Exp - 8 : Smart Surveillance and Tracking

Aditya Pulikal / MSc DSAI / L022

In [1]:
```
#pip install opencv-python
```

In [2]:
```
import cv2
```

In [4]:
```
face_classifier = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
)

video_capture = cv2.VideoCapture(0)

def face_detection(vid):
    gray_image = cv2.cvtColor(vid, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray_image, 1.1, 5, minSize=(40, 40))

    for (x, y, w, h) in faces:
        cv2.rectangle(vid, (x, y), (x + w, y + h), (0,255,255), 2)
    return faces

while True:
    result, video_frame = video_capture.read()  # Read frames from the video
    if not result:
        break  # Terminate the loop if the frame is not read successfully

    faces = face_detection(video_frame)  # Apply the function to the video frame

    # Resize the frame (e.g., doubling the size)
    height, width = video_frame.shape[:2]
    new_dimensions = (width * 2, height * 2)
    resized_frame = cv2.resize(video_frame, new_dimensions, interpolation=cv2.INTER_LINEAR)

    # Display the resized frame in a window named "Face Detection Window"
    cv2.imshow("Face Detection Window (Press X to exit)", resized_frame)

    if cv2.waitKey(1) & 0xFF == ord("x"):
        break

video_capture.release()
cv2.destroyAllWindows()
```

Explanation

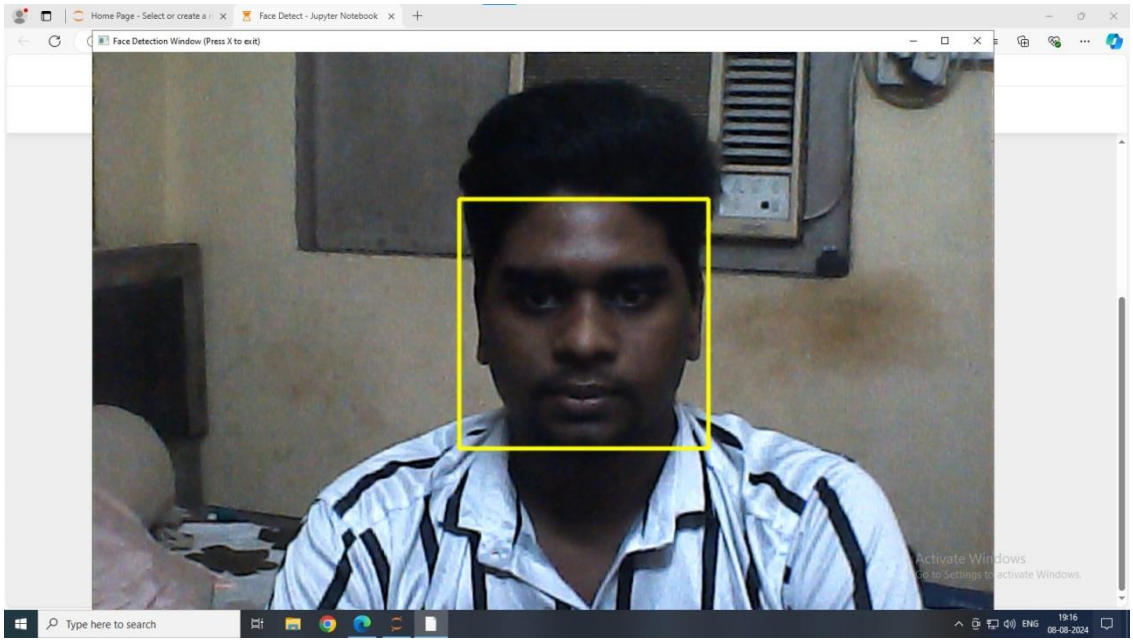
This program detects faces in real-time using our computer's camera. It starts by loading a pre-trained model, which acts like a digital template to recognize human faces.

Once the camera is activated, it captures the video feed and breaks it down into individual frames. For each frame, the image is converted to black and white, making it easier for the program to detect faces by comparing the image to the stored face pattern. If a face is found, a yellow box is drawn around it.

The program then enlarges the frame for better visibility and displays it in a window. This process repeats continuously, updating the display with each new frame. We can stop the program by pressing the "x" key, which will close the camera and the display window.

Output

The following output is generated:


Face Detection Window (Press X to exit)
Type here to search

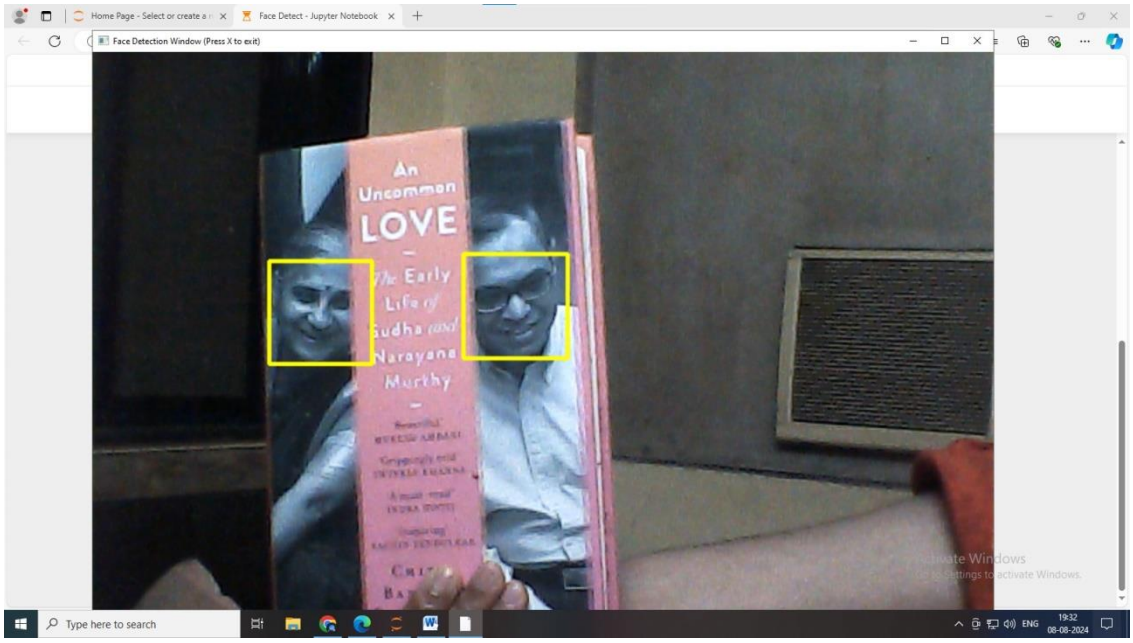

Face Detection Window (Press X to exit)
An Uncommon LOVE
Type here to search

Image & Video Analytics.

Experiment - 8.

Aditya Pulikal
MSc DSAI / L022.

A) • cv2.VideoCapture() is a class in OpenCV library that allows you to capture video from various sources. You can use it to read video files, capture video from a camera or even from a video stream.

• Syntax - cv2.VideoCapture(). / cap = cv2.VideoCapture()
0 - This opens the default camera, you can pass a diff index if you have multiple cameras.
path - pass a file path to open a video file.

• Along with cv2.VideoCapture(), we have cap.read(). This reads a frame from video source. It returns 2 values 'ret', a boolean value indicating if the frame was read correctly & 'frame', the actual frame image.

• Some of the other methods / functions used are :-

i) cap.isOpened() - checks if video capture is successfully initialized.

ii) cap.release() - releases the video capture object, freeing up resources.

iii) cap.get(propId) - retrives a property from the video capture, such as frame width, height, etc.

iv) cv2.waitKey() - waits for a specified amount of time for a key event. It is typically used in a loop where images / frames are being displayed, allowing to control display timing & response (i.e, to exit).

• The most common use cases would be :-

i) Video processing - Capture frames from a video / camera for detecting faces / objects.

ii) Real-time apps - Use it in real-time apps like video conferencing, surveillance, etc.