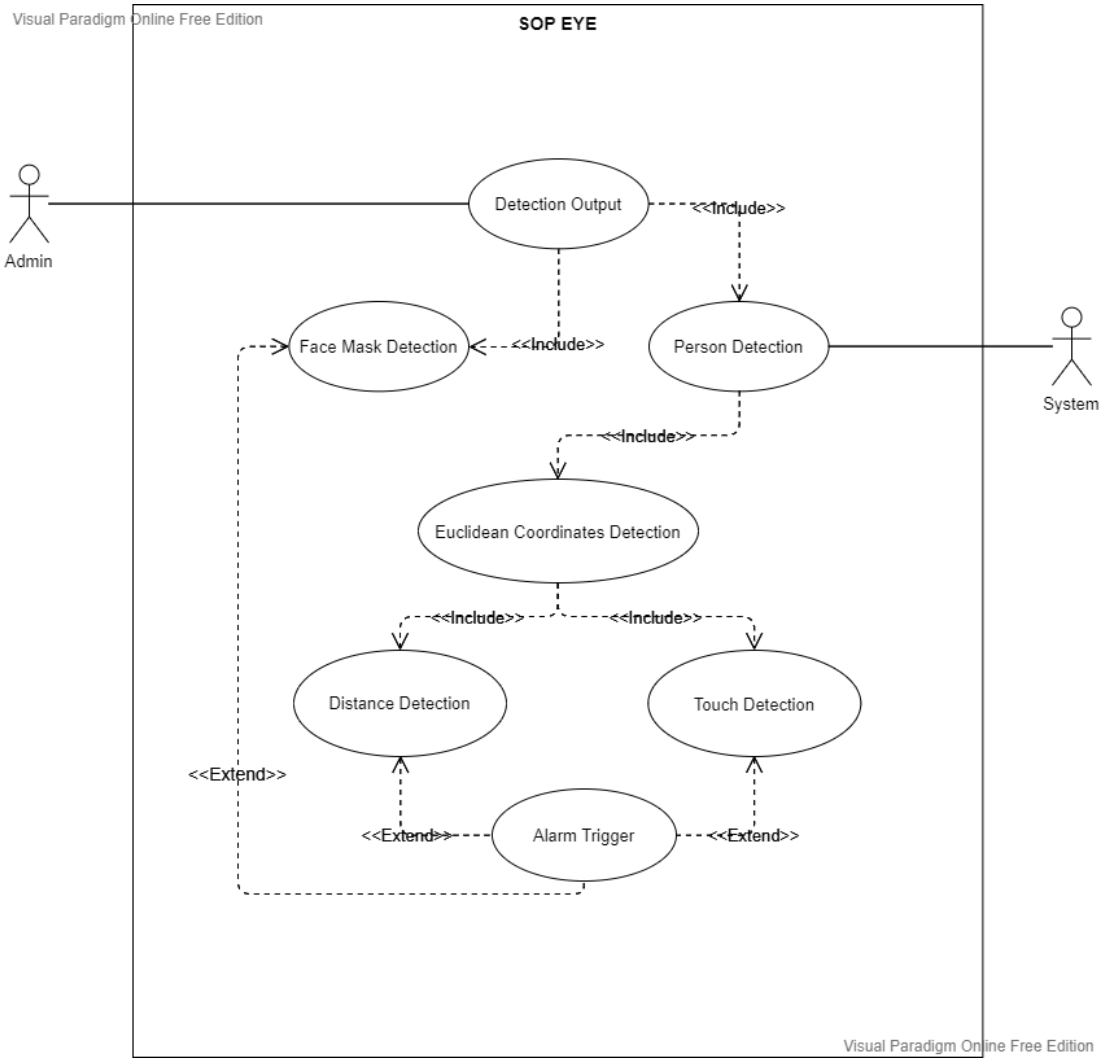

Figure 3-1: Use Case Diagram

There is only one primary actor "**Admin**" who will be only monitoring real time SOPs violation.

There is one secondary actor which is system itself which basically implements all the functionality and modules of SOPs detection.

Use cases include:

- Face-Mask Detection
- Social Distance Detection
- Touch Detection
- Person Detection
- Euclidean Coordinates Detection
- Alarm Trigger

Face-Mask Detection:

This use case defines that, it makes detections whether the person is wearing a face-mask or not.

Social Distance Detection:

This use case defines that, it makes detections whether the people are maintaining 6ft distance or not.

Touch Detection:

This use case defines that, it makes detections whether any one is making physical contact or not e.g., Hand shaking etc.

Person Detection:

This use case defines that, it makes person detections necessary for calculating their position and coordinates.

Euclidean Coordinates Detection:

This use case defines that, it makes calculations and measures the exact location of a person in a live camera feed.

Alarm Trigger:

This use case defines that, it calculates the ration of SOPs violation. If it exceeds the set threshold, people will respective area will be alarmed and asked to reduce violations through some audio.

3.2. Functional Requirements

- Face Mask Detection
- Social Distance Detection
- Touch Detection
- Alarm Trigger for critical situations

3.2.1. SOPs Detection

Table 3.1 This table defines necessary pre-conditions and the basic flow for "Face-Mask Detection"

Use Case ID	UC01
Use Case Name	Face-Mask Detection
Actors	Admin
Pre-Conditions	Monitoring agent should be running
Priority	High
Basic Flow	Admin opens SOP Eye and clicks on face-mask detection

Table 3.2 This table defines necessary pre-conditions and the basic flow for "Social Distance Detection"

Use Case ID	UC02
Use Case Name	Social Distance Detection
Actors	Admin
Pre-Conditions	Monitoring agent should be running
Priority	High
Basic Flow	Admin opens SOP Eyes and clicks on social distance detection

Table 3.3 This table defines necessary pre-conditions and the basic flow for "Touch Detection"

Use Case ID	**UC03**
Use Case Name	Touch Detection
Actors	Admin
Pre-Conditions	Monitoring agent should be running
Priority	High
Basic Flow	Admin opens SOP Eyes and clicks on Touch detection

Table 3.4 This table defines pre-conditions and the basic flow for "All SOPs Detection"

Use Case ID	**UC04**
Use Case Name	All SOPs Detection
Actors	Admin
Pre-Conditions	Monitoring agent should be running
Priority	High
Basic Flow	Admin opens SOP Eyes and clicks on detect all.

3.2.2. Alarm Trigger

Table 3.5 This table defines necessary pre-conditions and the basic flow for "Alarm Trigger"

Use Case ID	**UC01**
Use Case Name	Alarm Trigger
Actors	Admin
Pre-Conditions	Any of three SOPs detection must be running
Priority	High
Basic Flow	Admin opens SOP Eyes and clicks on any of SOPs detection button and system's detecting SOPs violations.

3.3. Interface Requirements

Let’s split interface into following three categories

- User Interface
- Hardware Interface
- Software Interface

3.3.1. User Interface

Basically, system is only to monitor SOPs violations so there is no such focus on user Interface. There would be a simple UI which will be showing live camera feed with couple of buttons.

There will be buttons to make individual SOP detections as well as button for detecting all SOPs violations all together.

3.3.2. Hardware Interfaces

Hardware Interface include interface between Computer (on which SOP EYE monitoring agent would be running) and the Security Camera (which will be used to get live camera feed)

3.3.3. Software Interfaces

Pre-trained models are used and trained on custom objects.

SSD MobileNet v2 FPN was used for Face-Mask detection whereas Yolov5 was used for social distance and touch detection.

Python was used as programming language to write all the necessary code including training, testing and real-time object detection.

Jupyter notebook was used to install and run all necessary libraries and models.

Alarm Trigger module had interface with Detection module, once violations exceeded, alarm triggers.

OpenCV was used to access live camera feed for real time object detection.

It could be trained and run on multiple Operating Systems such as Windows, Linux etc

3.3.4. Communications Interfaces

We will use CCTV cameras to capture live footages. It will send video footage via an IP network or some third-party app such as DroidCam to monitoring agent which might be running on Personal Computer over local network.

3.4. Database Requirements

There's no requirement of maintaining any type of data such as: number of violations, so no database is integrated.

3.5. Non-Functional Requirements

Non-functional requirements are selling point of a product. It defines the metrics that can be used to judge the functionality or performance of a system. The NFR for our system includes:

- Performance Requirements
- Safety Requirements

3.5.1. Performance Requirements

- The system must be reliable and work 24/7
- The Detections should be quick and accurate
- The results should be at least 90% accurate
- System must not label unwanted objects
- The system must perform detections in different circumstance (low light/high light)

3.5.2. Safety Requirements

- The average time to failure shall be a minimum of 1 month.
- In case of a server crash, a backup server will be up and running within half an hour.

