An Application for the Detection of COVID-19 Face Mask Using OpenCV

T. Venkat Narayana Rao[1], Gadige Vishal Sai[2], Manideep Chenna[3], M. Sahas Reddy[4]
[#1]Professor, Department of CSE, Sreenidhi Institute of Science and Technology, Ghatkesar, Hyderabad, Telangana, India
[#2] Student, CSE,Sreenidhi Institute of Science and Technology, Ghatkesar, Hyderabad, Telangana,
[#3] Student,Chaitanya Bharathi Institute of Technology, Gandipet, Hyderabad, Telangana, India
[#4] Student, Chaitanya Bharathi Institute of Technology, Gandipet, Hyderabad, Telangana, India

Abstract

Computer Vision can solve many inter-extensive applications that can range from agriculture to health care. It can also be implemented to solve many problems that humans could not. Even during this tough COVID-19 pandemic situation, Computer Vision can be employed to contain this novel coronavirus. To date, there has been no efficient vaccine to cure this disease. However, the chances of transmission can almost be nullified if there is an extensive spread usage of masks, proper sanitization, and maintaining social distancing. This paper focuses on creating an application that detects whether a person is wearing a face mask or not with Open Source Computer Vision library OpenCV using Python. Here, the user image is captured from the video stream, then preprocess it and later apply several haar cascade classifiers to detect face, eyes, nose, and mouth from the image. Based on the values obtained, we then apply decision logic to calculate whether a mask is present. This application can be applied in several use cases, such as industrial utility where there is a compulsion for usage of masks.

Keywords: COVID-19, Computer Vision, OpenCV, Python, Haar cascade classifiers.

I. INTRODUCTION

Computer Vision is a part of Computer Intelligence that trains the data processing machine to learn and interpret the visual world. With the help of digital feed in the form of images, videos, etc. from several capturing devices, a computer can precisely detect, examine, identify, and classify objects to respond to what it sees. Computer Vision can be coined as the future of many inter-extensive applications ranging from agriculture to health care. In many scenarios where a human is unable to fit in, to solve a particular problem, most people often raise their hands and turn towards computers seeking their help to solve the problem. One such scenario is the outbreak of the novel coronavirus COVID-19. This pandemic has affected more than 215 countries across the globe by infecting more than 13 million people. Since there is no ready vaccine available to cure this virus, this pandemic is cruising in at a

rapid pace. The only precautions that humanity can take are wearing a mask, maintaining proper social distancing, sanitizing workspace, etc.

In this paper, the main focus is on developing a Computer Vision application which helps us to identify whether an individual covered his face with a mask or not. In this system, an application is built using Python's version of Open Source Computer Vision Library (Open CV). Later haar cascade classifiers is applied to detect the face, eyes, nose, and a mouth, to identify whether a mask covers the individual's face or not. It is proposed to consider the following architectural diagram for our application using Fig 1.1

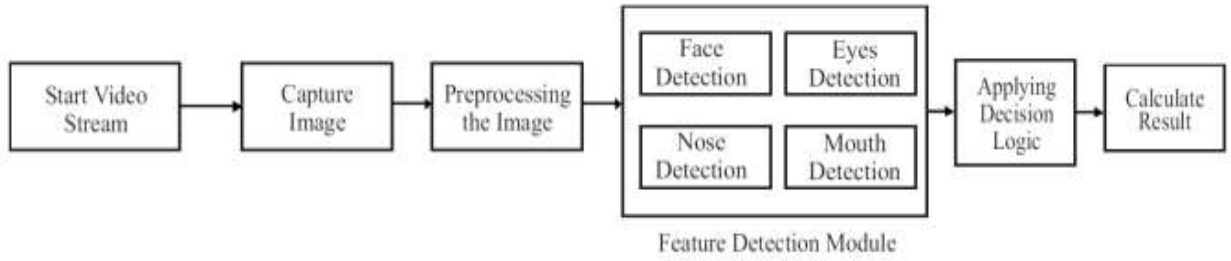

Fig 1.1 Architectural Diagram of Face Mask Detection Application

II. RELATED WORK

The term Computer Vision is deemed to be the future of many inter-extensive use cases, and its researches range from agricultural to health care applications. Computer Vision also finds its place in several Machine Learning (ML), Deep Learning (DL) applications making it an essential concept for all the researchers to work with.

In this paper, for this application, the related work includes an Automatic Face Detection System that can recognize a face, its expression, and also its interaction with the computer, which is known as Human-Computer Interaction. It also provides us with a comprehensive survey of various techniques implemented for face detection for digital images [1].

Similar research includes an application that provides a new method for the face recognized attendance management system using an algorithm named LBP – Local Binary Pattern, in addition to image processing techniques like image blending and histogram equalization to improve the accuracy of the system [2].

Another research presents a methodology to generate accurate face segmentation masks from any arbitrary sized input image [3]. Also, several research pieces include detecting multiple facial masks in a single frame using Deep Learning models built from Convolutional Neural Networks.

III. METHODOLOGY

In this paper, it proposes the following methodology to identify whether a person is wearing a face mask or not.

1. Start Video Stream
2. Capture Image
3. Preprocessing the Image
4. Feature Detection Module
   - 4.1 Face Detection
   - 4.2 Eyes Detection
   - 4.3 Nose Detection
   - 4.4 Mouth Detection
5. Applying Decision Logic
6. Calculate Result

Fig 3.1 Proposed Algorithm for Face Mask Detection Application

3.1. Start Video Stream

When this application run, the OpenCV module starts the camera connected to the device. This camera could be an integrated webcam or the in-built camera present on the device itself. The video stream is done using OpenCV's VideoCapture() method and proceeds continuously until the user positions himself according to the camera and is ready to capture his image.

3.2. Capture Image

The application continually reads the feed as received from the camera and shows it on an in-built User Interface that pops up on the monitor. When the user is ready with his relative position concerning the camera, he presses 'Q,' which captures the user's image, stores it under the name 'test.jpg' as an RGB image as shown in the Fig 3.2.1. He later releases the stream using release() method. Then, the application destroys all the unnecessary OpenCV windows that are present on the screen.

Fig 3.2.1 Captured Image from the stream

3.3. Preprocessing the image

For preprocessing the image, the below mentioned steps are followed:

1. Read the image 'test.jpg' from its storage path.
2. Note that the retrieved image has the dimensions of 640x480 pixels.
3. Convert the image from RGB to its equivalent grayscale.
4. After performing these steps, the image is said to be well-processed and is ready for the application to perform the next steps.

3.4. Feature Detection Module

The term feature corresponds to a unique attribute or a facet that can help us identify objects distinctively. In this paper, the primary objective which we are concerned about is the human face. A human look is a collection of features like eyes, ears, nose, mouth, jawline, eyebrows, face tone, etc. A person can be easily distinguished from other people using these features only. Here, the system take into consideration the most common features like eyes, nose, and mouth. Depending upon the existence of these features, it can ascertain that the person is wearing a face mask or not.

To achieve these, we used the haar cascade classifiers for face detection, eye detection, nose detection, and mouth detection. The process of implementation for each of these features is explained separately.

3.4.1 Face Detection

Face detection plays a significant role in finding whether a person is wearing a mask or not. To detect a face from an image, the implementation of the classifier 'haarcascade_frontalface_default.xml' using the following algorithm is given below:

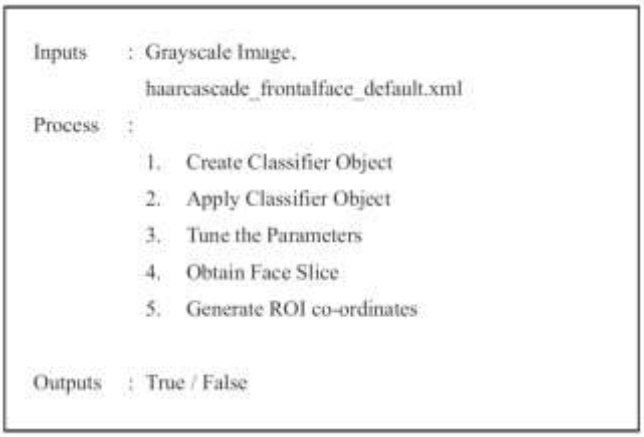

Fig 3.4.1.1 Algorithm for Face Detection

As per Fig 3.4.1.1, we fetch the grayscale image 'test.jpg' from its location, and we also load the classifier haarcascade_frontalface_default.xml. Later, we create a classifier object using cv2.CascadeClassifier() and using it, we tune the parameters like scaleFactor, minNeighbors,

minSize to obtain the face slice from the image. Once the face is detected, we store the coordinates as roi_coordinates (Region of Interest) to use these for other classifiers such as eyes, nose, and mouth. Then we either return True, in the case of the face is detected or False, elsewhere.

3.4.2 Eye Detection

Eye detection can also help us to identify whether a face has a mask or not in the absence of failed face detection. In cases where we cannot identify a face because of the mask coverage, we use eye detection to confirm whether there is a face. To detect eyes from an image, we implement the classifier 'haarcascade_eye.xml' using the following algorithm:

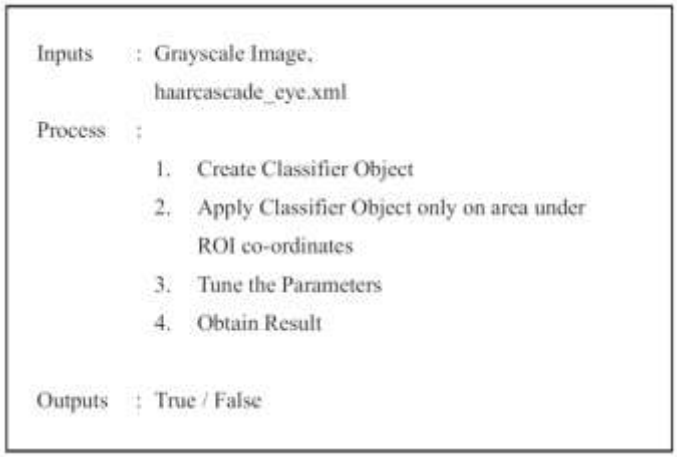

Fig 3.4.2.1 Algorithm for Eye Detection

As per Fig 3.4.2.1, we fetch the grayscale image from its location, and we also load the classifier haarcascade_eye.xml. Later we create a classifier object using cv2.CascadeClassifier() and using it, we try to obtain our image's eyes, especially under the area of the region of interest coordinates. Later we tune the parameters like scaleFactor, minNeighbors, minSize to achieve a better result. Then we either return True, in the case of eyes are detected or False, elsewhere.

3.4.3 Nose Detection

Nose detection is an essential attribute for detecting whether a face has a mask or not. If a mask covers the face, the nose is undetected. We use this same idea to detect whether a face has a mask or not. For this, we use the classifier 'haarcascade_mcs_nose.xml.' Here, we apply the following algorithm:

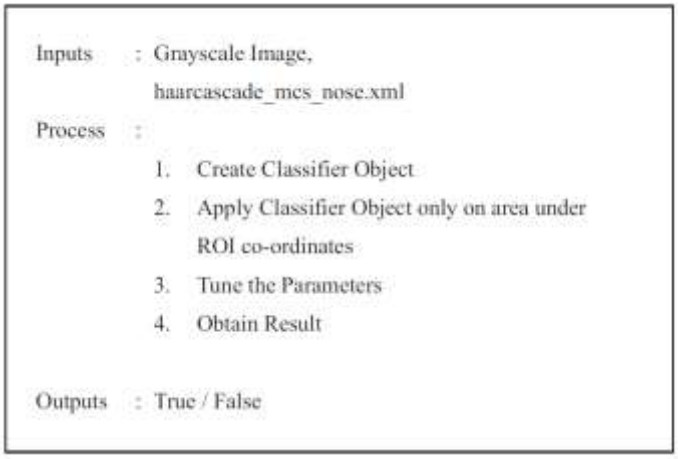

Fig 3.4.3.1 Algorithm for Nose Detection

As per Fig 3.4.3.1, we fetch the grayscale image from its location, and we also load the classifier haarcascade_mcs_nose.xml. Later we create a classifier object using cv2.CascadeClassifier() and using it, we try to obtain our image's nose, especially under the area of the region of interest coordinates. Later we tune the parameters like scaleFactor, minNeighbors, minSize to achieve a better result. Then we either return True, in the case of the nose is detected or False, elsewhere.

