INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 5, Issue 4)
Available online at: www.ijariit.com

Face detection method using OpenCV

Shubham Singh
shubhamsingh4874@outlook.com
Shri Ramdeobaba College of Engineering and Management, Nagpur, Maharashtra

Rainy Jain
jainrainy7044@gmail.com
Shri Ramdeobaba College of Engineering and Management, Nagpur, Maharasht

ABSTRACT

Currently, the OpenCV function library is being used and becoming more common and used in digital images processing to solve some image processing problems which can improve image processing efficiency. It is a computer vision method that helps to find and visualize the faces in digital images view. This method is a special use in the case for object detection methods that process instances of semantic objects that detect certain classes (for example, people, buildings, or cars, etc.) in digital images and video. This technology plays an important role in Improve the recognition rate of face recognition systems in noise environment and curse Communicate with others. In this research paper, we discussed the use of face detection method in classrooms for attendance as well as for safety, security reasons and method of the detection faces which can adopt the relative positions of every specific face present in the classroom. The experiments show that this method can quickly and efficiently extract the face region, and improve the face segmentation precisely.

Keywords— *OpenCV, Image processing, Haar feature-based cascade classifiers, Local Binary Pattern (LBP), Face detection*

1. INTRODUCTION

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. Face recognition has important considerations and is one of the most promising applications in image analysis. Face 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. Face 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:

(a) The knowledge-based method
(b) The feature-based method
(c) Template Matching method
(d) The appearance-based method.

In this research paper, 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. This paper is divided into five sections. Section 2 describes the study related to the research. Section 3 describes the methodology, Section 4 presents result, discussion, and section 5 draws the conclusion of this study.

2. RELATED WORK

The work is done for this research paper:

(a) All related libraries used in face detection where studied.
(b) For each student, around 10 pictures were grouped in folders.
(c) The Haar feature-based cascade classifiers are used and studied carefully.
(d) The Local Binary Pattern Histogram algorithm is used for the learning process in face detection.

2.1 Local Binary Patterns Histograms (LBPH)

A Local Binary Pattern (LBP) is a simple but very effective texture operator that identifies image pixels by holding back the vicinity of each pixel and considers the result as a binary number. Using the LBP combined with histograms we can represent the face images with a simple data vector. As LBP is a visual descriptor it can also be used for face recognition tasks. The LBPH uses 4 parameters:

- **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- **Neighbours**: the number of sample points to build the circular local binary pattern.
- **Grid X**: the number of cells in the horizontal direction. It is usually set to 8.
- **Grid Y**: the number of cells in the vertical direction. It is usually set to 8.

Training of algorithm: First, we have the training algorithm. To do this, we need to use a dataset to identify images of the face of the people we want. We need to set an ID for each image, so the algorithm will use this information to identify an input image and to give you an output. Images of the same person must have the same ID. With the training set already built, we see LBPH computer degrees. We have an image in grayscale, each histogram (from each grid) will contain only n positions representing the occurrences of each pixel intensity. Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 4 x4 grids, we will have 4x4xn=16n positions in the final histogram. The final histogram represents the characteristics of the image original image.

To find the best image for the input image we find the distance between any two histograms and the closet distance between histogram is selected as the best match for the input image. We can use various approaches to calculate the distance between two histograms. For example Euclidean distance and chi-square etc. In this research paper, we used the Euclidean distance based on the following formula.

$$D = \sqrt{\sum_{i=1}^{n}(x_1 - x_2)^2}$$

The LBPH is provided by the OpenCV library. The OpenCV library can be used by many programming languages (e.g. C++, Python, and Ruby).

2.2 The Haar Cascades techniques

A series of re-scaled "square shape" functions that form a harmony or foundation family. It is based on Haar Wavelet's technique for analyzing image pixels in function squares. This uses machine learning techniques to obtain a high level of accuracy from what is known as "training data". This uses the concepts of "true image" to calculate the "features" detected. Haar Cascades uses the AdaBoost learning algorithm which selects a small number of important elements from a large series to give the efficient outcome of classifications. For detail, explanation refers to research [2].

3. METHODOLOGY

A very basic method is used in this research paper,
This is the specific code which is used,

```
import cv2,numpy as np, os
subjects = ["","Rainy","Chris","Hemsworth","Scarlett","Robert","Other"]
def detect_face(img):
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
   face_cascade = cv2.CascadeClassifier('C:\opencv\sources\data\haarcascades\haarcascade_frontalface_default.xml')
   faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.035,minNeighbors = 5,minSize = (1000,1000))
   if faces == ():
      faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.035,minNeighbors = 5,minSize =( 600,600)}
                        if faces == ():
                           faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.035,minNeighbors = 5,minSize = (200,200))
                           if faces == ():
                              faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.035,minNeighbors = 5,minSize = (100,100))
                              if faces == ():
                                 faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.035,minNeighbors = 5)
   if (len(faces) == 0):
      return None, None
   (x, y, w, h) = faces[0]
   return gray[y:y+w, x:x+h], faces
def prepare_training_data(data_folder_path):
   dirs = os.listdir(data_folder_path)
   faces = []
   labels = []
   count = []
   for dir_name in dirs:
      if not dir_name.startswith("s"):
         continue;
      label = int(dir_name.replace("s", ""))
      subject_dir_path = data_folder_path + "/" + dir_name
      subject_images_names = os.listdir(subject_dir_path)
```



```
        i = 0
        for image_name in subject_images_names:
            if image_name.startswith("."):
                continue;
            image_path = subject_dir_path + "/" + image_name
            i += 1
            image = cv2.imread(image_path)
            face, rect = detect_face(image)
            if face is not None:
                faces.append(face)
                labels.append(label)
            cv2.destroyAllWindows()
        count.append(i)
    cv2.waitKey(1)
    cv2.destroyAllWindows()
    return faces, labels,count
print("Preparing data...")
faces, labels, count = prepare_training_data("D:\Training data")
print("Data prepared")
print("Total faces: ", len(faces))
print("Total labels: ", len(labels))
face_recognizer = cv2.createLBPHFaceRecognizer()
face_recognizer.train(faces, np.array(labels))
def predict(test_img):
    img = test_img.copy()
    face, rect = detect_face(img)
    label = face_recognizer.predict(face)
    t = label[0]
    label_text = subjects[t]
    print (label_text)
    face, rect = detect_face(img)
    labels.append(t)
    faces.append(face)
    count1 = count[t - 1]
    path = "D:/Training data/s" + str(t) + "/" + str(label_text) + str(count1 + 1) + ".jpg"
    cv2.imwrite(path,img)
    return t
var = 1
while (var == 1):
    x = raw_input("Press the Spacebar : ")
    if (ord(x) == 32):
        video = cv2.VideoCapture(0)
        check, test_img1 = video.read()
        video.release()
        t = predict(test_img1)
        predicted_img1 = cv2.resize(test_img1, (381,291))
        cv2.imshow("PIC", predicted_img1)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        face_recognizer.train(faces, np.array(labels))
        count[t - 1] += 1
        var = input("Press 1 to continue\nPress 2 to stop\n")
```

Only an image is a standard Numpy array containing pixels of data points. A number of pixels in an image, it is best to solve. You can think of pixels being small blocks of information organized in the form of a 2 D grid, and a pixel depth refers to the color information contained therein. In order to be processed by computer, the image must be converted to binary form. What we get as an output is slightly different in color. A bright colored image was expected but what we find is an element of bluish bias. This is because OpenCV and matplotlib have different basic color orders. While OpenCV reads images in BGR form, matplotlib, on the other hand, continues RGB sequence. Therefore, when we read a file through OpenCV, we will read it as if there were ways in blue, green and red. However, when we display the image by using a matplotlib, the channel is red and blue exchanged.

4. RESULT

Face detection technique using OpenCV is developed in Python's programming language in Anaconda software platform running on Windows operating system 10.1, 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 do mischievous behaviour or tries to break the rule in classroom.

5. CONCLUSION AND FUTURE WORKS

In this paper, 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.

6. REFERENCES

[1] Yang Pingxian, Guo Rong, Guo Peng, Fang Zhaoju "Research on lip detection based on OpenCV". Available: - 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE) December 16-18, Changchun, China

[2] Y ue Yaru, Zhu Jialin "Algorithm of Fingerprint Extraction and Implementation Based on OpenCV". Available: - 2017 2nd International Conference on Image, Vision and Computing

[3] Aftab Ahmed, Jiandong Guo, Fayaz Ali, Farha Deeba, Awais Ahmed "LBPH based improved face recognition at low resolution". Available: - IEEE

[4] Priyanka Goel, Suneeta Agarwal"Hybrid Approach of Haar Cascade Classifiers and Geometrical Properties of Facial Features Applied to Illumination Invariant Gender Classification System". Available:-2012 International Conference on Computing Sciences

