Index to Python/OpenCV Code Snippets

1 Introduction

NOTE: This document refers to the set of provided Python/OpenCV scripts and isn't much use on its own. They have been tested with Python 3.7.4 and OpenCV 4.2.0.

This document gives a description of what each of the small Python/OpenCV scripts do. You can search this document for keywords to find which script contains the operation you are looking for.

Because of the popularity of both Python and OpenCV, there is a vast amount of information and source code available on the Internet. An Internet search will usually provide you with snippets of code to incorporate into your programs.

Python3 is required to run the supplied scripts. They can be run with a terminal in the snippet directory with a command such as:

```
python3 snippet_01.py
```

If the script requires command-line parameters, they can be added to the end of the command. Remember that pressing Tab in a bash shell completes file names, so you may only need to type `python3 sn<Tab>` to get it to complete your command.

If you are new to Python, there are lots of tutorials and books to download on the internet. We will be using Python version 3 on this module. Here is a link to one:

An introduction to Python for absolute beginners by Bob Dowling.

2 Snippet 01

Purpose: Load an image, save it in a different format and display it.

OpenCV decides how to open image files based on the file extension (.jpg, .jpeg, .png, .bmp, .tiff, .tiff, etc.). Similarly, it writes image files in the format specified by the extension the output image file is given.

Keywords: imread, imwrite, namedWindow, imshow, waitKey

3 Snippet 02

Purpose: Load an image, check that it loaded successfully and save it in greyscale.

In OpenCV, cvtColor can be used to change the colour format of an image, such as between BGR, greyscale, HSV and so on. If you know that you are only going to operate on an image in greyscale, you can save a function call and open it in the desired format using imread with the IMREAD_GRAYSCALE parameter.

Keywords: cvtColor, COLOR_BGR2GRAY, None, sys.exit

4 Snippet 03

Purpose: Load a colour image and save the three colour channels.

The default colour format in OpenCV is BGR (blue, green, red). This is the reverse of the usual RGB format used by other programs. This only matters for manipulations of the images in OpenCV, it does not affect the way the images are stored in different file formats.

Keywords: split, RGB, BGR

5 Snippet 04

Purpose: Load an image as greyscale and perform inversion and brightness and contrast adjustments.

In Python, OpenCV stores its images as numpy arrays. So normal numpy operations can be performed such as slicing and performing arithmetic operations to every pixel with one statement.

Keywords: np.astype, np.clip

6 Snippet 05

Purpose: Change image brightness with a TrackBar (or slider).

Rather than changing functions in the script or via the command-line and re-running the script each time, trackbars can be attached to windows. These allow values to be varied while the program is running and their effect can be seen immediately. A callback function is specified for trackbars and when the trackbar is moved, the callback is called with the trackbar position. The callback can contain an image operation and imshow to view the effect as the slider is moved.

This script also demonstrates how to separate your Python scripts into a main 'function' and other user-defined functions. Note that it is also possible to create classes in Python for object-orientated programming.

Keywords: __name__, __main__, IMREAD_GRAYSCALE, createTrackbar

7 Snippet 06

Purpose: Access command-line arguments.

Any parameters that are required in the Python script can be supplied on the command-line. They are accessed via the array sys.argv. The number of parameters supplied can be found with `len(sys.argv)`. Remember that the first parameter sys.argv[0] is the name of the Python script and additional parameters will start at sys.argv[1].

Keywords: sys.argv, len, range

8 Snippet 07

Purpose: Create and display an image histogram using matplotlib.

There are a number of ways to create and display image histograms. For example, the OpenCV function calcHist can be used to calculate the histogram and a function can be written to draw the histogram into an image. A nicer way is to use a graph-plotting library to draw the histogram. This gives the option of making a nicer looking graph for use in documents. To simplify things even more, it is possible to use matplotlib to calculate the histogram and draw it in one statement.

The matplotlib graphs can be saved to a file using the Save button at the top of the graph's window. If you call it something like chart.png, it will save it as an image. If you call it chart.pdf it will save it in Acrobat format which allow it to be resized nicely.

A point to note about the syntax for calcHist as in the following example:

```
hist = cv2.calcHist([img], [0], None, [256], [0, 256])
```

The square brackets around the parameters make these single entry arrays. This is required as the function expects arrays because it is able to create multiple histograms. As a result of this, the returned value is an array of histograms. The reshape function can be used to rearrange the returned value if the function is only being used for one histogram.

Keywords: calcHist, reshape, linspace, matplotlib, pyplot, plt.bar

9 Snippet 08

Purpose: Access pixels and regions (slicing).

Individual pixels can be accessed by indexing the image array, for example img[row, col]. A for loop can be used to access a region of pixels, but it is much faster to use slicing. This accesses a square of pixels with the row and column indices both in the range [2, 4], that is {2, 3, 4}: `img[2:5, 2:5]`.

Note that these return references to the pixels in the image – if you change their value, they will change in the original image. To avoid this you can make a copy of the slice: `img[2:5, 2:5].copy()`.

Keywords: for, range, copy, indexing

10 Snippet 09

Purpose: Threshold a greyscale image.

Threshold an greyscale image by specifying a threshold value and also let OpenCV choose a threshold value for you via Otsu's method.

Keywords: threshold, THRESH_BINARY, THRESH_OTSU

11 Snippet 10

Purpose: Filter a greyscale image.

Using a number of filter functions and creating a user-defined kernel.

Keywords: GaussianBlur, Sobel, Laplacian, filter2D, medianBlur, convertScaleAbs, addWeighted, np.array

12 Snippet 11

Purpose: Add noise to an image.

Adding Gaussian noise with a specified μ and σ to an image. Also corrupting an image with salt and pepper noise: sets pixels to 0 or 255, both with a probability of p.

Keywords: randn, binomial, where, reshape

13 Snippet 12

Purpose: Erode, dilate, open and close.

Creating a structuring element and applying morphological operations to a binary image.

Keywords: threshold, getStructuringElement, erode, dilate

14 Snippet 13

Purpose: Find lines and circles in an image with the Hough Transform.

Use the Canny edge detector to find clean edges in an image, then apply the Hough Transform to find lines. There are a number of parameters that need to be adjusted for these two operations.

Keywords: Canny, cvtColor, HoughLinesP, HoughCircles, line, circle, format

15 Snippet 14

Purpose: Stitch images together.

Use the OpenCV stitcher class that allows a number of images to be stitched together to create panoramas

Keywords: append, Stitcher_create, stitch

16 Snippet 15

Purpose: Write your own functions and draw on an image.

When using OpenCV in Python, we will usually use a mix of the OpenCV and numpy libraries. The key thing to remember is that any coordinates and sizes in OpenCV are ordered (x, y) and (width, height). Whereas in in numpy and in Python slicing, they are (row, column) for coordinates and (rows, columns) for the size.

Keywords: def, return, np.zeros, rectangle, circle, ellipse

17 Snippet 16

Purpose: Find image features using ORB.

You will often hear the Scale-invariant Feature Transform (SIFT) discussed for feature detection. However, this algorithm is patented and licensed so OpenCV no longer includes it as standard in the OpenCV library. It does include other feature detectors and descriptors that are free to use, such as ORB, BRISK and AKAZE.

This snippet shows the basic use of ORB as a feature detector and marks the discovered features on the input image.

Keywords: ORB_create, detectAndCompute, drawKeypoints

18 Snippet 17

Purpose: Find and match image features using ORB.

This snippet requires two input images. It finds features and their descriptors in both images and assembles the two images into one and draws lines between the features that match.

Keywords: ORB_create, detectAndCompute, BFMatcher, match, drawMatches

19 Snippet 18

Purpose: Register images using ORB features.

This snippet finds matches between features in two images and then warps one of the images so that it aligns with the other image. This could be used for object detection where one of the images is a scene and the other is an object that you want to find in the scene. This snippet will align the object image to the scene image and also draw a rectangle in the scene image showing where the object image was found.

Keywords: ORB_create, detectAndCompute, BFMatcher, match, findHomography, warpPerspective, line

20 Snippet 19

Purpose: K-means clustering.

Source: `https://docs.opencv.org/4.2.0/d1/d5c/tutorial_py_kmeans_opencv.html`

Segmentation of images is performed using k-means clustering. It performs segmentation on a colour and greyscale image into a specified number of colours or intensities.

Keywords: reshape, float32, kmeans, uint8, flatten, reshape

21 Snippet 20

Purpose: Access a webcam and display the video stream.

Gets frames from the first camera found on the computer (you can specify another if you have got more than one). It processes each frame by shuffling the colour channels before displaying the frame in a window.

