Experiment 3

Image Processing Basics in OpenCV

Aim:

1) To implement the following basic functions on an image or a video in Open CV:Convert to Gray Scale, implement image intensity transformations, Blur, Draw shapes and adding Text, Mask or Crop, Histogram, Thresholding, Image Addition and Image Subtraction.

Software/ Packages Used:

1. Pycharm IDE
2. Libraries used:
   - NumPy
   - opencv-python
   - matplotlib
   - scipy

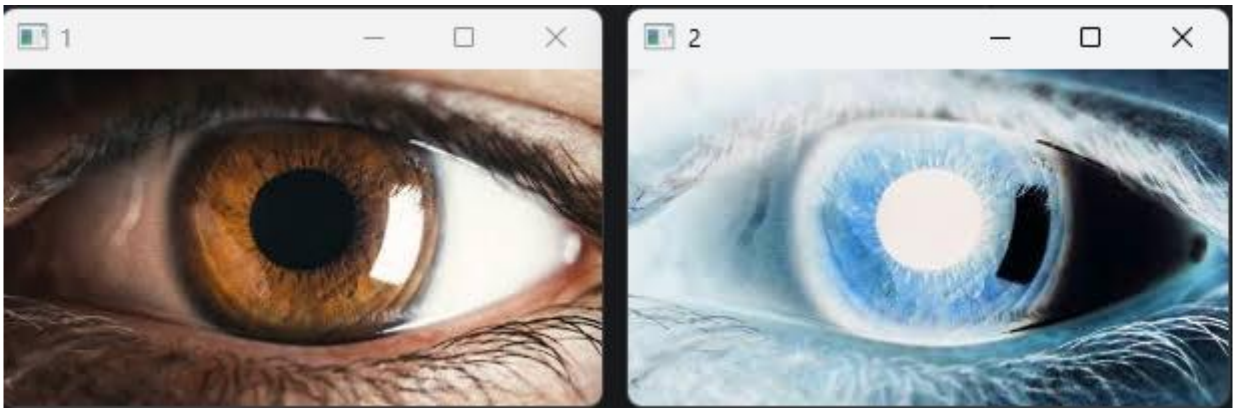
1. IMAGE TRANSFORMATIONS:

a) NEGATIVE:

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread('images.jpg')
print(img.dtype)
print(img)
img_neg = 255-img
cv2.imshow('1', img)
cv2.imshow('2', img_neg)
cv2.waitKey(0)
```

OUTPUT IMAGE:



COMMAND WINDOW:

```
uint8
[[[ 28  28  28]
  [ 28  28  28]
  [ 27  27  27]
  ...
  [194 213 234]
  [195 214 235]
  [196 215 236]]

 [[ 27  27  27]
  [ 27  27  27]
  [ 26  26  26]
  ...
  [189 208 229]
  [190 209 230]
  [191 210 231]]
```

b)LOG:

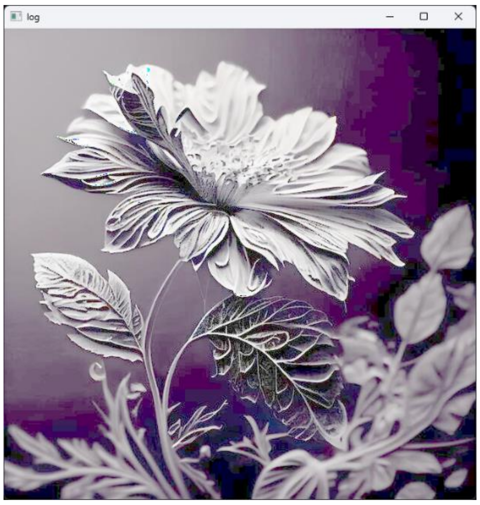
PROGRAM:

```
import cv2
import numpy as np
# Open the image.
img = cv2.imread('sample.jpg',0)
print(img)
# Apply log transform.
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)
#MATRIX
# Specify the data type.
log_transformed = np.array(log_transformed, dtype = np.uint8)
print(log_transformed)
# show
cv2.imshow('log', log_transformed)
cv2.waitKey(0)
# Save the output.
cv2.imwrite('log_transformed.jpg', log_transformed)
cv2.imwrite('log_transformed.jpg', log_transformed)
```

COMMAND WINDOW:

