Detecting Face and Eyes

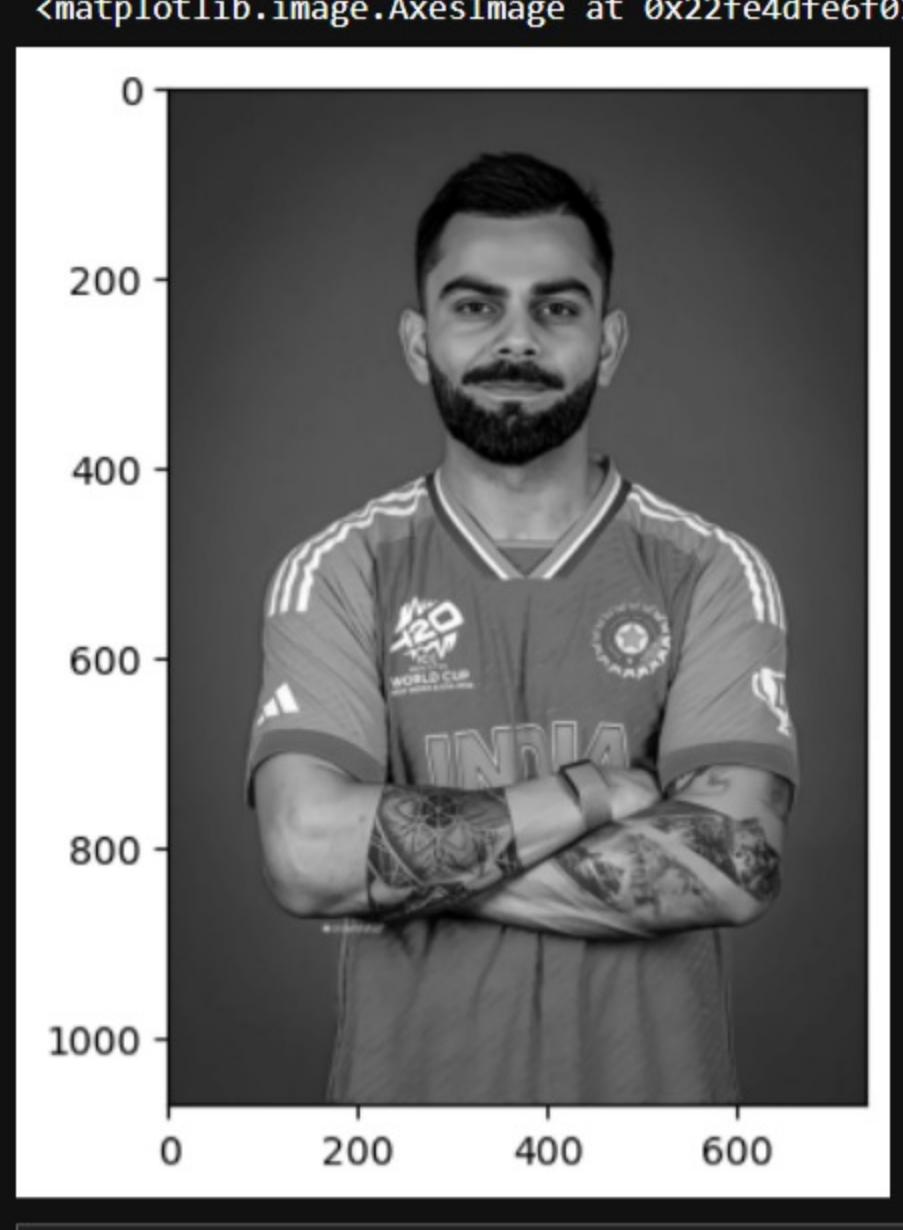
[1]:
```
import cv2
from matplotlib import pyplot as plt
```

[3]:
```
crick1 = cv2.imread('virat.jpg',1)
plt.imshow(crick1[:,:,::-1])
```

[3]:
```
<matplotlib.image.AxesImage at 0x22fe24b6870>
```

[4]:
```
import cv2
from matplotlib import pyplot as plt

crick1 = cv2.imread('virat.jpg',1)
plt.imshow(crick1[:,:,::-1])
gray_img = cv2.cvtColor(crick1, cv2.COLOR_BGR2GRAY)
plt.imshow(gray_img, cmap='gray')
```

[4]:
```
<matplotlib.image.AxesImage at 0x22fe4dfe6f0>
```



[7]:
```
haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
faces = haar_cascade.detectMultiScale(gray_img, 1.3, 5)
```

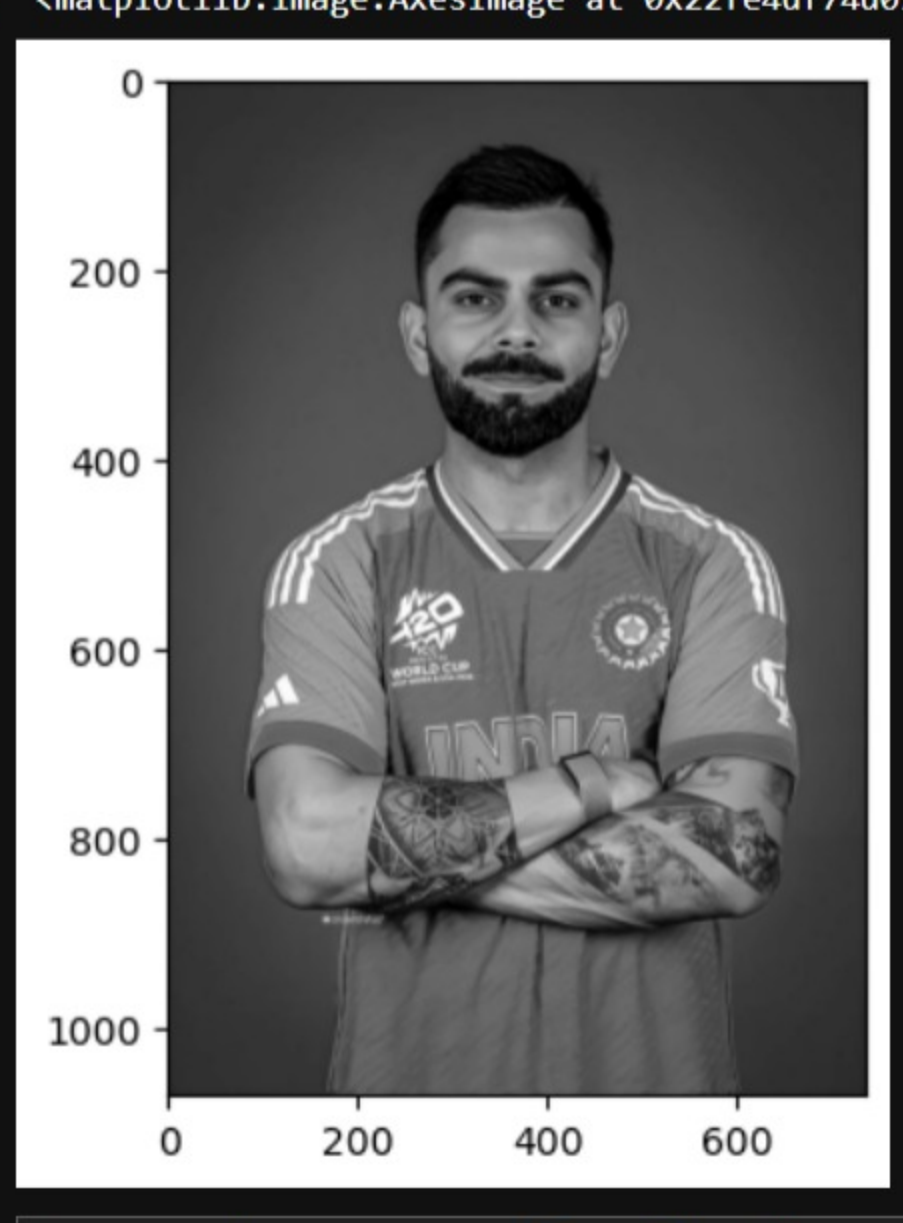
[9]:
```
crick1 = cv2.resize(crick1,(0,0),fx=0.75,fy=0.75)
```

[11]:
```
for (x, y, w, h) in faces:
    cv2.rectangle(crick1, (x, y), (x+w, y+h), (0, 255, 0), 2)

    roi_gray = gray_img[y:y+h, x:x+w]
    roi_color = crick1[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)

    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(255,0,0),2)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cv2.imshow('Detected Faces', crick1)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Detecting Face and Nose

[13]:
```
import cv2
from matplotlib import pyplot as plt

crick1 = cv2.imread('virat.jpg',1)
plt.imshow(crick1[:,:,::-1])

gray_img = cv2.cvtColor(crick1, cv2.COLOR_BGR2GRAY)
plt.imshow(gray_img, cmap='gray')
```

[13]:
```
<matplotlib.image.AxesImage at 0x22fe4df74d0>
```



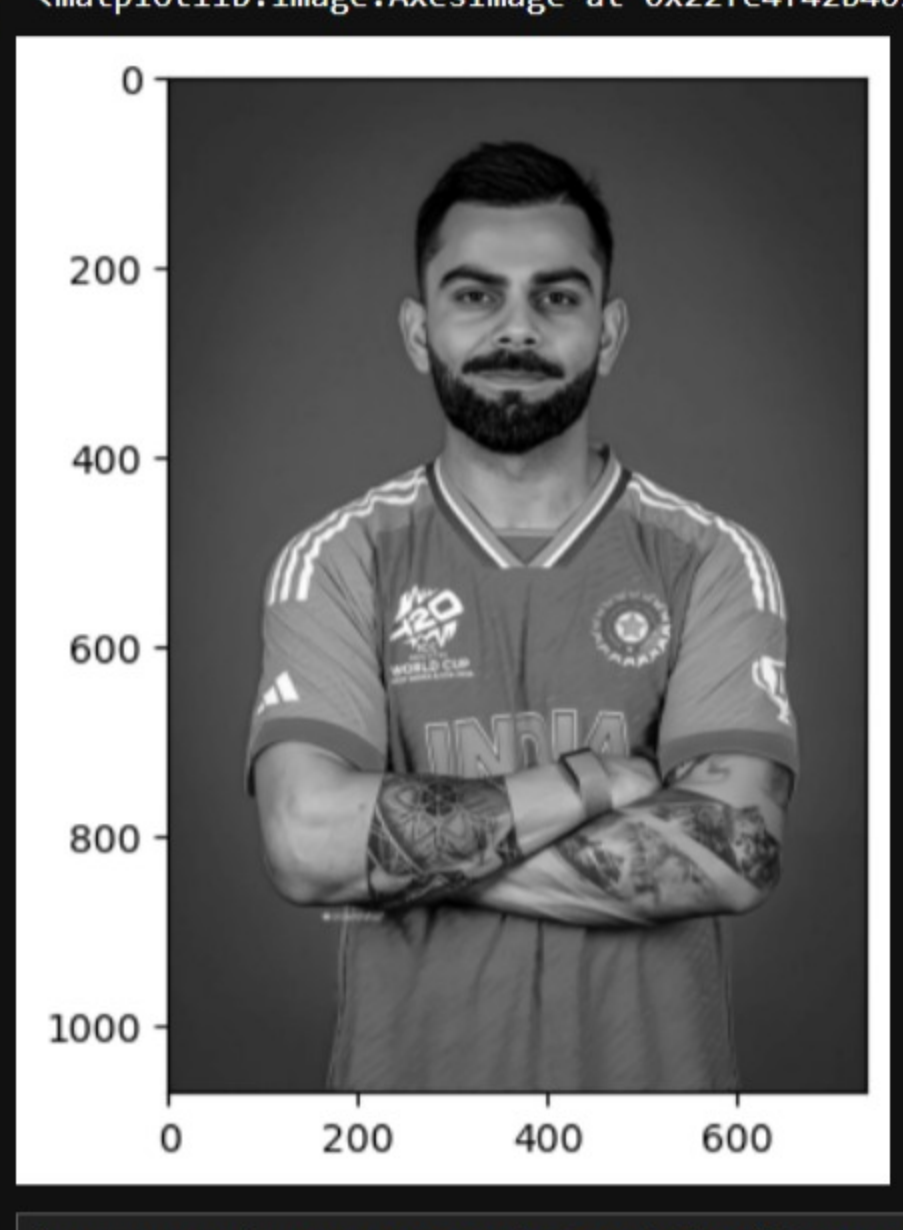
[15]:
```
haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
nose_cascade = cv2.CascadeClassifier('haarcascade_nose.xml')
faces = haar_cascade.detectMultiScale(gray_img, 1.3, 5)
```

[17]:
```
for (x, y, w, h) in faces:
    cv2.rectangle(crick1, (x, y), (x+w, y+h), (0, 255, 0), 2)

    roi_gray = gray_img[y:y+h, x:x+w]
    roi_color = crick1[y:y+h, x:x+w]

    nose = nose_cascade.detectMultiScale(roi_gray)

    for (ex,ey,ew,eh) in nose:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(255,0,0),2)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cv2.imshow('Detected Faces', crick1)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Detecting Face and Mouth

[20]:
```
import cv2
from matplotlib import pyplot as plt

crick1 = cv2.imread('virat.jpg',1)
plt.imshow(crick1[:,:,::-1])

gray_img = cv2.cvtColor(crick1, cv2.COLOR_BGR2GRAY)
plt.imshow(gray_img, cmap='gray')
```

