European Journal of Interdisciplinary Research and Development
Volume-10 Dec. - 2022
Website: www.ejird.journalspark.org ISSN (E): 2720-5746

FACE IN PYTHON PROGRAMMING LANGUAGE DETERMINATION AND IDENTIFICATION

Daniyor Quvondikovich Ibadullayev
Senior teacher of Chirchik State Pedagogical University
Ibadullaev.d@gmail.com

ABSTRACT
Python is becoming an increasingly popular programming language. It's a free, high-level language with a very flat learning curve. It has an extensive collection of libraries that can be used for free. This article will first discuss computer vision libraries. Then the face detection and face recognition capabilities of existing libraries are analyzed. The basic description of the algorithm used in libraries is given. An example of the resulting image is provided for each key step. Although only two sample images are given in the paper, the algorithm has been analyzed on many images. The analysis confirmed that Python is indeed the tool of choice for face detection and recognition tasks.

KEYWORDS: Python, Image Processing, OpenCV, Face Detection, Face Recognition.

Python is a high level general purpose programming language created by Guido van Rossum in 1991. It has a design philosophy that focuses on code readability. It supports several programming paradigms, including object-oriented, imperative, functional, and procedural, and has a large standard and 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 article, the latest version was Python 3.10 . Python is a good choice for all researchers in the scientific community because it [1]:

- *free and open source*
- *scripting language, meaning it is interpreted*
- *modern language (object-oriented, exception handling, dynamic typing, etc.)*
- *compact, easy to read and quick to learn*
- *full of freely available libraries, especially scientific ones (linear algebra, visualization tools, graphing, image analysis, solving differential equations, symbolic calculations, statistics, etc.).*
- *useful in a wider range of settings: scientific computing, scripting, websites, text analysis, etc*
- *widely used in industrial applications*

Compared to other programming languages like C/C++, Java, and Fortran, Python is a high-level language. Therefore, the computation time is usually slightly longer, but it is much easier to program. C and Fortran example also have wrappers. On the other hand, PHP and Ruby are both high-level languages. Ruby is comparable to Python, but lacks scientific libraries. PHP, on the other hand, is a more web-oriented language. Python can be compared to Matlab which has a really extensive scientific library. However, it is not open source and free. Scilab and

Octave are open source environments similar to Matlab. However, their language features are much lower than those available in Python. Generally, people think that complex problems require complex processes to produce complex solutions. Python was developed with the exact opposite philosophy. It has a very flat learning curve and development process for software engineers [4]. It is used by NASA for system control tasks, both for development and as a scripting language for a number of its systems, Industrial Light & Magic uses Python to produce special effects for big-budget feature films, Yahoo! uses it (among other things) to manage newsgroups, and Google has used it to implement many components of its web browser and search engine [3]. Because Python is both easy to learn from scratch, powerful and convenient [2] , we may soon be asking who isn't using it. As mentioned above, Python has an extensive set of libraries that can be imported into a project to perform specific tasks. NumPy and SciPy are the ones that should definitely be mentioned in any scientific paper about mathematics. NumPy is a library that supports large, multidimensional arrays. Since images are essentially large two-dimensional (gray) or three-dimensional (color) matrices, this library is essential in all image processing tasks. It should also be noted that many other libraries (not limited to image processing) use the NumPy array representation. SciPy is a library built on the NumPy array object, with modules for signal and image processing, linear algebra, fast Fourier transforms, and more. The last library mentioned in this introduction is Matplotlib. As the name suggests, this library is a drawing library. Although it is widely used in all areas of science, image processing relies heavily on it. This article reviews the libraries and their capabilities in image processing, image analysis, and computer vision in general. The execution of the most common algorithms in this field is also demonstrated along with the captured images.

There are several Python libraries related to image processing and computer vision. The ones presented in this article are:

• *PIL/ Pillow*

This library is mainly suitable for simple image manipulation (rotate, resize, etc.) and very simple image analysis (eg histogram).

• *SimpleCV*

Its library (as the name suggests) is designed to be a simplified version of OpenCV. It doesn't offer all the features of OpenCV, but it's easier to learn and use.

• *OpenCV*

It is by far the most capable and most widely used computer vision library. Written in UC/C++, but Python bindings are added during installation. It also focuses on real-time image processing. Among those that are not presented, it is worth mentioning Ilastik. It is a simple, easy-to-use tool for classifying, segmenting and analyzing interactive images.

Python Imaging Library (PIL) is a library written by Fredrik Lund. The last edition of PIL dates back to 2009, as it is somewhat out of date [5]. A successor that also supports Python 3 is called Pillow [6]. Therefore, both cannot be installed at the same time. At the time of writing the latest version of Pillow is 5.1.0. The most trivial program (show image) can be written as:

```
from PIL import Image
im=Image.open('/path/to/the/image/')
im.show()
```

Pillow is capable of obtaining a lot of information from an image and allows you to perform several standard procedures for image manipulation, including:

- manipulations per pixel,
- work with masking and transparency,
- image filtering such as blurring, contouring, smoothing or edge detection;
- image enhancements such as sharpening, brightness, contrast or color adjustments

Some of them are shown. An image, for example, can be easily rotated to a certain angle (45◦ in our case) and then saved with the following code:

```
rotated_image=im.rotate(45)
rotated_image.save('rotated.jpg')
```

It is also possible to separate a color image into different components (red, green and blue).

```
r, g, b = im.split() r.show()
```

The image can also be easily sharpened or blurred. In this case, it is important that we also import the ImageFilter library. The required code is shown below.

```
from PIL import ImageFilter sharp=im.filter(ImageFilter.SHARPEN)
blur=im.filter(ImageFilter.BLUR)
```

For example, an image can be easily cropped with the following command:

```
cropped_im=im.crop((100,100,400,400))
```

If a basic image processing task is needed, PIL/Pillow has been shown to be very easy. For more detailed analysis and computer vision, SimpleCV and OpenCV are more suitable.

2.2 SimpleCV: SimpleCV is an interface to open source machine vision libraries in Python. As the name suggests, it is a simplified version of OpenCV. It is relatively easy to learn and use because it has a compact, camera-readable interface and allows image manipulation, feature extraction, and format conversion. At the time of writing, it is still limited to Python2. Since most Python users have already moved to Python3, SimpleCV will not be described in detail. A simple program can be seen below.

```
from SimpleCV import Image
img = Image('lena.jpg')
img.show()
```

Since everything included in SimpleCV can still be implemented in the native OpenCV, the focus is on OpenCV.

OpenCV is an open source computer vision library written in C and C++ and runs on Linux, Windows, Mac OS X, iOS and Android. Interfaces are available for Python, Java, Ruby, Matlab and other languages. A very simple program used to display an image can be written as:

```
import numpy as np import cv2
img = cv2.imread('lena-color.jpg') cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Since OpenCV is the most suitable library for computer vision today, we use it in the rest of the paper.

OpenCV allows us to perform more complex tasks relatively easily. For example, there are routines that detect face(s) (eyes in an image). The following sequence of commands will do this.

```
face_cascade=cv2.CascadeClassifier ('C:\\Users\\...\\haarcascade
_frontalface_default.xml') eye_cascade=cv2.CascadeClassifier
('C:\\Users\\...\\haarcascade_eye.xml') img=cv2.imread('lena.jpg')
gray=cv2.cvtColor
(img, cv2.COLOR_BGR2GRAY)
faces=face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w] roi_color = img[y:y+h, x:x+w]
eyes = eye_cascade.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',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The result is shown in Figure 1. For the image in Figure 1, the algorithm works perfectly.

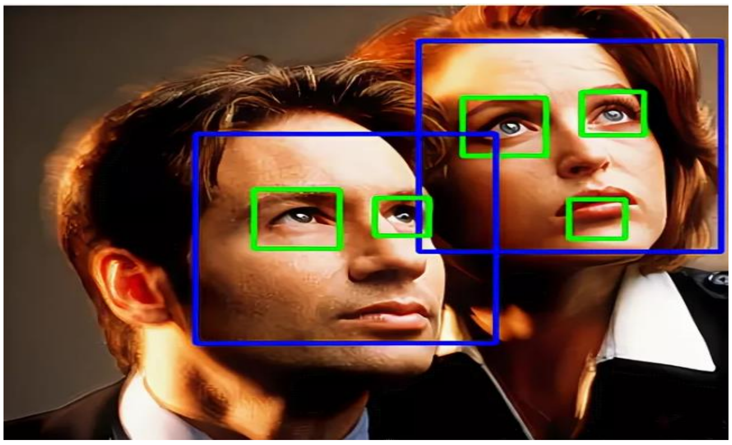
Figure 1. Face and eye detection in Python.

If a more complex image is used, the result (especially for the eyes) is not so good. For example, see Figure 2. The algorithm itself uses cascade classifiers based on the Haar feature. It was proposed by Paul Viola and Michael Jones as an effective method for object detection [9]. The number of these features can be huge. But most of them are insignificant. A good feature, for example, is that the eye area is usually darker than the nose and cheek area.

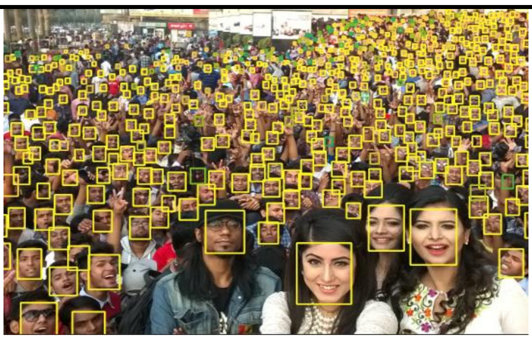

Figure 2. Face detection in Python.

This feature can be based, for example, on the fact that the eyes are usually darker than the bridge of the nose. With the increase of such features, we can increase the reliability of the algorithm. Misclassifications are, of course, always possible. It should also be noted that reliability decreases with a decrease in the number of pixels in the face area.

REFERENCES USED

1. ML Hetland, Beginning Python: from novice to professional, 3rd Ed., Apress, 2017.
2. RV 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.
3. Sultanov, RO, Yusupov, MR (2020). Problems in teaching mathematics in education and the importance of information and communication technologies in their solution. News of UzMU, 2(1/2/1), 144-147.
4. Sultanov, R. O. (2020). Methods of improving the Idea block encryption algorithm. Academic Research in Educational Sciences, 1(3), 397-404.
5. Ibadullayev, DK (2022). Research and analysis of the solution of Hamilton-Jacobi equation. Science and Education, 3(9), 8-15.
6. Khurramov, A.J., Rajabov, O.T., S ultonov, R.O., (2021). Use of animation and computer graphics in the educational process . Academic research in educational sciences, 2(11), 1382-1388.
7. Olimov IS, Karimov AA, Yoldoshev Sh.Z. (2021). Authentication and permission management in the Internet of Things. Problems of modern information, communication technologies and IT-education implementation. 3, 236-239.
8. Sultanov, RO, (2021). Issues of digital technology of education. Academic research in educational sciences, 2(CSPI conference 3), 804-807.
9. Sultanov, RO, (2022). Use of information technology in making function graph. Economics and society, 1(92), 645-650.
10. Ibadullayev, DK, Atajonov, MN, (2022). The most secure encryption algorithms. Academic research in educational sciences, 3 (4), 848-854.