**Count number of Faces using Python – OpenCV**

Prerequisites: [Face detection using *dlib* and *openCV*](https://www.geeksforgeeks.org/opencv-facial-landmarks-and-face-detection-using-dlib-and-opencv/)

Image processing is used to detect and count the number of faces. We are not supposed to get all the features of the face. Instead, the objective is to obtain the bounding box through some methods i.e. coordinates of the face in the image, depending on different areas covered by the number of the coordinates, number faces that will be computed.

Required libraries:

* [OpenCV](https://www.geeksforgeeks.org/opencv-python-tutorial/) library in python is a computer vision library, mostly used for image processing, video processing, and analysis, facial recognition and detection, etc.
* Dlib library in python contains the pre-trained facial landmark detector, that is used to detect the (x, y) coordinates that map to facial structures on the face.
* [Numpy](https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/) is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays.

Count the number of faces:

1. Capture the frames continuously.

2. Convert the frames to grayscale (not necessary).

3. Take an iterator ***i***and initialize it to zero.

4. Each time you get the coordinates to the face structure in the frame, increment the iterator by 1.

5. Plot the box around each detected face along with its face count.

The code you provided captures video from the default camera, detects faces in real-time, and draws a rectangle around each face. Additionally, it displays the face number above each face.

**Code Explanation**

1. **Libraries Import**:
   * cv2: OpenCV library used for image processing and working with video frames.
   * numpy: A library used for numerical computations (not used explicitly here, but commonly used with OpenCV).
   * dlib: A library used for face detection, with the function get\_frontal\_face\_detector() for detecting faces.
2. **Initialize Video Capture**:
   * cap = cv2.VideoCapture(0): This opens the webcam (default device 0). If you want to use another camera, you can change the index to 1 or a different number.
3. **Face Detector**:
   * detector = dlib.get\_frontal\_face\_detector(): This initializes a pre-trained face detector from dlib.
4. **Capture Video Frames**:
   * Inside the while loop, cap.read() captures frames continuously.
   * cv2.flip(frame, 1) flips the frame horizontally to create a mirror effect.
5. **Convert Frame to Grayscale**:
   * gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY): Converts the frame from BGR to grayscale. Face detection typically works better on grayscale images.
6. **Face Detection**:
   * faces = detector(gray): This detects faces in the grayscale image.
7. **Face Coordinates**:
   * The detected faces are iterated over, and for each face, the coordinates (x, y) for the top-left corner and (x1, y1) for the bottom-right corner are obtained.
   * cv2.rectangle(frame, (x, y), (x1, y1), (0, 255, 0), 2) draws a rectangle around each detected face using the green color (0, 255, 0).
8. **Display Face Count**:
   * A counter i is used to keep track of the number of faces detected, and cv2.putText() is used to display the face number.
9. **Display Frame**:
   * cv2.imshow('frame', frame): Displays the processed video frame with detected faces.
10. **Exit Condition**:
    * The loop continues until the user presses the "q" key, which triggers the cv2.waitKey(1) & 0xFF == ord('q') check.
11. **Release and Clean Up**:
    * cap.release() stops the video capture.
    * cv2.destroyAllWindows() closes all OpenCV windows.

**Potential Issues & Solutions**

* **Face detection not working properly**: Ensure that your webcam is functional and that dlib and opencv-python are properly installed.
* **Error in face detection**: If there’s an issue with dlib (e.g., installation errors), you can try installing older versions of dlib or check system dependencies for building dlib.

**Required Libraries**

As mentioned earlier, the following libraries need to be installed:

pip install opencv-python numpy dlib

**Running the Code**

Once everything is installed and the code is running, it will continuously detect faces in the webcam feed, drawing rectangles around each face and displaying the face number on the screen. To stop the script, press the **"q"** key.