[20]:
```
<matplotlib.image.AxesImage at 0x22fe4f42b40>
```

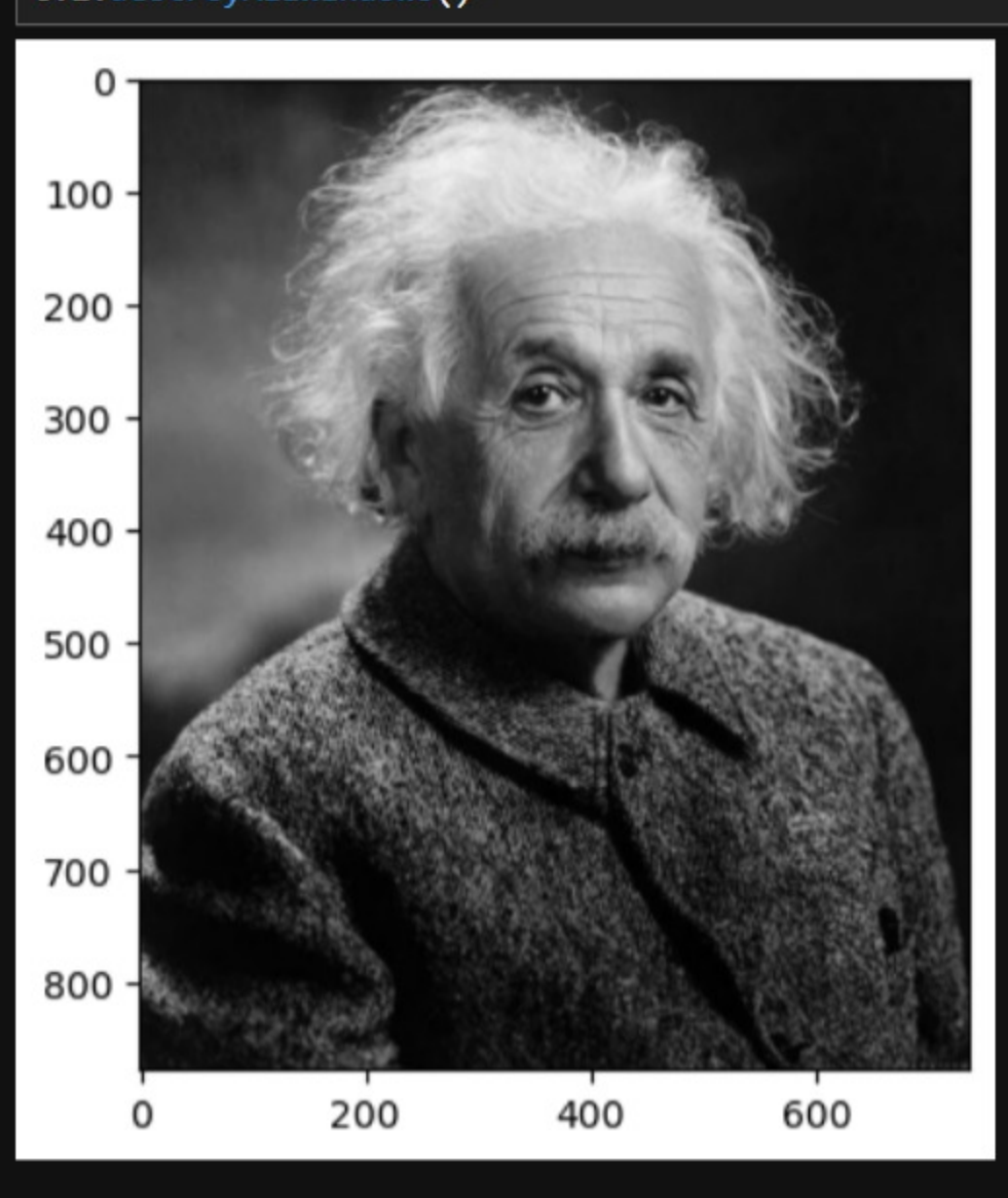


[22]:
```
haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
mouth_cascade = cv2.CascadeClassifier('haarcascade_mouth.xml')
faces = haar_cascade.detectMultiScale(gray_img, 1.3, 5)
```

[ ]:
```
for (x, y, w, h) in faces:
    cv2.rectangle(crick1, (x, y), (x+w, y+h), (0, 255, 0), 2)

    roi_gray = gray_img[y:y+h, x:x+w]
    roi_color = crick1[y:y+h, x:x+w]

    mouth = mouth_cascade.detectMultiScale(roi_gray)

    for (ex,ey,ew,eh) in mouth:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(255,0,0),2)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cv2.imshow('Detected Faces', crick1)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Face, Eye and PUPIL Detection

[27]:
```
import cv2
from matplotlib import pyplot as plt
crick1 = cv2.imread('albert.jpg',1)
plt.imshow(crick1[:,:,::-1])
gray_img = cv2.cvtColor (crick1, cv2.COLOR_BGR2GRAY)
plt.imshow(gray_img,cmap='gray')
import cv2
crick1 = cv2.imread('albert.jpg',1)
gray_img = cv2.cvtColor (crick1, cv2.COLOR_BGR2GRAY)
haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml.')
faces = haar_cascade.detectMultiScale(gray_img,1.3,5)
for (x, y, w, h) in faces:
    cv2.rectangle(crick1, (x, y), (x+w, y+h), (255, 0, 0), 2)
    roi_gray = gray_img[y:y+h, x:x+w]
    roi_color = crick1[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)

    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(255,0,0),2)

    eye_roi_gray = roi_gray[ey:ey+eh, ex:ex+ew]
    eye_roi_color = roi_color[ey:ey+eh, ex:ex+ew]

    _, eye_thresh = cv2.threshold(eye_roi_gray, 50, 255, cv2.THRESH_BINARY_INV)
    contours, _ = cv2.findContours(eye_thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) > 0:
        pupil = max(contours, key=cv2.contourArea)
        x1, y1, w1, h1 = cv2.boundingRect(pupil)
        center = (int(x1 + w1/2), int(y1 + h1/2))
        cv2.circle(eye_roi_color, center, 3, (0, 0, 255), -1)

cv2.imshow('Detected faces', crick1)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



License Plate Detection

[29]:
```
import cv2
import numpy as np

faceCascade = cv2.CascadeClassifier('haarcascade_russian_plate_number.xml')

img = cv2.imread('car1.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(gray,scaleFactor=1.2,
    minNeighbors = 5, minSize=(25,25))

for (x,y,w,h) in faces:
    cv2.rectangle(gray,(x,y),(x+w,y+h),(255,0,0),2)
    plate = gray[y: y+h, x:x+w]
    plate = cv2.blur(plate,ksize=(20,20))
    # put the blurred plate into the original image
    gray[y: y+h, x:x+w] = plate

cv2.imshow('plates',gray)
if cv2.waitKey(0) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
```

License Vidio Detection

[ ]:
```
import cv2
import numpy as np

carPlatesCascade = cv2.CascadeClassifier('haarcascade_russian_plate_number.xml')

cap = cv2.VideoCapture('vid.mp4')

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 80)

if not cap.isOpened():
    print('Error Reading video')

while True:
    ret, frame = cap.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    car_plates = carPlatesCascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(25, 25))

    for (x, y, w, h) in car_plates:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
        plate = frame[y: y + h, x: x + w]
        plate = cv2.blur(plate, ksize=(20, 20))

        # Replace the entire plate area (all 3 channels) in the original frame
        frame[y: y + h, x: x + w] = plate

    if ret:
        cv2.imshow('Video', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

cap.release()
cv2.destroyAllWindows()
```

[ ]:

Live Web Cam

[1]:
```
import cv2
print(cv2.__version__)
```

```
4.10.0
```

•[1]:
```
# First Method.

import cv2
video_capture = cv2.VideoCapture(0, cv2.CAP_DSHOW)

while True:
    _, img = video_capture.read()
    cv2.imshow("face detection", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

•[3]:
```
# Second Method.

import cv2

camera_index = 0  # Change to 1 or 2 if needed
video_capture = cv2.VideoCapture(camera_index, cv2.CAP_DSHOW)  # Use CAP_DSHOW for Windows

# Check if the camera opened successfully
if not video_capture.isOpened():
    print(f"Error: Could not open webcam at index {camera_index}")
    exit()

while True:
    ret, img = video_capture.read()

    if not ret:
        print("Error: Could not read frame")
        break

    cv2.imshow("Face Detection", img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release resources
video_capture.release()
cv2.destroyAllWindows()
```

•[5]:
```
# Third Method.

import cv2

# Initialize video capture with an explicit backend
video_capture = cv2.VideoCapture(0, cv2.CAP_DSHOW)  # Use CAP_V4L2 for Linux

# Check if the camera opened successfully
if not video_capture.isOpened():
    print("Error: Could not open webcam")
    exit()

while True:
    ret, img = video_capture.read()

    if not ret:
        print("Error: Could not read frame")
        break

    cv2.imshow("Face Detection", img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release resources
video_capture.release()
cv2.destroyAllWindows()
```

[ ]: