Face Detection Python

Files used in this project:

1. Haarcade_frontalface_default.xml

    - file is a pre-trained classifier file used with OpenCV, a popular computer vision library. This XML file contains data that defines the parameters of a Haar Cascade classifier specifically trained for detecting frontal faces

2. facedet.py

    - This python file contains the code for the implementation of the face detection

Module used:

1. OpenCV

    - This module provides functions and classes for various computer vision tasks such as image and video processing, object detection, machine learning,

<u>Explanation of the code:</u>

1. import cv2 :
    - Imports the OpenCV library for image and video processing.
2. cv2.CascadeClassifier('haarcascade_frontalface_default.xml:
    - Initializes a Cascade Classifier object using the Haar Cascade XML file. This file contains the trained data for detecting frontal faces.
3. **cv2.VideoCapture(0):**
    - Creates a VideoCapture object (cv2.VideoCapture) to access the default camera (index 0). This allows capturing frames from the webcam.
4. **b.read():**
    - Captures a frame (d_img) from the webcam using the VideoCapture object (b). Returns a tuple (c_rec, d_img), where c_rec is a boolean indicating success (True if frame was read successfully) and d_img is the captured frame.
5. **cv2.cvtColor(d_img, cv2.COLOR_BGR2GRAY):**
    - Converts the captured frame (d_img) from BGR (Blue-Green-Red) color space to grayscale. Grayscale conversion is typically used for face detection as it reduces computational complexity.
6. **a.detectMultiScale(e, 1.3, 6):**
    - Detects objects (faces in this case) in the grayscale image (e) using the Cascade Classifier (a). Returns a list of rectangles (f), where each rectangle represents the detected face's bounding box.
7. **cv2.rectangle(d_img, (x1, y1), (x1+w1, y1+h1), (255, 0, 0), 5):**
    - Draws a rectangle on the original color frame (d_img) around each detected face.
        - (x1, y1) and (x1+w1, y1+h1) specify the opposite corners of the rectangle.

- (255, 0, 0) represents the color of the rectangle border in BGR format (blue, green, red).
- 5 is the thickness of the rectangle border in pixels.

8. **cv2.imshow('img', d_img):**

   - Displays the image (d_img) in a window titled 'img' using cv2.imshow. This function is crucial for visualizing the detection results.

9. **cv2.waitKey(40):**

   - Waits for a key press for up to 40 milliseconds (40). If a key is pressed during this time, it returns the ASCII value of the key. If no key is pressed, it returns -1.

10. **b.release():**

   - Releases the VideoCapture object (b). This is important to free up resources associated with the webcam after the program finishes.

Output: