International Journal of Education, Social Science & Humanities.
Finland Academic Research Science Publishers
ISSN: 2945-4492 (online) | (SJIF) = 7.502 Impact factor
Volume-11| Issue-11| 2023 Published: |22-11-2023|

OPENCV LIBRARY IN PAYTON AND COMPARATIVE ANALYSIS OF THE METHODOLOGY OF USE IN COMPUTER VISION

https://doi.org/10.5281/zenodo.10138625

D.A.Khalilov
Fergana branch of TUIT named after Muhammad al-Kharizmi, Professor of "Information Technology", Ph.D.,

Introduction

In this article, we will learn how to use the OpenCV library in Python.

OpenCV is an open-source library supporting many platforms including Windows, MacOS and Linux. This library also exists for many programming languages. But most often it is used to write machine learning applications in Python, especially in the field of computer vision.

In addition to being cross-platform and supporting many programming languages that allow applications to be used on various systems, the OpenCV library is very efficient (compared to other similar libraries) in terms of calculations, since almost all functions and operators in it are vectorized.

In this article, we will teach you how to install the OpenCV library on Windows, MacOS and Linux platforms. We will also talk about image operations, image arithmetic, image smoothing and geometric image transformations using the OpenCV library. Let's get started!

Library installation

Note: since we are considering using the OpenCV library for the Python language, it is implicitly assumed that you already have Python (version 3) installed. To install the OpenCV library, run one of the following commands, depending on your operating system.

Windows
```
$ pip install opencv-python
```
macOS
```
$ brew install opencv3 --with-contrib --with-python3
```
linux
```
$ sudo apt-get install libopencv-dev python-opencv
```
To check the installation result, simply enter the following command in the Python terminal:

International Journal of Education, Social Science & Humanities.
Finland Academic Research Science Publishers
ISSN: 2945-4492 (online) | (SJIF) = 7.502 Impact factor
Volume-11| Issue-11| 2023 Published: |22-11-2023|

```
import cv2
```

If no error message appears, then the library was installed successfully.

Basic Image Operations

Having installed OpenCV, let's now, so to speak, feel the main functionality of this library.

Screen output

The process of displaying an image on the screen consists of two steps. First, we have to load the image and then display it on the screen. These operations are performed sequentially, and a separate function is assigned to each of them.

To display an image on the screen, we need to set two things:

1. Path to the file containing the image (both relative and absolute paths will do).
2. File read mode (read only, write, etc.).

The function with which we read the image is called cv2.imread(). She has three modes of operation. The first one is IMREAD_GRAYSCALE. As the name suggests, it converts an image to black and white with grayscale. The second one is IMREAD_UNCHANGED, it loads the image without clipping the alpha channel. And the third one, used by default, is IMREAD_COLOR. It simply loads a color image using the RGB channels.

Here is an example code:

```
import cv2
my_bike = cv2.imread('bike.png')
```

Thus, we load the bike image from the bike.png file and save it to the my_bike variable for further work.

Note: If this code resulted in an error, there are only three possible reasons for this. The first is that you incorrectly set the path to the file. The second is that such a file simply does not exist, and the third is that the image type (jpg/jpeg/png) is set incorrectly.

Now let's display the image we just loaded on the screen. For this, the cv2.imshow() function is used. If you have used Matlab, you should be familiar with how it works.

```
cv2.imshow('my_bike', my_bike)
```

The first parameter of the imshow() function is the string we want to use as the caption for our image. The second parameter is a variable containing the image we uploaded.

Saving images

To save the results of our work with images in the OpenCV library, there is a cv2.imwrite() function.

Here is an example of its use:

```
cv2.imwrite('bike.png', my_bike)
```

Here we set the name of the file and the variable that contains the image. It will be saved to the current working directory.

Image arithmetic

Image arithmetic includes addition, subtraction, division, and multiplication of various images and is used to obtain a new image by arithmetic operations on input images. Image arithmetic has many practical applications, such as watermarking an image, blending two images, applying various filters to images, and so on.

Of the many possible operations, we will consider only two examples that will help us understand the concept of arithmetic operations in the OpenCV library. As the first example, we will take the addition of two images, and as the second, their blending (blending).

Let's look at the code for these examples.

Image stacking

```
import cv2
# Read two images
image_1 = cv2.imread('bike.jpg')
image_2 = cv2.imread('car.jpg')
# Sum the arrays of two images across all channels
result = cv2.add(image_1, image_2)
cv2.imshow('result', result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The waitkey command is used to stop code execution until any key on the keyboard is pressed. This is very convenient, because otherwise the image displayed on the screen would only flash on the screen for a fraction of a second, and then the program would complete its execution.

Blending Images

Blending images is very similar to adding them, except that now we can control the contribution of each of the input images to the resulting image. In general, if we want one of the input images to be more contrasty and the other to be blurrier when they are merged, we should use blending instead of image stacking.

To clarify this, let's look at the code:

```
import cv2
# Read two images
image_1 = cv2.imread('bike.jpg')
image_2 = cv2.imread('car.jpg')
result = cv2.addWeighted(image_1, 0.9, image_2, 0.1)
cv2.imshow('result', result)
cv2.waitKey(0) # Program stops until any key is pressed
cv2.destroyAllWindows()
```

The sum of the weights passed to the cv2.addWeighted() function must equal 1. You can also pass a scalar value to the function at the end, which will be added to the value of each pixel in the resulting image.

Note: images can be of any type, but the type of all images must be the same. For example, if you use the PNG format, then all images must be in this format.

Image smoothing

Image smoothing is an extremely useful operation and is very often used before passing an image to a machine learning model for processing. Basically, this should be done to filter out high-frequency noise, using a low-pass filter for this. There are many different filters, such as the averaging filter (box filter), median filter (median filter), wave type filter (mode filter), Gaussian filter and many others. But in order to understand image smoothing and its application in the OpenCV library, we will consider only the first, averaging filter (box filter).

Let's say you have a 10X10 image and want to run it through a 3X3 averaging filter. How will you act?

We'll start at the far left of our image, place a 3x3 filter there, and replace the center element with the average of all nine elements (the sum of the elements that hit the filter divided by 9). This will only be the first step. Next, we will move the filter one step to the right and repeat this procedure. This example is illustrated below.

Filter or mask:

Applying a filter to a 10X10 image:

Let's now see how we can organize image filtering using the OpenCV library. Please carefully read all the comments for each line of this code.

```
import cv2
# Load initial image
original_image = cv2.imread('my_bike.png')
```

Volume-11| Issue-11| 2023 Published: |22-11-2023|

```
# Filtering the image with a 3X3 averaging filter
average_image = cv2.blur(original_image,(3,3))
# Apply Gaussian filter to original image
gaussian_image = cv2.GaussianBlur((original_image,(3,3),0))
# Apply median filter to original image
median_image = cv2.medianBlur(original_image,3)
```

To see the result on the screen, run the following helper code:

```
import matplotlib.pyplot as plt
plt.imshow(average_image)
plt.show()
```

Image conversion.

The last, but one of the most important topics we raised in this review of the OpenCV library is image transformation. This topic is used in a variety of applications, but the problem of data augmentation for machine learning models should be mentioned separately. We are talking about situations where there is not enough data in our dataset for full-fledged training, and we, supplementing and modifying existing pictures, increase the initial dataset to the desired size. This helps us to seriously increase the accuracy of the trained model.

The list of possible transformations is very long and includes scaling, image affine transformation, rotation, transposition and much more. We will briefly cover only scaling and rotation, but the OpenCV library has support for all possible transformations. Let's start with scaling.

Scaling.

Simply put, scaling is nothing more than resizing an image, making it larger or smaller. In the OpenCV library, there is a resize function for this. This function, in turn, has three methods: INTER_CUBIC, INTER_LINEAR and INTER_AREA. Let's take a look at a specific code example and see how it all works. Please carefully study the code, comments to it and the description below.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
image = cv2.imread('my_bike.jpg')
# Zoom in/expand 2 times in width and height
result_1 = cv2.resize(image, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)
```

```
# Zoom out/shrink 2 times in width and height
result_2 = cv2.resize(image, None, fx=2, fy=2, interpolation=cv2.INTER_AREA)
# Display the resulting images
plt.imshow(result_1)
plt.imshow(result_2)
plt.show()
```

Here, in the resize function, the fx parameter determines the scale of changes in width, fy - in height, and the interpolation parameter is responsible for the method of changes itself (that is, expansion or contraction).

Rotation

Rotation allows us to move the image around a certain axis by a given angle.

Before we learn how to rotate our images using the OpenCV library, let's remember that there is a linear operator called the rotation matrix, which just performs this kind of transformation. We will not go into mathematical details, since in the OpenCV library this matrix is calculated with a single function call. You will see this in the following code:

```
import cv2
import matplotlib.pyplot as plt
# Upload the bike image
image = cv2.imread('my_bike.jpg',0)
# rows and columns
r, c = image.shape
matrix = cv2.getRotationMatrix2D((cols/2,rows/2), 180, 1)
result = cv2.warpAffine(image,matrix,(c,r))
# Display the rotated image
plt.imshow(result)
plt.show()
```

In the getRotationMatrix2D function, 180 is the angle by which our image should be rotated, 1 is the scale factor. This function returns the rotation matrix, which is stored in the matrix variable.

Next, the warpAffine function, using the rotation matrix calculated in the previous step, rotates our image in accordance with the specified specification.

Conclusions

Summing up, let's once again go through the most important places in our article. The OpenCV library is available in many languages and is often used in conjunction with the NumPy, SciPy and Matplotlib libraries, as we have seen in the examples. Some library functions are taken from Matlab and also support vector calculations, which greatly improves computational efficiency.

Also, OpenCV is one of the best libraries for computer vision and after reading this article, you will be able to find many machine learning applications that have been developed using OpenCV.

It should also be said that this article reveals only a very small part of what is in the OpenCV library. Reading it should inspire you to dive even deeper into this library and learn about the many other advanced features in it.

REFERENCES:

1. OpenCV library [Electronic resource]. – Access mode: https://opencv.org/.

2. Xalilov, D. Сунъий интеллект ва радиал нейрон тармоқларнинг математик асослари. Science and innovation, 1(A6), 664-671.

3. Khalilov, D. A., Jumaboyeva, N. A. K., & Kurbonova, T. M. K. (2021). Advantages and Applications of Neural Networks. Academic research in educational sciences, 2(2), 1153-1159.

4. Kolesnik A. V., Ladyzhensky Yu. V. Distributed face recognition system based on geometric characteristics [Electronic resource] // DonNTU Masters Portal. – Access mode: http://masters.donntu.org/2010/fknt/kolesnik/library/index.htm.

5. D.A.Khalilov Lecture notes. FF TUIT.2020.