OpenCV-Python is the Python API of OpenCV.

This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

1. VideoCapture():

To capture a video, you need to create a VideoCapture object. Its argument can be either the device index or the name of a video file.Device index is just the number to specify which camera. Normally one camera will be connected. You can select the second camera by passing 1 and so on.

2. CascadeClassifier()

// for more refer http://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_objdetect/py\_face\_detection/py\_face\_detection.html#face-detection

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier.

Then we need to extract features from it.

Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it.We don’t consider remaining features on it. If it passes, apply the second stage of features and continue the process. OpenCV already contains many pre-trained classifiers.

First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

3. read()

Grabs, decodes and returns the next video frame.

4. detectMultiScale()

Detects objects of different sizes in the input image. The detected objects are returned as a Vector of rectangles where each rectangle contains the detected object.

cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors)

scaleFactor – Parameter specifying how much the image size is reduced at each image scale. helps decide what size cars to detect. In short. Your model has a fixed size defined during training. This means that this size of face is detected in the image if occuring. However, by rescaling the input image, you can resize a larger face towards a smaller one, making it detectable for the algorithm.

minNeighbors: Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality.

5. rectangle()

Draws a simple, thick, or filled up-right rectangle.

cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]])

img – Image.

pt1 – Vertex of the rectangle.

pt2 – Vertex of the rectangle opposite to pt1

6. imshow()

to update the content of the OpenCV window with a new image use the imshow function. Specify the OpenCV window name to update and the image to use during this operation:

color – Rectangle color in BGR

thickness – Thickness of lines that make up the rectangle.

Negative values, like CV\_FILLED , mean that the function has to draw a filled rectangle.

minNeighbors – Parameter specifying how many neighbors each candidate rectangle should have to retain it.

7. waitkey()

Because we want our window to be displayed until the user presses a key,

we use the waitKey function whose only parameter is just how long should it wait for a user input

(measured in milliseconds). Zero means to wait forever.