International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Face Detection and Recognition using Python Language

Rishabh Kalra, Jaspreet Singh
Ms. Gurpreet Kaur, Assistant Professor
Department of Computer Science
Mahavir Swami Institute of Technology, Jagdishpur, Haryana, India
Affiliated to Guru Gobind Singh Indraprastha University, New Delhi, India

ABSTRACT

Python is changing into a well-liked programming language. it's a free, high-quality language with a flat learning curve. it's a large assortment of free libraries. during this paper computer vision libraries are mentioned first. Then the face detection and face recognition capabilities available within the libraries are analyzed. A basic description of the algorithm employed in the library is provided. for every major step is given an example of the ensuing image. although 2 sample pictures are provided on paper, the algorithm was analyzed in most images. The analysis confirmed that Python is a tool of choice for face recognition and recognition functions.

KEYWORDS

Python, Image Processing, OpenCV, Face Detection, Face Recognition

1 INTRODUCTION

Python is a masterpiece of standard programming developed by Guido van Rossum in 1991. It has a design philosophy that emphasizes the readability of the code. It supports a wide range of program integration, priority, practical and process integration paradigms and has a comprehensive library. The first release was followed by Python 2.0 in 2000 and Python 3.0 in 2008. At the time of writing this latest version is Python 3.7. Python is a good choice for all researchers in the scientific community because [1]:

- Free and open source
- a scripting language, meaning that it is interpreted
- a modern language (object oriented, exception handling, dynamic typing etc.)
- short, easy to read and fast to learn
- full of freely available libraries, especially scientific libraries (linear algebra, visualization tools, plotting, image analysis, differential equations solving, symbolic computations, statistics etc.)
- useful for a wide range of applications: computer science, text, web sites, text classification, etc.
- widely used in industrial applications

Compared to other programming languages such as C / C ++, Java, and Fortran, Python is a higher-level language. So, the computation takes typically a little longer time, but it is very easy to set it up. In the case of C and Fortran, wrappers are also available. PHP and Ruby on the other hand are high-level languages as well. Ruby can be compared to Python but has no scientific libraries. PHP on the other hand is a very web-based language. Python can also be compared to MATLAB, which has a very extensive scientific library. However, it is not open source and free. Scilab and Octave are open-source environments such as MATLAB. Their language features however are lower than those found in Python. People often think that complex problems require complex processes in order to produce complex solutions. Python was developed with

the exact opposite philosophy. It has a very flat learning curve and software development process for developers [4]. Used for program management activities, by NASA both in development and as a writing language in a few of its programs, Industrial Light & Magic uses Python in its production of special film effects for a large budget feature, Yahoo! uses it (among other things) to manage its discussion groups and Google uses it to make the most of its web search and search engine [3]. Since Python is also an easy-to-learn language that is both powerful and relevant from the beginning [2], we may soon ask, who does not use it. As already mentioned, Python has a wide range of libraries that can be imported into a project for specific tasks. Not to be overlooked in any mathematical science paper NumPy and SciPy. NumPy is a library that provides support for large, multi-dimensional collections. Since images are actually two large (grayscale) or three (color) three dimensions, this library is essential for all image processing activities. It should also be noted that many other libraries (not just image processing) use NumPy list representation. SciPy is a library built on the same NumPy object and contains signal and image processing modules, line algebra, Fourier transform instant, etc. The last library mentioned in this introductory section is Matplotlib. As the name implies, this library is an editing library. Although it is widely used in all scientific fields, photographic processing relies heavily on it. This paper discusses libraries and their strengths in the field of image processing, image analysis and general computer perspective. The implementation of some of the most common field algorithms is also displayed along with the resulting images.

2 PYTHON'S IMAGE PROCESSING LIBRARIES

- There are several Python libraries related to image processing and computer vision. What will be presented in this paper are:
- PIL/Pillow

  This library is best suited for image manipulation (rotation, resize, etc.) and very basic image analysis. (For example, histogram)

- SimpleCV

  Its library is intended (as the name suggests) to be a simplified version of OpenCV. It does not offer all OpenCV opportunities, but it is easy to read and use.

- OpenCV

  It is a very powerful and widely used computer library. It is written in C / C ++, but Python bindings are added during installation. It also gives emphasis to real-time image processing. Among these, which will not be presented may need to be discussed in Ilastik. It is a simple, easy-to-use tool for classifying images, segmentation and analysis.

2.1 Python Imaging Library (PIL)

The Python Imaging Library (PIL) is a library written by Fredrik Lundh. Since the last issue of PIL began in 2009 it has become somewhat outdated [5]. The follower who also supports Python 3 is called the Pillow [6]. As a result, not all of them can be installed at the same time. At the time of writing, the latest version of Pillow is 5.1.0. The smallest program (pictured) can be labeled as follows:

```
from PIL import Image
img=Image.open('/image_path')
img.show()
```

The pillow is able to extract a lot of information from the image and make it possible to perform a few general image conversion procedures, including:

- manipulations per pixel,
- handling masking and transparency,
- filtering an image, such as blurring, contouring, smoothing, or finding edges,

- enhances image, such as sharpening, adjusting brightness, color or contrast.

Some of them will be displayed. An image for example can be easily rotated at a certain angle (for us 45◦) and saved with the following code:

```
image_rotated=img.rotate(45)
image_rotated.save('imagerotated.jpg')
```

The color image can also be divided into different parts (red, green and blue).

```
red,green,blue= img.split()
red.show()
```

An image can be easily sharpened or blurred. In this case it is important, however, that we submit an ImageFilter library. The required code is shown below.

```
from PIL import ImageFilter
sharp=img.filter(ImageFilter.SHARPEN)
blurring=img.filter(ImageFilter.BLUR)
```

The image can be easily cut with the following command

```
crop_img=img.crop((100,100,400,400))
```

It was shown that PIL / Pillow is much easier when only basic image processing work is required. For detailed analysis and computer perspective SimpleCV and OpenCV are very suitable.

2.2 SimpleCV

SimpleCV is an interface for open-source machine library in Python. As the name suggests is a simplified version of OpenCV. It is easy to read and use, as it is short, has a readable camera interface, and makes it possible to perform image manipulation, feature extraction, and format conversion. At the time of writing however it is still limited to Python2. Since most Python users have now transferred to Python3, SimpleCV will not be described in detail. Example given below.

```
from SimpleCV import Image image
= Image('ronaldo.jpg')
image.show()
```

As everything embedded in SimpleCV can be done in OpenCV, which is still being developed, OpenCV will be emphasized.

2.3 OpenCV

OpenCV is an open-source library of available computer labs written in C and C ++ running under Linux, Windows, Mac OS X, iOS, and Android. Links are available in Python, Java, Ruby, MATLAB, and other languages. The simplest system used to display an image can be written as follows:

```
import numpy as np
import cv2
image = cv2.imread(
'ronaldo.jpg')
cv2.imshow('image',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Since OpenCV is now the most convenient computerized library we will use it for the rest of the paper.

3 FACE DETECTION

OpenCV enables us to perform even more complex tasks with ease. There are for example processes, which get the face (eyes in the picture). The following sequence of instructions does just that.

```
Cascade_face=cv2.CascadeClassifier
('C:\\Users\\...\\Cascade_Face.xml')
cascade_eye=cv2.CascadeClassifier
('C:\\Users\\...\\Cascade_Eye.xml')
image=cv2.imread('ronaldo.jpg')
gray=cv2.cvtColor (image,
cv2.COLOR_BGR2GRAY)
faces=cascade_face.detectMultiScale
(gray, 1.3, 5)
for (x,y,w,h) in faces:
cv2.rectangle(image,(x,y),(x+w,y+h),(255
,0,0),2)
roi_gray = gray[y:y+h, x:x+w] roi_color
= image[y:y+h, x:x+w]
eyes = cascade_eye.detectMultiScale
(roi_gray)
for (ex,ey,ew,eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),
(ex+ew,ey+eh),(0,255,0),2)
```

```
cv2.imshow('img',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The result is shown in Fig. 1. For the picture in Fig. 1 algorithm works perfectly. However, if a complex image is used, the effect (especially of the eyes) is not so good. See example Figure 2. The algorithm itself uses so-called Haar feature-based cascade classifiers. It was suggested as an effective way to get something done by Paul Viola and Michael Jones [9]. The number of these features can be quite large. But most of them are not important. A good example is that the eye area is usually darker than the nose and cheeks. A second positive factor may be for example based on the fact that the eyes are usually darker than the bridge of the nose.

