Retraction

Retraction: A Hybrid Model to Ensure Biosecurity during Pandemic Situations Using OpenCV (J. Phys.: Conf. Ser. 1916 012129)

Published 23 February 2022

This article (and all articles in the proceedings volume relating to the same conference) has been retracted by IOP Publishing following an extensive investigation in line with the COPE guidelines. This investigation has uncovered evidence of systematic manipulation of the publication process and considerable citation manipulation.

IOP Publishing respectfully requests that readers consider all work within this volume potentially unreliable, as the volume has not been through a credible peer review process.

IOP Publishing regrets that our usual quality checks did not identify these issues before publication, and have since put additional measures in place to try to prevent these issues from reoccurring. IOP Publishing wishes to credit anonymous whistleblowers and the Problematic Paper Screener [1] for bringing some of the above issues to our attention, prompting us to investigate further.

[1] Cabanac G, Labbé C and Magazinov A 2021 arXiv:2107.06751v1
Retraction published: 23 February 2022
A Hybrid Model to Ensure Biosecurity during Pandemic Situations Using OpenCV

Syeda Arbeena Kausar, Roopa K, Kaushik S Arakalgud and Anupama B C

1Department of Electronics and Communication Engineering, The National Institute of Engineering, Mysuru-570008, Karnataka, India
2Tally Solutions Pvt. Ltd., Bengaluru-560068, Karnataka, India
aarbeena0@gmail.com, roopakodnad@gmail.com, anubc52@gmail.com
ska.kaushik@gmail.com

Abstract. This paper presents a hybrid model consisting of hardware, software subsystems and publicly available trained feature sets. The developed hybrid model is useful in automated contactless collection and analysis of employees’ and visitors’ data in an organization especially during pandemic situations to ensure biosecurity. Such data include temperature and face mask status. If the set norms are not satisfied, the entry into the premises will be restricted or denied. The status is also updated in the corresponding record in the organization database. The hardware subsystem includes an arduino nano, sensors and audio visual alarms. The software subsystem was developed using OpenCV in Python and VSCode editor. Both offline and real time implementations were carried out. The model was validated using real time images and online data sources. The system was tested and found to work satisfactorily under practical input conditions.

1. Introduction
The main objective of the project is to build an automated contactless system to screen employees and visitors, collect and save their data for management and analytics purpose. This system will be useful in organisations like educational institutions, health care centres and industries. The system involves two levels of scanning. In the first level the employee and visitor are verified. In the second level their temperature and mask status are checked. The project gains more importance in the aftermath of Covid19 pandemic wherein the need for contactless processes in day to day activities has increased.

2. Related Work
This section describes some of the recent works in the related field. [1] propose a system to identify people who are not wearing protective facial mask in a smart city network monitored by ClosedCircuit Television (CCTV) cameras and then informing the in charge authority for necessary action. A deep learning Convolutional Neural Network (CNN) architecture [2] is used and they have reported 98.7% recognition accuracy. They collected dataset from [3, 4] for training and testing the model. They used a total of 1539 samples with 80% and 20% for training and testing phases respectively.

[5] discuss masked face recognition using CNN. They used AR face database [6], IIIT-Delhi Disguise Version 1 Face Database [7] and their own Masked Face Dataset (MFD). The MFD has 45 subjects in seven disguises and various backgrounds with a total of 990 images. They reported accuracies...
ranging from 47.43% to 98.50%. [8] discuss various techniques of implementing face detection and tracking like Adaptive Boosting (AdaBoost) and Haar cascades approaches using Open source Computer Vision (Open CV).

[9] discuss a system to identify suspicious individuals, which consists of a Raspberry Pi Zero, a camera module, capacitive touch sensor and organic light-emitting diode (OLED) display. This system uses Haar Cascade Classifier (HCC) for face detection followed by Local Binary Pattern Histogram (LBPH) for facial recognition using OpenCV. They used an image dataset of four subjects with ten images of each subject. They reported recognition accuracy in the range of 55 - 65 %. Alcantara et al. [10] discuss head detection and tracking using OpenCV. They obtained an accuracy of 71.40% to 90%.

[11] describe various face recognition algorithms like Haar Cascade, Eigen faces, Fisherface and Local Binary Pattern (LBP). Also described in the paper is a Principal Component Analysis (PCA) based facial recognition system which can provide a reduced representation of the data. A deep learning based facemask and physical distancing detection is discussed [12].

[13] describes an implementation for detection of objects using HCC [14] and OpenCV library [1517]. HCC is based on a machine learning approach where a cascade function is trained from both positive and negative images. The trained function is then used to detect the objects in unseen or test images. It uses haarcascade_frontalface_default.xml [18] to detect individuals’ faces.

[19] discuss a facemask-wearing condition identification method by combining image Super Resolution and Classification Networks (SRCNet). The system was trained and evaluated on the public domain Medical Mask Dataset containing 3835 images with 671 images of not wearing facemask, 134 images of incorrectly wearing facemask and 3030 images of correctly wearing facemask. They have reported 98.7% accuracy of detection.

[20] discuss a binary face classifier to detect face in an image irrespective of its alignment. Training is performed through fully convolutional networks to semantically segment the faces present in an image. Semantic segmentation is the process of assigning a label to each pixel of the image, i.e., either face or no-face. The authors describe that their model works well for both frontal and nonfrontal face images. The Visual Geometric Group VGG 16 architecture is used as the base network for face detection. The initial image size fed to the model is 224×224×3. After the final max pooling layer – the image size is reduced to 28×28×2. Experiments were performed on Multi Human Parsing Dataset containing about 5000 images. They have reported an accuracy of 93.8%.

A mechanism to implement real time face mask detection and alert is given in [21]. The datasets for ‘with_mask’ and ‘without_mask’ categories are available in [22]. A few more online data sources can be found in [23].

In the work reported in this paper the authors present two models both utilising, hardware and software systems, hence called hybrid, for safe analysis of data by its contactless collection and processing. Model1 uses a single trained feature set, assuming that the input is invariably a face image. Model2 is more robust. It is built on a multi-layered filtering approach where each layer filters out non facial features. This model uses five different independently trained feature sets. The collective decision is used to calculate the accuracy. The models are tested for both offline and online operations. The validation of the model is done using online image datasets. A subset of this dataset as described in section 4 of this paper was also used for validation. The authors also created their own small dataset of 20 images with and without mask for both offline and real time testing of the model.

3. Methodology
The overall block diagram of the hybrid model is shown in Figure 1. The safe data analysis model consists of two levels of scanning. In the first level of scanning, contactless checking of employees and visitors is done using Quick Response Code (QR Code) scanning which uses a 2D pictorial code to store the data. Python supports QR Code scanning with the help of built in libraries like pyzbar and zbar.
After the first level of scanning, the employee/visitor enters a lobby where there is a camera to scan the face image. The second level of scanning is used to check for the mask and to ensure that the temperature is within the safe limits. A database is used to store and retrieve the data for analysis.

![Block Diagram](image)

**Figure 1.** The overall block diagram of the hybrid model

### 3.1. Steps involved in QR code scanning for the Employee
- a) Each employee has his identity card (ID) with a built-in unique QR code containing his employee id.
- b) Employee scans the QR code on his ID card against a camera connected to the computer.
- c) A Python code is written in the computer to read this code and obtain the employee id.
- d) The computer code fetches employee details from the database and verifies his identity.
- e) The computer recognizes the internal employee as the details like identity number, name, designation, department, mobile number and email id are already available in the organization’s database.
- f) The employee is then allowed for the second level of scanning.

### 3.2. Steps involved in QR code scanning for the Visitor
- a) The visitor scans the static QR code meant for the visitor (displayed at the main entrance) using his mobile.
- b) Once scanned, an electronic form (e-form) opens in his mobile. This operation requires internet connectivity.
- c) Visitor fills the e-form with his name, organisation, purpose of visit, department to visit, mobile number, email id and submits the filled e-form.
- d) On successful submission, the visitor receives an id as QR code in his mobile. This is a temporary id for the visitor.
- e) The visitor scans the allotted temporary id (or token), against the camera which is connected to the computer.
f) Again the Python code as mentioned in subsection 3.1 reads the visitor id and verifies his details in the database.
g) On successful verification, the visitor is allowed to enter the lobby for second level of scanning.

3.3. Hardware details
Figure 2 shows the block diagram of hardware used in the model. It consists of an arduino nano [24], an ultrasonic sensor; HCSR04 [25] and a temperature sensor; MLX90614 [26] which is an infrared thermometer used for non-contact temperature measurements. The range of the sensor lies between 2cm and 400cm. The ranging accuracy can be up to 3mm. An OLED [27] is used for displaying the temperature.

For face mask recognition, the images are processed in the computer using the algorithm for mask detection, implemented in Python [28, 29] programming language with OpenCV [30] library functions. The temperature and mask status are updated in the database against the corresponding record. This data may be monitored regularly by the administrative authority for necessary actions and decision making. A piezo buzzer [31] is also ‘on’ whenever the temperature is not in the safe limits.

![Figure 2: Block diagram of hardware for Second level of Scanning.](image)

3.4. Software details
Figure 3 shows the block diagram for software processes involved in the implementation of the model. The input images are preprocessed, like flipping, resizing and gray scale conversion. Features from the test image are compared with those in the trained classifier and object classification is done. The program is written in Python Version 3.8.5 with OpenCV functions. The Microsoft Visual Studio Code (MS VS Code) [32, 33] Release 1.52.1 is used as editor for Python code. Computer with Windows 10 Pro with 64-bit Operating System, Intel(R) Core i5, 2.60 GHz, x64-based processor and 8 GB RAM is used for executing the code. Both online and offline implementations are carried out in this work. Some of the functionalities of OpenCV used in this work are: cv2.resize(), cv2.imread(), cv2.imshow(), cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), vid = cv2.VideoCapture(0), vid.read().
mouth_cascade = cv2.CascadeClassifier("<path to the classifier>");
mouth_cascade.detectMultiScale(gray, 1.5, 6); where 1.5 is the scale factor and 6 is the number of minimum neighbours that each candidate rectangle must consider. Suitable values are found by experimentation. Some of the modules imported for Python program execution are cv2, time, winsound, webbrowser and tkinter.

4. Results

Figure 4(a) and Figure 4(b) show the flow charts for model 1 and model 2 respectively. In model 1, the input should be a face image with ‘mask’ or ‘no mask’ condition. Model 1 uses haarcascade_mcs_mouth.xml [34]. Model 2 uses haarcascade_mcs_mouth.xml [35], haarcascade_eye.xml [36], haarcascade_eye_tree_eyeglasses.xml [37], haarcascade_righteye_2splits.xml [38] and haarcascadelefteye_2splits.xml [39].

Two datasets are used for validating the model; Dataset1 and Dataset2. Dataset1 is a set of Dataset2. It consists of randomly picked 270 images each in the ‘with mask’ and ‘without mask’ categories. Dataset2 consists of 2160 images in the ‘with mask’ category and 1930 images in the ‘without mask’ category. The average response time is approximately 0.38s in real time operation. Tables 1 to 6 show the performance efficiency for different scenarios. Table 7 shows the summary of performances of Model1 (M1) and Model2 (M2). As shown in Table 7, for M2, the accuracy is higher for ‘WT’ compared to ‘M’. This is a logically required outcome as the basic intention is more towards identifying the
defaulter. For M2 with D2, the accuracy for M is the least with 53.2% as it contains many complex images. However, for practical purposes for which it is intended, people are expected to wear a mask properly and show frontal face to the camera. Therefore, the real time implementation accuracy was satisfactory.

Table 8 shows the comparison of the implemented work with two recent research works. The present work used a publicly available larger dataset with a large number of complex images, for validation. It works well for real time operation and is suitable for implementation in organisations at check-in. Temperature checking is also done. Additional trained feature sets can be used to improve the robustness of the model.

Table 1. Performance efficiency of model 1 with dataset 1 for images ‘with mask’.

| S. No | No of images ‘with mask’ detected | Time Required for testing (s) | Efficiency of the model (%) For ‘with mask’ category |
|-------|---------------------------------|-------------------------------|-----------------------------------------------------|
| 1     | 216                             | 13                            | (216/270) x100 = 80                                  |

Table 2. Performance efficiency of model 1 with dataset 1 for images ‘without mask’.

| Sl. No | No of images ‘without mask’ detected | Time Required for testing (s) | Efficiency of the model (%) for ‘Without Mask’ category | Average efficiency of the model (%) |
|--------|-------------------------------------|-------------------------------|---------------------------------------------------------|-----------------------------------|
| 1      | 203                                 | 14                            | (203/270) x100 =75.19                                   | (75.2+80)/2 =77.6                 |

Table 3. Performance efficiency of model 2 with dataset 1 for images ‘with mask’.

| Sl. No | No of images ‘with mask’ Detected | Time Required for testing (s) | Efficiency of the model (%) for ‘with mask’ category |
|--------|----------------------------------|-------------------------------|------------------------------------------------------|
| 1      | 172                              | 19                            | (172/270) x100 = 63.7                                 |

Table 4. Performance efficiency of model 2 with dataset 1 for images ‘without mask’.

| S. No | No of images ‘without mask’ Detected | Time required for testing (s) | Efficiency of the model (%) for ‘without mask’ category | Average efficiency of model 2 (%) with dataset 1 |
|-------|-------------------------------------|-------------------------------|----------------------------------------------------------|-----------------------------------------------|
| 1     | 203                                 | 14                            | 203/270 =75.2                                           | (63.7+75.2)/2 =69.5                           |

Table 5. Performance efficiency of model 2 with dataset 2 for images ‘with mask’.

| Sl. No | No of images ‘with mask’ Detected | Time Required for testing (s) | Efficiency of the model 2 (%) for ‘With Mask’ category |
|--------|----------------------------------|-------------------------------|-------------------------------------------------------|
Table 6. Performance efficiency of model 2 with dataset 2 for images ‘without mask’.

| Sl. No. | No of images ‘without mask’ | Time Required for testing (s) | Efficiency of the model 2 (%) for ‘Without Mask’ category | Average efficiency of the model 2 (%) With dataset 2 |
|---------|----------------------------|-------------------------------|----------------------------------------------------------|--------------------------------------------------|
|         | Detected | Not detected | Face not detected |                                      |                                                   |
| 1       | 1562     | 277          | 91               | 97                                   | (1562/1930) x 100 = 81                           | (53.2+81)/2 = 67                                  |

Table 7. Summary of the performance of M1 and M2.

| Sl. No. | Model Type | Dataset (D1 or D2) | Type of data | % efficiency | Average Efficiency (%) |
|---------|------------|--------------------|--------------|--------------|------------------------|
| 1       | M1         | D1                 | M: With mask, WT: Without mask | 80          | 77.6                   |
|         |            |                    | M: With mask, WT: Without mask | 75.2        |                        |
| 2       | M2         | D1                 | M: With mask, WT: Without mask | 63.7        | 69.5                   |
|         |            |                    | M: With mask, WT: Without mask | 75.2        |                        |
| 3       | M2         | D2                 | M: With mask, WT: Without mask | 53.2        | 67                     |
|         |            |                    | M: With mask, WT: Without mask | 81          |                        |

Table 8. Comparison of results with latest research papers.

| Sl. No. | Year   | Technique used                  | Dataset | Total no. of samples | Accuracy (%) | Remarks                                      |
|---------|--------|---------------------------------|---------|----------------------|--------------|-----------------------------------------------|
| 1       | 2020   | CNN Collection from (3, 4)      | D1      | 1539                 | 98.7         | Smaller dataset.                              |
| 2       | 2020   | SuperResolution and Classification Networks | D2: with mask; 2165 images, without mask: 1930 images | 3835 images (Without mask: 671, Incorrectly wearing: 134, Correctly wearing: 3030) | 98.7 | Smaller dataset. Larger number of ‘correctly wearing’ category compared to the other two categories. |
| 3       | Proposed work/2021 | Open CV                  | i) D2: with mask; 2165 images, without mask: 1930 images | i) M1, D1: 77.6 | Larger dataset. Result reported for both large and small datasets. Checks temperature also. Suitable for |
5. Conclusion
To ensure biosecurity during pandemic situations, the paper discussed an automated system for contactless collection of employees’ and visitors’ data using a QR code, camera, ultrasound sensor and temperature sensor. Hybrid models were built, validated and tested successfully for offline and online operations using Python and OpenCV. The model worked satisfactorily for the intended application. As a future work, this working model must be integrated and developed into a deployable model.

References
[1] Rahman M M, Manik M M H, Islam M M, Mahmud S and Kim J H 2020 An automated system to limit covid-19 using facial mask detection in smart city network 2020 IEEE Int. IOT, Electronics and Mechatronics Conf. (IEMTRONICS), 1-5, Vancouver, BC, Canada, 2020. doi: 10.1109/IEMTRONICS51293.2020.9216386. Available on https://ieeexplore.ieee.org/document/9216386.

[2] Khan A, Sohail A, Zahoora U and Qureshi A S 2020 A survey of the recent architectures of deep convolutional neural networks. 21-04-2020, Issue 8/2020.

[3] https://www.kaggle.com/andrewmvd/face-mask-detection, accessed July 27, 2020.

[4] https://github.com/prajnasb/observations, accessed July 27, 2020.

[5] Ejaz M S and Islam M R 2019 Masked face recognition using convolutional neural network 2019 Int. Conf. on Sustainable Technologies for Industry 4.0 (STI), 1-6, Dhaka, Bangladesh, 2019. doi: 10.1109/STI47673.2019.9068044.

[6] Aleix M Martinez, 2019 AR face database. Available on http://www2.ece.ohiostate.edu/~aleix/ARdatabase.html, accessed Sept. 2019.

[7] Image Analysis and Biometrics Lab @ IIIT Delhi, Available on http://www.iabrubric.org/resources/facedisguise.html, accessed Sept. 2019.

[8] M. Suganya and H. Anandakumar, Handover based spectrum allocation in cognitive radio networks, 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), Dec. 2013. doi:10.1109/icgce.2013.6823431. doi:10.4018/978-1-5225-5246-8.ch012

[9] Haldorai and A. Ramu, An Intelligent-Based Wavelet Classifier for Accurate Prediction of Breast Cancer, Intelligent Multidimensional Data and Image Processing, pp. 306–319.

[10] Alcantara G K L, Evangelista I D J, Malinao J V B, Ong O B, Rivera R S D and Ambata E L U 2018 Head detection and tracking using openCV 2018 IEEE 10th Int. Conf. on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 1-5, Baguio City, Philippines, 2018. doi: 10.1109/HNICEM.2018.8666374.
[11] Khan M, Chakraborty S, Astya R and Khepra S 2019 Face detection and recognition using openCV. 2019 Int. Conf. on Computing, Communication, and Intelligent Systems (ICCCIS), 116-9, Greater Noida, India, 2019. doi: 10.1109/ICCCIS48478.2019.8974493.

[12] Militante S V and Dionisio N V 2020: Deep Learning Implementation of Facemask and Physical Distancing Detection with Alarm Systems 2020 3rd International Conference on Vocational Education and Electrical Engineering (ICVEE), 1-5, Surabaya, Indonesia, 2020. doi: 10.1109/ICVEE50212.2020.9243183.

[13] Tanvir Khan 2019 Computer vision — detecting objects using Haar cascade classifier, Dec. 2019. Available on https://towardsdatascience.com/computer-vision-detecting-objectsusinghaar-cascade-classifier-4585472829a9.

[14] Paul Viola and Michael Jeffrey Jones 2001 Rapid object detection using a boosted cascade of simple features IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Feb. 2001. doi: 10.1109/CVPR.2001.990517. Available on https://ieeexplore.ieee.org/document/990517.

[15] https://opencv.org/, accessed Feb 2021.

[16] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html, accessed March 2021.

[17] Maël Fabien 2019 A guide to face detection in Python (with code). Apr. 2019. Available on https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f69fc1.

[18] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml, accessed March 2021.

[19] Bosheng Qin and Dongxiao Li 2020 Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19. Sensors (2020), 20, 5236. doi: 10.3390/s20185236.

[20] Meenpal T, Balakrishnan A and Verma A 2019 Facial mask detection using semantic segmentation 2019 4th Int. Conf. on Computing, Communications and Security (ICCCS), 1-5, Rome, Italy, 2019. doi: 10.1109/CCCS.2019.8888092.

[21] https://www.youtube.com/watch?v=B-ws7J8On3E&feature=youtu.be, accessed Feb. 2021.

[22] https://drive.google.com/drive/folder/1Z8oqvnvDZ79QyD-R0r1VZ9fvtck-GM, accessed Feb. 2021.

[23] https://medium.com/the-programming-hub/worlds-most-complete-masked-facerecognitiondataset-is-for-free-10d780eed512, accessed March 2021.

[24] https://store.arduino.cc/usa/arduino-nano, accessed March 2021.

[25] https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf, accessed March 2021.

[26] https://components101.com/sensors/melexis-mlx90614-contact-less-ir-temperature-sensor, accessed March 2021.

[27] https://www.winstar.com.tw/products/oled-module/graphic-oled-display/4-pin-oled.html, accessed March 2021.

[28] https://www.python.org/, accessed March 2021.

[29] https://docs.python.org/3/tutorial/index.html, accessed March 2021.

[30] https://docs.opencv.org/master/d9/d08/tutorial_root.html, accessed March 2021.

[31] https://components101.com/buzzer-pinout-working-datasheet, accessed March 2021.

[32] https://code.visualstudio.com/docs, accessed March 2021.

[33] https://code.visualstudio.com/docs/introvideos/basics, accessed March 2021.

[34] https://jardownload.com/artifacts/com.neuronrobotics/BowlerScriptingKernel/0.4.27/sourcecode/haar/haarcascade_mcs_mouth.xml, accessed March 2021.

[35] https://github.com/AlexeyAB/OpenCVdetectionmodels/blob/master/haarcascades/haarcascade_mcs_mouth.xml, accessed March 2021.
[36] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_eye.xml, accessed March 2021.
[37] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_eye_tree_eyeglasses.xml, accessed March 2021.
[38] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_righteye_2splits.xml, accessed March 2021.
[39] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_lefteye_2splits.xml, accessed March 2021.