```
[[ 92  95 100 ...  22  30  32]
 [ 78 105 102 ...  31  31  30]
 [101 144 139 ...  30  29  29]
 ...
 [ 45  46  43 ...  83  75  61]
 [ 43  43  41 ...  82  74  76]
 [ 43  43  41 ...  76  80  91]]
```

OUTPUT IMAGE:



C)POWER LAW:

PROGRAM:

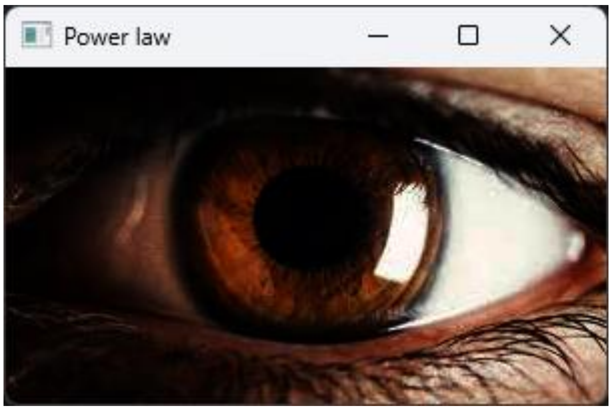
```
import cv2
import numpy as np
# Open the image.
img = cv2.imread('images.jpg')
print(img)
# Trying 4 gamma values.
for gamma in [0.1, 0.5, 1.2, 2.2]:
    # Apply gamma correction.
    gamma_corrected = np.array(255 * (img / 255) ** gamma, dtype='uint8')
    print(gamma_corrected)
    #show
    cv2.imshow('Power law', gamma_corrected)
    cv2.waitKey(0)
    # Save edited images.
    cv2.imwrite('gamma_transformed' + str(gamma) + '.jpg', gamma_corrected)
```

COMMAND WINDOW:

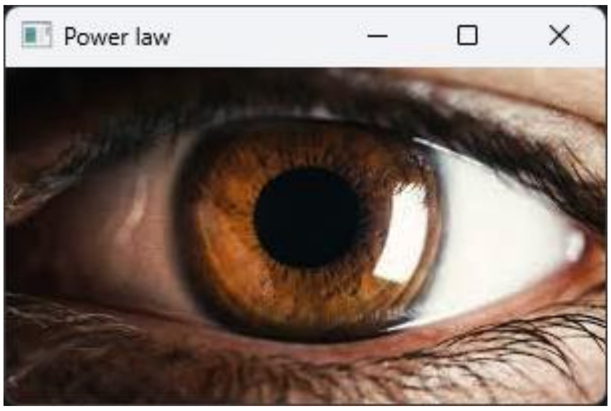
```
[[[ 28  28  28]
  [ 28  28  28]
  [ 27  27  27]
  ...
  [194 213 234]
  [195 214 235]
  [196 215 236]]
```

OUTPUT IMAGE:

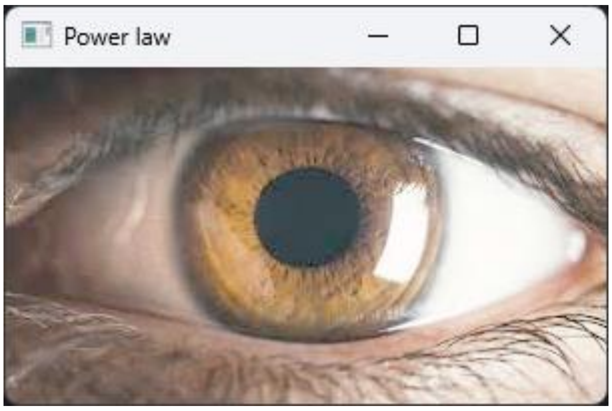
GAMMA 0.1



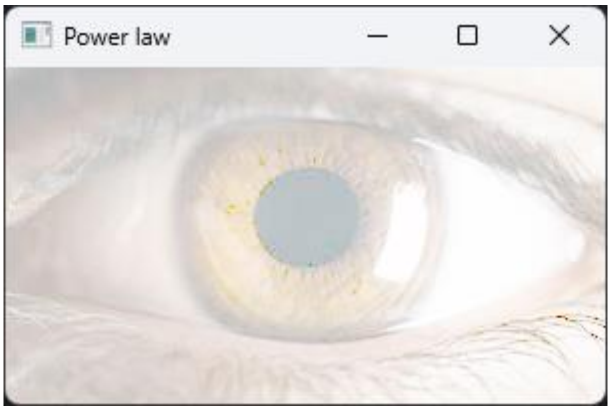
GAMMA 0.5



GAMMA 1.2



GAMMA 2.2



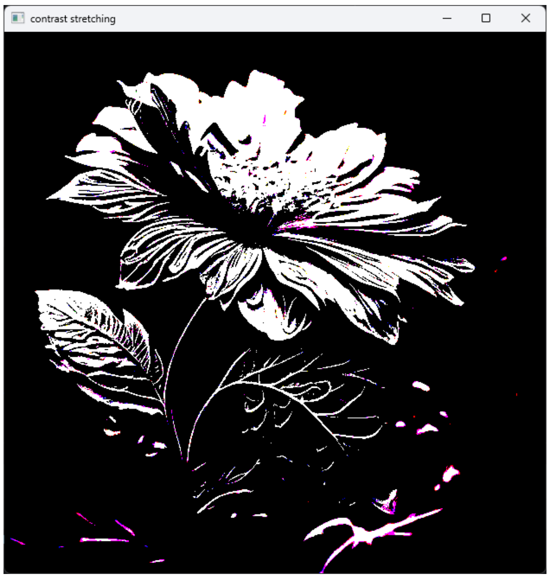
d)CONTRAST STRETCHING:

PROGRAM:

```
import cv2
import numpy as np
# Function to map each intensity level to output intensity level.
def pixelVal(pix, r1, s1, r2, s2):
    if (0 <= pix and pix <= r1):
        return (s1 / r1)*pix
    elif (r1 < pix and pix <= r2):
        return ((s2 - s1)/(r2 - r1)) * (pix - r1) + s1
    else:
        return ((255 - s2)/(255 - r2)) * (pix - r2) + s2
# Open the image.
img = cv2.imread('1.jpg')
# Define parameters.
r1 = 70
s1 = 0
r2 = 140
s2 = 255
```

```
# Vectorize the function to apply it to each value in the Numpy array.
pixelVal_vec = np.vectorize(pixelVal)
# Apply contrast stretching.
contrast_stretched = pixelVal_vec(img, r1, s1, r2, s2)
# show
cv2.imshow('contrast stretching', contrast_stretched)
cv2.waitKey(0)
```

OUTPUT IMAGE:



e)BIT PLANE SLICING

PROGRAM:

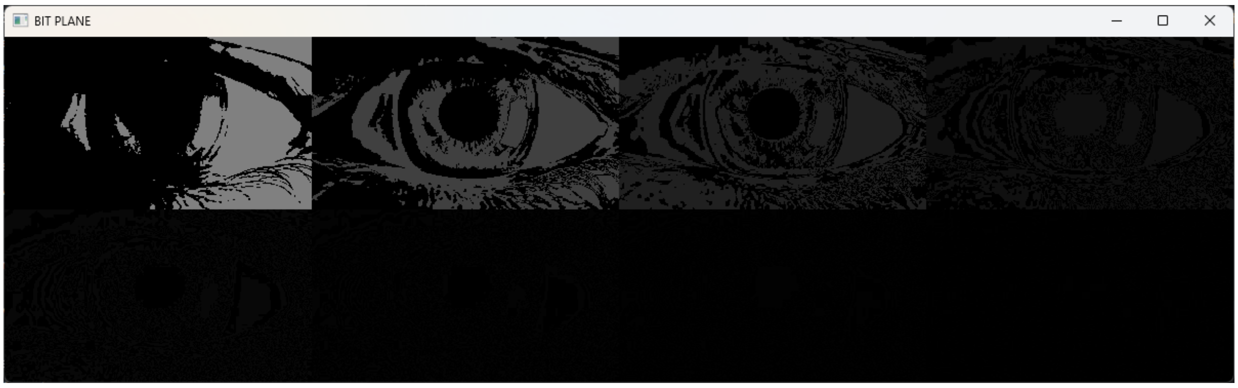
```
import numpy as np
import cv2
# Read the image in greyscale
img = cv2.imread('images.jpg',0)

