TABLE OF CONTENTS

S.No	Experiment Title	Date	Sign

Date:……./……./…………

Experiment - 1

Experiment 1: Write a code/program in OpenCV(Using Python) to read and display an image.

Aim: Using OpenCV library of python , insert an image and display it.

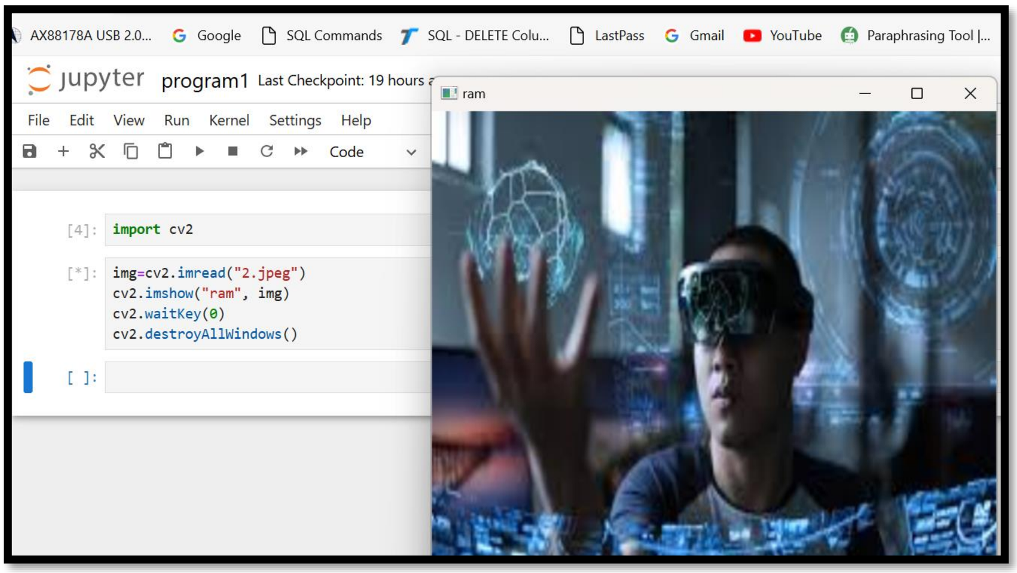
Algorithm/ Procedure:

1. *Start Program by opening Jupiter Navigator*
2. *Open Jupiter Notepad*
3. *Import CV2 library*
4. *Insert path of an image to read.*
5. *Show image*
6. *Use waitKey( ) function for exit*
7. *Exit*

Program:

```
Import cv2
img=cv2.imread("2.jpeg")          #reading an image by using a path
cv2.imshow("ram", img)            #display image by using imshow function
cv2.waitKey(0)                    #  display a window for given milliseconds or until any key is pressed
cv2.destroyAllWindows()           #allows users to destroy or close all windows
```

Output:



```
[4]: import cv2

[*]: img=cv2.imread("2.jpeg")
     cv2.imshow("ram", img)
     cv2.waitKey(0)
     cv2.destroyAllWindows()

[ ]:
```

Result: program successfully executed, Image successfully read and displayed.

.

Date:……./……./…………

Experiment -2

Experiment 2: Write a code/program in OpenCV(Using Python) to read , resize and display an image.

Aim: Using OpenCV library of python , insert an image and display it.

Algorithm/ Procedure:

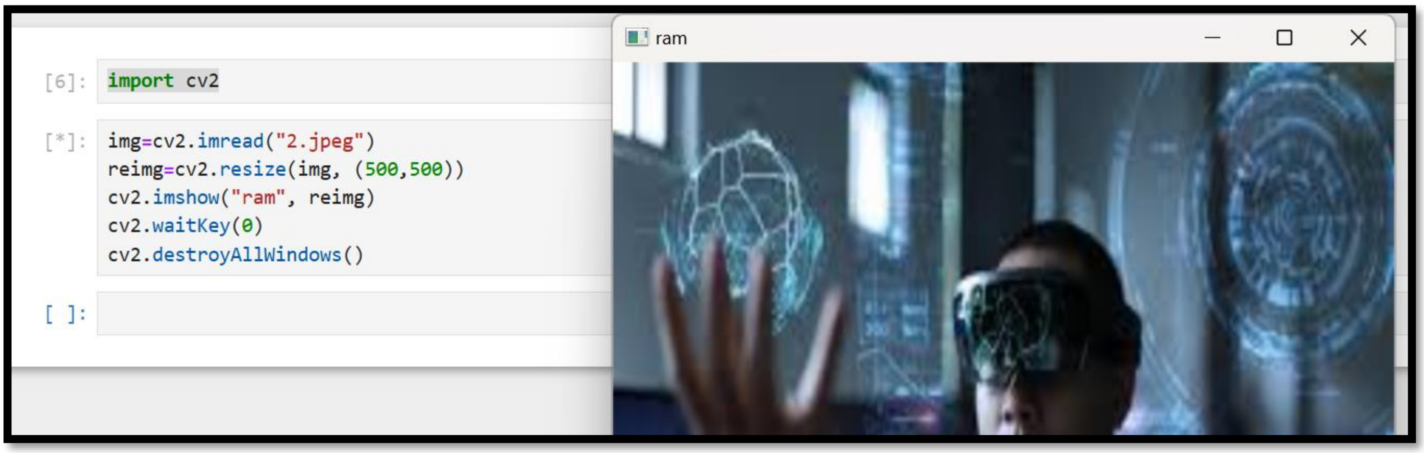
1. *Start Program by opening Jupiter Navigator*
2. *Open Jupiter Notepad*
3. *Import CV2 library*
4. *Insert path of an image to read.*
5. *Put resize function for resizing image*
6. *Show image*
7. *Exit*

Program:

```
import cv2
img=cv2.imread("1.jpg")
reimg=cv2.resize(img, (500,500))     #used function for resizing its dimension
cv2.imshow("ram", reimg)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input/ Output

Input an image after the program executed , image displayed into a resized form



```
[6]: import cv2

[*]: img=cv2.imread("2.jpeg")
     reimg=cv2.resize(img, (500,500))
     cv2.imshow("ram", reimg)
     cv2.waitKey(0)
     cv2.destroyAllWindows()

