EXPERIMENT - 1

- Shubh Agarwal
- AIML B2
- 22070126108

AIM: To recognize objects in consumer images and analyze human activity in videos using OpenCV and Haar-Cascade classifiers.

Objectives: 1) Access the camera integrated with a Desktop/Laptop using OpenCV in Python. 2) Capture video and store it. 3) Perform different operations frame-wise on captured images/video using Haar-Cascade. 4) Operations include blurring, negative, enhancement, etc.

Theory:

- **Camera Basics**:
  - OpenCV allows interfacing with a webcam or external camera using `cv2.VideoCapture(0)`.
  - Frames are read using `cam.read()`, where `ret` indicates success, and `frame` stores the captured image.
  - `cv2.VideoWriter` is used to save videos.
- **Haar Cascade**:
  - A machine learning object detection method used to identify objects in images.
  - Uses XML files for detecting predefined objects (e.g., face, eyes, full body, etc.).
  - Works in multiple stages to filter features progressively.
  - Positive images – These images contain the images which we want our classifier to identify.
  - Negative Images – Images of everything else, which do not contain the object we want to detect.

In [1]:
```
!pip3 install opencv-python
```

```
Requirement already satisfied: opencv-python in /opt/anaconda3/lib/python3.
12/site-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in /opt/anaconda3/lib/python3.
12/site-packages (from opencv-python) (1.26.4)
```

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

In [3]:
```
# open the default camera
cam = cv2.VideoCapture(0)
```

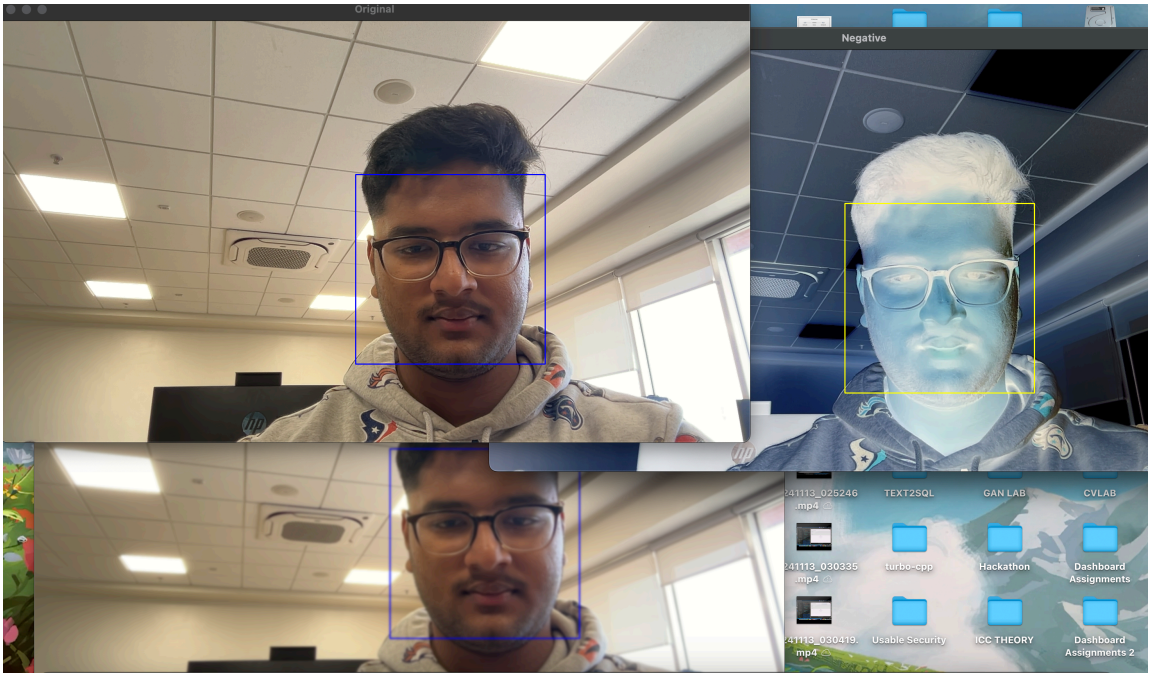
In [4]:
```
# Get the default height and width
frame_width = int(cam.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cam.get(cv2.CAP_PROP_FRAME_HEIGHT) )
```

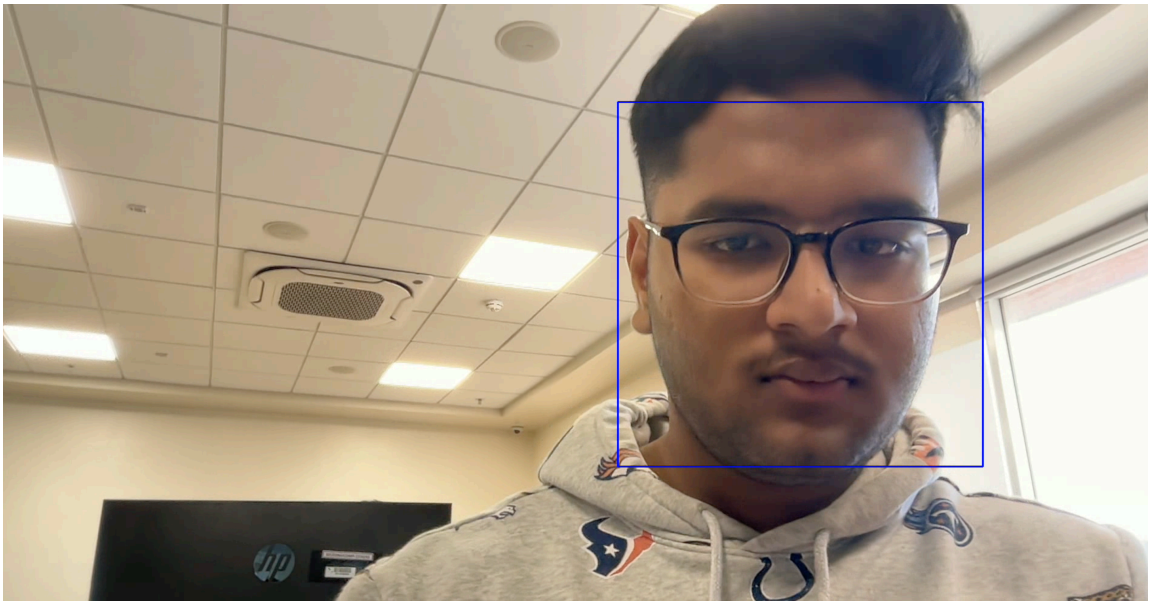
In [5]:
```
# Define the codec and create VideoWrite Object
fourcc= cv2.VideoWriter_fourcc(*'mp4v')
out =cv2.VideoWriter('/Users/shubh/Desktop/CVLAB/output.mp4',fourcc , 20.0 ,
```

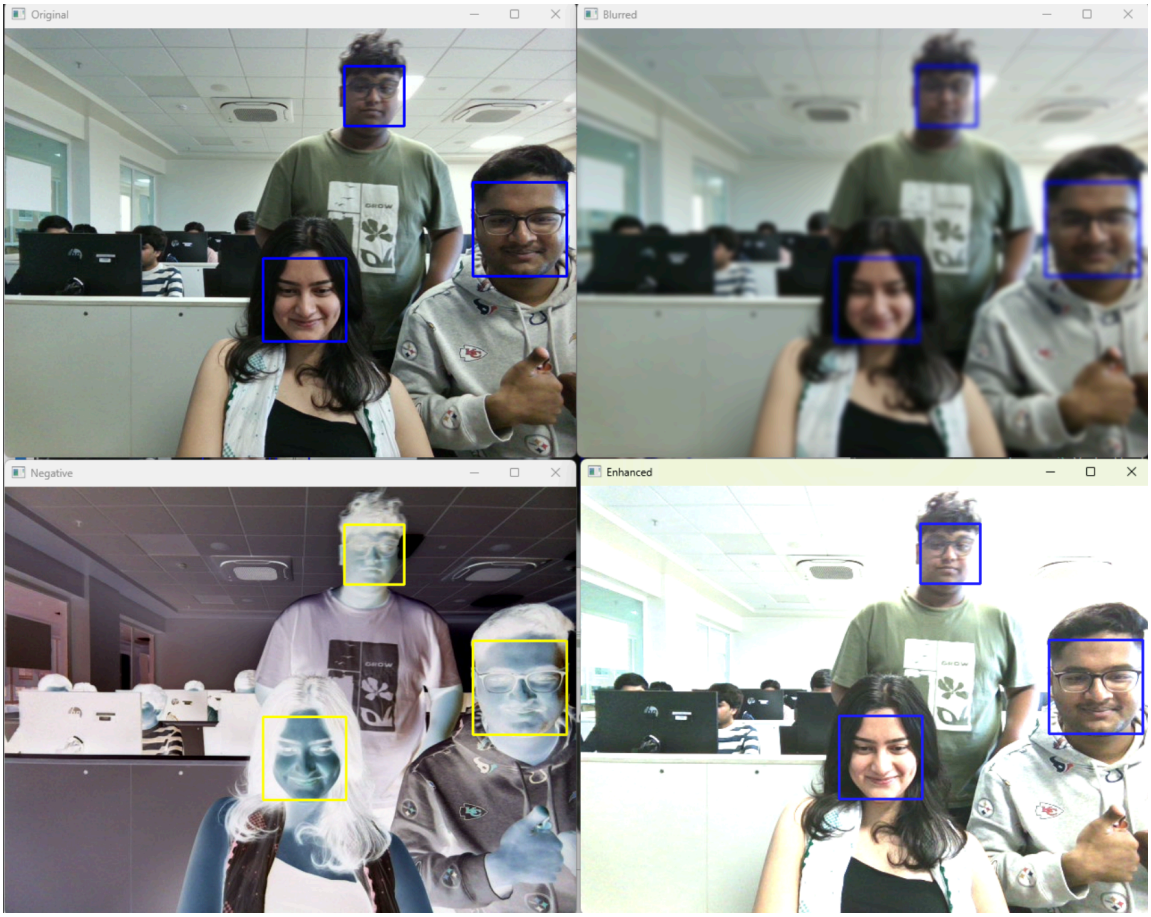
In [6]:
```
# Load Haar Cascade for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_fr
```

In [ ]:
```
while True:
    ret, frame = cam.read()
    if not ret:
        break

    # Convert frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Apply Haar Cascade for face detection
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbor
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

    # Apply Blurring
    blurred = cv2.GaussianBlur(frame, (15, 15), 0)

    # Create Negative effect
    negative = cv2.bitwise_not(frame)

    # Show the different effects
    cv2.imshow('Original', frame)
    cv2.imshow('Blurred', blurred)
    cv2.imshow('Negative', negative)

    out.write(frame)

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

cam.release()
out.release()
cv2.destroyAllWindows()
```

Output:


Original
Negative
TEXT2SQL
GAN LAB
CVLAB
turbo-cpp
Hackathon
Dashboard Assignments
Usable Security
ICC THEORY
Dashboard Assignments 2





Conclusion: This experiment successfully demonstrates the ability to capture and process video using OpenCV in Python. It integrates Haar Cascade for object detection, allowing real-time identification of faces. Additionally, various image processing techniques such as blurring and negative transformation enhance the video frames. The implementation provides a foundation for further advancements in object recognition and activity analysis in video streams.