#Iterate over each pixel and change pixel value to binary using np.binary_repr() and
store it in a list.
lst = []
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        lst.append(np.binary_repr(img[i][j] ,width=8)) # width = no. of bits

# We have a list of strings where each string represents binary pixel value. To extract bit
planes we need to iterate over the strings and store the characters corresponding to bit
planes into lists.
# Multiply with 2^(n-1) and reshape to reconstruct the bit image.
eight_bit_img = (np.array([int(i[0]) for i in lst],dtype = np.uint8) *
128).reshape(img.shape[0],img.shape[1])
print(eight_bit_img)
```

```
seven_bit_img = (np.array([int(i[1]) for i in lst],dtype = np.uint8) *
64).reshape(img.shape[0],img.shape[1])
print(seven_bit_img)
six_bit_img = (np.array([int(i[2]) for i in lst],dtype = np.uint8) *
32).reshape(img.shape[0],img.shape[1])
print(six_bit_img)
five_bit_img = (np.array([int(i[3]) for i in lst],dtype = np.uint8) *
16).reshape(img.shape[0],img.shape[1])
print(five_bit_img)
four_bit_img = (np.array([int(i[4]) for i in lst],dtype = np.uint8) *
8).reshape(img.shape[0],img.shape[1])
print(four_bit_img)
three_bit_img = (np.array([int(i[5]) for i in lst],dtype = np.uint8) *
4).reshape(img.shape[0],img.shape[1])
print(three_bit_img)
two_bit_img = (np.array([int(i[6]) for i in lst],dtype = np.uint8) *
2).reshape(img.shape[0],img.shape[1])
print(two_bit_img)
one_bit_img = (np.array([int(i[7]) for i in lst],dtype = np.uint8) *
1).reshape(img.shape[0],img.shape[1])
print(one_bit_img)

#Concatenate these images for ease of display using cv2.hconcat()
finalr = cv2.hconcat([eight_bit_img,seven_bit_img,six_bit_img,five_bit_img])
finalv =cv2.hconcat([four_bit_img,three_bit_img,two_bit_img,one_bit_img])

# Vertically concatenate
final = cv2.vconcat([finalr,finalv])

# Display the images
cv2.imshow('BIT PLANE',final)
cv2.waitKey(0)
```

COMMAND WINDOW:

```
[[  0   0   0 ... 128 128 128]
 [  0   0   0 ... 128 128 128]
 [  0   0   0 ... 128 128 128]
 ...
 [  0   0   0 ... 128 128 128]
 [  0   0   0 ... 128 128 128]
 [  0   0   0 ... 128 128 128]]
[[ 0  0  0 ... 64 64 64]
 [ 0  0  0 ... 64 64 64]
 [ 0  0  0 ... 64 64 64]
```

OUTPUT IMAGE:

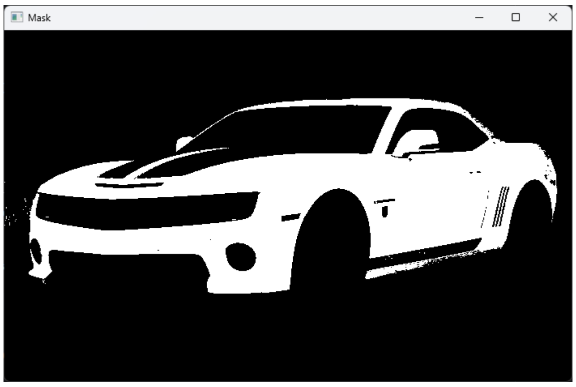


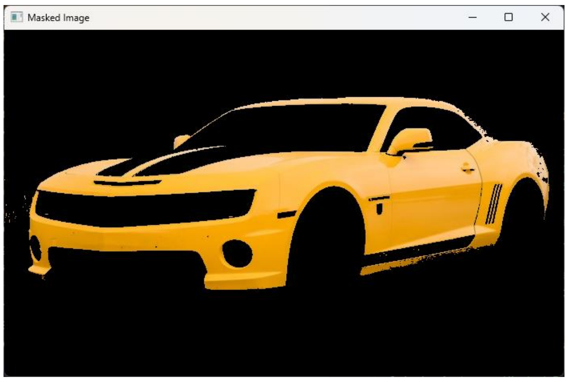
2.MASK AN IMAGE

PROGRAM:

