1) Program to color to grayscale image converstion.

2) Program for ostu thresholding.

---

1) Pro for simple thresholding.

2) pro read and display images

---

1) program for subtraction of two images

2) pro to performs bitwise operations on images

---

1) program for addition of two images

2) pro to filter the images using a sobel filter

1) Program to color to grayscale image converstion.

```
import cv2
image = cv2.imread('C:\\Documents\\full_path\\tomatoes.jpg')
cv2.imshow('Original', image)
cv2.waitKey()
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Grayscale', gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2) Program for ostu thresholding

```
import cv2
import numpy as np
image1 = cv2.imread('input1.jpg')
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY +
                        cv2.THRESH_OTSU)
cv2.imshow('Otsu Threshold', thresh1)
if cv2.waitKey(0) & 0xff == 27:
  cv2.destroyAllWindows()
```

1)) Pro for simple thresholding

```
import cv2
import numpy as np
image1 = cv2.imread('input1.jpg')
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY_INV)
ret, thresh3 = cv2.threshold(img, 120, 255, cv2.THRESH_TRUNC)
ret, thresh4 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO)
ret, thresh5 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO_INV)
cv2.imshow('Binary Threshold', thresh1)
cv2.imshow('Binary Threshold Inverted', thresh2)
cv2.imshow('Truncated Threshold', thresh3)
cv2.imshow('Set to 0', thresh4)
cv2.imshow('Set to 0 Inverted', thresh5)
if cv2.waitKey(0) & 0xff == 27:
  cv2.destroyAllWindows()
```

2) pro read and display images

```
import  cv2
import matplotlib.pyplot as plt % matplotlib inline
first_img = cv2.imread("C://gfg//image_processing//players.jpg")
cv2.imshow("first_img", first_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1) program for subtraction of two images

```
# Two images size must be same
import  cv2
import matplotlib.pyplot as plt % matplotlib inline
first_img = cv2.imread("C://gfg//image_processing//players.jpg")
second_img = cv2.imread("C://gfg//image_processing//tomatoes.jpg")
print(first_img.shape)
print(second_img.shape)
# we need to resize, as they differ in shape
dim =(544, 363)
resized_second_img = cv2.resize(second_img, dim, interpolation = cv2.INTER_AREA)
print("shape after resizing", resized_second_img.shape)
subtracted = cv2.subtract(first_img, resized_second_img)
cv2.imshow("first_img", first_img)
cv2.waitKey(0)
cv2.imshow("second_img", resized_second_img)
cv2.waitKey(0)
cv2.imshow("subtracted image", subtracted)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2) pro to performs bitwise operations on images

```
import cv2
import numpy as np
img1 = np.zeros((300, 300), dtype="uint8")
cv2.rectangle(img1, (100, 100), (200, 200), 100, -1)
cv2.imshow("Image 1", img1)
img2 = np.zeros((300, 300), dtype="uint8")
cv2.circle(img2, (100, 100), 90, 100, -1)
cv2.imshow("Image 2", img2)
rect_and_circle = cv2.bitwise_and(img1,img2)
cv2.imshow("AND operation",rect_and_circle)
rect_or_circle = cv2.bitwise_or(img1,img2)
cv2.imshow("OR operation",rect_or_circle)
rect_xor_circle = cv2.bitwise_xor(img1,img2)
cv2.imshow("XOR Operation",rect_xor_circle)
rect_not_circle = cv2.bitwise_not(img1,img2)
cv2.imshow("not Operation",rect_not_circle)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1) program for addition of two images

```
# Two images size must be same
import  cv2
import matplotlib.pyplot as plt % matplotlib inline
# matplotlib can be used to plot the images as subplot

first_img = cv2.imread("C://gfg//image_processing//players.jpg")
second_img = cv2.imread("C://gfg//image_processing//tomatoes.jpg")

print(first_img.shape)
print(second_img.shape)

# we need to resize, as they differ in shape
dim =(544, 363)
resized_second_img = cv2.resize(second_img, dim, interpolation = cv2.INTER_AREA)
print("shape after resizing", resized_second_img.shape)

added_img = cv2.add(first_img, resized_second_img)

cv2.imshow("first_img", first_img)
cv2.waitKey(0)
cv2.imshow("second_img", resized_second_img)
cv2.waitKey(0)
cv2.imshow("Added image", added_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2) pro to filter the images using a sobel filter

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img0 = cv2.imread('SanFrancisco.jpg',)
img0 = cv2.imread('C:/Users/HP_LAPTOP/Desktop/images/8.jpg')
gray = cv2.cvtColor(img0, cv2.COLOR_BGR2GRAY)
img = cv2.GaussianBlur(gray,(3,3),0)
laplacian = cv2.Laplacian(img,cv2.CV_64F)
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=5)  # x
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=5)  # y
plt.subplot(2,2,1),plt.imshow(img,cmap = 'gray')
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
plt.title('Laplacian'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,3),plt.imshow(sobelx,cmap = 'gray')
plt.title('Sobel X'), plt.xticks([]), plt.yticks([])
plt.subplot(2,2,4),plt.imshow(sobely,cmap = 'gray')
plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])
plt.show()
```