3.6. Project Feasibility

Performing this project was technically feasible. We had enough time to deliver all the deliverables and enough help, tools, resources and expert guidance to carry out this project in time.

3.6.1. Technical Feasibility

To produce this product was technically feasible. We've all the required tools and experts available. There was a lot of help from the research paper which helped us understand technical difficulties. Our supervisor helped us a lot with all the uncertainties.

3.6.2. Operational Feasibility

The project is also operationally feasible. This project was taken considering the ever increasing and alarming situation of spread of COVID-19 19. The world has controlled its spread to a large extent but still there are COVID-19 cases and areas where SOPs are being followed. Yet 5th layer of COVID-19 (Omicron) case is also reported in Pakistan. Moreover, this project isn't static and can be modified if, God-forbid, any other pandemic occurs.

3.7. Conclusion

In this chapter, we have illustrated different system requirements necessary to develop and carry out the project. It includes all the functional, non-functional, and interface requirements. We also discussed project feasibility.

CHAPTER – 4

SYSTEM DESIGN

Chapter 4

System Design

All the modules were implemented separately at the first place. After successful implementation of each module, one by one all modules were integrated at single place and merged into the final product.

Whole System was implemented iteratively. Each module was implemented one by one and was iteratively evolved time to time until the complete implementation of the system. Modules were divided and implemented by different persons. However, system was tested by both members to maximize the quality.

Idea was to use pre-trained models with custom objects to train on our dataset. Research was made and different models were tested and compared. Model with appropriate speed and COCO mAP was selected.

For face-mask detection module we used SSD Mobilenet v2 FPN.

Yolov5 was used to make person detections and different techniques were applied to implement social distance and touch detection.

4.1. Design Approach

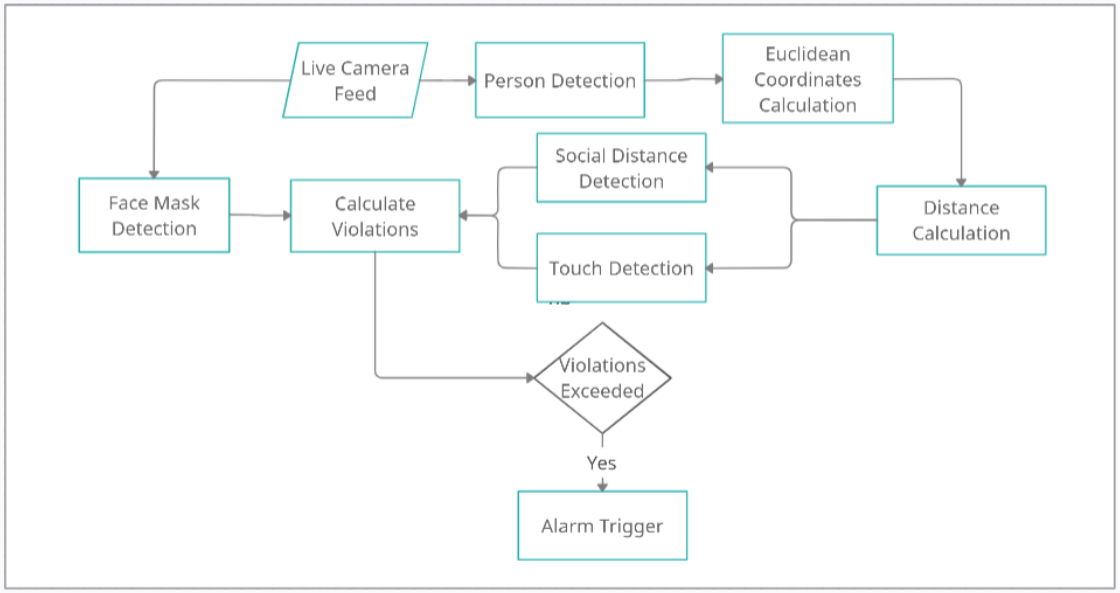

Figure 4-1: Design Approach of SOP EYE

Figure 4-1 shows the design approach for SOP EYE. This shows the main modules and the basic workflow of the application.

4.2. Design Constraints

The constraints of software caused by hardware:

- High quality camera
- GPU with good computing power and memory

Dataset constraints:

- The model was trained using dataset from
    - CCTV footages
    - Self-collected data
    - Kaggle

Toolkits Compatibility:

- CUDA and cuDNN versions must be compatible in order to use GPU for training model. In our case we used cuDNN 8.4.1 for CUDA 10.2.
- Libraries and models used must be compatible. We used v2 models compatible with TensorFlow v2 from TF model zoo.

Other constraints:

- The model made better detections for good quality video so obviously there was constraint of having good quality footage in order to produce better result.
- We got more accurate results at areas where backlight was higher as compared to areas where backlight was low.

4.3. Methodologies

For SDLC, waterfall model approach was used. All the phases were performed in sequence and were connected in a sequential order. The steps were taken in following sequence:

- Requirements Gathering
- System Designing
- System Implementation
- Integration and Testing
- System Deployment
- System Maintenance

4.4. System Architecture

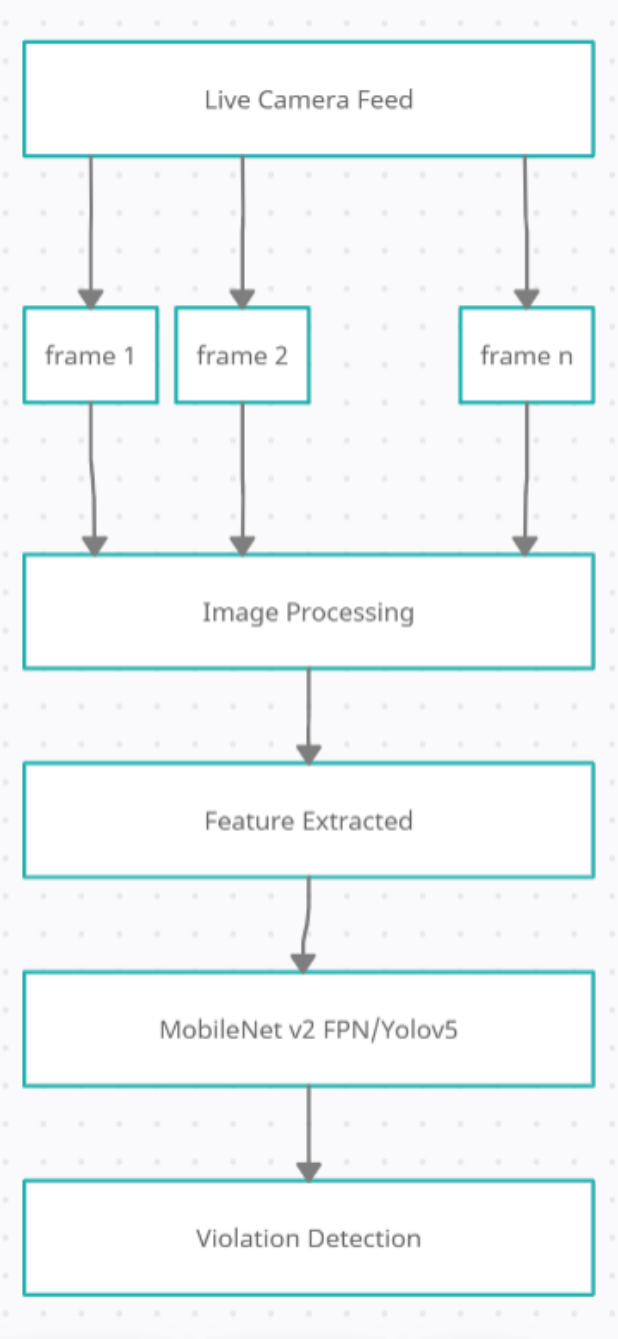

Figure 4-2 System Architectural Design

4.4.1. Data Layer

We opted for real-time object detection so we used CCTV to get video footage. Video footage is sent to Monitoring system via IP address protocol. SOPs Detection are made in real time.

4.4.2. Processing Layer

All the work between Data layer and Presentation layer is performed by Processing layer. This layer includes image and video processing techniques necessary to make predictions.

4.4.3. Presentation Layer

The final System architecture layer in which we'll the results which is basically the output of processing layer. For this system, the final layer will represent the violation detections of respective SOPs.

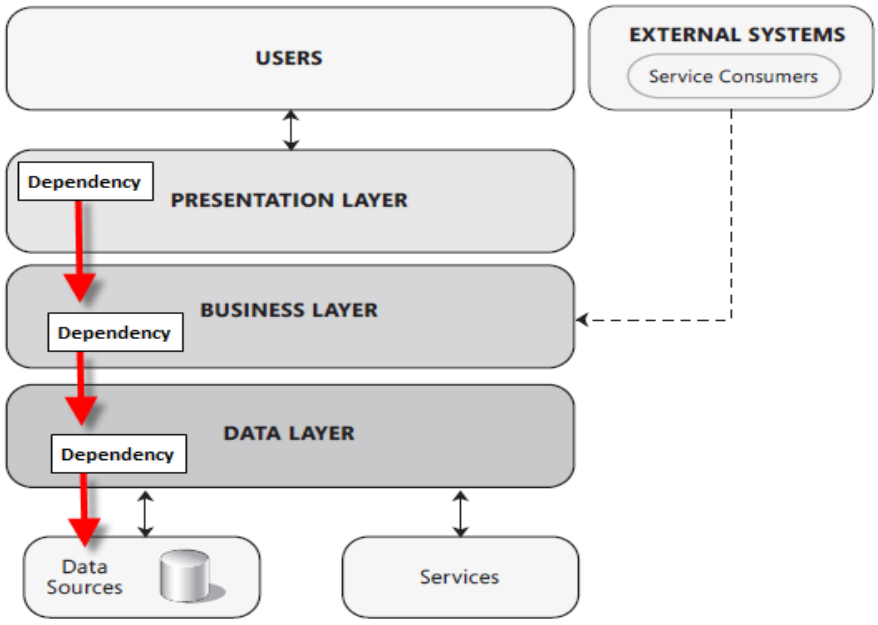

Figure 4-3 System Layers

4.5. Logical Design

4.5.1. Class Diagram

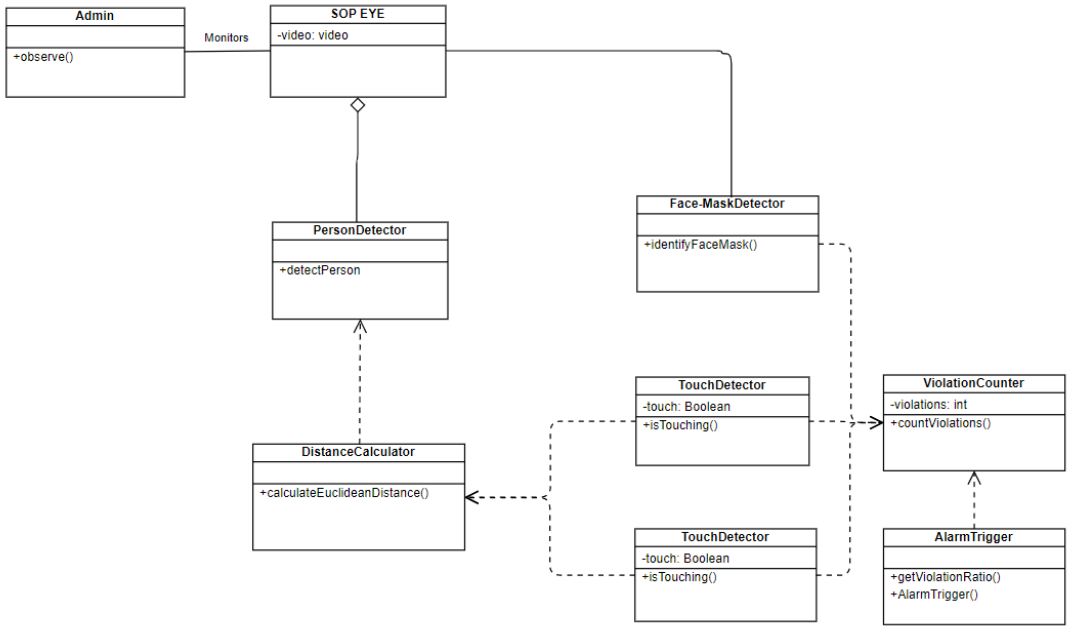

Figure 4-4: Class Diagram

4.6. Dynamic View

4.6.1. Activity Diagram

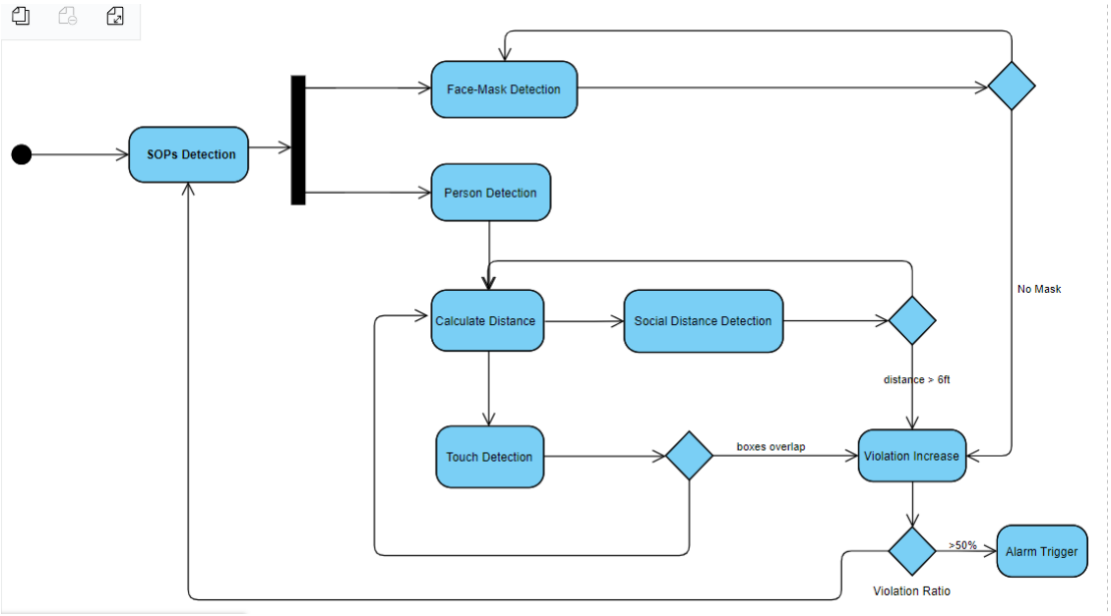

Figure 4-5 Activity Diagram

4.7. Component Design

4.7.1. Component Diagram

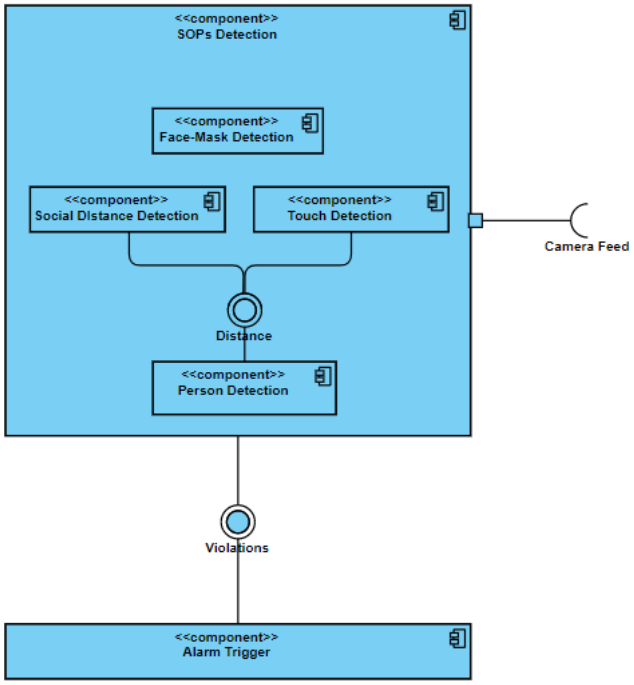

Figure 4-6 Component Diagram

4.7.2. Deployment Diagram

Figure 4-7 Deployment Diagram

4.8. Data Models

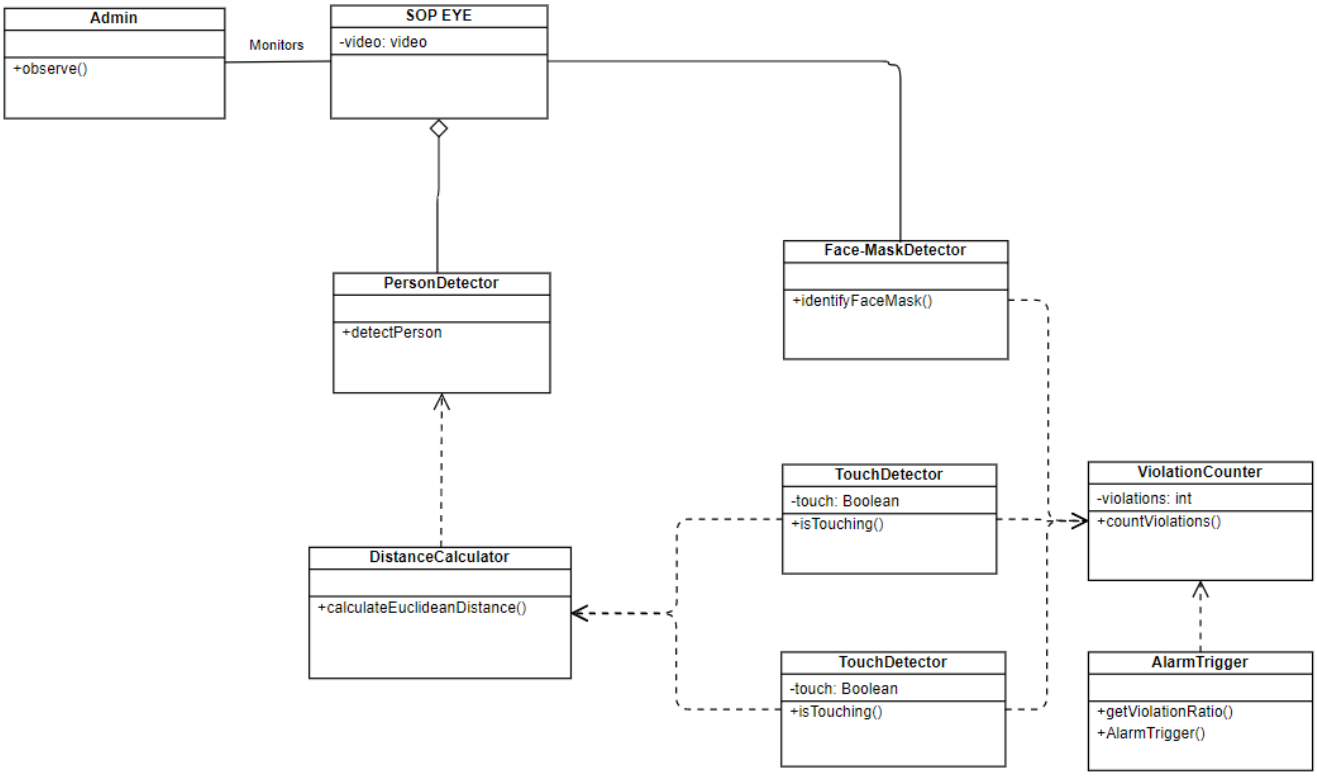

Figure 4-8 Data Models

4.9. User Interface Design

4.9.1. Animation Screen

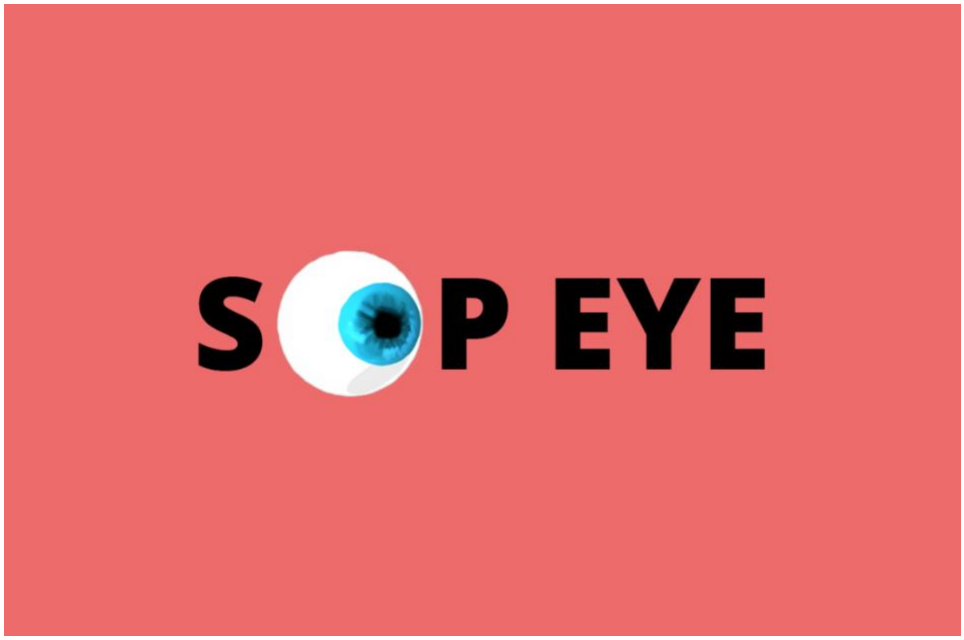

Figure 4-9 Animation Screen UI

4.9.2. Face Mask Detection

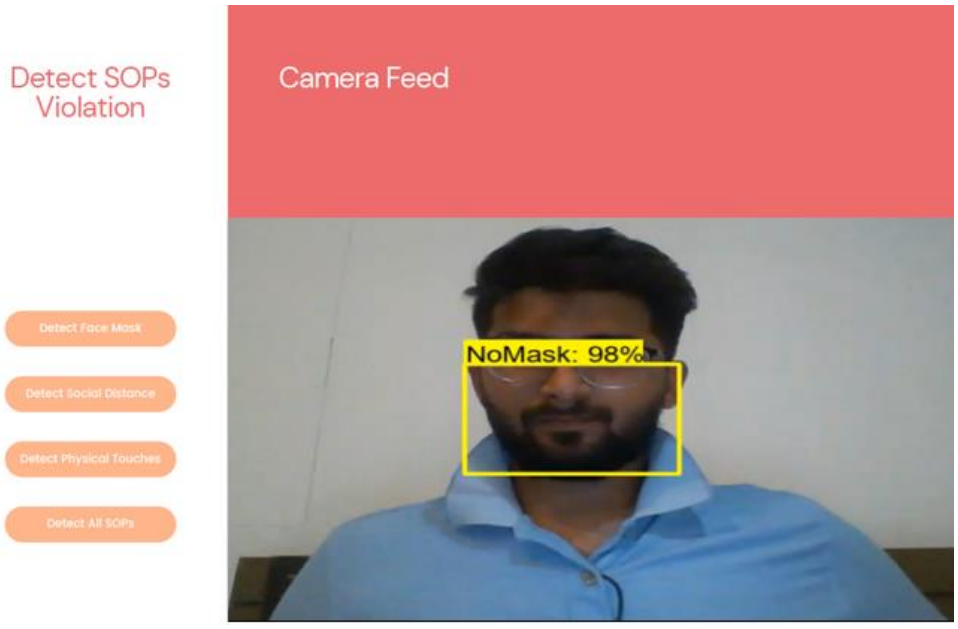

Figure 4-10 Face Mask Detection UI

4.9.3. Distance Detector

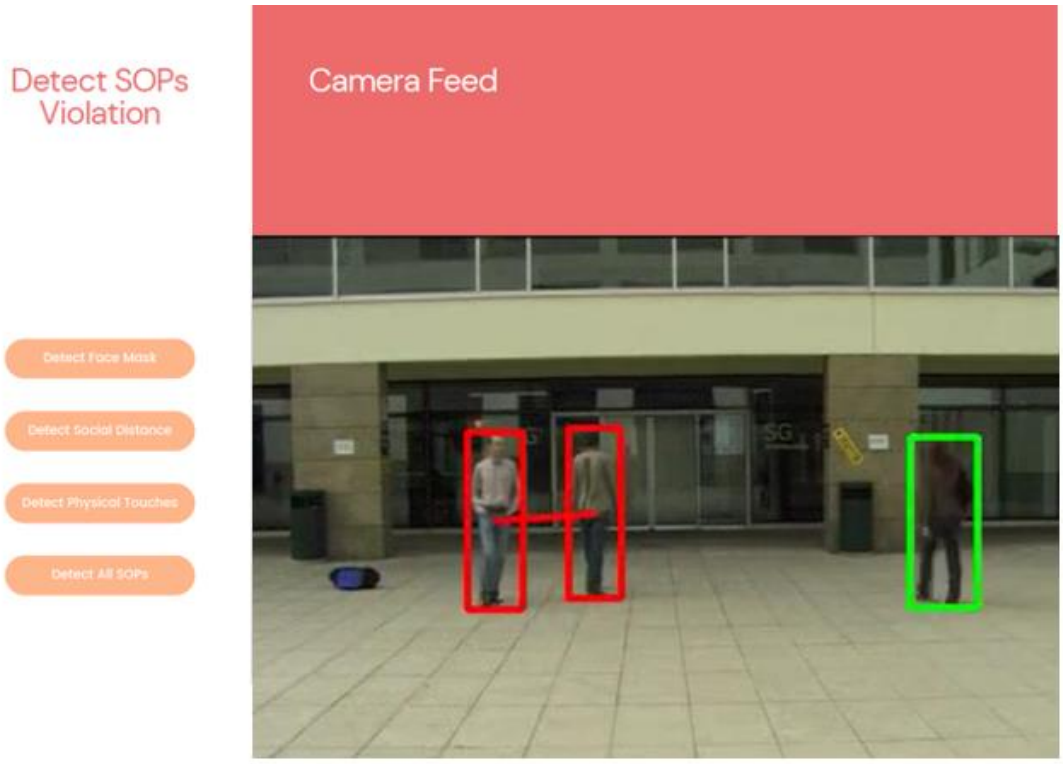

Figure 4-11 Distance Detector UI

4.9.4. Touch Detector

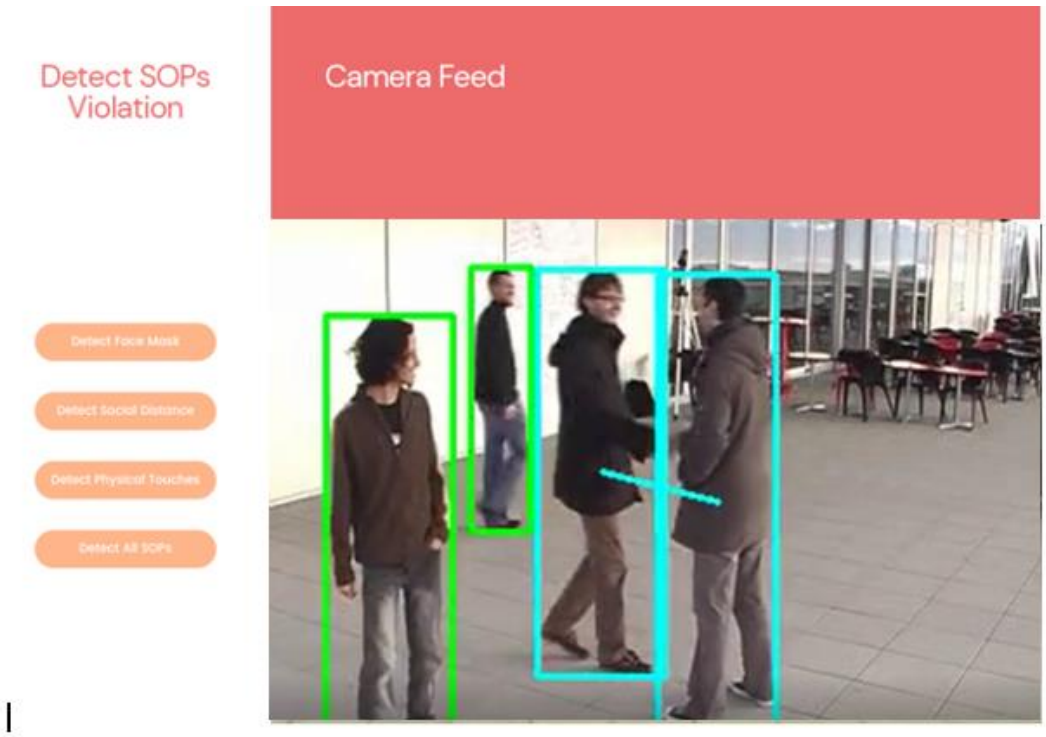

Figure 4-12 Touch Detector UI

4.10. System Prototype

4.10.1. Main Screen

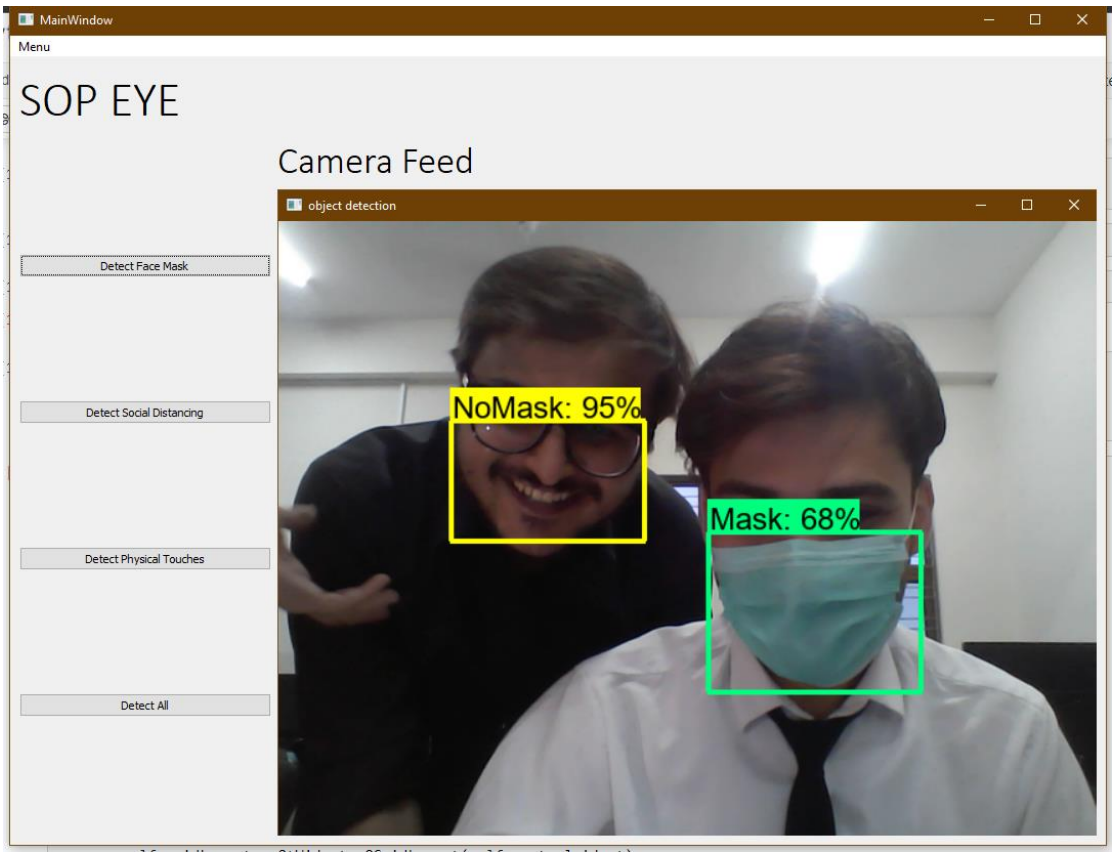

Figure 4-13 Prototype Screen 1

Figure 4-14 Prototype Screen 2

Figure 4-15 Prototype Screen 3

4.11. Conclusion

In this chapter we've discussed the system design approach and methodologies to implement the system. The design constraints and problems faced with design phase were mentioned and discussed. Logical Design, dynamic view and component view were demonstrated with the help of different UML Diagrams. System UI Design and prototype was also displayed in this chapter.

CHAPTER – 5

SYSTEM IMPLEMENTATION

Chapter 5

System Implementation

5.1. Tools

We've researched a lot and tried out most suitable methods and tools to deal with the issues and problems we've faced during implementation.

5.1.1. Jupyter Notebook

Jupyter Notebook is a web-based open-source software which provides interactive computing approaches across different programming languages. It was used to implement python code.

5.1.2. PyQt5

PyQt helps develop Python binding GUI across different platforms. It was used to design GUI for SOPs detection.

5.1.3. Anaconda

Anaconda framework is provided by Python. Generally, you've to manually provide path for your Python interpreter as it's required by most of **IDEs**. We used Anaconda to use packages like conda and virtual environments.

5.1.4. Python

Python is one of the most vastly used programming language. We used Python v 3.10.2. It is used to write all the necessary code for numerical computations, distance calculations and training, testing and detection of SOPs.

5.1.5. CUDA

CUDA is an open-source tool kit developed by NVIDIA. It's designed to enhance the performance of training models a whole heap faster and a lot easier.
To utilize GPU resources, CUDA 10.2 was used along with cudNN 8.4.1.

5.2. Models Used

We've tried multiple pretrained models and trained them on our custom objects in order to get acceptable results. Some of pretrained models are mentioned below that we used for Face-Mask Detection.

- SSD MobileNet V2 FPNLite 320x320
- Faster R-CNN ResNet152 V2 640x640
- Yolov5

We gained different COCO mAP and speed for different models.

Table 5.1 Models Speed and Accuracy Comparison

Model	Speed (ms)	COCO mAP
SSD MobileNet v2 FPN 320*320	22	22.2
Yolov5	38	56
FasterR-CNN ResNet152 V2 640x640	101	37.5

5.2.1. SSD MobileNet V2 FPN 320x320

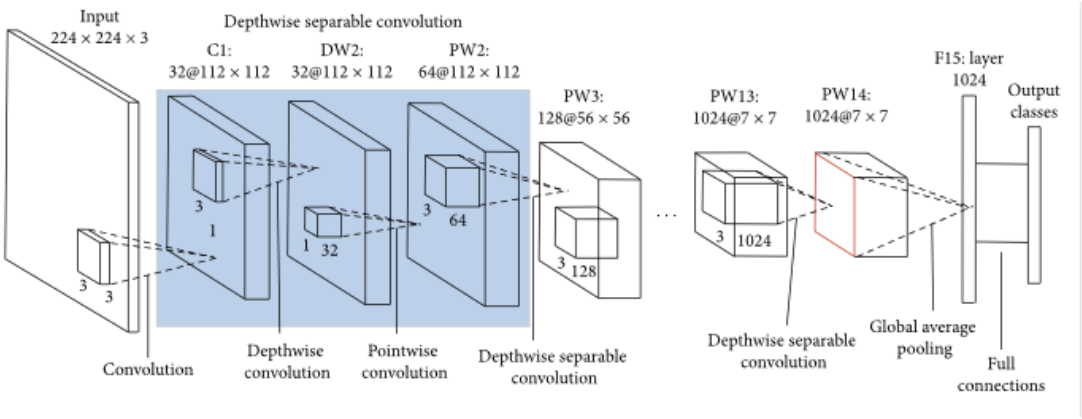

Figure 5-1 SSD MobileNet v2 Architecture

This is one-stage object detection model which means it has higher inference speed with relatively higher accuracy model. It is also compatible with devices with low computational power.

As the requirement was to make detections in real-time so it was best suited for detection of Face masks and it produced satisfactory results.

Table 5.2 Parameters of MobileNet V2

Parameters	**Values**
Learning Rate Base	0.079
Total Steps	50000
Warmup Learning Rate	0.026
Warmup Steps	1000
Batch Norm	
Decay	0.997
Scale	True
Epsilon	0.001

5.2.2. Faster R-CNN ResNet152 V2 640x640

Faster R-CNN is dual stage object detection model. It is Region-Based Convolution Neural Network whose region of interest is to produce accurate results using deep ConvNet. Earlier versions had disadvantages like slower object detection and its space expensive nature. However, they were improved in advance version named as Faster R-CNN, which was introduced in 2015 by Shaoqing Ren et al.

Basically, Faster R-CNN is composed of two modules,

- **Fast R-CNN**: In Fast R-CNN the whole image and passed it to several convolutional and max pooling layers which generates a conv feature map.
- **RPN network**: The feature map is passed to small RPN network to produce region proposals. This method introduces a concept of pre-defined anchor boxes of k sizes. This method speeds up the process of detection, and it produces two outputs K number of Bounding box and information regarding presence of object in bounding box

Later ROI layer fixes reshaping of bounding boxes then will finally use fully connected layer to classify objects.

This model was also used to make face-mask detections. It produced more accurate results as compared to SSD MobileNet but it was much slower in making real time detections.

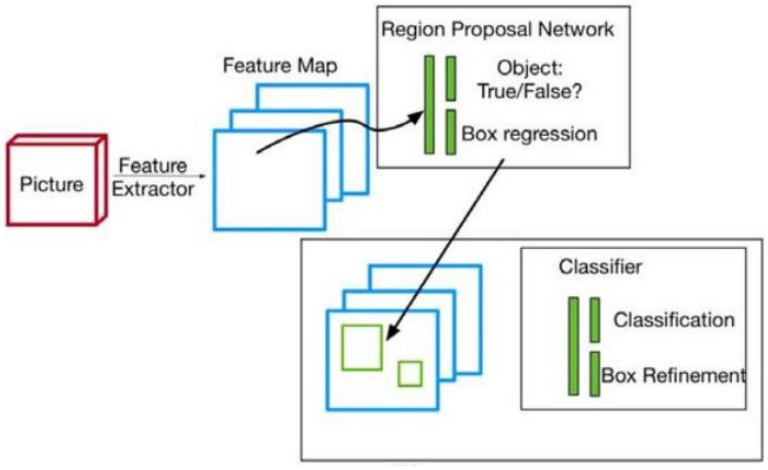

Figure 5-2 Faster R-CNN Architecture

5.2.3. Yolo

5.2.3.1. Architecture

Yolo architecture consists of minimum twenty-four convolutional layers followed by two fully connected layers. It uses reduction layers of 1 × 1 along with 3 × 3 convolutional layers. Fast YOLO version uses a neural network and consists of nine convolutional layers. Except the size of the network, all the other training and testing parameters remain same between both versions.

5.2.3.2. Yolov5

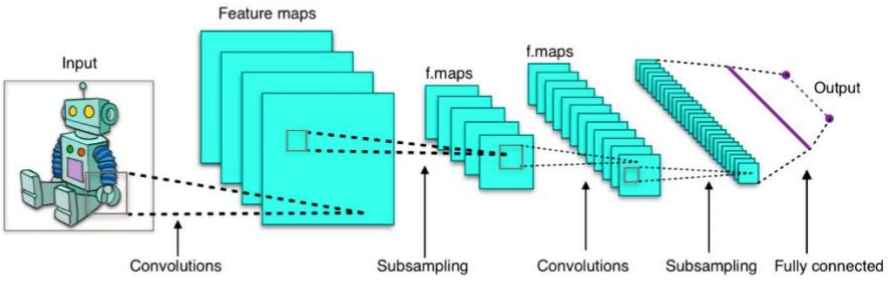

Figure 5-3 Yolov5 Architecture

Latest version of Yolo that is YOLOv5 was released by Glenn Jocher in 2020. YOLOv5 is written in the Pytorch framework. It is lightweight and easy to use.

As compared to previous versions of Yolo, Yolov5 provided auto anchoring step which allows training on custom dataset. The code will automatically look into anchors and start comparing it with the data if they fall below threshold, then the algorithm will start training new anchors automatically. It will use a k-mean with and some initial guesses and generate new anchors using genetic algorithm. Later it will automatically place these new anchors in the training model.

The decrease in inference time of the YOLOv5 models because of PyTorch framework. As it allows to half the floating-point precision in training and inference from 32-bit to 16-bit precision. YOLOv5 creates **.yaml** format file for model configuration this file specifies the different layers of network and then it multiplies those by the number of layers in the block.

We've used Yolov5x for calculating and solving social distancing and touching between two persons.

Table 5.3 Yolov5 Parameters

Parameters	Values
Learning Rate	0.01
Weight Rate Decay	0.0005
Momentum	0. 937
Batch Size	1
Number of Iterations	300
Number of Anchors	3
Anchor Size	[149,82, 160,104, 214,78] [207,105, 196,130, 265,110] [244,142, 338,132, 324,178]
IoU Training Threshold	0.20
Final One Cycle Learning Rate	0.2

5.3. Libraries Used

5.3.1. Pip

It is one of the most immensely used packages installer powered by Python. It usually comes built-in with Python Binary Installers. It enables you to effortlessly install and assemble packages.

5.3.2. Numpy

Numpy is free and easily available extension module used for large scientific computing which uses Python programming language. It is suitable for implementing multi-dimensional arrays and matrices to operate on these arrays.

5.3.3. Conda

It is a management tool to keep and manage Anaconda Python Installations. Conda tool is totally different from pip. It helps in managing virtualenv and deployment of binary extensions.

5.4. Other Tools

5.4.1. TensorFlow v2

TensorFlow is an open-source library used for implementing wide range of machine learning and AI tasks. TensorFlow provides an interface for interactive ML calculations, and a usage for executing such calculations.

TensorFlow library has been used to implement face mask detection module.

5.4.2. PyTorch

PyTorch is free and open-source framework for machine learning project which is provided by Torch library.

Yolov5 works perfectly with PyTorch framework so that's why we implemented remaining two modules with PyTorch framework.

5.4.3. OpenCV

OpenCV provides a wide range of functions to develop real-time computer vision applications, so we've used OpenCV library to get the live camera feed in order to make detections on them.

5.5. Problems faced and Solutions

5.5.1. Issue while detecting social distance violations in 2D:

While getting distance with Yolo bounding boxes there were chances that objects might get behind one another through certain angles and might not properly detect if the SOP violation happened or not. In order to solve the issue, we implemented bird eye's view.