```
# import required libraries
import cv2
import numpy as np

# read input image
img = cv2.imread('CAR.jpg')

# Convert BGR to HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# define range of blue color in HSV
lower_yellow = np.array([15,50,180])
upper_yellow = np.array([40,255,255])

# Create a mask. Threshold the HSV image to get only yellow colors
mask = cv2.inRange(hsv, lower_yellow, upper_yellow)

# Bitwise-AND mask and original image
result = cv2.bitwise_and(img,img, mask= mask)

# display the mask and masked image
cv2.imshow('Mask',mask)
cv2.waitKey(0)
cv2.imshow('Masked Image',result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT IMAGE:



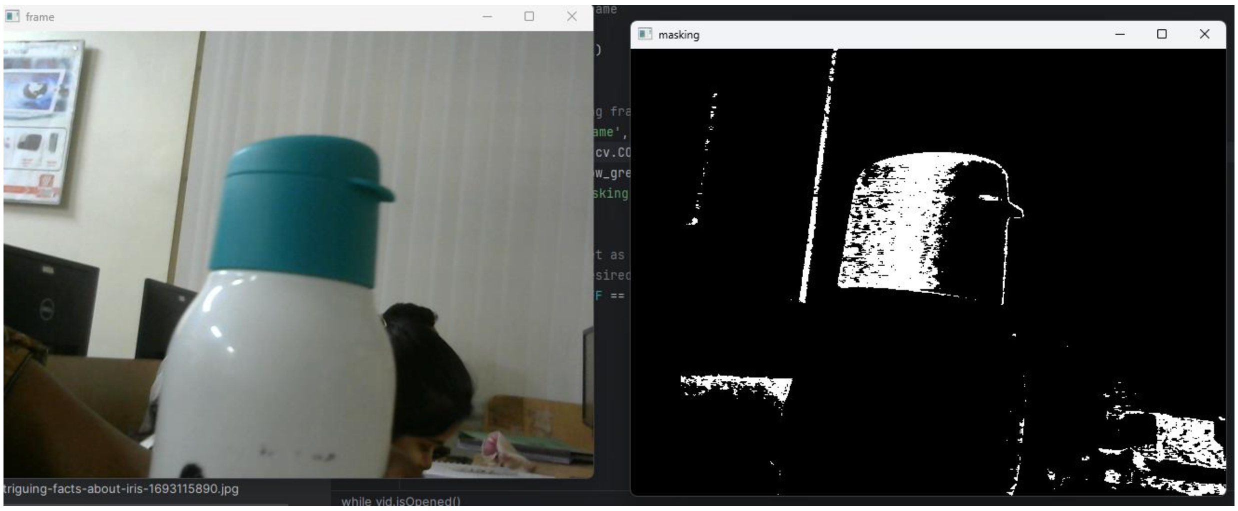


3.MASK A VIDEO:

PROGRAM:

```
import cv2 as cv
import numpy as np
low_green=np.array([25,52,72])
high_green=np.array([102,225,255])
# define a video capture object
vid = cv.VideoCapture(0)
while vid.isOpened():
   # Capture the video frame
   # by frame
   ret, frame = vid.read()
   if not ret:
      break
   # Display the resulting frame
   cv.imshow('frame', frame)
   hsv=cv.cvtColor(frame,cv.COLOR_RGB2HSV)
   mask=cv.inRange(hsv,low_green,high_green)
   cv.imshow('masking', mask)
   # the 'q' button is set as the
   # quitting button # desired button
   if cv.waitKey(1) & 0xFF == ord('q'):
      break
vid.release()
```

OUTPUT IMAGE:

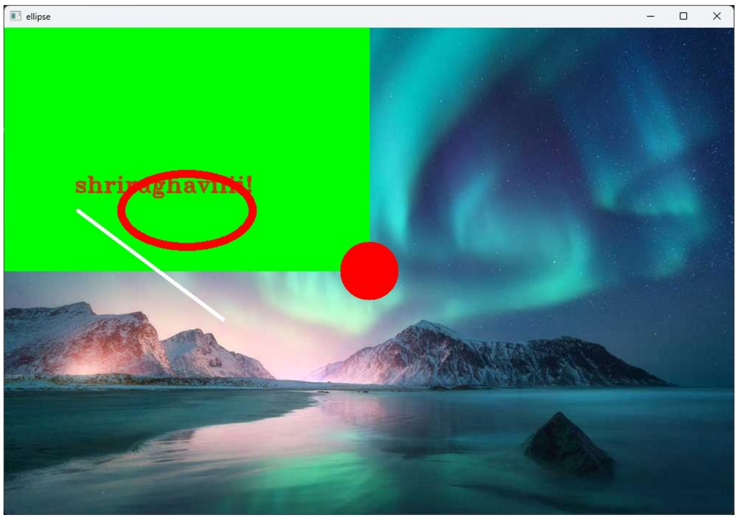


4.DRAW ON AN IMAGE:

PROGRAM:

```
import cv2 as cv
import numpy as np

blank = cv.imread('AURB_08_1000by667px_1000x.jpg')
cv.imshow('Blank', blank)

# 1. Paint the image a certain colour
blank[200:300, 300:400] = 0,0,255
cv.imshow('Green', blank)

# 2. Draw a Rectangle
cv.rectangle(blank, (0,0), (blank.shape[1]//2, blank.shape[0]//2), (0,255,0), thickness=-
1)
cv.imshow('Rectangle', blank)

# 3. Draw A circle
cv.circle(blank, (blank.shape[1]//2, blank.shape[0]//2), 40, (0,0,255), thickness=-1)
cv.imshow('Circle', blank)

# 4. Draw a line
cv.line(blank, (100,250), (300,400), (255,255,255), thickness=3)
cv.imshow('Line', blank)

# 5. Write text
cv.putText(blank, '      shriraghaviiii!', (0,225), cv.FONT_HERSHEY_TRIPLEX, 1.0,
(0,65,190), 2)
cv.imshow('Text', blank)
cv.ellipse(blank, (250,250), (90,50), 0, 0, 360, (0,0,255), 10)
cv.imshow('ellipse', blank)

cv.waitKey(0)
```

OUTPUT IMAGE:

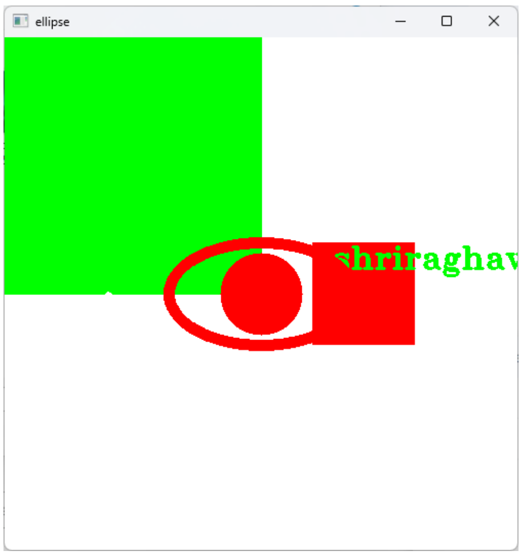


5.DRAW ON A WHITE BLANK IMAGE:

PROGRAM:

```
import cv2 as cv
import numpy as np

blank = np.zeros((500,500,3), dtype='uint8')
blank.fill(255)
cv.imshow('Blank', blank)

# 1. Paint the image a certain colour
blank[200:300, 300:400] = 0,0,255
cv.imshow('Green', blank)

# 2. Draw a Rectangle
cv.rectangle(blank, (0,0), (blank.shape[1]//2, blank.shape[0]//2), (0,255,0), thickness=-1)
cv.imshow('Rectangle', blank)

# 3. Draw A circle
cv.circle(blank, (blank.shape[1]//2, blank.shape[0]//2), 40, (0,0,255), thickness=-1)
cv.imshow('Circle', blank)

# 4. Draw a line
cv.line(blank, (100,250), (300,400), (255,255,255), thickness=3)
cv.imshow('Line', blank)

# 5. Write text
cv.putText(blank, ' SHRI RAGHAVI!', (0,225), cv.FONT_HERSHEY_TRIPLEX, 1.0, (0,255,0), 2)
cv.imshow('Text', blank)
cv.ellipse(blank, (250,250), (90,50), 0, 0, 360, (0,0,255), 10)
cv.imshow('ellipse', blank)

cv.waitKey(0)
```

OUTPUT IMAGE:



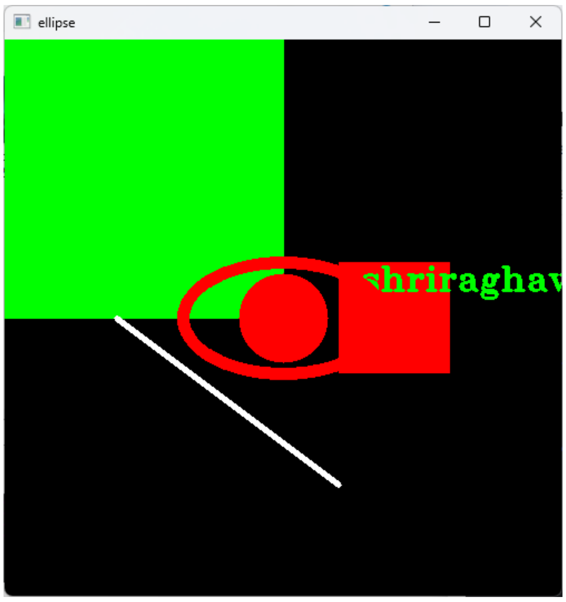
6.DRAW ON A BLACK BLANK IMAGE:

PROGRAM:

```
import cv2 as cv
import numpy as np
blank = np.zeros((500,500,3), dtype='uint8')
cv.imshow('Blank', blank)

# 1. Paint the image a certain colour
blank[200:300, 300:400] = 0,0,255
cv.imshow('Green', blank)

# 2. Draw a Rectangle
cv.rectangle(blank, (0,0), (blank.shape[1]//2, blank.shape[0]//2), (0,255,0), thickness=-1)
cv.imshow('Rectangle', blank)

# 3. Draw A circle
cv.circle(blank, (blank.shape[1]//2, blank.shape[0]//2), 40, (0,0,255), thickness=-1)
cv.imshow('Circle', blank)

# 4. Draw a line
cv.line(blank, (100,250), (300,400), (255,255,255), thickness=3)
cv.imshow('Line', blank)
```

```
# 5. Write text
cv.putText(blank, '                shriraghavi!', (0,225), cv.FONT_HERSHEY_TRIPLEX, 1.0, (0,255,0), 2)
cv.imshow('Text', blank)

cv.ellipse(blank, (250,250), (90,50), 0, 0, 360, (0,0,255), 10)
cv.imshow('ellipse', blank)

cv.waitKey(0)
```

OUTPUT IMAGE:

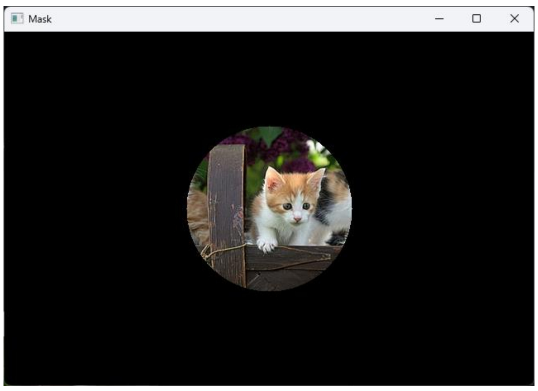


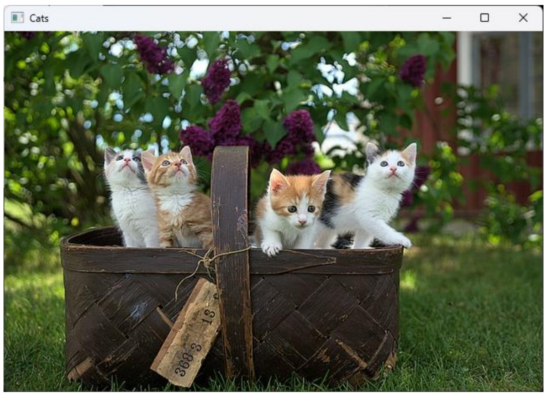
7.HISTOGRAM:

