Improvement of Haar Feature Based Face Detection in OpenCV Incorporating Human Skin Color
Mohit Admane¹, Hirendra Hajare²
M. Tech Student¹, Assistant Professor²
Department of CSE
Ballarpur Institute of Technology (BIT), Ballarpur, India

Abstract:
Face detection from a digital image or video stream is used often for various purposes. But sometimes a system detects an object or area as a face where there is no face at all. This paper presents a technique to reduce such wrong detection rate introducing human skin color (HSC) characteristic. The general property of human skin in RGB color space is that it possesses R>G>B (i.e., red values are higher than green value and green value is higher than blue). In this study, such HSC property has been incorporated with the popular Haar Feature Based Face Detection (HFFD) in OpenCV, to reduce wrong detection of faces. Proposed HFFD with HSC (HFFD-HSC) has been tested and compared with standard HFFD rigorously on large number images with single and multiple faces. Experimental results identified the effectiveness of HSC incorporation in HFFD to improve its performance reducing wrong detection of faces.

Keywords: Haar feature based face detection; OpenCV; skin color analysis; RGB;

I. INTRODUCTION

Face detection is a very important and challenging matter in the field of image processing. It is also a crucial step of face recognition. Face recognition has distinct advantages over other biometrics systems (e.g., finger print, palm print etc.) because of its non-contact process. Face images can be captured from a distance without touching the person who is being identified and the identification does not require interacting with the person. In addition, face recognition serves the crime deterrent purpose because face images that have been recorded and archived can later help to identify a person. Face detection is one of the tasks that can be done easily by human vision but very complex for a machine. Several studies are available on face detection which discusses different techniques like feature-based face detection, example-based face detection, geometric based face detection, neural network-based face detection and many more [1-3]. Now a day’s almost every smart phone uses face detection for automatic focusing on the face area when capturing a photo. The main goal of any face detection system is to achieve a very high detection rate along with low computational cost. A face detection method detects facial features and ignores anything else, such as building, trees and also bodies. However, many face detection methods fail to detect correct faces from images. David, Kriegman, and Ahuja presented a survey of face detection and presented the trends of researches in face detection [4]. In the survey authors categorized and evaluated different face detection algorithms. Some limitations of those algorithms were also discussed in a brief. A common problem of the existing methods is that they treat non facial area as a facial area. The popular Haar like feature-based face detection [9, 12] also suffers from the same problem.

II. LITERATURE REVIEW

Author K.kadir, M.k.Kamaradduin, H.Nasir, S.I.Saife, Z. A. Bakti proposed a paper in which they describes Comparative study between LBP an HAAR Like features using opencv [1].Comparison has been made between haar features and LBP in three different databases to calculate detection speed. Haar like feature are effective for frontal face detection but ideal for arbitrary or moving positions.

Author Licuimei, Qi Zhiliang, Jia Nan,Wu Jinhua Bakti proposed a paper in which they describes Haar cascade by classifier combine with additional classifiers technique for implementations[2].Viola –Jones’ detector uses Adaboost, series of nodes with each node being a definite multi-tree Ada boost classifier. The key advantage of Haar cascade classifier over others is its calculation speed.

Author by Souhail Guennouni, AliAhaitouf, Anass Mansouri proposed a paper in which they describes two algorithms; edge orientation matching algorithm and Haar like features combined with classifiers are compared[3]. The key advantage of haar cascade classifier over Edge orientation algorithm is its high detection accuracy.

Author Zheng jun,HuaJizhaq,Wang proposed a paper in which they describesColor space Model and local binary pattern. Colour space Model is uses various colour models for skin detection whereas LBP used for texture extraction[4].LBP gives simple and efficient way for face detection but takes more time than Haar cascade.

The various methods today being used for video processing are Frame differencing, Optical flow and Background subtraction. To detect the moving objects the Frame differencing method uses subtraction of successive frames. This approach is straightforward to implement and easily adaptable to dynamic environments, but it cannot always extract the complete edges of the object. Another popular technique is the optical flow method [2]. This method has two steps. First finding the image optical flow and then performing clustering process with the obtained optical flow characteristics. It performs accurately.
III. PROPOSED SYSTEM

The integral part of this research paper is a face detection method using the OpenCV library. The OpenCV was launched in 1999 by Gary Bradsky at Intel. The first problem appeared later in the year 2000. In fact, OpenCV stands for Open Source Computer Vision Library. Although it is written in optimized C/C++, it has interfaces to Python and Java as well as C++. OpenCV has an active user base around the world, and its use is growing due to the growing number of computer vision applications. Object (Face) recognition has important considerations and is one of the most promising applications in image analysis. Object detection can take into account the main part of the facial recognition operation. Computing resource based on the part of the image, the intensity of which is concentrated on the face. Object detection methods in images are complex due to changes in the face, such as posture, expression, position and orientation, skin colour, presence of glasses or facial hair, differences in camera gain, lighting conditions and image resolution, the face detection method is divided into four categories and these categories are as follows:

Figure.1. Types of Object (Face) Detection Methods

In this research, we used the appearance-based method. The appearance-based method relies on several face images used for delegate training to identify face models. An appearance-based approach is superior to other types of performance. In general, appearance-based techniques rely on statistical analysis and machine learning techniques to find relevant features of facial images. This method is also used to extract facial features. Identification or facial recognition: it basically compares the input facial image with all facial images from a dataset with the aim to find the user that matches that face. It is basically a 1xN comparison.

IV. PROJECT DESIGN

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features". It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Haar Feature Based Face Detection in OpenCV

OpenCV is a very popular tool for object detection. Any types of objects including human faces can be detected by it. Currently OpenCV is using Haar feature based cascaded classifier for face detection [10]. At first the classifier is trained with a lot of positive images (the images containing particular object like car or face we are interested to detect) scaled to same size say 20x20 resolution. And then the classifier is trained with some negative images (arbitrary images that does not contain that particular object like car or face) of same size. After completion of the training process the classifier capture frequently happening features or pattern throughout the whole training images. Then the classifier can be applied to the region of interest to detect that particular object. In the detecting process the classifier finds those features throughout the region of interest and returns the coordinates of that particular object in the input region. Otherwise it does not return any coordinate. OpenCV contains many pre-trained classifier for detecting face, eye, upper body, smile, nose etc. Following subsections briefly explains HAAR feature selection and detecting process of cascade classifier those are used by OpenCV for face detection.

Haar-Cascade Detection in OpenCV

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc. Those XML files are stored in the opencv/data/haarcascades/ folder. Let's create a face and eye detector with OpenCV.

OpenCV

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

OpenCV-Python

Python is a general-purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

Haar features: OpenCV's algorithm is currently using the following Haar-like features which are the input to the basic
classifiers:
1. Edge features
(a) (b) (c) (d)
2. Line features
(a) (b) (c) (d) (e) (f) (g) (h)
3. Center-surround features
(a) (b)

Opencv’s Pre-Trained Classifiers
OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder:

```
~/opencv/opencv/data/haarcascades
haarcascade_eye.xml
haarcascade_mcs_lefteye.xml
haarcascade_frontalface_alt2.xml
haarcascade_mcs_mouth.xml
haarcascade_frontalface_alt.xml
haarcascade_mcs_rightear.xml
haarcascade_frontalface_default.xml
haarcascade_mcs_righteye.xml
```

Opencv’s Face Detection: Let’s load the required XML classifiers.

```python
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
```

Then, we need to load input image in grayscale mode:

```python
img = cv2.imread('X11es4.jpg')
grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

We use `cv2.CascadeClassifier.detectMultiScale()` to find faces or eyes, and it is defined like this:

```python
cv2.CascadeClassifier.detectMultiScale(image, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]])
```

Where the parameters are:

**image**: Matrix of the type CV_8U containing an image where objects are detected.

**scaleFactor**: Parameter specifying how much the image size is reduced at each image scale. This scale factor is used to create scale pyramid as shown in the picture. Suppose, the scale factor is 1.03, it means we're using a small step for resizing, i.e. reduce size by 3%. We increase the chance of a matching size with the model for detection is found, while it's expensive.

**minNeighbors**: Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality. We're using 5 in the code.

**flags**: Parameter with the same meaning for an old cascade as in the function `cv Haar Detect Objects`. It is not used for a new cascade.

**minSize**: Minimum possible object size. Objects smaller than that are ignored.

**maxSize**: Maximum possible object size. Objects larger than that are ignored.

If faces are found, it returns the positions of detected faces as `Rect(x,y,w,h)`.

```python
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
```

V. EXPERIMENTAL RESULTS

Snapshot
1. Object Detection with one person

```
```
2. Object Detection with many persons

VI. FUTURE SCOPE

In this research, we introduced the topic of human faces and dealt with the factors that affect face detection operation. We discussed the solution that improves the face segmentation precisely. Here using OpenCV as it can large training data, if want to get better results, we can collect a large number of face images and nose images, and train them which can improve the accuracy of the face location.
VII. CONCLUSION

Object (Face) detection technique using OpenCV is developed in Python’s programming language in Anaconda software platform (or IDLE) running on Windows 10 operating system, is created using Numpy, functions, Matplotlib and OpenCV libraries. This application detects the faces and gives the information about the student whether she/he is present in the classroom and also gives the faces of all those who try to mischievous behaviour or tries to break the rule in classroom.

VIII. REFERENCES