Keywords: VideoCapture, read, split, merge

22 Snippet 21

Purpose: Read and write a video file.

Read frames from a video file and write them to another video file, but changing the frame rate so the video runs at half speed.

Keywords: VideoCapture, isOpened, CAP_PROP_FRAME_WIDTH, CAP_PROP_FRAME_HEIGHT, CAP_PROP_FPS, read

23 Snippet 22

Purpose: Very basic motion tracking.

Track an object moving in a video. This works by subtracting the previous frame from the current one to find out what is different. It then thresholds the result and finds the centre of mass of the white pixels, which it takes as being the moving object.

Using this method, only the object should be moving and not the background and also it can only detect one moving object. Note also that, for the example ball rolling video, the centre of mass is detected at the centre of the combination of the ball, its shadow and its motion blur. Obviously this example would not be good enough for a real world application.

Keywords: VideoWriter, VideoWriter_fourcc, abs, read, write

24 Snippet 23

Purpose: Face detection with Viola-Jones.

Using OpenCV's Haar Cascade Classifier to detect faces in images. The details for the classifier are held in an xml file which is a text file that can be viewed in an editor. This example uses the frontal face classifier, but there are others available in OpenCV such as for car number plates (see the directory opencv-4.2.0/data/haarcascades, for example, in the OpenCV source code).

Keywords: cvtColor, CascadeClassifier, detectMultiScale, rectangle, haarcascade_frontalface_default.xml, destroyAllWindows

25 Snippet 24

Purpose: Camera calibration.

When you take a phone a photo of a scene, there will be some distortion to the image. You may have seen videos from wide-angle lenses, such as on a GoPro, where the horizon is curved.

When you process images, you will normally want to undistort them so that straight lines in the scene appear as straight lines on the image.

This is normally done by taking a number of images from different angles of a chessboard image with your camera set at the same focal length as it was when you took the images.

This folder contains the chessboard image called pattern.png which is taken from the OpenCV library (opencv-4.2.0/doc/pattern.png).

Because the OpenCV calibration functions know that the lines on the chessboard should be straight, it can calculate matrices that can be used to undistort other images.

This snippet folder contains two Python scripts: one to find the camera calibration and one to undistort images.

An example of using this code

Paths will be relative to `snippet_24/basic` and I will refer to this particular calibration as canon_18mm. You will use whatever name means something to you.

Calibrate

Take a number of images of your printed out chessboard and place them in the `canon_18mm_cal` folder (say 5 or 6 images).

Run the calibration script with:

```
python3 calibrate.py 'canon_18mm_cal/*.JPG' canon18mm
```

(If it doesn't work, make sure you have used normal single quotes around the file filter.)

The `canon18mm` on the end is a name to give the calibration files. You should now have two files called `canon18mm_matrix.txt` and `canon18mm_distortion.txt`. These are plain text files with the matrices in.

Undistort

Take your distorted image and place it in `canon_18mm_test` folder (here it is called `test.jpg`).

Run the calibration script with:

```
python3 undistort.py canon_18mm_test/test.jpg canon18mm
```

The `canon18mm` on the end will be used to find the calibration files.
You should now have an undistorted image in the `canon_18mm_test` folder called `calibresult.jpg`.

NOTE: Some cameras and phones detect their own orientation and rotate the image so that it is portrait or landscape. You should ensure that all of your calibration images are in the same orientation, rotating them yourself if necessary.

NOTE: The code for this snippet is in the `basic` folder. For those who want to investigate a bit more, want to calibrate fish-eye lenses or are running into problems, see the `advanced` folder which also contains an explanatory document.

Keywords: chessboard, checkerboard, findChessboardCorners, cornerSubPix, drawChessboardCorners, calibrateCamera, savetxt, loadtxt, getOptimalNewCameraMatrix, undistort

26 Snippet 25

Purpose: Image segmentation with Mean-Shift.

This code performs image segmentation using the Mean-Shift algorithm as developed by D. Comaniciu and P. Meer in 'Mean Shift: A robust approach toward feature space analysis' and also by C. Christoudias, B. Georgescu and P. Meer in 'Synergism in low level vision.'

Christoudias and Georgescu developed the C++ library and the Python wrapper was developed by Frederic Jean (`https://github.com/fjean/pymeanshift`).

The source code is included in this snippet directory and can be compiled for different operating systems.

Keywords: pymeanshift, pms.segment