Bird Eye View:

To handle above mentioned issue, we implemented the concept of **Bird Eye View**. It is a built in tool with OpenCV which allows users to add perspective view to an area. By selecting an ROI, the user can easily manipulate the ROI and apply perspective view. This gets the job done by taking the top-down view of certain scene, however it has some disadvantages, we've to statically select the ROI each time wherever such situation arises.

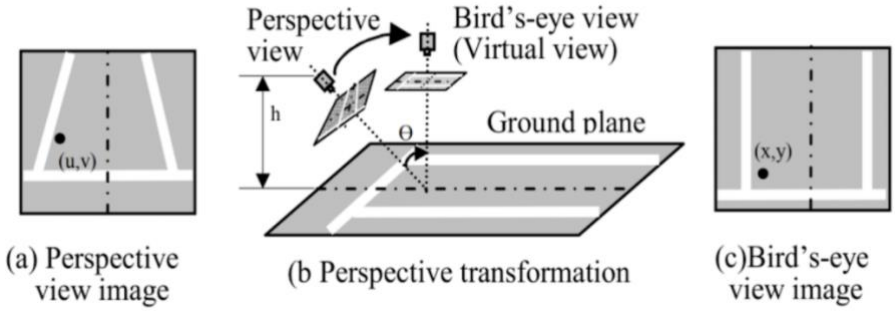

Figure 5-4 Birds Eye View

Figure 5-4 Birds Eye View shows the proposed methodology to implement bird's eye view using OpenCV's perspective functionality.

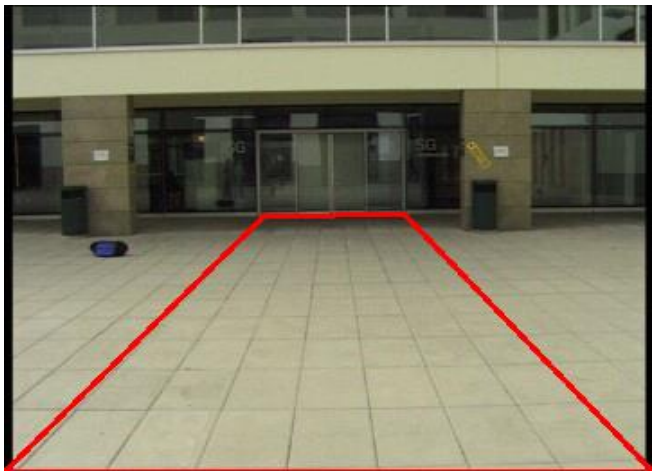

Figure 5-5: Example of ROI

Figure 5-5 Shows how the marked ROI space would look like in the video frame.

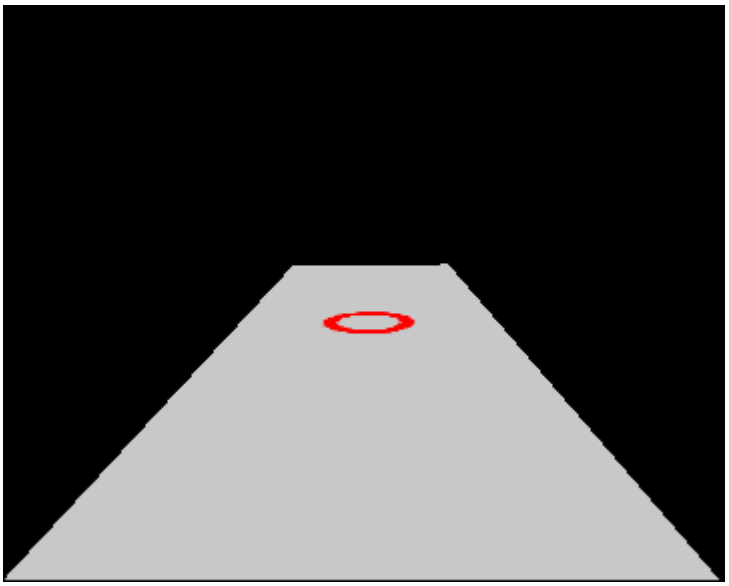

Figure 5-6 Applying perspective on ROI

Figure 5-6 shows how the marked ROI would look like in virtual space and how an object would look like in perspective of a birds eye view.

Figure 5-7 Example of Bird eye view

Figure 5-7 shows how the birds eye view would look like in real time with green box meaning safe distance (more than 6ft) and red boxes meaning not safe distance (less than 6ft)

5.6. Conclusion

This chapter discussed all necessary tools, libraries, models, methods and techniques applied to overcome the problems. Different models were discussed in detail and their comparison was made; thus, reasons were given for using any particular model.
Different problems that were faced were discussed and their proposed solution was also debated in detail.

CHAPTER – 6

SYSTEM TESTING & EVALUATION

Chapter 6

System Testing & Evaluation

Necessary measures were taken and system was test against all possible combinations of input and output. To ensure that system performs as it should under given circumstances, we've performed different level of testing, necessary for development side, including:

- Unit testing
- Integration testing
- System testing.

In following chapter, we'll discuss testing practises which we used to measure the system's credibility and test-cases that we designed to evaluate system performance.

6.1. Test Strategy

Our project test strategy ran from following three phases:

- Experiment phase
- Development phase
- Production phase

The **Experiment phase:** It is the core of a Machine Learning Project development as data science process is very research centric. Different algorithms and models have been tried throughout the experiment phase until they reach a satisfied result.

The **Development phase:** In this phase the finalised model from experiment phase has been used for production usage. After that unit, differential and integration tests are performed to evaluate the performance of the model as found in the experiment phase.

The **Production phase**: The final stage in which the system has been used to make predictions against real-time data. The focus of this testing phase is to evaluate the performance and accuracy of the system.

6.2. Unit Testing

First, we implemented a single module, tested against each possible scenario and then moved to the next module.

All the three modules were unit tested:

- Face-Mask Detection
- Social Distance Detection (**6ft**)
- Touch Detection

6.3. Integrated Testing

Each module that was tested individually was then integrated and tested again whether it gives the expected output for given data.

6.4. System Testing

After implementing the whole system, the system was again tested for all possible set of data and was matched against expected result and performance of system was evaluated.

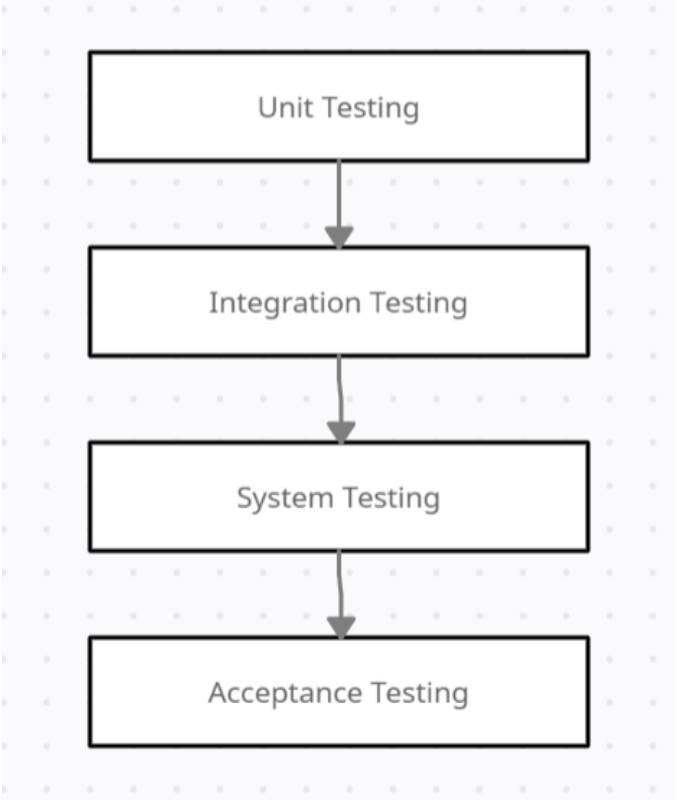

Figure 6-1 System Testing

6.5. Test Cases

Table 6.1 Test Case for Running Application Successfully

TC ID	**TC01**
Description	Running Application
Initial State	Idle
Input	Double click on app icon
Expected Output	Application should launch successfully
Output	Application launched successfully
Status	Pass

Table 6.2 Test Case for Making No Face-Mask Detection Successfully

TC ID	**TC02**
Description	Face-Mask Detection
Initial State	Application running
Input	Person Without face-mask (from live camera feed)
Expected Output	NoMask label must be shown
Output	System shows NoMask label
Status	Pass

Table 6.3 Test Case for Making Face-Mask Detection Successfully

TC ID	**TC03**
Description	Face-Mask Detection
Initial State	Application running
Input	Person wearing face-mask (from live camera feed)
Expected Output	Mask label must be shown
Output	System shows Mask label
Status	Pass

