Computer Systems Science & Engineering
DOI: 10.32604/csse.2022.024149
Article

Tech Science Press

Primary Contacts Identification for COVID-19 Carriers from Surveillance Videos

R. Haripriya[*] and G. Kousalya

Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore, 641014, India
*Corresponding Author: R. Haripriya. Email: priyarr@gmail.com
Received: 06 October 2021; Accepted: 30 November 2021

Abstract: COVID-19 (Coronavirus disease of 2019) is caused by SARS-CoV2 (Severe Acute Respiratory Syndrome Coronavirus 2) and it was first diagnosed in December 2019 in China. As of 25th Aug 2021, there are 165 million confirmed COVID-19 positive cases and 4.4 million deaths globally. As of today, though there are approved COVID-19 vaccine candidates only 4 billion doses have been administered. Until 100% of the population is safe, no one is safe. Even though these vaccines can provide protection against getting seriously ill and dying from the disease, it does not provide 100% protection from getting infected and passing it on to others. The more the virus spreads; it has more opportunity to mutate. So, it is mandatory to follow all precautions like maintaining social distance, wearing mask, washing hands frequently irrespective of whether a person is vaccinated or not. To prevent spread of the virus, contact tracing based on social distance also becomes equally important. The work proposes a solution that can help with contact tracing/identification, knowing the infected persons recent travel history (even within the city) for few days before being assessed positive. While the person would be able to give the known contacts with whom he/she has interacted with, he/she will not be aware of who all were in proximity if he/she had been in public places. The proposed solution is to get the CCTV (Closed-Circuit Television) video clips from those public places for the specific date and time and identify the people who were in proximity—i.e., not followed the safe distance to the infected person. The approach uses YOLO V3 (You Only Look Once) which uses darknet framework for people detection. Once the infected person is located from the video frames, the distance from that person to the other people in the frame is found, to check if there is a violation of social distance guideline. If there is, then the people violating the distance are extracted and identified using Facial detection and recognition algorithms. Two different solutions for Face detection and Recognition are implemented and results compared—Dlib based models and OpenCV (Open Source Computer Vision Library) based models. The solutions were studied for two different CCTV footages and the results for Dlib based models were better than OpenCV based models for the studied videos.

Keywords: COVID-19; social distancing; object detection; facial detection; facial recognition; contact tracing; artificial intelligence

1 Introduction

Coronavirus disease is an infectious disease caused by a newly discovered coronavirus. The virus spreads primarily through droplets of saliva or discharge from the nose when an infected person coughs or sneezes. Some of the methods suggested by WHO (World Health Organization) to reduce the spread of infection include practicing Social Distancing, wearing masks, avoiding crowded areas, frequent cleaning of hands and staying at home, going out only for important or essential needs. As per WHO guidelines, it is suggested to maintain at least 1 meter distance from others while being outdoors and greater distance when indoors, to prevent the spread of infection. Given that infected people may be asymptomatic as well, it becomes particularly important to track the positive cases and quarantine them to prevent spread. Also, so far, though vaccines have started becoming available, they are still not readily available to larger population. With a fast-moving pandemic, no one is safe, unless everyone is safe [1]. Hence it becomes important to focus more on building immunity and preventing further spread by taking precautionary measures suggested by WHO. Given the importance of social distancing, most of the countries had taken decision to implement lockdown to prevent the outbreak. Travel restrictions were implemented, educational institutions and most of the industries excluding essential services were closed. Due to this there was an impact to the economy. After a prescribed period of lockdown, the countries gradually started relaxing the restrictions and letting industries open with minimal staff with enough social distance etc. Even during this stage social distance was perceived as a key factor to be adhered to. Even though the guideline has been set in place, people missed out to follow the social distance in public spaces and this had led to further spread of infection and wave 2 in many countries. Hence once a person is tested COVID-19 positive, it becomes very important to do contact tracing and test and quarantine all primary contacts as well. This is difficult if the person has been to public places and doesn't know who all have violated the safe distance from the infected person. This is the reason why Indian Government had released mobile applications like Arogya Setu which helps track proximity to infected person through Bluetooth and GPS (Global Positioning System). In this research work, couple of AI based solutions are proposed to do the contact tracing using YOLO V3 (You Only Look Once) for Object detection and two methods for Facial detection and Facial recognition algorithms which are dlib based models and OpenCV (Open Source Computer Vision Library) based models. The results of both the methods are compared with and without image preprocessing and with and without some additional training. This solution can be used on any CCTV (Closed Circuit Television) footages which are readily available in many public places to identify the person who tested COVID-19 positive, and their primary contacts based on the prescribed social distance. To evaluate the effectiveness of the solutions the accuracy, precision and recall parameters are arrived at and compared for Object detection, Facial detection, Facial recognition as well as calculation of Social Distance between people.

2 Background Study and Related Work

Many countries including India and South Korea are utilizing GPS to track the movements of the suspected or infected persons to monitor any possibility of their exposure among healthy people. In India, the government is using the Arogya Setu App, which worked with the help of GPS and Bluetooth to locate the presence of COVID-19 patients in the vicinity area. On the other hand, some law enforcement departments have been using drones and other surveillance cameras to detect mass gatherings of people and taking regulatory actions to disperse the crowd. Such manual intervention in these critical situations might help flatten the curve, but it also brings a unique set of threats to the public and is challenging to the workforce.

As a technology advancement, computer vision and deep learning are being looked at, to create solutions for identifying social distancing violations, violations in wearing masks etc. The first set of references discussed here are the works studied for object detection.

In the work of authors Kajabad et al. [2], the application of different deep learning algorithms like YOLO, SSD (Single-Shot Detection), R-CNN (Region Based Convolutional Neural Networks) for object detection were studied. As per this study, the YOLO performed faster than R-CNN and Faster R-CNN. This is especially true for running in real-time applications with a capable GPU (Graphics processing unit). Although it is not as fast as SSD, YOLOv3 algorithm has higher accuracy in people identification. Background subtraction was done with Gaussian Mixture algorithms and heatmap color technique to identify dense areas in the picture. The experimental results have shown that the accuracy and the performance of both algorithms are quite good.

A Comparative Study of Custom Object Detection Algorithms blog by Intellica.AI [3] discussed the performance of various object detection algorithms. According to the study R-CNN, Fast RCNN were developed initially but were much slower. Faster RCNN was developed to improve the speed and was the best of the three algorithms on COCO (Common Objects in Context) dataset. But all the above algorithms have some demerits like not looking at the complete image and also using only the sections of image which have higher probability of containing the object to locate the object. Due to this they were still slow. SSD outperformed the above three algorithms in terms of speed but compromised on the accuracy. The latest version i.e., the V3 of YOLO outperformed the Faster R-CNN and SSD in terms of accuracy and inference time on the benchmark dataset. Since the number of layers for YOLO V3 are more than that of YOLO V2, it is bit slower than YOLO V2, though it is better than YOLO V2 in most aspects. In the blog on YOLO v3 by Li [4], he has discussed that the key performance improvement is achieved by its ability to detect multi-scale objects, better feature extraction capability and the loss function. Based on the above background study on object detection models YOLO V3 is chosen for detecting people from the video frames for this work.

Works related to social distancing were studied and listed here. Prem et al. [5] study the effects of social distancing measures on the spread of the COVID-19 epidemic and the effects of sudden lifting of lockdown. Authors used synthetic location-specific contact patterns to simulate the ongoing trajectory of the outbreak using SEIR (susceptible-exposed-infected-removed) models. It was also suggested that premature and sudden lifting of social distancing could lead to an earlier secondary peak, which could be flattened by relaxing the interventions gradually. Adolph et al. [6] highlighted the situation of the United States of America, where due to lack of common consent among all policymakers' lockdowns could not be adopted at an early stage, which resulted in harm to public health. Although social distancing impacted economic productivity, many researchers are trying hard to overcome the loss.

Punn et al. [7] have suggested a solution to monitor social distancing with the help YOLO V3 object detection and other Deep sort techniques. The solution was a real-time deep learning-based framework for automating the monitoring of social distancing using surveillance videos. The work computes the number of violations. Various state-of-the-art object detection models like Faster RCNN, SSD, and YOLO v3 were compared, out of which YOLO v3 illustrated the most efficient performance. Ahmed et al. [8] proposed an overhead perspective to monitor social distance in surveillance videos. Pre-trained YOLOV3 was used, and transfer learning was done to additionally train the model for overhead perspective. Euclidean distance between the pairwise centroids (of the bounding boxes) was used to monitor the social distance violations. For the distance calculation, an approximation of physical distance to the pixel was used and for the social distance, threshold was defined. In the solution proposed for this work as well, an approximation of physical distance to pixel is used. The transfer learning helped increase the accuracy of the model. Saponara et al. [9] have proposed an intelligent surveillance system based on thermal images for social distance measurement. YOLO V2 was used for detection and tracking of people. Thermal images are used to monitor the temperature of the person. The distance between the pixels is translated into a metric distance, after knowing the range and field of view covered by the camera and then compared with the threshold. Since CCTV cameras provide angle views on the ground, homograph transformation has been used to get the corresponding points on the ground to calculate the correct distance. The works discussed so far focus on live tracking of the social distance from surveillance videos. In the next set some extensive works done in the field of Facial detection and recognition are discussed next.

Some of the blogs by Geitgey [10] and Rosebrock [11] were referred to understand more about other models available for face detection, pre-processing and recognition. Few of the face detection models available are Haar Cascade Face Detector in OpenCV, Deep Learning based Face Detector in OpenCV, HOG (Histogram of Oriented Gradients) face Detector in Dlib, Deep Learning based Face Detector in Dlib. Haar Cascade face detector was developed by Viola and Jones and was the best for many years [12]. It is capable of detecting at multiscale and has a simple architecture. But some of the major drawbacks are, it doesn't work on non-frontal images, doesn't work under occlusion and has lot of false predictions. This was tried for this work for few of the CCTV videos and there were large number of false predictions and hence was not chosen. DNN (Deep Neural Network) is the deep learning-based face detector in OpenCV. It is based on Resnet-10 architecture. This is the most accurate out of the four methods, works well for different face orientations, works under substantial occlusion and capable of multi-scale detection. HOG Face detector is based on HOG features and SVM (Support Vector Machine). This is the fastest on CPU (Central processing unit), works well for frontal and slightly non-frontal faces, works under small occlusion. It doesn't work well for smaller faces. However, it can be trained with smaller faces. CNN (Convolutional Neural Networks) Face detector in DLIB uses a MMOD (Maximum-Margin Object Detector) with CNN based features. Though it works for different face orientations and fast on GPU, it works very slow on CPU. Also, this was tried out for one of the CCTV videos and it was not able to detect the faces—maybe due to the reason that the resolution was quite low. For the same reason this face detector was not chosen for this work. Based on the above study, DNN face detector in Open CV and HOG face detector in Dlib were chosen for this work.

Face recognition algorithms help in identification or verification of a person. Google's Facenet and Facebook's Deepface are some of the best performing algorithms [13]. But both of them are not open source. OpenBR (Open Source Biometric Recognition) is another option but licensed as per Apache2.0 and implements 4SF2 (Spectrally Sampled Structural Subspace Features 2) for face recognition. OpenFace is one the best performing open-source algorithm for the Facial Recognition. It is based on Python and Torch implementation of face recognition with deep learning. It can be run on CPUs and the model can be trained with high accuracy with little data. Schroff et al. [14] proposed Facenet as a solution for face recognition/verification. The method proposes to directly learn from the facial embedding into Euclidean space for face verification. The key differentiator from other previously proposed CNN based solutions is that the suggested end to end triplet loss-based training simplifies the setup and improves the performance. The same method of training is used by the proposed model for face recognition in this work. Amos et al. [15] proposed OpenFace which is a face recognition library, as the solution for Facial recognition. Facenet's loss function was used in the solution. The nn4. small2v1 Openface model was used. Affine transformation was used to pre-process the input image, in order to align the faces and the neural network. Openface model was used to generate the 128d facial embeddings. Linear SVM was used for classification/Facial recognition. This method was evaluated using the standard LFW (Labelled Faces in the Wild) dataset and compared with eigenfaces, fisherfaces, LBPH (Local Binary Pattern Histogram). Openface had the fastest training time and better accuracy among them. Adam Geitgey's blog helped to understand more about the Adam Geitgey's face recognition library built using Davis King's dlib. This model is able to perform with an accuracy of 99.38% on the standard LFW dataset. This library has lot of useful tools for face detection, pre-processing like aligning faces, finding the facial landmarks, finding the 128-d embeddings etc. Based on the above study it was decided to use both the Openface model and dlib based model for the chosen work and compare the results.

After obtaining the results, in order to understand if the performance of the proposed solution was good enough, William crumplers blog [16] was referred on How accurate are Facial recognition systems. According to this blog, when the Facial recognition on surveillance videos and crowded places were analyzed, it was found that the accuracy varies with a huge range (as low as 36% to 87%) depending on a lot of parameters. Whereas in a controlled setting the accuracy could increase to 94.4% NIST's (National Institute of Standards and Technology) 2017 FIVE (Face in Video Evaluation) validated the performance of Facial recognition algorithms on videos captured in settings like airport boarding gates and sports venues. The test found that when using footage of passengers entering through boarding gates —a relatively controlled setting—the best algorithm had an accuracy rate of 94.4%. Whereas when tested for the sporting venue, even leading algorithms had accuracies ranging between 36% and 87% only, depending on camera placement. For the application chosen for this work, since the environments may vary from closed spaces like shops, shopping malls, hotels to open spaces like open auditoriums or parking lots, the clarity of the videos will vary between medium to low resolution and very rarely will have higher resolution. With this limitation, the facial recognition accuracy would be limited. But even with this limitation, it would be much better than manual process, to identify as many contacts as possible from public places.

Though there are extensive works done in identifying social distance violations or in identifying whether people are wearing masks, to our knowledge this is the first effort to identify the primary contacts and recognizing their face (by measuring the social distance between people) from CCTV footage videos.

3 Proposed Methodology for Object Detection, Face Recognition and Social Distance Calculation

3.1 Object Detection

As discussed in previous section YOLO v3 is used for detecting the people from video frames for the proposed solution.

YOLO object detection algorithm was developed using the darknet framework. YOLO looks at the complete image only once and does the object detection and hence it is much faster. It works by splitting the entire image into multiple grids and extracting the feature from each grid. Both feature extraction and object detection are done multi-scale. The features extracted from the last three layers from each of the feature extractor are fed into different detection heads. Residual blocks are utilized for feature learning. These blocks have many convolutional layers and skip connections. An input image is first subject to feature extraction to extract the feature embeddings at multiple scale all at once. YOLO V3 uses Darknet-53 as feature extractor [17]. The extracted features are fed to corresponding detectors to do the object detection as well at different scales. These Key processing steps in YOLO v3 are summarized in the Fig. 1 below.

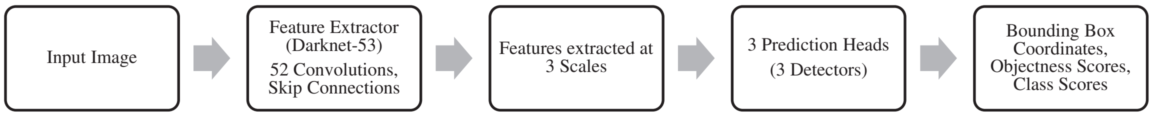
Input Image
Feature Extractor (Darknet-53)
52 Convolutions, Skip Connections
Features extracted at 3 Scales
3 Prediction Heads
(3 Detectors)
Bounding Box Coordinates, Objectness Scores, Class Scores

Figure 1: Key processing steps in YOLO v3

YOLO predicts multiple bounding boxes for each grid. But only one box per scale is needed. So, while training, this is identified by selecting the one which has highest Intersection Over Union (IoU) value with the ground truth box. [8] If BoxT is ground truth box and BoxP is predicted Box then IoU is calculated as in Eq. (1)

$$IoU(pred,\ actual) = area\frac{BoxT\ \ \cap\ BoxP}{BoxT\ \ \cup\ BoxP} \tag{1}$$

In YOLO v3, there are three bounding boxes for each grid cell, one for each scale. There would be 52 × 52 × 3, 26 × 26 × 3 and 13 × 13 × 3 boxes for each scale. Each box would have predictions for box coordinates, objectness score and the class probability. The coordinates contain predicted information for bounding box: x, y, w, h. The objectness score indicates if the box has an object or not. This will be 1 if an object is present. The class probabilities represent the probability that the object predicted is present in the box. This has class number value.

Mean squared error is used to calculate the error in each of the predictions. The loss function consists of Localization loss $\mathcal{L}_{loc}$ which includes both centroid (xy) loss & width-height (wh) loss, objectness loss and classification loss. Localization Loss is calculated using Eq. (2)

$$\mathcal{L}_{loc} = \lambda_{Co-ord} 1_{ij}^{obj} \sum_{i=0}^{s\wedge 2} \sum_{j=0}^{B} (x_i - x_i^*)\wedge 2 + (y_i - y_i^*)\wedge 2 + \left(\sqrt{w_i} - \sqrt{w_i^*}\right)^2 + \sqrt{h_i} - \sqrt{h_i^*})\wedge 2 \tag{2}$$

where i refers to the grid number and j is the bounding box number, λ_{Co-ord} is the weight of the coordinate error, S^2 is the number of grids in the image, B is the number of generated bounding boxes per grid 1_{ij}^{obj} will be 1 when the jth bounding box in grid i has an object. x_i, y_i, w_i, h_i are the positions of the predicted bounding box. $x_i^*,\ y_i^*,\ w_i^*,\ h_i^*$ are the actual positions of the bounding box of the ith grid.

The Objectness loss $\mathcal{L}_{Obj}$ has two terms as given by Eq. (3) given C_{ij}^*-the objectness score for jth bounding box in the ith grid cell, 1_{ij}^{obj}–which will be 1 when in the ith cell, the jth bounding box is responsible for object detection; otherwise, it is equal to 0, λ_{noobj} is a constant that is used to reduce the loss during background detection. The first term for the case when an object is detected and other for case the object not detected.

$$\mathcal{L}_{Obj} = \sum_{i=0}^{s\wedge 2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_{ij} - C_{ij}^*)\wedge 2 + \lambda_{noobj} \sum_{i=0}^{s\wedge 2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_{ij} - C_{ij}^*)\wedge 2 \tag{3}$$

Given $p_i^*(c)$ as the conditional class probabilities for class c in ith grid cell, the classification loss $\mathcal{L}_{cls}$ for each grid is computed as the squared error of the conditional class probabilities $p_i(c)$ as shown in Eq. (4)

$$\mathcal{L}_{cls} = \sum_{i=0}^{s\wedge 2} 1_{ij}^{obj} \sum_{c\,\in\,class} 1_i^{obj} (p_i(c) - p_i^*(c))\wedge 2 \tag{4}$$

Overall Loss is calculated as Eq. (5)

$$\mathcal{L} = \mathcal{L}_{loc} + \mathcal{L}_{Obj} + \mathcal{L}_{cls} \tag{5}$$

YOLO v3 model used has already been pre-trained on COCO dataset. The dataset had 80 different categories such as humans, cats, dogs, traffic lights, etc., and almost 300,000 images.

3.2 Face Detection and Recognition

Facial Detection helps in identifying all the faces from an image. This can be achieved by training the algorithm with different types of faces which may have millions of samples.

Facial Recognition algorithms on the other hand are used to identify individuals from photos or videos. These algorithms as any other Machine/deep learning algorithms need to be trained with a set of known images of the people who are to be identified. Once done, they would be able to identify the faces of the persons on which they were trained on. These algorithms work by passing an image through multiple layers to extract facial features like the position of eyes, nose, mouth, distance between these features etc. to create a feature vector. The extracted feature vector is compared with the trained dataset to identify the closest match. In general, using surveillance video images to locate/identify a person tend to have lower accuracies compared to the applications in which they are used for Facial verification systems, which have fixed higher resolution cameras which could capture high-quality images in the required angle and lighting. Whereas in the case of surveillance videos, the lighting and visibility of individuals moving freely through public spaces cannot be controlled.

First step in any face recognition pipeline is to locate the face in the image using facial detection algorithms. Next step is extracting the feature vectors. The final step is the facial recognition which is done by comparing the output from previous step with the feature vectors extracted by trained model to recognize the person with whom the vector closest matches with.

For the work two methods for face detection and recognition are chosen and the results compared from these two approaches. HOG face detector with Face recognition using dlib and DNN face detector in OpenCV with OpenFace using OpenCV.

3.2.1 Method 1-HOG Face Detector with Adam Geitgey's Face Recognition Using Dlib

HOG detector extracts the key features from the image by understanding the distribution of directions of gradients. Most of the information about the image is packed in the edges and corners rather than flat regions. So, these are used by the detector to extract useful information from the image. In the first step, the image is pre-processed. For face detection, each the pre-processing involves extracting the face by cropping the image and then resizing the image by modifying the aspect ratio as expected by the detector. In the work this is made easier by passing the box contents of each person. Next step is calculation of gradients—As mentioned earlier, the detector understands the image by analyzing the gradient information. To find the histogram of gradients, the horizontal and vertical gradients are calculated. This is easily achieved by filtering the image with the two kernels $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1\\0\\1\end{bmatrix}$. This helps in firing the magnitude of gradient where ever there is a sharp change in intensity. This helps in extracting only the relevant information from the image while eliminating the rest. From the below Fig. 2 it is easy to identify that it is a face.

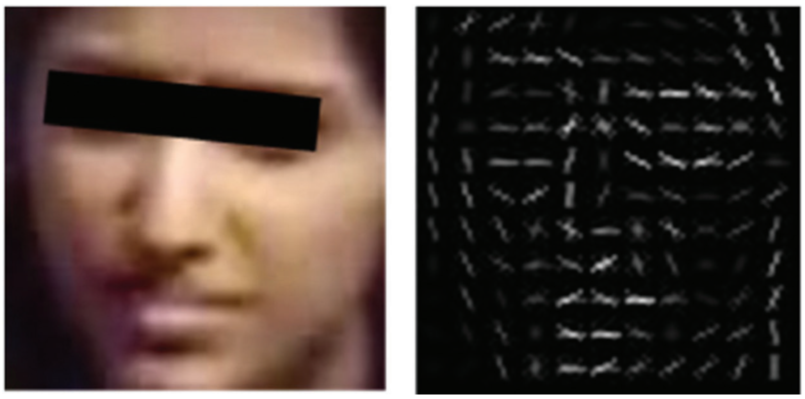

Figure 2: Gradients calculated to extract the relevant information from face

The x-gradient fires on vertical lines and the y-gradient fires on horizontal lines. The magnitude fires in places of sharp intensity changes. None of them fire when the region is smooth. If g_x *and* g_y are the horizontal and vertical components of the gradient, then the magnitude g and direction θ of the gradient is calculated using Eq. (6)

$$g = \sqrt{g_x^2 + g_y^2} \qquad \theta = arctan\frac{g_y}{g_x} \tag{6}$$

The values for each pixel over 8 × 8 block are quantized into 9 bins [18]. The bin represents a directional angle of gradient and value in that bin. The value is calculated as the summation of the magnitudes of all pixels with the same angle. The histogram is then normalized over a 16 × 16 block size. This is done by normalizing four blocks of 8 × 8 together which helps minimize the effects of over and under exposure. With the above steps, the face can be detected. Next step is to recognize the same.

This is done by using Facial encodings. These are 128 measurements that are extracted from each face. The face encoder is pre-trained on public datasets to generate the 128 embeddings. This is used to construct face embeddings used for the actual recognition process. This pre-trained model is used to extract the embeddings for the image database used for this work. Multiple images should be available for each person to be trained. The image for every person is stored in separate folder with the folder name being the person's name. The algorithm goes through each face, extracts embeddings and associates it with the person's name. All the embeddings along with the names for the trained faces are stored in a face encodings file. This file is later used by the recognizer to match a new image and find the closest match. The input image is taken, and the face location identified first followed by extraction of the 128d embeddings. These embeddings are compared with the encodings in the face encodings file. If there is a close match, then the name of the person to whom the face belongs is identified. If there is more than one match, then one which has the maximum number of matches is selected.

The above face recognition process can also be done using aligned faces as input. In some frames the faces may be slightly turned towards left or right or may be tilted up or down rather than looking straight. Face Alignment is achieved using warping, so that the eyes, nose and lips are always in the same position in the image. For this purpose, face landmark estimation [19] is used. The Face Alignment algorithm uses dlib facial landmark predictor which estimates 68 specific points on any face which are called as the landmarks. These landmarks represent key regions in the face around eyes, nose, mouth and jawline The predictor used is pre-trained to identify these 68 points. These points are used to align the face. The steps involved in the process can be understood by Fig. 3.

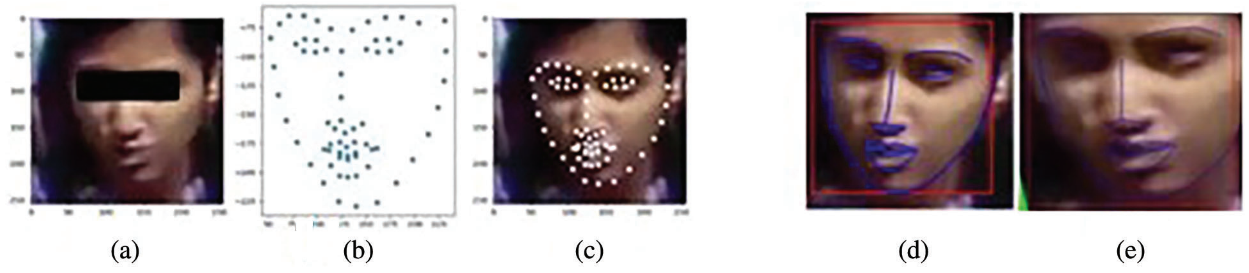

Figure 3: Face alignment using facial landmarks. (a) Aligned face (b) 68 facial landmarks (c) Landmarks overlayed on aligned face (d) Original face (e) Aligned face

3.2.2 Method 2-DNN Face Detector in OpenCV with OpenFace Using OpenCV

The Deep Neural Network (DNN) face detector is built on Caffe model. It is uses ResNet-10 architecture and is based on the Single Shot Detector (SSD). The neural network training and inference portions use

Torch, Lua and luajit. The DNN detector detects if a face is present in the image and if present the location of the face within the image is returned.

The input image is pre-processed to perform mean subtraction, scaling and in some cases, channel swapping as required. Mean Subtraction is performed to minimize the effects of changes in lighting conditions [11]. If R_i, G_i, B_i are the Red, Green and Blue Values of the ith pixel in the image and mean values of the Channels are given by μ_R, μ_G, μ_B then the new RGB values for the ith pixel is calculated as in Eq. (7)

$$R_i = R_i - \mu_R; \quad G_i = G_i - \mu_G; \quad B_i = B_i - \mu_B \tag{7}$$

Scaling can be done for the purpose of normalization by a factor σ as shown in Eq. (8), which may represent the standard deviation across the training dataset. For this work no scaling option is chosen i.e., scale factor is chosen as one.

$$R_i = (R_i - \mu_R)/\sigma; \quad G_i = (G_i - \mu_G)/\sigma; \quad B_i = (B_i - \mu_B)/\sigma \tag{8}$$

The swapping option is available as OpenCV expects the image in BGR (Blue Green Red) order. But the mean value expects in RGB (Red Green Blue) order. To resolve this R and B channels in image are to be swapped.

The above pre-processed image is passed through to the Deep Neural network for detection in the form of a blob. The face detector gives a list of detections which contains the information about the bounding boxes of the faces in the image. The problem with using these bounding boxes directly as an input into the neural network is that faces may be looking in different directions. So, the faces are aligned using the face aligner from dlib. Testing is also done without aligning step and the results are tabulated.

For the next step face recognition, the face image cannot be directly used by the recognizer. This is too high-dimensional for a classifier to take as input and recognize the face. So, to simplify the same, a low dimensional representation is created. In this work, the openface.nn4.small2. v1-face embedding deep learning model helps with feature extraction using a low-dimensional representation also known as 128-d embeddings for the face blob. The pre-trained Torch Deep Learning model is used to generate the 128 d embeddings. The simplified low dimensional representation generated by the embedder helps the neural network in making efficient classification. The extracted embeddings are passed on to Linear SVM model to perform the desired Facial recognition.

For training purpose, the pre-trained OpenFace model is used to generate the embeddings. This model uses FaceNet's triplet loss to extract the embeddings on the unit hypersphere. Euclidean distance between the embeddings represents the similarity between faces. The triplets have two faces of the same person (Anchor and Positive) and one face of a different person (negative). The goal of the triplet loss function is to separate the positive pairs from the negative by a distance margin while minimizing the distance between the anchor and the positive image.

[14] The facial embeddings $f(x)$ for the image x embedded on d-dimensional Euclidean space can be represented as $f(x) \in \mathbb{R}^d$. The embedding should be on the d-dimensional hypersphere, $\|f(x)\|_2 = 1$.

The goal of the training is to minimize the distance between the image x_i^a (anchor) of a person and all other images x_i^p (positive) of the same person compared to the distance between the anchor to an image x_i^n (negative) of any other person. All the triplets should satisfy the condition given in Eq. (9)

$$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \ \forall \left(x_i^a, \ x_i^p, \ x_i^n\right) \in \mathcal{T} \tag{9}$$

α is the margin that is enforced between positive and negative pairs and $\mathcal{T}$ is the set of all possible triplets in the training dataset and with a cardinality N. The goal of the training is to minimize triplet loss function given by (10).

$$\mathcal{L} = \sum_{i}^{N} [\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha] \tag{10}$$

The embeddings extracted by OpenFace model are used to train the linear SVM. For training the SVM, a local image database was created using multiple images for each person. The image for every person is stored in separate folder with the folder name being the person's name. The OpenFace model goes through each face, extracts embeddings and associates it with the person's name. All the embeddings along with the names which are the labels for the trained faces are extracted and stored as embeddings and label pickle files. The key steps involved in the face recognition pipeline proposed in this method are summarized as in Fig. 4.

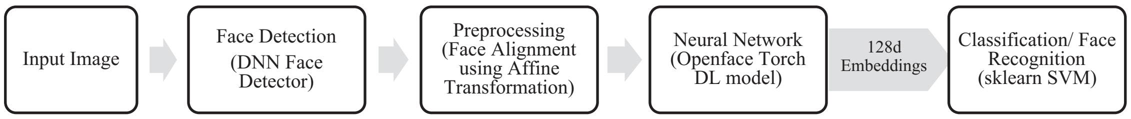
Input Image
Face Detection (DNN Face Detector)
Preprocessing (Face Alignment using Affine Transformation)
Neural Network (Openface Torch DL model)
128d Embeddings
Classification/ Face Recognition (sklearn SVM)

Figure 4: Summary of the OpenCV face recognition pipeline

Testing was done with and without aligning the faces and in this method aligned faces gave better accuracy. Face detection has already been explained in detail earlier. The faces are aligned using FaceAligner from imutils. The alignment is done by finding the centroid for each eye as well as the angle between the centroids [20]. Also, the midpoint between the eyes is computed. Once the angle of required rotation is found the entire face is rotated considering this midpoint as the top of the nose using affine transformation. The Fig. 5 below explains how the face alignment was performed in this method.

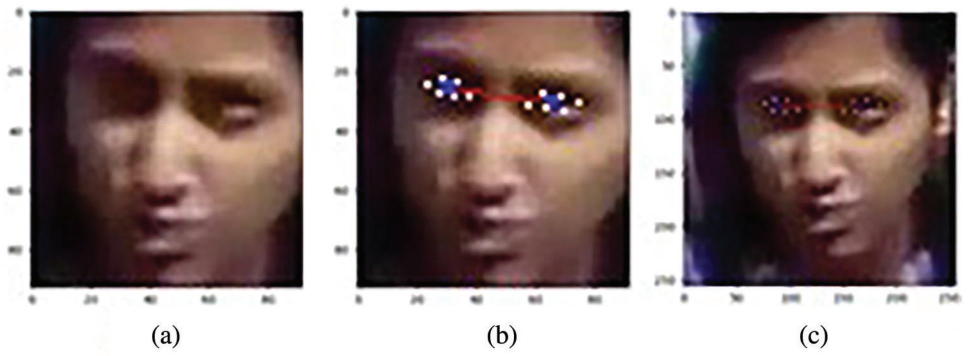

Figure 5: Face alignment using affine transformation. (a) Original face (b) Eye landmarks overlayed on original face (c) New eye landmarks on aligned face

3.3 Social Distance Calculation

The goal is to find distance between the given seed person and other people in the frame, to identify the primary contacts. For this first all the people in the image need to be detected. If the seed person is available in the frame, then the distance between this person to each of the other person is to be calculated. This is done by calculating the Euclidean distance between the centroid of the persons bounding boxes. The minimum distance to be maintained is set as the threshold value. If there is a violation, then the other person is identified using the Facial detection and recognition algorithms and stored as primary contact. If no violation, then we proceed to find the distance with other persons in the image until the distance from seed person to all other persons in the frame is calculated. The distance between the pixels is converted to measurable metric distance such as meters based on the typical distance of an object from the camera.

4 Proposed Solution

This application doesn't require real time processing. Hence offline processing is proposed. The proposed solution helps identify the primary contacts after a person is tested Covid positive. While it is easy to track the contacts if the subject has not been out of home, most cases they would have gone to several public places before they knew that they were infected. So, it is important to gather information from person on the recent public visits and collate the CCTV videos available for the date and time from those places. These videos can be used as input to the solution to identify the primary contacts who had been in close proximity with the subject/seed person. We train the Face detection and recognition algorithms with few photos of the seed person. The videos collated are first converted into frames. For practical purposes it is enough if we take 1 FPS (Frame Per Second), as there won't be much change in the movement of people within a second especially when they are walking around in public places. So 1 FPS input video used for testing. The overall approach is explained briefly in Fig. 6.

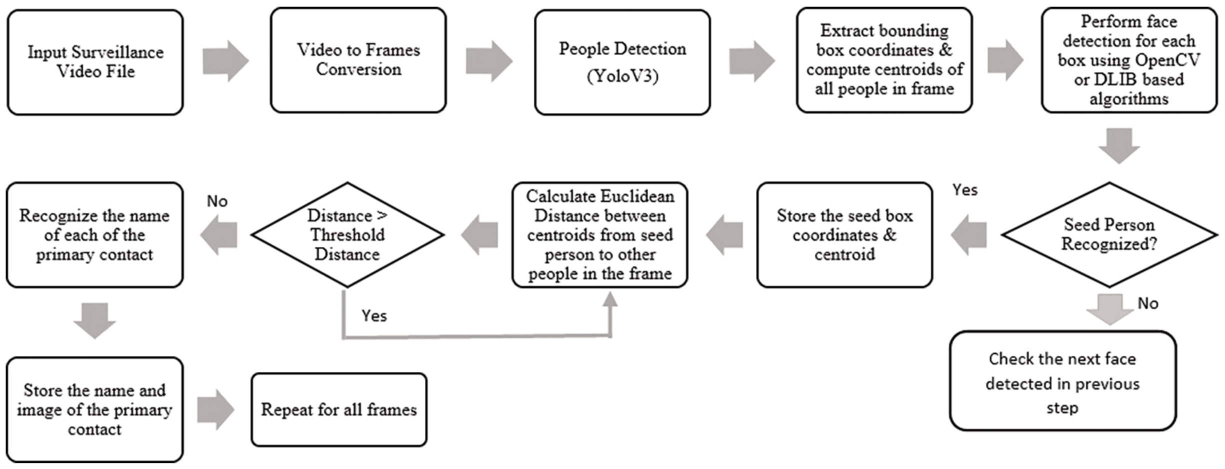
Input Surveillance Video File
Video to Frames Conversion
People Detection (YoloV3)
Extract bounding box coordinates & compute centroids of all people in frame
Perform face detection for each box using OpenCV or DLIB based algorithms
Seed Person Recognized?
Yes
No
Check the next face detected in previous step
Store the seed box coordinates & centroid
Calculate Euclidean Distance between centroids from seed person to other people in the frame
Distance > Threshold Distance
Yes
No
Recognize the name of each of the primary contact
Store the name and image of the primary contact
Repeat for all frames

Figure 6: Proposed solution

From each frame detect all people using YOLO V3 algorithm and store their bounding box coordinates in a list. Use the images from each of these bounding boxes to detect the faces using HOG detector or DNN detector and then recognize them using dlib face recognition or Openface and SVM. If the seed person is present, then we calculate the Euclidean distance from the centroid of the seed person bounding box to the centroid each of other persons bounding box. If this distance is less than the prescribed distance, then we use face detect and recognize algorithms to find the name of the contact and store it in our database. Irrespective of whether the algorithm is able to recognize the name of the contact or not, still the image of the contacts are extracted and stored in a separate path for our perusal later. If the distance is the and for every frame. By the end of the processing, we would have the names of the primary contacts in a separate file and also their images extracted from the frames in a separate path that we specified.

Training and Testing

For YOLO v3 we just utilized a pretrained algorithm trained on COCO dataset and this worked well for identifying the persons in the frame. The face detection models are also pre-trained on public image databases. For the extraction of Facial embeddings, the models are pre-trained on public image databases as well. OpenFace is trained with 500k images from combining the two largest labelled face recognition datasets for research, CASIA-WebFace (Chinese Academy of Sciences' Institute of Automation) and FaceScrub. For the face recognition the classifier can initially be trained with some public image databases for the specific region or city. To check if the accuracy improves, we trained the face detection and recognition algorithms with few images of the seed person from the input video and studied the results. Results were also compared with and without face alignment. For the purpose of this study, the training was done with multiple images for each person in the videos. Though the initial analysis was done for a wedding video shot with professional camera and some videos shot on the road with mobile camera initially, further testing was done using CCTV videos used for home surveillance. Two different videos from cameras installed at different views and heights were covered to check the variation in results.

5 Results, Analysis and Conclusion

The above approach was tested with a 2 different CCTV footage videos with low resolution (**960 × 576**). We have tested the performance with YOLOV3 for object detection and two methods of HOG face detector with face recognition algorithm using dlib and DNN face detector with Openface face recognition algorithms. The same was tested with and without pre-processing of faces by performing alignment before recognition. Testing was also done to check if there was any improvement in recognition if the algorithms were tried by additional training with few images from input video. The performance of the model was evaluated by calculating the accuracy, precision and recall understanding how well they were able to deliver the expected performance. To calculate these, we determine the True Positives, True Negatives, False Positives and False negatives for each of these. True Positives (TP) and True Negatives (TN) are the count of correctly predicted values—Yes predicted as yes and no predicted as No. False Positives (FP) and False Negatives (FN) are wrong predictions. Accuracy is the ratio of correct prediction to the total observations as in Eq. (11)

$$Accuracy = (TP + TN)/(TP + FP + FN + TN \tag{11}$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations given by Eq. (12). High precision relates to the low false positive rate.

$$Precision = TP/(TP + FP) \tag{12}$$

Recall which also refers to sensitivity is the ratio of correct positive predictions to all observations in actual class—yes as given by below Eq. (13).

$$Recall = TP/(TP + FN) \tag{13}$$

5.1 Performance of Object Detector

YOLO V3 performed well for the videos tested it was able to identify most of the people from all video frames accurately with very minimal miss. The True Positives and Negatives here refer to right prediction of all the people present or absent in each frame. The results are tabulated in Tab. 1.

5.2 Performance of Facial Detector

Face detection and recognition efficiency drops if the key facial features are not fully visible. So, we have tried to give the accuracy, precision and recall parameters in two ways—1) Considering only the count of faces with the key features fully visible and 2) Without eliminating any faces.

Table 1: Results for YOLO V3

Input file	Objects detected	People present	People detected	Accuracy	Precision
CCTV Video 1	71	71	70	98.6%	100%
CCTV Video 2	97	83	82	99.0%	100%

The HOG face detector in most cases was able to identify the person if full view of the face was available. The accuracy increased only slightly if the algorithm was trained with few images of seed person from the input video. The accuracy was better for HOG detector when the faces were not aligned before detection. When the Facial detection was attempted after training with two or three additional photos of the people from the video file and there was an improvement in identification as can be seen in the same table. As can be seen in the results as tabulated in Tab. 2, HOG detector performed much better than DNN face detector. The DNN detector performs slightly better when the faces were aligned before recognition compared to when they were not. The accuracy of face detection for DNN without alignment was quite low, even when trained with images from input video.

Face detection results-considering only properly visible faces

Table 2: Face detection results for CCTV video1

Face detection algorithm	Face aligned	Trained with images from input	Accuracy	Precision	Recall
HoG	No	Yes	97.80%	100%	97.80%
HoG	No	No	95.70%	100%	95.70%
HoG	Yes	No	71.70%	100%	71.70%
DNN	Yes	No	39.10%	100%	39.10%
DNN	Yes	Yes	25.00%	92%	23.90%
DNN	No	Yes	17.40%	100%	17.40%

Tabs. 2 and 3 are the results considering only the faces that are properly visible as the base count i.e., cases for which the key facial features are visible to CCTV 1 and 2 after eliminating the remaining from calculation. The corresponding graphical representations are given by Figs. 7 and 8.

Table 3: Face detection results for CCTV video2

Face detection algorithm	Face aligned before processing	Trained with input images	Accuracy	Precision	Recall
HoG	No	Yes	75.80%	100%	75.80%
HoG	No	No	75.80%	100%	75.80%
HoG	Yes	No	51.50%	100%	51.50%
DNN	Yes	No	39.40%	100%	39.40%
DNN	Yes	Yes	40.00%	93%	39.40%
DNN	No	Yes	39.40%	100%	39.40%

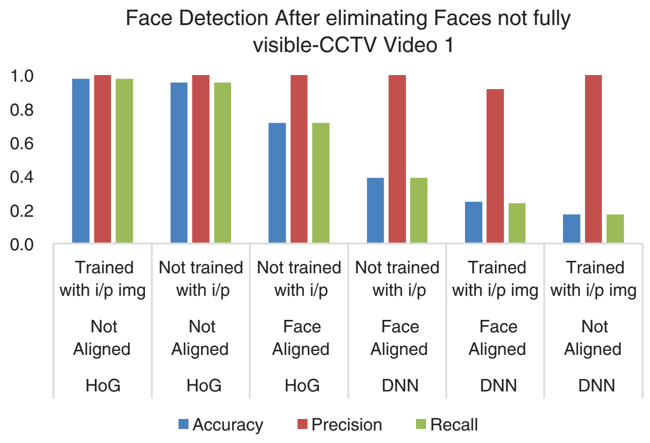

Figure 7: Face detection for CCTV video1

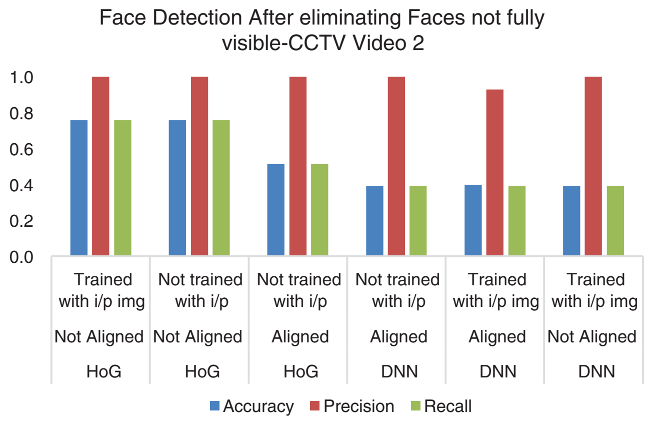

Figure 8: Face detection for CCTV video2

Tabs. 4 and 5 are the results considering all the faces in the frames irrespective of whether they are properly visible to CCTV 1 and 2. The corresponding graphical representations are given in Figs. 9 and 10.

Face detection results-considering all faces

Table 4: Face detection results for CCTV video1

Face detection algorithm	Face aligned before processing	Trained with input images	Accuracy	Precision	Recall
HoG	No	Yes	64.30%	100%	64.30%
HoG	No	No	62.90%	100%	62.90%
HoG	Yes	No	47.10%	100%	47.10%
DNN	Yes	No	25.70%	100%	25.70%
DNN	Yes	Yes	16.70%	92%	15.70%
DNN	No	Yes	11.40%	100%	11.40%

Table 5: Face detection results for CCTV video2

Face detection algorithm	Face aligned before processing	Trained with input images	Accuracy	Precision	Recall
HoG	No	No	30.50%	100%	30.50%
HoG	No	No	30.50%	100%	30.50%
HoG	Yes	No	20.70%	100%	20.70%
DNN	Yes	No	15.90%	100%	15.90%
DNN	Yes	Yes	16.70%	93%	15.90%
DNN	No	Yes	15.90%	100%	15.90%

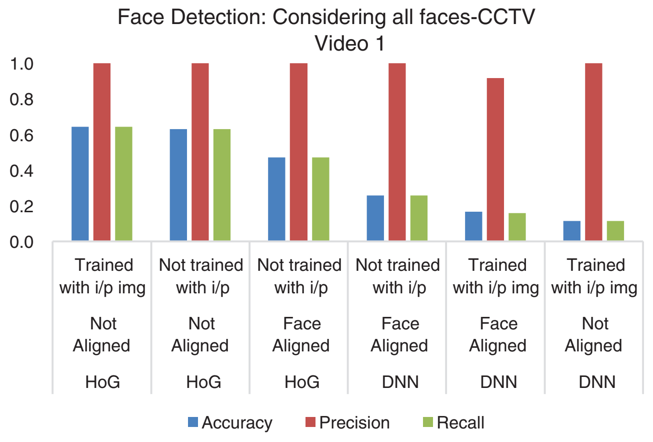

Figure 9: Face detection for CCTV video1

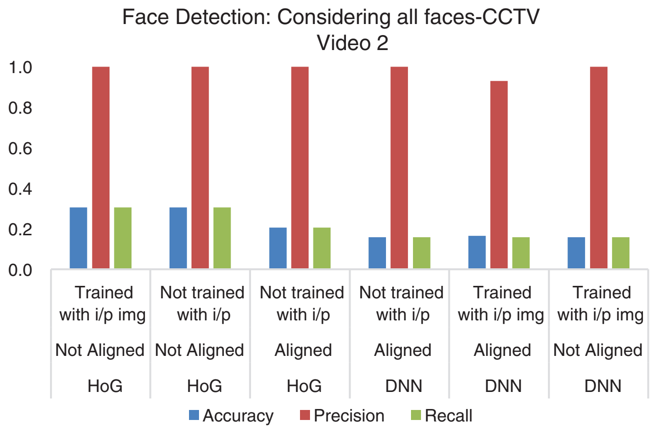

Figure 10: Face detection for CCTV video2

5.3 Performance of Facial Recognition

Face Recognition was tried out with Openface and dlib based face recognition algorithm. Dlib based algorithm performed better for the tested CCTV input videos. These algorithms were also tested by training with few additional photos from the input video to check if there is an improvement in recognition. The results obtained for the different input videos are tabulated and discussed below.

Tabs. 6 and 7 are show the results for the face recognition algorithms for CCTV Video 1 and 2 respectively. The corresponding graphs are shown in Figs. 11 and 12. For the Recognition algorithm only the detected faces come in as input and hence we have considered all the calculations only based on that count. While Dlib based algorithm worked much better without aligning faces before recognition, the accuracy of OpenFace improved considerably by training them and processing them after alignment. Training them with few images from input video improved the accuracy for both the algorithms.

Table 6: Face recognition results for CCTV video1

Face recognition algorithm	Face aligned before processing	Trained with input images	Accuracy	Precision	Recall
dlib based	No	Yes	97.78%	98%	100.00%
dlib based	No	No	86.36%	86%	100.00%
dlib based	Yes	No	12.12%	12%	100.00%
OpenFace	Yes	No	61.11%	61%	100.00%
OpenFace	Yes	Yes	66.67%	64%	100.00%
OpenFace	No	Yes	37.50%	38%	100.00%

Table 7: Face recognition results for CCTV video2

Face recognition algorithm	Face aligned before processing	Trained with input images	Accuracy	Precision	Recall
dlib based	No	Yes	100.00%	100%	100.00%
dlib based	No	No	96.00%	96%	100.00%
dlib based	Yes	No	5.88%	6%	100.00%
OpenFace	Yes	No	61.54%	62%	100.00%
OpenFace	Yes	Yes	100.00%	100%	100.00%
OpenFace	No	Yes	76.92%	77%	100.00%

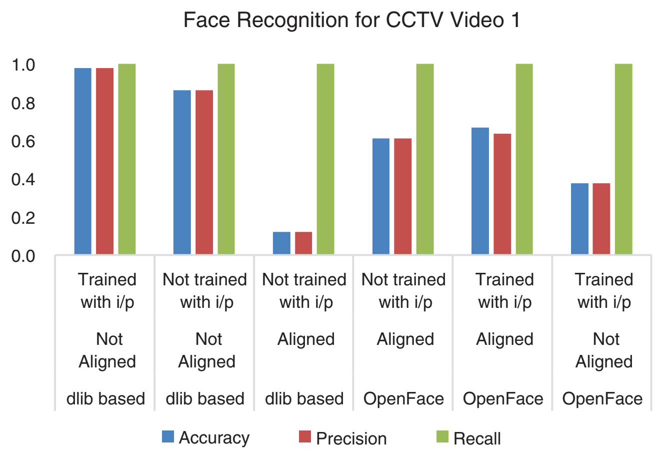

Figure 11: Face recognition for CCTV video1

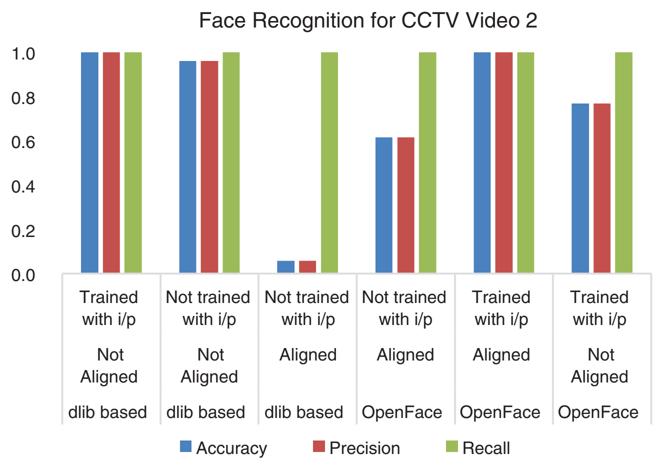

Figure 12: Face recognition for CCTV video2

5.4 Performance of Social Distance Calculation Logic

The performance of the social distance calculation is equally important in identification of the primary contacts. This was tested for an input video shot from a professional camera in a wedding ceremony. From the testing done, it was concluded that none of the primary contacts were missed out. But there were few non-primary contacts identified as primary, as the depth from the camera is not properly visible in a 2D image. This was due to the fact that camera calibration was not done, and we have used standard ratio to convert pixels to measurable metric. Tab. 8 shows the accuracy, precision and recall results for the Social Distance calculation based on the testing done.

Table 8: Results for social distance calculation

TP	TN	FP	FN	Accuracy	Precision	Recall
71	45	39	0	75%	65%	100%

5.5 Output

Few sample output images showing the contacts identified and showing the social distance between the seed and other people are given below. As discussed earlier we have used the CCTV videos for testing purpose, and these are taken from the ones typically fitted in houses. First image is from CCTV Video 1, in which there is one seed and one primary contact. The second and third images are from the CCTV Video 2, in which there is one seed person and two primary contacts. The processed outputs from these images are seen in Fig. 13.

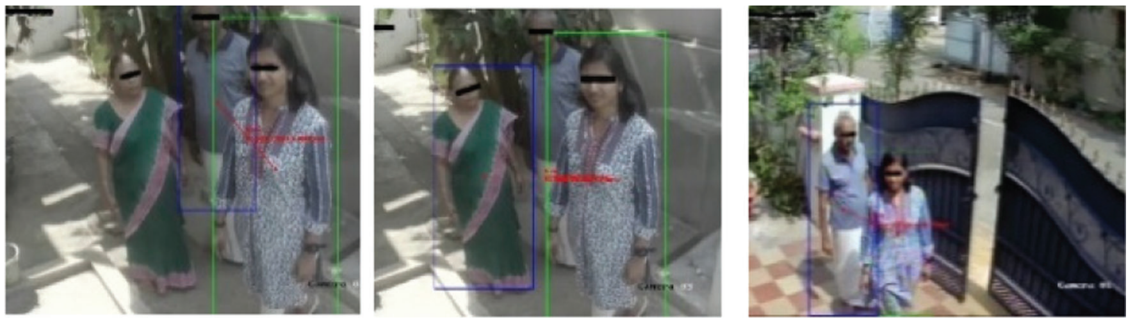

Figure 13: Output for CCTV videos

Acknowledgement: None other than those mentioned in the Authors section have contributed to this article.

Funding Statement: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] World Health Organization, Information on Covid-19. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019 [Online Accessed August 25, 2021].

[2] E. N. Kajabad and S. V. Ivanov, "People detection and finding attractive areas by the use of movement detection analysis and deep learning approach," *Procedia Computer Science*, vol. 156, pp. 327–337, 2019.

[3] Intellica.AI, *A Comparative Study of Custom Object Detection Algorithms*. Intellica.AI. Available: https://intellica-ai.medium.com/a-comparative-study-of-custom-object-detection-algorithms-9e7ddf6e765e [Online published: 2019, December 13].

[4] Y. E. Li, *Dive Really Deep into YOLO v3: A Beginner's Guide*. Towards datascience. Available: https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e [Online published: 2019, December 31].

[5] K. Prem and Y. Liu, "The effect of control strategies to reduce social mixing on outcomes of the COVID-19 epidemic in wuhan, China: A modelling study," *The Lancet*, vol. 5, no. 5, pp. E261–E270, 2020.

[6] C. Adolph, K. Amano, B. Bang-Jensen, N. Fullman and J. Wilkerson, "Pandemic politics: Timing state-level social distancing responses to COVID-19," *Journal of Health Politics, Policy and Law*, vol. 46, no. 3, pp. 211–233, 2021.

[7] N. S. Punn, S. K. Sonbhadra, S. Agarwal and G. Rai, *Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and deepsort techniques*. arXiv.org. Available: https://arxiv.org/abs/2005.01385 [Online published: 2021, April 27].

[8] I. Ahmed, M. Ahmad, J. J. P. C. Rodrigues, G. Jeon and S. Din, "A deep learning-based social distance monitoring framework for COVID-19," *Sustainable Cities and Society*, vol. 65, Feb 2021, article number 102571.

[9] S. Saponara, A. Elhanashi and A. Gagliardi, "Implementing a real-time, AI-based, people detection and social distancing measuring system for COVID-19," *Journal of Real-Time Image Processing*, vol. 18, no. 6, pp. 1937–1947, 2021.

[10] A. Geitgey, *Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning*. Medium. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78 [Online published: 2016, July 24].

[11] A. Rosebrock, *Deep learning: How OpenCV's blobFromImage works*. pyimagesearch. Available: https://www.pyimagesearch.com/2017/11/06/deep-learning-opencvs-blobfromimage-works/ [Online published: 2017, November 6].

[12] V. Gupta, *Face Detection–OpenCV, Dlib and Deep Learning (C++/Python)*. Available: https://learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/ [Online published: 2018, October 22].

[13] O. Kharkovyna, *Facial recognition: 8 Open-source tools to detect faces*. Available: https://medium.datadriveninvestor.com/facial-recognition-8-open-source-tools-to-detect-faces-4ec8e37bfcc6 [Online published: 2021, March 1].

[14] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, pp. 815–823, 7-12 June 2015.

[15] B. Amos, B. Ludwiczuk and M. Satyanarayanan, "*OpenFace: A general-purpose face recognition library with mobile applications*," Available: http://reports-archive.adm.cs.cmu.edu/anon/anon/2016/CMU-CS-16-118.pdf [Online published: 2016, June].

[16] W. Crumpler, *How Accurate are Facial Recognition Systems–and Why Does It Matter?*. CSIS. Available: https://www.csis.org/blogs/technology-policy-blog/how-accurate-are-facial-recognition-systems-%E2%80%93-and-why-does-it-matter [Online published: 2020, April 14].

[17] J. Redmon and A. Farhadi, *YOLOv3: An incremental improvement*. arXiv.org. Available: https://arxiv.org/abs/1804.02767 [Online published: 2018, April 08].

[18] R. Thaware, *Real-Time Face detection and Recognition with SVM and HOG Features*. EEWeb. Available: https://www.eeweb.com/real-time-face-detection-and-recognition-with-svm-and-hog-features/ [Online published: 2018, May 28].

[19] A. Rosebrock, *Facial landmarks with dlib, OpenCV, and Python*. pyimagesearch. Available: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/?_ga=2.268801905.1858494572.1636108281-1176519729.1617967486 [Online Published: 2017, April 3].

[20] A. Rosebrock, *Face Alignment with OpenCV and Python*. pyimagesearch. Available: https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/ [Online published: 2017, May 22].