3.4.3 Mouth Detection

Mouth detection is an essential attribute for detecting whether a face has a mask or not. If a mask covers the face, the mouth is undetected. We use this same idea to detect whether a face has a mask or not. For this, we use the classifier 'haarcascade_mcs_mouth.xml.' Here, we apply the following algorithm:

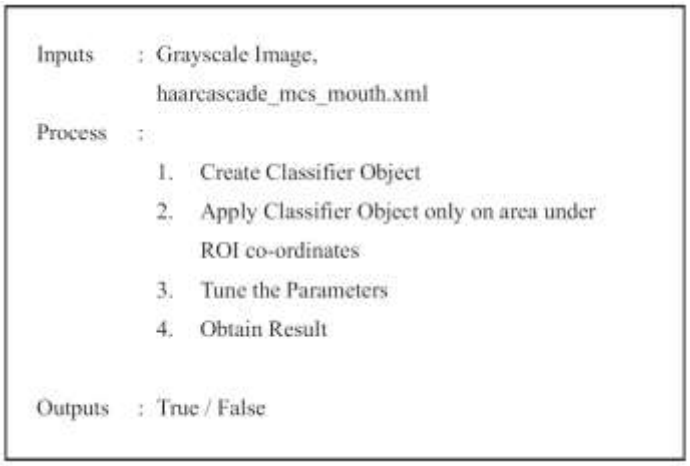

Fig 3.4.4.1 Algorithm for Mouth Detection

As per Fig 3.4.4.1, we fetch the grayscale image from its location, and we also load the classifier haarcascade_mcs_mouth.xml. Later we create a classifier object using cv2.CascadeClassifier() and using it, we try to obtain our image's mouth, especially under the

area of the region of interest coordinates. Later we tune the parameters like scaleFactor, minNeighbors, minSize to achieve a better result. Then we either return True, in the case of the mouth is detected or False, elsewhere.

After detecting these features from the face, the image would be like, as shown in Fig 3.4.4.2

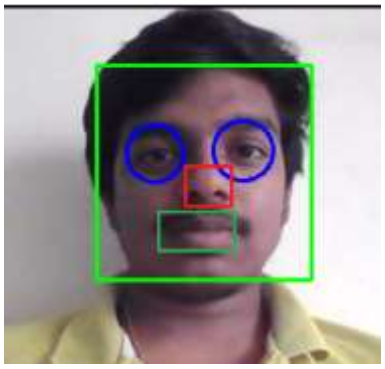

Fig 3.4.4.2 Image detecting Face, Eyes, Nose and Mouth

3.5 Apply Decision Logic

After the completion of Feature Detection module, we have the values for the questions:

1. Is face present in the image?
2. Are our eyes present in the image?
3. Is the nose present in the image?
4. Is mouth present in the image?

Based on these results obtained, we can tell whether a face is covered by a mask or not, using the following decision logic as shown in the Fig 3.5.1

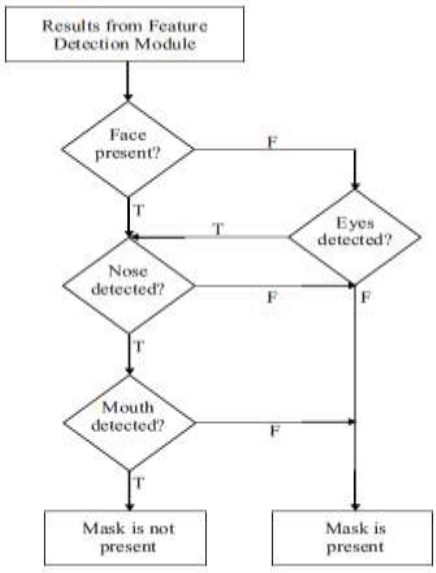

Fig 3.5.1 Decision Logic for the application

3.6 Calculate Result

Using the above decision logic and the values obtained from the Feature Detection module, we can say that

1. The face mask is not present if the face is detected, the nose is detected, and the mouth is detected.
2. The face mask is not present if the face is not detected, eyes are detected, the nose is detected, and the mouth is detected.
3. In all the remaining cases, we can say that Face mask is present.

IV. RESULTS AND DISCUSSIONS

In this paper, we successfully built the feature detection module, to find out the important attribute detection results and later applied our decision logic, to find out whether a person is wearing a mask. Here, we tabulate two different results, one with a mask and another without a mask with their corresponding features detected or not detected as follows:

Fig 4.1 Sample image test1.jpg

Fig 4.2 Sample image test2.jpg

Table 4.3 Comparison between test images

Image	Face detected	Eyes detected	Nose detected	Mouth detected	Result
test1.jpg	TRUE	TRUE	TRUE	TRUE	Without Mask
test2.jpg	FALSE	TRUE	FALSE	FALSE	With Mask

For the first image, the values returned from the Feature Detection module are as follows:

1. Face detected? = True
2. Eyes detected? = True
3. Nose detected? = True
4. Mouth detected? = True

Applying the decision logic for these values, we can say that "Mask is not present" for image 1.

Similarly, in the case of the second image, the values returned by Feature Detection module are False, True, False, False, respectively. Applying the decision logic for these values, we can say that "Mask is present" for image 2.

Similarly, for testing purposes we used Prajna Bhandary's face mask dataset which consists of 1376 images, with 690 images containing a face mask and remaining 686 images not containing a face mask.

Fig 4.4 Test Data containing Face Masks (690 images)

Fig 4.5 Test Data without Face Masks (686 images)

We performed testing of our application using the above data set as shown in figures 4.4 and 4.5 and tabulated the results as follows:

Fig 4.5 Test Data without Face Masks (686 images)

	Correctly Detected	Total Images	Accuracy
With Mask	641	690	92.8%
Without Mask	669	686	97.5%

The application returned the accuracies of 92.8 percent and 97.5 percent in the cases of detecting a face mask and not detecting a face mask respectively when applied on the dataset produced by Prajna Bhandary.

V. CONCLUSIONS AND FUTURE SCOPE

In this paper, we successfully built a Feature Detection module and also created a decision logic using the values as returned by the Feature Detection module. However, during the coronavirus pandemic, it is advised that wearing a mask properly can reduce the coronavirus transmission. Since COVID-19 is transmitted mainly through an infected person's respiratory droplets, the usage of masks on a large scale can suppress this virus from propagation. For instance, this application can be employed in a mall where the doors open automatically to detect a person wearing a mask. The future scope of the application would be using classification models in Machine Learning, along with boosting techniques and the implementation of neural networks in Deep Learning to identify whether a person is wearing a mask or not.

References

[1] Ashu Kumar, Amandeep Kaur, Munish Kumar, "Face Detection Techniques: A Review", Article in Springers on Artificial Intelligence Review-July 2018.

[2] Serign Modou Bah, Fang Ming, "An improved Face Recognition Algorithm and its application in attendance management system", published by Elsevier Inc. in December 2019.

[3] Toshan Meenpal, Ashutosh Balakrishnan, Amith Verma, "Face Mask Detection Using Semantic Segmentation", 2019 4th International Conference on Computing, Communications, and Security(ICCCS) on October 2019.

[4] S. Kumar, A. Negi, J. N. Singh, and H. Verma, "A deep learning for brain tumor mri images semantic segmentation using fcn," in 2018 4th International Conference on Computing Communication and Automation (ICCCA), Dec 2018, pp. 1–4.

[5] K. Li, G. Ding, and H. Wang, "L-fcn: A lightweight fully convolutional network for biomedical semantic segmentation," in 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Dec 2018, pp. 2363–2367.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2015.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014.

[9]T.-H.Kim,D.-C.Park,D.-M.Woo,T.Jeong,andS.-Y.Min,"Multi-class classifier-based adaboost algorithm," in Proceedings of the Second Sinoforeign-interchange Conference on Intelligent Science and Intelligent Data Engineering, ser. IScIDE'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 122–127.

[10] P. Viola and M. J. Jones, "Robust real-time face detection," Int. J. Comput. Vision, vol. 57, no. 2, pp. 137–154, May 2004.