Table 6.4 Test Case for Making Face-Mask Detection Successfully

TC ID	**TC04**
Description	Face-Mask Detection
Initial State	Application running
Input	Person has covered faced with something other than face-mask (from live camera feed)
Expected Output	Mask label must be shown
Output	System shows Mask label
Status	Pass

Table 6.5 Test Case for Making Social Distance Detection Successfully

TC ID	**TC05**
Description	Social Distance Detection
Initial State	Application running
Input	Distance between persons is less than 6ft (from live camera feed)
Expected Output	Bounding Boxes must turn blue
Output	Bounding boxes turned blue
Status	Pass

Table 6.6 Test Case for Detecting no Social Distance Violation Successfully

TC ID	**TC06**
Description	Social Distance Detection
Initial State	Application running
Input	People are maintaining 6ft distance (from live camera feed)
Expected Output	Bounding Box shouldn’t change colour and stay green
Output	Bounding Boxes stayed Green
Status	Pass

Table 6.7 Test Case for Making Touch Detection Successfully

TC ID	**TC07**
Description	Touch Detection
Initial State	Application running
Input	People make physical contact (from live camera feed)
Expected Output	Bounding box must turn red
Output	Bounding box turned red
Status	Pass

Table 6.8 Test Case for Detecting no Touch Violation Successfully

TC ID	**TC08**
Description	All SOPs Detection (Integrated)
Initial State	Application running
Input	People make SOP violation (Live camera feed or CCTV)
Expected Output	Agent must show and highlight each violation
Output	All violations detected successfully
Status	Pass

Table 6.9 Test Case for Triggering Alarm Successfully

TC ID	**TC09**
Description	Alarm Trigger on Excessive violations
Initial State	Application running
Input	More than 40% people are violation SOPs (from live camera feed)
Expected Output	Alarm should be triggered
Output	Alarm Triggered
Status	Pass

6.6. Results & Evaluation

To evaluate the system performance, we've put system under different tests such as providing high- and low-quality footage to check whether it produces accurate results. System was put under stress by providing footages with excessive violations. The system was able to do multiple detections at the same time. Following are the results of the detections made by SOP Eye.

6.7. Conclusion

This chapter concludes that the system is now verified and is validated through different testing techniques and is running smoothly. In this chapter we have performed complete testing of our system starting from unit testing and then moving on to the integration testing to check that modules are working collectively and in the end by performing the system testing. Our system altogether is working perfectly.

CHAPTER – 7

CONCLUSION

Chapter 7

Conclusion

To conclude we've made a lot of efforts and spent a lot of time in research in order to successfully complete this project. By supervision and support of our supervisor we completed and submitted every deliverable of this project before given deadline. We've learnt a whole lot of new methods, tools, technologies and problem-solving techniques which will help us in future

7.1. Contributions

COVID-19 hit and turned the world in global crisis as it affected everyone single person in terms of jobs, health and other outdoor activities. The main purpose for designing this machine learning product was to contribute in this pandemic by automating the detection of COVID-19 SOPs Violation.

It has capability to alarm the people automatically, if number of violations goes above set threshold/ratio.

It'll help security staff to deal with SOPs regularity in indoor areas. It'll eliminate the need of some security individual to observe SOPs obedience by detecting:

- Face Mask
- Social Distancing
- Touching

Of course, there is a lot of work already done in this domain and there are a lot published research papers but we've integrated all the SOPs detection at one place and given a final product for SOPs detection which is never done before.

To achieve Quality is the most important trait in software project lifecycle and software project itself. This quality is achieved by meeting all the requirements and needs of market and customer. We've tried to gain maximum level accuracy and quality of the product i.e., by making accurate detections of SOPs in real-time.

7.2. Reflections

SOP Eye is able to run on multiple Operation Systems. It's influence on society will result effectively in SOPs obedience as it'll automate SOPs violation detection. It is effective in a manner that it can make several detections at the same time that a naked eye couldn't possibly do. This project has enabled us to learn new technologies, methods and problem-solving techniques. It improved our skills in Machine Learning and Artificial Intelligence domain.

7.3. Future work

As we know COVID-19 virus isn't completely eradicated and there are some sectors where SOPs are still followed and taken care of, such as: medical centres. So, it is still useable in hospitals.

This system can be advanced by training it more and more with large dataset and new models to get better accuracy and speed.

Our product is completely extensible and flexible to new technologies and models and it also has capability to integrate more SOPs. SOP EYE was designed to monitor COVID-19 SOPs but it is not static. It could be modified, if God-forbid, any other pandemic occurs.

Using bigger model require more computation power and CUDA memory for example: Large Yolov5I and XLarge Yolov5x.

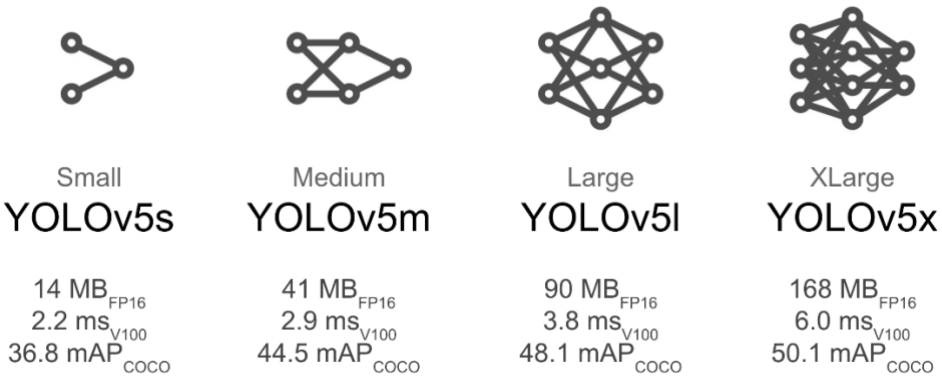

Figure 7-1 Yolov5 models

If we've finitely large dataset and enough Video memory we can train models better and get more accurate results.

REFERENCES

[1] T. Ikram, A. Saeed, N. Ayn, M. A. Tahir and R. Mumtaz, "A review of the prevalent ICT techniques used for COVID-19 SOP violation detection," 2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET), 2020, pp. 194-198, doi: 10.1109/HONET50430.2020.9322821.

[2] X. Wang et al., "A Weakly-Supervised Framework for COVID-19 Classification and Lesion Localization From Chest CT," in IEEE Transactions on Medical Imaging, vol. 39, no. 8, pp. 2615-2625, Aug. 2020, doi: 10.1109/TMI.2020.2995965.

[3] A. A. R. Alsaeedy and E. K. P. Chong, "Detecting Regions At Risk for Spreading COVID-19 Using Existing Cellular Wireless Network Functionalities," in IEEE Open Journal of Engineering in Medicine and Biology, vol. 1, pp. 187-189, 2020, doi: 10.1109/OJEMB.2020.3002447.

[4] S. Zhang and X. Wang, "Human detection and object tracking based on Histograms of Oriented Gradients," 2013 Ninth International Conference on Natural Computation (ICNC), 2013, pp. 1349-1353, doi: 10.1109/ICNC.2013.6818189.

[5] Hong Han, Youjian Fan and Zhichao Chen, "Human detection based on Curvelet transform," 2011 International Conference on Multimedia Technology, 2011, pp. 356-359, doi: 10.1109/ICMT.2011.6003080.

[6] G. Deore, R. Bodhula, V. Udpikar and V. More, "Study of masked face detection approach in video analytics," 2016 Conference on Advances in Signal Processing (CASP), 2016, pp. 196-200, doi: 10.1109/CASP.2016.7746164.

[7] W. Bu, J. Xiao, C. Zhou, M. Yang and C. Peng, "A cascade framework for masked face detection," 2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2017, pp. 458-462, doi: 10.1109/ICCIS.2017.8274819.

[8] Punn, Narinder Singh, et al. "Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques." *arXiv preprint arXiv:2005.01385* (2020).

[9] Rasheed R, Rizwan A, Javed H, Sharif F, Zaidi A. Socio-economic and environmental impacts of COVID-19 pandemic in Pakistan-an integrated analysis. Environ Sci Pollut Res Int. 2021 Apr;28(16):19926-19943. doi: 10.1007/s11356-020-12070-7. Epub 2021 Jan 6. PMID: 33410007; PMCID: PMC7787403.

APPENDIX A

Abbreviations	Meaning
SOPs	Standard Operating Procedures
AI	Artificial Intelligence
ML	Machine Learning
OS	Operating System
GUI	Graphical User Interface
UI	User Interface
WHO	World Health Organization
YOLO	You Only Look Once
R-CNN	Region-based Convolutional Neural Network