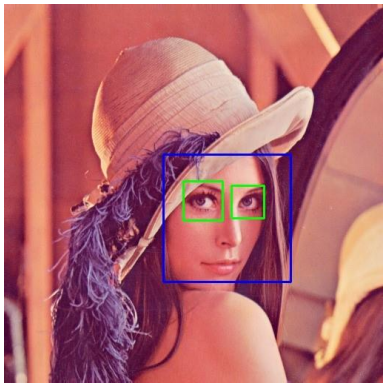

Figure 1. An example of face detection

Figure 2. An example of face detection

With a growing number of such features, we can increase the reliability of the algorithm. Classification is always true and possible. It should also be noted, that the reliability is reduced by decreasing the number of pixels in the surface area.

4 FACE RECOGNITION

The facial attention field can be defined as a study area where facial images are collected into sets that are part of a single person. Perhaps it is easier to understand if we use Facebook as an example. In the past, Facebook could see faces (see previous section), but then the user had to tag someone by clicking on the image and specifying its name. Facebook now has the ability to tag everyone in the photo automatically. This is achieved by face recognition algorithms. In Python this function can be achieved using convolutional neural networks and OpenCV. The whole system relies heavily on various libraries. Therefore, module methods, face recognition, argparse, picle and os should be included in the Python project.

Initially, some photographs of a person we wish to know should be collected. This can be done manually, or by using Microsoft's Bing API search. Ideally, a data set should contain at least 30 images per person. No other people should be present in the photos used for the training. Two sample images (one for each individual) are shown in Fig. 3.

Figure 3. Sample images of Bob Kelso and John Dorian

The network structure used for face recognition is based on the ResNet-34 neural network [10]. The Python Visual Library however has a few layers and the number of filters is reduced by half. The network was trained in a database of about 3 million images (mainly VGG data set [11] and data set of scrubs [12]). The algorithm is built in four steps:

1. Find all faces:
   In the first step we can use the face detection algorithm described in the previous section [9], which is the most widely used. The facial library however uses the most advanced History of the Oriented Gradients (HOG) method [13]. Colored images should first be converted into greyscale. Then with each pixel in the picture we look at where the image turns black. So, we get a matrix of gradients (see Fig. This matrix is largely independent of a different light from the first image. It is but much larger that it can be changed. So 16x16 size submatrices were built. Then the main direction of each submatrix is found.

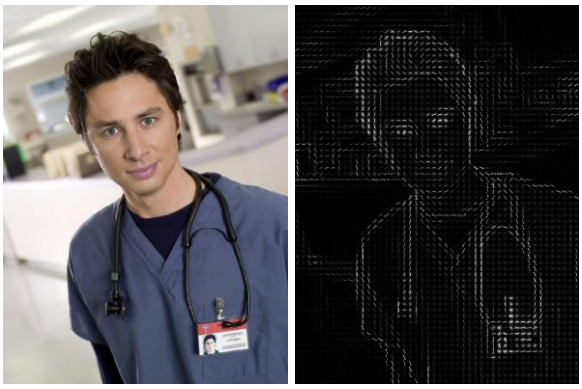

Figure 4. A sample of an original image and its histogram of gradients

2. Posing and Projecting Faces
   This step deals with the problem facing the image which may be pointing elsewhere and not directly to the camera. There are several solutions to this problem. The Python Library uses a method with 68 landmarks present on any surface [14]. An example of such a picture is shown in Fig. 5. Then the machine learning algorithm is trained to detect these 68 local symbols on any surface. The face is then altered using an affine alignment so that the eyes and mouth are as focused as possible.

Figure 5. A face with the landmarks

3. Encoding Faces
   The Deep Convolutional Neural Network is trained to produce 128 measurements per face. The training process uses three images at a time (photo of a celebrity, another photo of the same person and a photo of another person) [15]. This step requires a large database and a lot of computer power. But it should be done only once. Some pre-trained neural networks are also available online.

4. Finding a person's name in encoding
   The last step is actually very basic. The analytical face is compared to the face we have on our website. The Python Library uses a Support vector (SVM) machine to do just that. In fact, any other division algorithm can be used.

The performance of the algorithm is illustrated in the diagram shown in Fig. 6. It can be seen that it works well, despite the fact that both Bob Kelso and John Dorian do not look directly at the camera. In John Dorian's case the head is placed almost perpendicular to the camera axis. It should also be noted that the whole image is very dark and proportional. Despite this the algorithm was able to detect both completely. Another example is shown in Fig. 7). In this case there is no problem with light. John Dorian

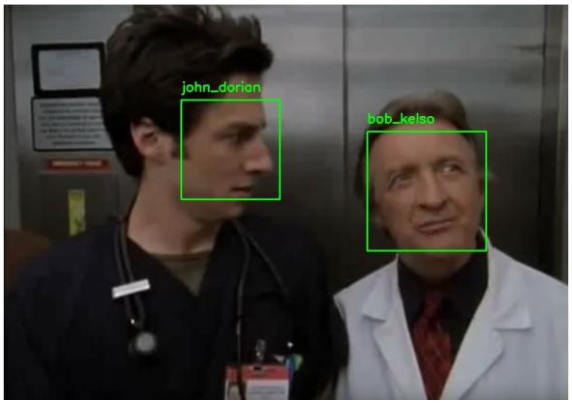

Figure 6. An example of the Face detection algorithm

is once again looking at Bob Kelso. And his emotions are reflected in his facial expressions. In the case of Bob Kelso, we see that he smokes pipe and has a strange face. Otherwise, the algorithm is able to detect both correctly.

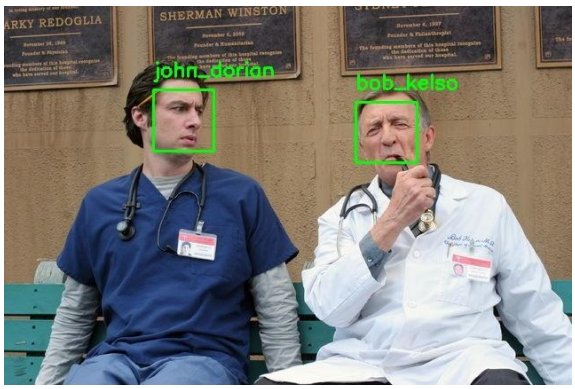

Figure 7. Another example of the Face detection algorithm

5 CONCLUSION

The paper is divided into two parts. In the first a brief overview of the most common Python libraries related to image processing and computer view is provided. The second part uses the OpenCV library. In this section libraries for face detection and face recognition are described and analyzed. Face detection and face recognition are two areas of intensive research, as they enable better interaction between computer systems or robots on the one hand and humans on the other. The Python Library is becoming the fastest and most reliable tool for face detection and face recognition. Since Python is an advanced programming language the library is well suited to be used as a face recognition (recognition) for a wide-ranging project without the need for detailed theater knowledge of the theater algorithms used. Thus, in our view it has a bright future.

In the future it would be very interesting to explore Python's opportunities and its emotional libraries. This field is the hottest topic in the study of the human machine interface. By using the results of this study, you may have significantly improved the social features of robots or software packages that the user can adapt to. It provides the most reliable answer to human machine interaction.

REFERENCES

[1] C. Fuhrer, J. E. Solem, and O. Verdier, Scientific Computing with Python 3, Packt Publishing Ltd, 2016. *(references)*

[2] S. Nagar, Introduction to Python: For Scientists and Engineers, Bookmuft, 2016.

[3] M. L. Hetland, Beginning Python: from novice to professional, 3rd Ed., Apress, 2017.

[4] R. V. Hattem, Mastering Python: master the art of writing beautiful and powerful Python by using all of the features that Python 3.5 offers, Packt Publishing, 2016.

[5] http://www.pythonware.com/products/pil/

[6] http://python-pillow.org/

[7] https://en.wikipedia.org/wiki/Python Imaging Library

[8] A. Kaehler, and G. Bradski, Learning OpenCV: computer vision in C++ with the OpenCV library, 2nd Ed.," O'Reilly, 2016.

[9] P. Viola, and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. I-511-I-518, 2001.

[10] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition," In Proceedings of the

IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016.

[11] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," In British Machine Vision Conference, vol. 1, no. 3, p. 6, September, 2015.

[12] H. W. Ng, and S. Winkler, "A data-driven approach to cleaning large face datasets," IEEE International Conference on Image Processing (ICIP), pp. 343347, 2014.

[13] N. Dalal, and B. Trigs, "Histograms of oriented gradients for human detection," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 886-893, 2005.

[14] V. Kazemi, and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1867-1874, 2014.

[15] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815-823, 2015.