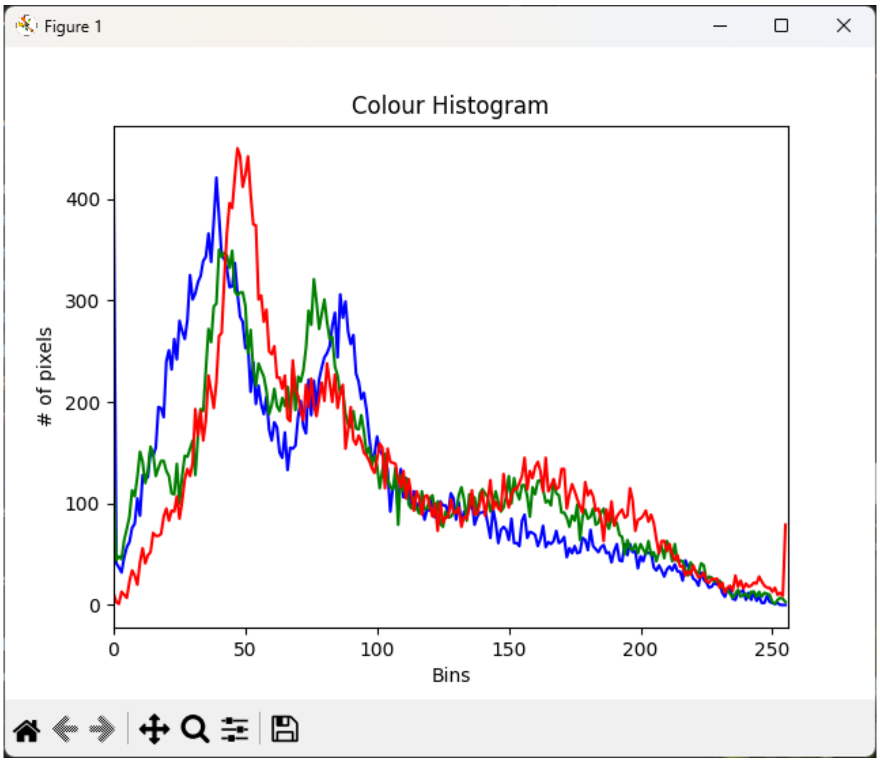
PROGRAM:

```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
img = cv.imread('cats.jpg')
cv.imshow('Cats', img)

blank = np.zeros(img.shape[:2], dtype='uint8')

mask = cv.circle(blank, (img.shape[1]//2,img.shape[0]//2), 100, 255, -1)
masked = cv.bitwise_and(img,img,mask=mask)
cv.imshow('Mask', masked)

# Colour Histogram
plt.figure()
plt.title('Colour Histogram')
plt.xlabel('Bins')
plt.ylabel('# of pixels')
colors = ('b', 'g', 'r')

for i,col in enumerate(colors):
    hist = cv.calcHist([img], [i], mask, [256], [0,256])
    plt.plot(hist, color=col)
    plt.xlim([0,256])
plt.show()

equ = cv.equalizeHist(img)
res = np.hstack((img,equ))
#stacking images side-by-side
cv.imwrite('res.png',res)
cv.imshow("result",res)
cv.waitKey(0)
```

OUTPUT IMAGE:







8.FINDING COORDINATES OF AN IMAGE:

PROGRAM:

```
import cv2 as cv
# Function to handle mouse events
def click_event(event, x, y, flags, param):
    if event == cv.EVENT_LBUTTONDOWN:
        print(f"Coordinates: ({x}, {y})")
# Read an image from file
image = cv.imread("41-KzR9ASUL.jpg")
# Display the image
cv.imshow("Image", image)
# Set the callback function for mouse events
cv.setMouseCallback("Image", click_event)
# Wait for a key press and close the window
cv.waitKey(0)
cv.destroyAllWindows()
```

OUTPUT:

```
Coordinates: (218, 195)
Coordinates: (161, 148)
Coordinates: (190, 209)
Coordinates: (247, 95)
Coordinates: (194, 92)
```

Department of RAE			
Criteria	**Excellent (75% - 100%)**	**Good (50 – 75%)**	**Poor (<50%)**
Preparation (30)			
Performance (30)			
Evaluation (20)			
Report (20)			
Sign:		**Total (100)**	

Result:

Thus the basic image processing concepts were learnt using OpenCV in Pycharm IDE.