[ ]:
```

Result: *program successfully executed, Image successfully resized and displayed*.

.

Experiment -3

Date:……./……./…………

Experiment 3: Write a program in OpenCV(Using Python) to show multiple images in a single frame.

Aim: Using OpenCV and python, show multiple images into a single frame.

Algorithm/ Procedure:

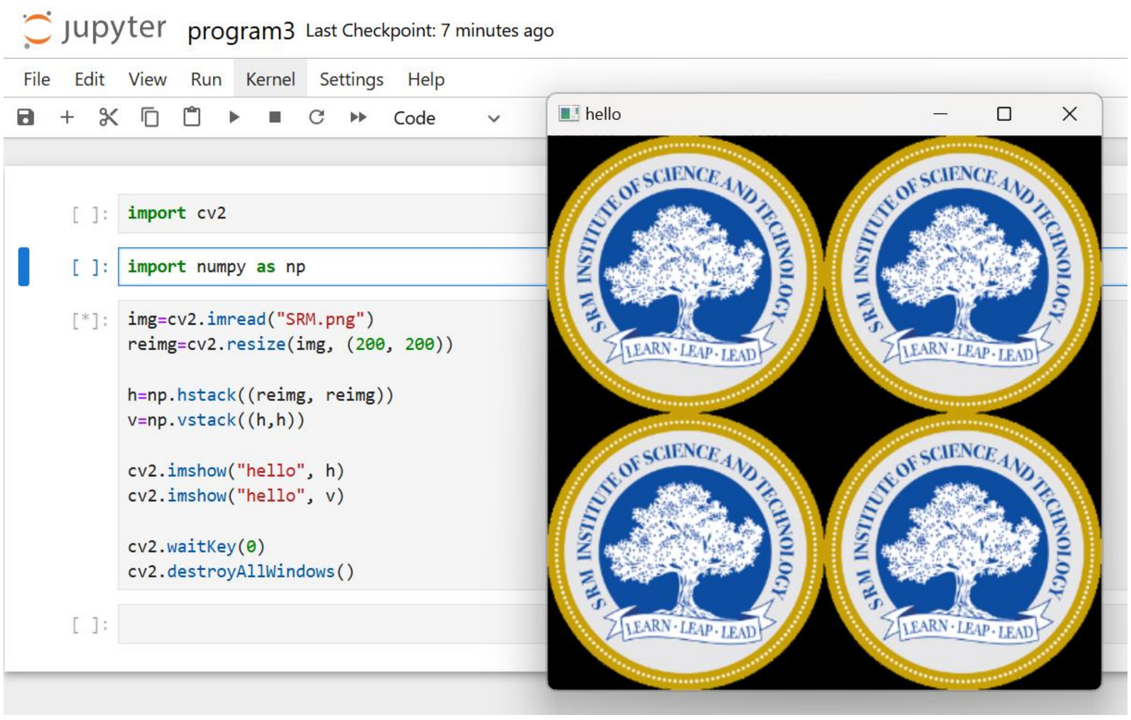
1. *Start Program by opening Jupiter Navigator*
2. *Open Jupiter Notepad*
3. *Import CV2 library*
4. *Import numpy variable*
5. *Insert path of an image to read.*
6. *Put resize function for resizing image*
7. *Adjust images horizontally and vertically*
8. *Use waitKey( ) function for exit*
9. *Exit*

Program:

```
import cv2
import numpy as np
img=cv2.imread("SRM.png")
reimg=cv2.resize(img, (200, 200))

h=np.hstack((reimg, reimg))
v=np.vstack((h,h))

cv2.imshow("hello", h)
cv2.imshow("hello", v)

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

Input/Output:

Input a figure (SRM Seal) and adjust it into frame 4x4


jupyter program3 Last Checkpoint: 7 minutes ago
File Edit View Run Kernel Settings Help
Code
hello

```
[ ]: import cv2

[ ]: import numpy as np

[*]: img=cv2.imread("SRM.png")
     reimg=cv2.resize(img, (200, 200))

     h=np.hstack((reimg, reimg))
     v=np.vstack((h,h))

     cv2.imshow("hello", h)
     cv2.imshow("hello", v)

     cv2.waitKey(0)
     cv2.destroyAllWindows()

[ ]:
```

Result: *program successfully executed, image successfully framed.*

Experiment -4

Date:……./……./………..

Experiment 4: Write a program in OpenCV(Using Python) crop an image also show its original image.

Aim: Using OpenCV and python, crop image from original image.

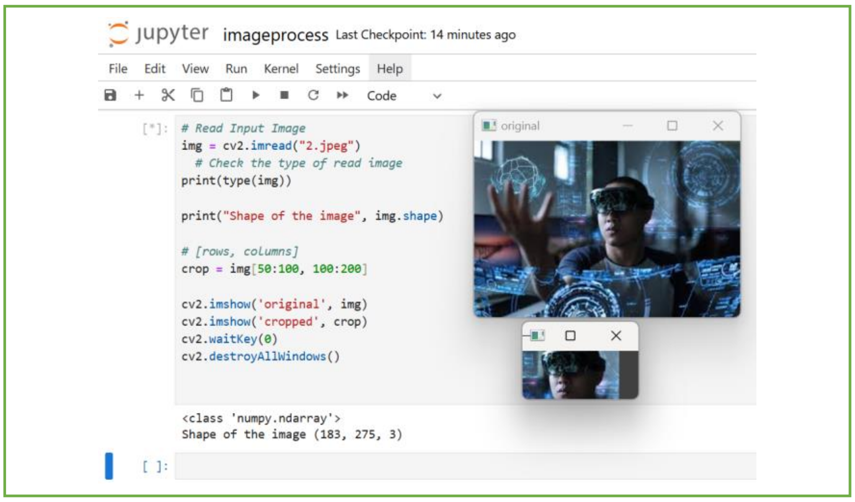
Algorithm/ Procedure:

1. *Start Program by opening Jupiter Navigator*
2. *Open Jupiter Notepad*
3. *Import CV2 library*
4. *Print type of image*
5. *Print image original size*
6. *Crop image by using crop variable and ratio*
7. *Use waitKey( ) function for exit*
8. *Exit*

Program:

```
Import cv2
# Read Input Image
img = cv2.imread("2.jpeg")
 # Check the type of read image
print(type(img))
print("Shape of the image", img.shape)
# [rows, columns]
crop = img[50:180, 100:300]

cv2.imshow('original', img)
cv2.imshow('cropped', crop)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input/output



```
[*]: # Read Input Image
     img = cv2.imread("2.jpeg")
       # Check the type of read image
     print(type(img))

     print("Shape of the image", img.shape)

     # [rows, columns]
     crop = img[50:100, 100:200]

     cv2.imshow('original', img)
     cv2.imshow('cropped', crop)
     cv2.waitKey(0)
     cv2.destroyAllWindows()

     <class 'numpy.ndarray'>
     Shape of the image (183, 275, 3)

[ ]:
```

Result: Image cropped successfully

Experiment -5

Date:……./……./…………

Experiment 5: Write a program in OpenCV using python to detect edges of an image.

***Aim: Using OpenCV* to find the edges of an image.**

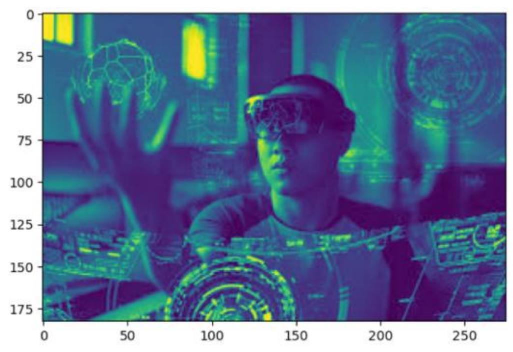
Algorithm/ Procedure:

1. *Start Program by opening Jupiter Navigator*
2. *Import cv2*
3. *Read image*
4. *Use canny function*
5. *Result image*
6. *Exit*

Program:

```
import cv2
import matplotlib.pyplot as plt
img=cv2.imread('ss.jpeg', 0)
plt.imshow(img)
```

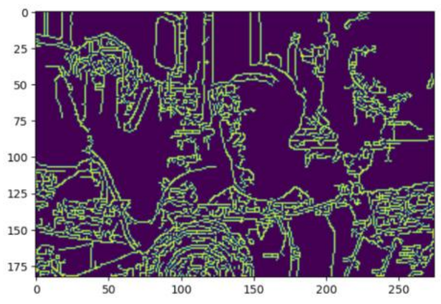
Input of image with name ss.jpg



```
img1=cv2.Canny(img,10,400)
plt.imshow(img1)
```

Output:

```
<matplotlib.image.AxesImage at 0x1b5b7e2c150>
```



Result: Edges detected successfully

Experiment -6

Date:……./……./…………

Experiment 6: Write a program in OpenCV for detecting a face by using AI code.

***Aim: Using OpenCV* to detect a face using AI cascade.**

Procedure:

1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Import image that is required to detect.*
4. *Convert image into grayscale*
5. *Import cascade XML cascade file for detect face.*
6. *Use for loop for creating square on face*
7. *Display image*
8. *End program*

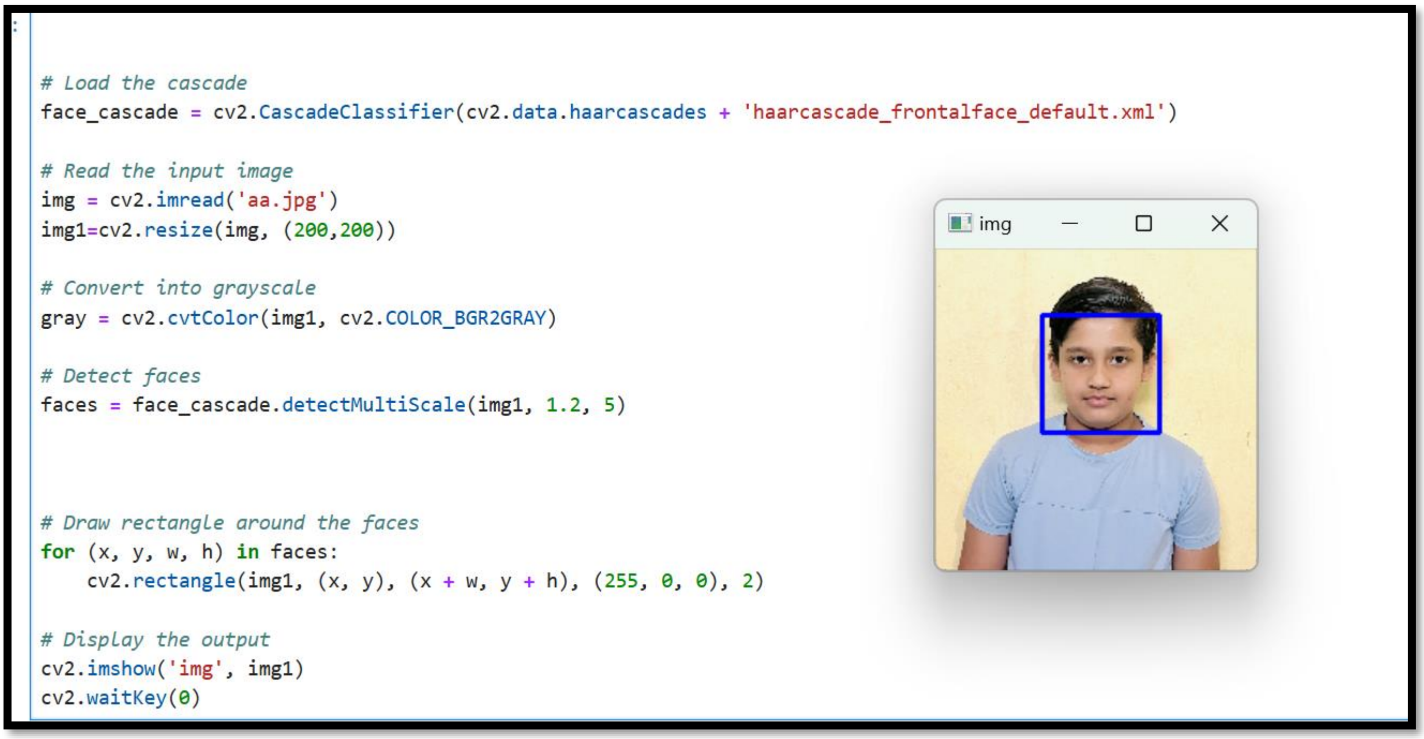
Program/ Code :

```
import cv2
import matplotlib.pyplot as plt

# Load the cascade
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Read the input image
img = cv2.imread('aa.jpg')
img1=cv2.resize(img, (200,200))

# Convert into grayscale
gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

# Detect faces
faces = face_cascade.detectMultiScale(img1, 1.2, 5)

# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img1, (x, y), (x + w, y + h), (255, 0, 0), 2)

# Display the output
cv2.imshow('img', img1)
cv2.waitKey(0)
```

Input / Output

Input an image with name aa.jpg

Output:



```
# Load the cascade
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Read the input image
img = cv2.imread('aa.jpg')
img1=cv2.resize(img, (200,200))

# Convert into grayscale
gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

# Detect faces
faces = face_cascade.detectMultiScale(img1, 1.2, 5)



# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img1, (x, y), (x + w, y + h), (255, 0, 0), 2)

# Display the output
cv2.imshow('img', img1)
cv2.waitKey(0)
```

Result:

Face detected successfully using opencv programming

Experiment -7 **Date:……./……./…………**

Experiment 7: Write a program in OpenCV for detecting eyes on face by using AI code.

***Aim: Using OpenCV* to detect eyes using AI cascade.**

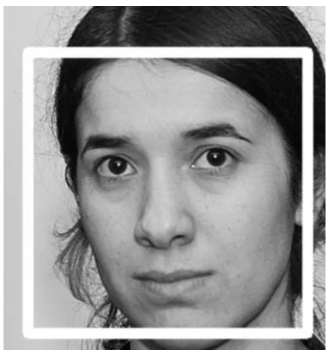
Procedure:

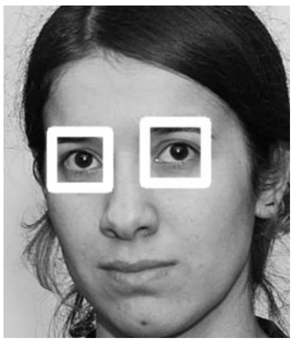
1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Import image that is required to detect.*
4. *Convert image into grayscale*
5. *Import cascade XML cascade file for detect face and eyes.*
6. *Use for loop for creating square on eyes*
7. *Display image*
8. *End program*

Program/ Code :

```
import cv2

eye_cascade = cv2.CascadeClassifier('.\Data\haarcascade_eye.xml')
nadia = cv2.imread('./Data/Nadia_Murad.jpg',0)

def detect_eyes(img):
    face_img = img.copy()
    eyes = eye_cascade.detectMultiScale(face_img,scaleFactor=1.2,minNeighbors=5)

    for (x,y,w,h) in eyes:
        cv2.rectangle(face_img, (x,y), (x+w,y+h), (255,255,255), 10)
    return face_img

result = detect_eyes(nadia)

while True:
    cv2.imshow('',result)
    code = cv2.waitKey(10)
    if code == ord('q'):
        break
```

Input / Output





Result : Eyes detected successfully by using OpenCV

Experiment -8 **Date:……./……./…………**

Experiment 8: write a program in OpenCV for erosion and dilation of an image.

Aim: Using OpenCV **erosion and dilation of an image.**

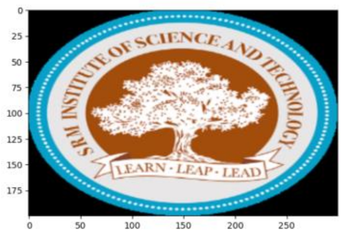
Procedure:

1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Import image that is required to erosion and dilation.*
4. *Resize image*
5. *Use erode function for filling pixels*
6. *Show image*
7. *Dilate image or remove pixels.*
8. *Display image*
9. *End program*

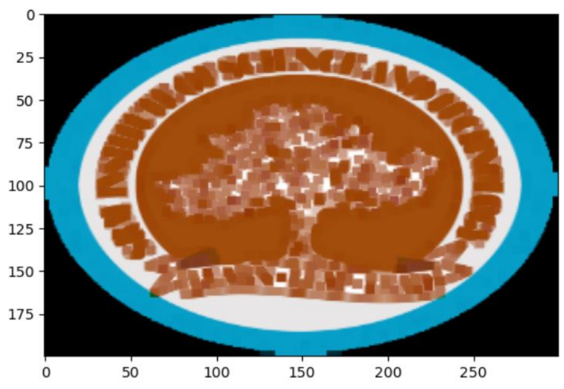
Program/ Code :

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

i=cv2.imread('SRM.png')
img=cv2.resize(i,(300,200))
plt.imshow(img)
```

Input Image:

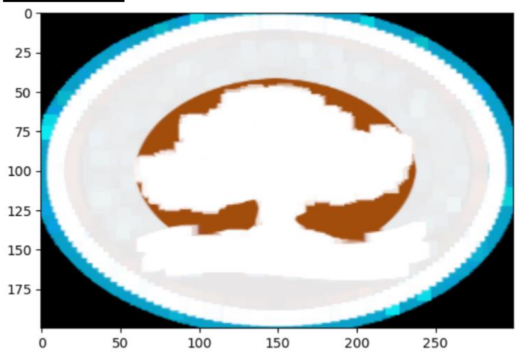


```
kernel = np.ones((5,5), np.uint8)
img_erosion = cv2.erode(img, kernel, iterations=1)
plt.imshow(img_erosion)
```



```
img_dilation = cv2.dilate(img, kernel, iterations=2)
plt.imshow( img_dilation)
```

Output:-



Result : erosion and dilation of images successfully

Date:……./……./…………

Experiment -9

Experiment 9: Write a computer vision program to find a figure's corners.

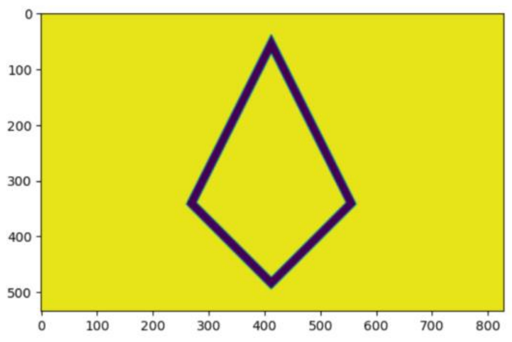
Aim: computer vision program to find the corners of an image.

Procedure:

1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Import numpy*
4. *Import image*
5. *Convert to grayscale*
6. *apply the cv2.cornerHarris method to detect the corners*
7. *Threshold for an optimal value.*
8. *Show images*
9. *End program*

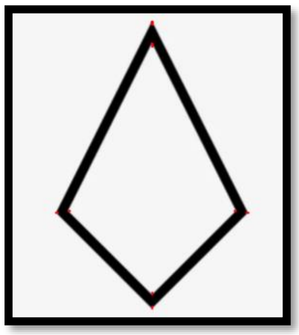
Program/ Code :

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('pentagon.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray)
plt.show()
```



```
# apply the cv2.cornerHarris method to detect the corners
corners = cv2.cornerHarris(gray, 2, 3, 0.05)

#result is dilated for marking the corners
corners = cv2.dilate(corners, None)

# Threshold for an optimal value.
img[corners > 0.01 * corners.max()]=[0, 0, 255]

#  window showing output image with corners
cv2.imshow('Image with Corners', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

<u>Output</u>



Result:

***Corner found successfully of a pentagon image*.**

Experiment -10 **Date:……./……./…………**

Experiment 10: Write a program of computer vision to detect multiple faces.

Aim: computer vision program to detect multiple faces.

Procedure:

1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Insert image and read image*
4. *Convert image to grayscale*
5. *Detect faces in image*
6. *Draw rectangle around the image*
7. *Show image*
8. *End program*

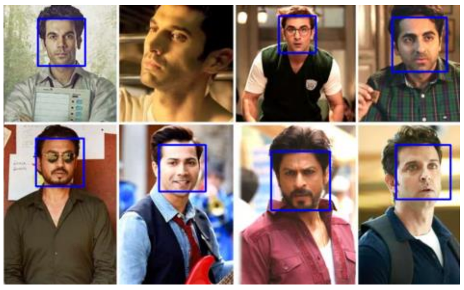
Program/ Code :

```
Import cv2

# Load the pre-trained face detection classifier
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

# Read the image
img = cv2.imread("h1.jpg")

# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect faces in the image
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(50, 50))

# Draw rectangles around the detected faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

# Display the image with detected faces
cv2.imshow("Faces", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input / Output:

Input

Output



Result: *face detected successfully*

<u>**Experiment -11**</u> **Date:……./……./…………**

Experiment 11: Write a program in computer vision using tesseract for text detection and print it.

Aim: using tesseract library to detect text from image and print it.

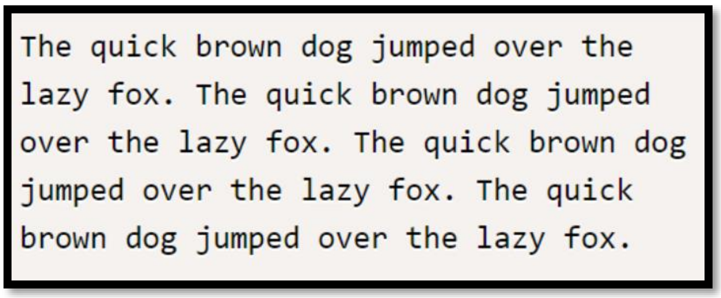
Procedure:

1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Import pytesseract*
4. *Import pytesseract exe from C drive*
5. *Import image*
6. *Convert image to grayscale*
7. *Apply thresholding to preprocess the image*
8. *Perform OCR*
9. *Print text*
10. *End program*

<u>Program/ Code :</u>

```
import cv2

import pytesseract


# Install tesseract-ocr and add it to path

pytesseract.pytesseract.tesseract_cmd = 'C:/Program Files/Tesseract-OCR/tesseract.exe'


# Read the image

img = cv2.imread('imm.png')


# Convert to grayscale

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


# Apply thresholding to preprocess the image

thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]


# Perform OCR

text = pytesseract.image_to_string(thresh)

#Print detected text

print(text)
```

Input



```
The quick brown dog jumped over the
lazy fox. The quick brown dog jumped
over the lazy fox. The quick brown dog
jumped over the lazy fox. The quick
brown dog jumped over the lazy fox.
```

Output:

The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox. The quick brown dog jumped over the lazy fox.

Result:

Text Detected Successfully.

Experiment -12 **Date:……./……./…………**

Experiment 12: Write a program in computer vision for contour detection and masking.

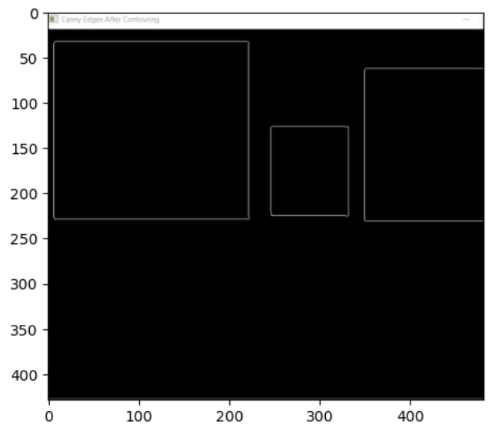
***Aim: using opencv application in* computer vision for contour detection and masking**

Procedure:

1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Import pytesseract*
4. *Import pytesseract exe from C drive*
5. *Import image*
6. *Convert image to grayscale*
7. *Apply thresholding to preprocess the image*
8. *Perform OCR*
9. *Print text*
10. *End program*

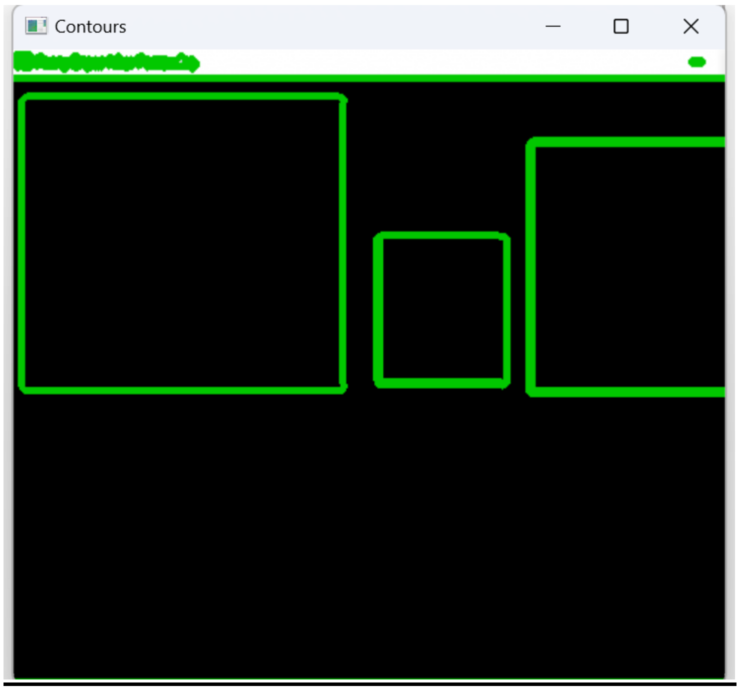
Program/ Code :

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

img=cv2.imread('111.png')
plt.imshow(img)
```



```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Find Canny edges
edged = cv2.Canny(gray, 30, 200)
cv2.waitKey(0)

# Finding Contours
# Use a copy of the image e.g. edged.copy()
# since findContours alters the image
contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

#cv2.imshow('Canny Edges After Contouring', edged)
#cv2.waitKey(0)

#print("Number of Contours found = " + str(len(contours)))

# Draw all contours
# -1 signifies drawing all contours
cv2.drawContours(img, contours, -1, (0, 200, 2), 3)

cv2.imshow('Contours', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output



Result: **represent the boundaries of the object(contour) successfully.**

Experiment -13 Date:……./……./………..

Experiment 13: Write a program in computer vision for object detection.

Aim: using YOLO(You Look Only) code , detect an object

Procedure:

1. *Start Jupyter Notebook*
2. *Import CV2*
3. *Import numpy, yaml, os*
4. *Load yaml*
5. *Load yolo model*
6. *get the YOLO prediction from the the image*
7. *Load image*
8. *Convert image to grayscale*
9. *Draw bounding*
10. *Show image*
11. *End program*

Program/ Code :

```
import cv2
import numpy as np
import os
import yaml
from yaml.loader import SafeLoader
```

```
# load YAML
with open('data.yaml',mode='r') as f:
    data_yaml = yaml.load(f,Loader=SafeLoader)

labels = data_yaml['names']
print(labels)

['person', 'car', 'chair', 'bottle', 'pottedplant', 'bird', 'dog', 'sofa', 'bicycle', 'horse', 'boat', 'motorbike', 'cat',
'tvmonitor', 'cow', 'sheep', 'aeroplane', 'train', 'diningtable', 'bus']
[43]:
```

```
# load YOLO model
yolo = cv2.dnn.readNetFromONNX('./Model/weights/best.onnx')
yolo.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
yolo.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

# load the image
img = cv2.imread('./streetIndia.jpg')
image = img.copy()
row, col, d = image.shape


# get the YOLO prediction from the the image
# step-1 convert image into square image (array)
max_rc = max(row,col)
```

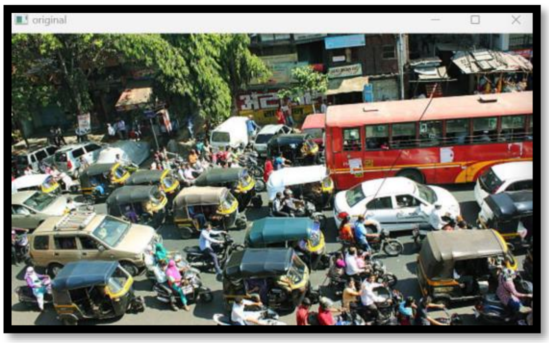
```
input_image = np.zeros((max_rc,max_rc,3),dtype=np.uint8)
input_image[0:row,0:col] = image
# step-2: get prediction from square array
INPUT_WH_YOLO = 640
blob =
cv2.dnn.blobFromImage(input_image,1/255,(INPUT_WH_YOLO,INPUT_WH_YOLO),swapRB=True,cro
p=False)
yolo.setInput(blob)
preds = yolo.forward() # detection or prediction from YOLO

# Non Maximum Supression
# step-1: filter detection based on confidence (0.4) and probability score (0.25)
detections = preds[0]
boxes = []
confidences = []
classes = []

# widht and height of the image (input_image)
image_w, image_h = input_image.shape[:2]
x_factor = image_w/INPUT_WH_YOLO
y_factor = image_h/INPUT_WH_YOLO

for i in range(len(detections)):
    row = detections[i]
    confidence = row[4] # confidence of detection an object
    if confidence > 0.4:
        class_score = row[5:].max() # maximum probability from 20 objects
        class_id = row[5:].argmax() # get the index position at which max probabilty occur

        if class_score > 0.25:
            cx, cy, w, h = row[0:4]
            # construct bounding from four values
            # left, top, width and height
            left = int((cx - 0.5*w)*x_factor)
            top = int((cy - 0.5*h)*y_factor)
            width = int(w*x_factor)
            height = int(h*y_factor)

            box = np.array([left,top,width,height])

            # append values into the list
            confidences.append(confidence)
            boxes.append(box)
            classes.append(class_id)

# clean
boxes_np = np.array(boxes).tolist()
confidences_np = np.array(confidences).tolist()

# NMS
index = cv2.dnn.NMSBoxes(boxes_np,confidences_np,0.25,0.45).flatten()


# Draw the Bounding
```

```
for ind in index:
    # extract bounding box
    x,y,w,h = boxes_np[ind]
    bb_conf = int(confidences_np[ind]*100)
    classes_id = classes[ind]
    class_name = labels[classes_id]

    text = f'{class_name}: {bb_conf}%'

    cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)
    cv2.rectangle(image,(x,y-30),(x+w,y),(255,255,255),-1)

    cv2.putText(image,text,(x,y-10),cv2.FONT_HERSHEY_PLAIN,0.7,(0,0,0),1)
```

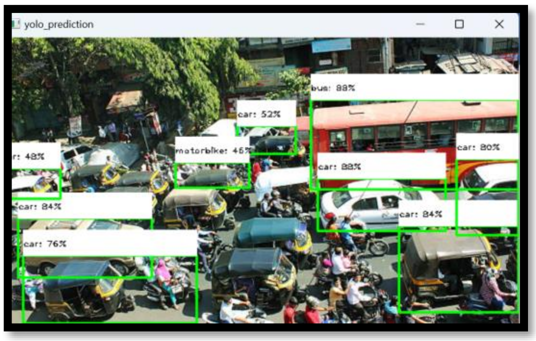
```
cv2.imshow('original',img)
cv2.imshow('yolo_prediction',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

<u>Input/ output:</u>

<u>Orignal Image</u>



Yolo pridictions



Result: Object Detected Successfully

Experiment -14 **Date:……./……./…………**

Experiment 14: Write a program in computer vision for real time object detection from a video using YOLO

Aim: using YOLO(You Look Only) code , detect an object

Procedure:

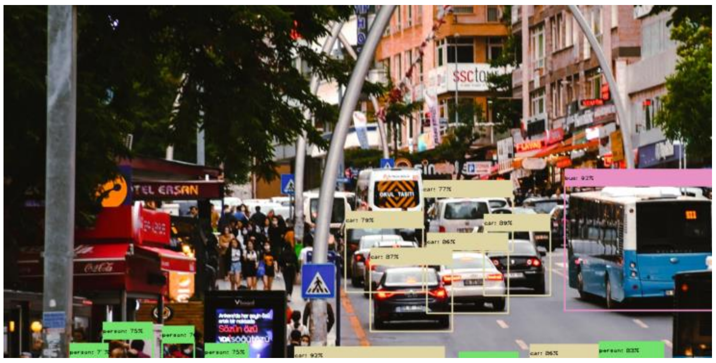
1. *Step 1: Setup the Environment.*
2. *Step 2: Importing Necessary Libraries.*
3. *Step 3: Define Function to Get Class Colours .*
4. *Step 4: Load YOLO Model.*
5. *Step 5: Open Video Capture.*
6. *Step 6: Process Video Frames.*
7. *Step 7: Main Loop for Real-time Object Detection.*
8. *Step 8: Display Image and Break Loop.*

Program/ Code :

```
import cv2
from yolo_predictions import YOLO_Pred

#load dataset
yolo = YOLO_Pred('./Model/weights/best.onnx','data.yaml')

#load image

img = cv2.imread('./street_image.jpg')

#show image
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
# predictions
img_pred = yolo.predictions(img)
```

<u>output preditions</u>

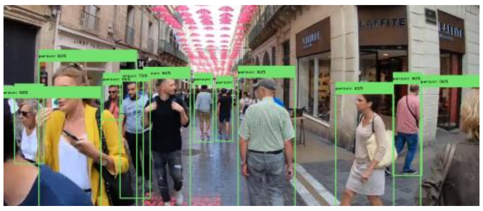


#Now real time object detection using a video (MP4 video)

#Capturing video from database:

```
cap = cv2.VideoCapture('video.mp4')

while True:
    ret, frame = cap.read()
    if ret == False:
        print('unable to read video')
        break

    pred_image = yolo.predictions(frame)

    cv2.imshow('YOLO',pred_image)
    if cv2.waitKey(1) == 10:
        break

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

Output:



Result: Object detected from streaming video successfully

Experiment -15 **Date:……./……./…………**

Experiment 15: Write a program in computer vision for live streaming or opening a camera.

Aim: Program for opening a camera using CV code.

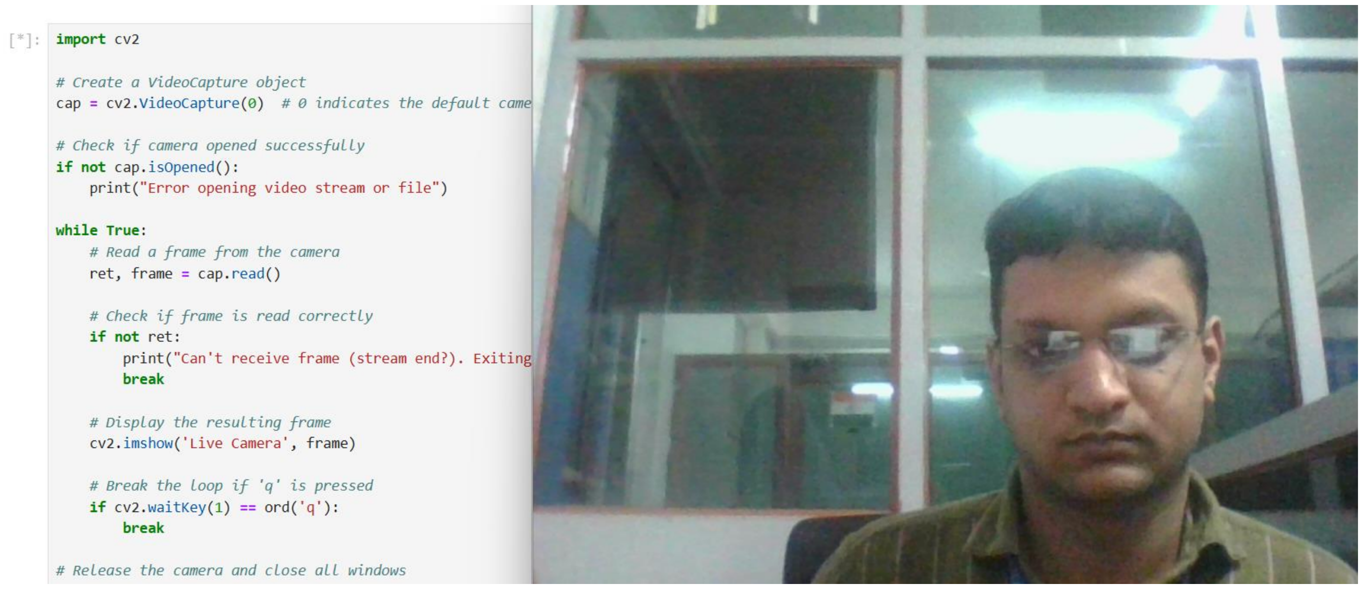
Procedure:

1. *Step 1: Setup the Environment.*
2. *Step 2: Importing Necessary Libraries.*
3. *Step 3: code for live camera Video Capture object.*
4. *Step 4: Check if camera opened successfully or not.*
5. *Step 5: Check that frame is reading correctly*
6. *Step 6: display the camera or resultant frame*
7. *Step 7: display the camera or resultant frame*
8. *Step 8: press C for close the camera*

Program/ Code :

```
import cv2

# code for live camera Video Capture object
cap = cv2.VideoCapture(0)                    # Here 0 means default camera

# Check if camera opened successfully or not
if not cap.isOpened():
   print("Error opening video stream or file")

while True:
   # Read a frame from the camera
   ret, frame = cap.read()

   # Check that frame is reading correctly
   if not ret:
      print("Can't receive frame (stream end?). Exiting ...")
      break

   # display the camera or resultant frame
   cv2.imshow('Live Camera', frame)

   # press c for close the camera

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

# Release the camera and close all windows
cap.release()
cv2.destroyAllWindows()
```

Output:



```
[*]: import cv2

     # Create a VideoCapture object
     cap = cv2.VideoCapture(0)  # 0 indicates the default came

     # Check if camera opened successfully
     if not cap.isOpened():
         print("Error opening video stream or file")

     while True:
         # Read a frame from the camera
         ret, frame = cap.read()

         # Check if frame is read correctly
         if not ret:
             print("Can't receive frame (stream end?). Exiting
             break

         # Display the resulting frame
         cv2.imshow('Live Camera', frame)

         # Break the loop if 'q' is pressed
         if cv2.waitKey(1) == ord('q'):
             break

     # Release the camera and close all windows
```

Result: ***Camera opened successfully and live streaming successfully.***

Experiment -16

Date:……./……./………..

Experiment 16 : Write a program in computer vision for face detection using camera (Live face detection).

Aim: Using CV code , detect faces (use haarcascades frontface dataset)

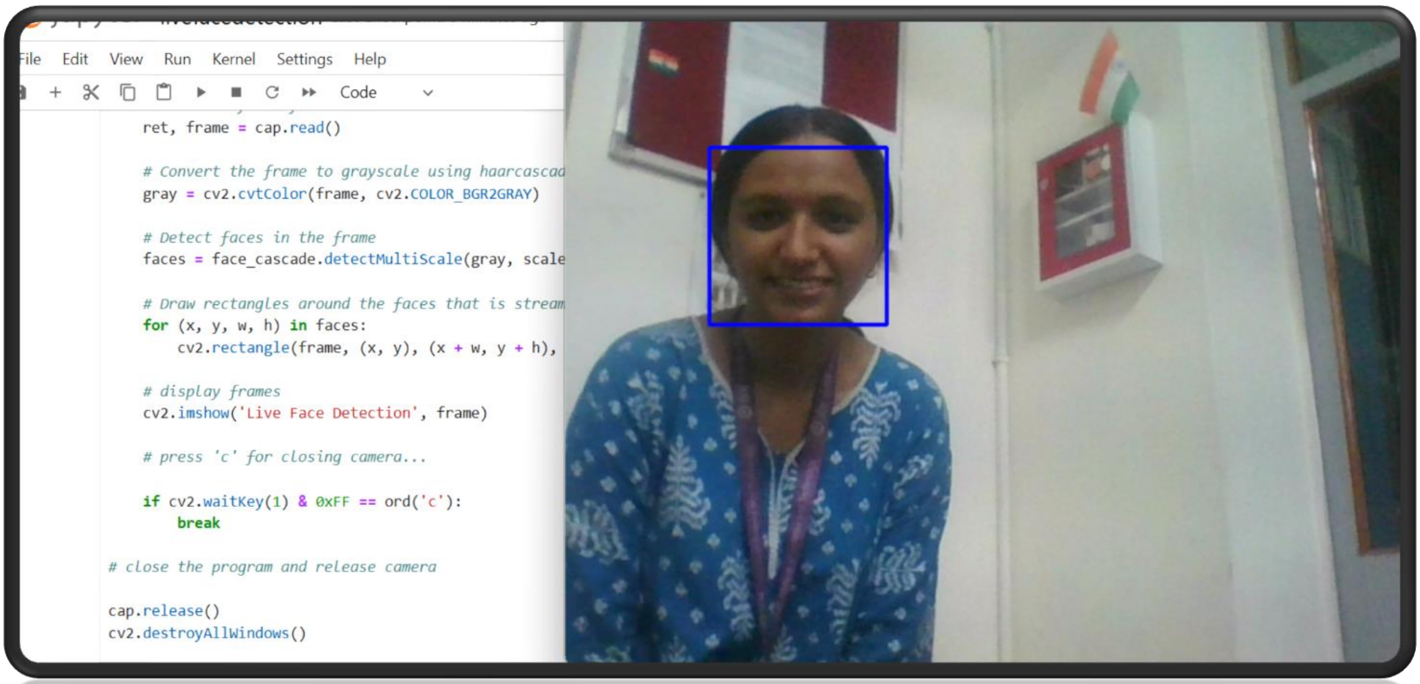
Procedure:

1. *Step 1: Setup the Environment.*
2. *Step 2: Importing Necessary Libraries.*
3. *Step 3: Read the frame from the webcam.*
4. *Step 4: Convert the frame to grayscale using haarcascade dataset.*
5. *Step 5: Detect faces in the frame*
6. *Step 6: Draw rectangles around the faces that is streaming*
7. *Step 7: display the camera or resultant frame*
8. *Step 8: press C for close the camera*

Program/ Code :

```
# Import libraries that is required
import cv2

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')


cap = cv2.VideoCapture(0)  # 0 for the primary webcam, 1 for an external webcam

while True:
    # Read the frame from the webcam
    ret, frame = cap.read()

    # Convert the frame to grayscale using haarcascade dataset
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the frame
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    # Draw rectangles around the faces that is streaming
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

    # display frames
    cv2.imshow('Live Face Detection', frame)

    # press 'c' for closing camera...

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

# close the program and release camera

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

Result:



```
        ret, frame = cap.read()

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

        # Detect faces in the frame
        faces = face_cascade.detectMultiScale(gray, scale

        # Draw rectangles around the faces that is stream
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h),

        # display frames
        cv2.imshow('Live Face Detection', frame)

        # press 'c' for closing camera...

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

    # close the program and release camera

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

Result: Face detected successfully