In-depth Learning of Face Recognition Based on Python

Bojun Zhu

Zhengzhou University of Industry Technology, Department of Information Engineering, Zhengzhou, China

Houyanyang521@sina.com

Keywords: Face recognition; Python; OpenCV

Abstract: Thanks to the progress of computer hardware technology and the improvement of software algorithm, artificial intelligence technology has been developed unprecedented in recent years. Face recognition technology based on in-depth learning has been widely used, especially in security. Finance and other areas requiring a higher safety factor. A face recognition tracking method based on machine learning and implemented by Python programming language and third-party OpenCV library is proposed.

1. Introduction of Face Recognition

The process of face recognition through biological features is called face recognition[1], which is a research problem in the field of computer vision. In the 1960s, Bradsoe and Chen designed a face recognition system that was representative of the origin of face recognition research. Now the issue of strengthening anti-terrorism security has become the consensus of governments, especially in the United States. This global security issue has become more prominent since the events of 9.11. Since the year of 2010, ICAO has required its member states and regions to use machine-readable passports, in which face recognition is its first model[2].From the use of face recognition system in Beijing Olympic Games to the application of railway identification system today, it marks the large-scale use of face recognition system in China. Compared with other biometric features, such as iris and fingerprint, face recognition technology has the advantage of information acquisition and verification. With the maturity of technology, face recognition has been widely used in various fields.

2. The Process of Face Recognition

Observation of a person's face can be direct, natural access to a person's many news, such as: age, gender, mood, etc. Although the algorithms used in different face detection systems differ in principle, most of them can be summed up in the following steps:

(1) Face detection: The face is detected from the image information and the location range is returned[3].

(2) Face normalization: correction of face changes due to light intensity, image pixels, etc.

(3) the extraction of facial features: the recognition of the face feature extraction[4].

(4) Face recognition: The extracted features are matched with the existing features in the database and face recognition is carried out.

This article uses a method of face recognition based on depth learning and using Python programming languages and OpenCV third-party libraries.

3. Process of Realization

3.1. Introduction of OpenCV Library

The use of Python is highly efficient. Thanks to its rich third-party libraries, Python can connect quickly and loosely blocks of its other language. The OpenCV used in this article is a third-party visual library that can be invoked through the Python language. OpenCV was founded by Intel in

  DOI: 10.25236/isrme.2019.039

1999 and has become a powerful and universal library of image and vision in the field of computer vision[5]. In this paper, we use CascadeClassifie in the CV2 library in OpenCV to check and measure the human face, and make use of LBPHFaceRecognizer, LBPH (Local Binary Pattern Histogram) face.

The recognizer performs face matching.

3.2. Principle of Face Detection

OpenCV provides three face recognizers, which are PCA-based feature discriminator, Fisherface classifier based on LDA and LBPH classifier used in this paper. Compared with the other two classification algorithms, LBPH has the advantage of being less influenced by the light source. In this method, the eigenvalue is obtained by detecting the local information of the image, and the characteristic information is obtained by comparing the gray value between each pixel and the adjacent pixel in the image[6]. The RGB image is converted to a matrix of 3x3 after it is converted to a grayscale image. When the grayscale value is larger than the central image, it is represented by 0. When it is less than the central image, it is represented by 1. In order to get a binary list of 0 and 1. After processing each pixel in the image and converting it into decimal, the histogram is a feature histogram[7].

3.3. Method Realization

Before implementing the method, it needs to configure the Python development environment and the OpenCV library (this article uses version 3.1) and download the training set that CascadeClassie needs to categorize. The realization process of this article mainly has the following steps:

(1) Collecting the face information of the subject under test;

(2) Process the traversal image by gray level;

(3) Identify the face area of the image for interception;

(4) Each pixel of the intercepted region generates a binary matrix;

(5) Generate the feature histogram of the image;

(6) Label matching of the face of the subject after LBPH conversion.

The steps of the program are as follows: first, using the camera to pick up the image information of the object, using the cascadeclassier classifier to detect each frame in the video. After recognizing the face, the image will be labeled as a training set to save. The image was analyzed by LPBH algorithm, and the characteristic histogram was extracted. By using the camera to capture the image, the face information of each frame of the video is analyzed, then the feature histogram is generated and compared with the training set. The following are the main functions used to implement this method.

1) The function of this function is to recognize the effective area of the face.

```
def function 1 (img):
gray = cv2.CvtColor (img, cv2.COLOR _ BR2GRAY )
# to the image for grayscale transformation
img_two = img
cap = cv2.CascadeClassifier (‘face_discren． xml ')
# Identify whether there are valid faces
faceRects = cap.detectMultiScale(gray, scaleFactor=1． 2)
if len (faceRects) = = 0:
if_have_face = 0
if len (faceRects):
if_have_face = 1
return img _ two, if _ have _ face
# returns the original image and the confirmation value of whether anyone has a face or not
```

2) The function is to collect the image of the object under test.

```
def function 2 (number)
# number is the number of photos taken the_wh_number =1
```

```
while (1):
ret, imga = cap.read () # reads the camera image
img_add = cv2．flip (imga,1) # to flip the image
original_img, if_face=ceshiceshi．function1(img_add)
if if_face == 1:
# Decide if the camera recognizes a face Store the image after the current time
the_now_time=time.strftime("%Y-%m-%d-%H: %M: %S″,time.localtime())
the_picture_name = str(the_now_time)
old_filname = the_picture_name + ′．jpg′
filname = old_filname．replace(″:″, ″ ″)
cv2．imwrite(filname,imga)
the_wh_number +=1
if the_wh_number == number:
break
```

3) The image collected by the function is traversed to mark the photo from each subject, and take out the image region of the face part as a return value and transfer to the LBPH classifier.

```
def function 3 (img):
gray=cv2.CvtColor (img, cv2.COLOR_BGR2GRAY)
face_cascade=cv2.CascadeClassifier(′ face discren．xml ')
face_data =face_cascade.detect MultiScale (gray, scaleFactor =1.2)
(x, y, w, h) = face_ data[0]
return gray[y:y + w, x:x +h], faces_data[0],1
```

4) The main procedure is as follows:

```
while(1):
ret, imga = cap.read()
face, rect, c = function 3 (imga)
a, b, = face, rect
label = face_recognizer．predict(face)
label_text = subjects[label[0]]
if c!=0:
draw_rectangle(imga, rect)
draw_text(imga, label_text, rect[0], rect[1]-5)
cv2．imshow ('image', imga)
```

3.4. Methods of argumentation

In the form of recording 20 photos per person of 5 subjects,10 sets of recognition results and acquaintance degree were randomly selected and put into Table 1.

Table 1 10 recognition results and similarity of random selection

label	5	5	5	5	5	5	5	5	5	5
degree of similarity	40.26	41.22	44.19	46.94	44.77	39.78	38.82	45.89	39.19	44.06

After testing, the method can correctly identify and track specific faces.

4. Conclusion

This paper introduces a method of face recognition and tracking based on in-depth learning, and provides a simple and easy solution for the related demand field. In this method, there is still a problem of low detection success rate in dynamic detection of face area, which requires more data set to learn, and the relevant algorithm is improved.

References

[1] Sunan, Wu Bing, Xu Wei, etc. Human face recognition different syncretic technique hair exhibition [J].A Study on the Complete Technology of Sanctuary,2016 (1):33-38

[2] A Python Security Analysis Framework in Integrity Verification and Vulnerability Detection[J]. PENG Shuanghe, LIU Peiyao, HAN Jing. Wuhan University Journal of Natural Sciences.

[3] Li Bing, Wu Song, Zeng Fantao. The implementation of face recognition in smartphones[J]. Computer Engineering,2017 (09):10-11.

[4] Li Wei. Face recognition algorithm implemented on a smartphone[J]. Journal of Lanzhou Institute of Technology,2016 (08):11-17

[5] Danelljan M, Hager G, Khan F S, et al. Accurate Scale Estimation for Robust Visual Tracking[C]. british machine vision conference, 2014.

[6] Feng X, Mei W, Hu D. A Review of Visual Tracking with Deep Learning[C].International Conference on Artificial Intelligence and Industrial Engineering. 2016.

[7] Zhou X, Xie L, Zhang P, et al. An ensemble of deep neural networks for object tracking[C]//IEEE International Conference on Image Processing. IEEE, 2015:843-847.