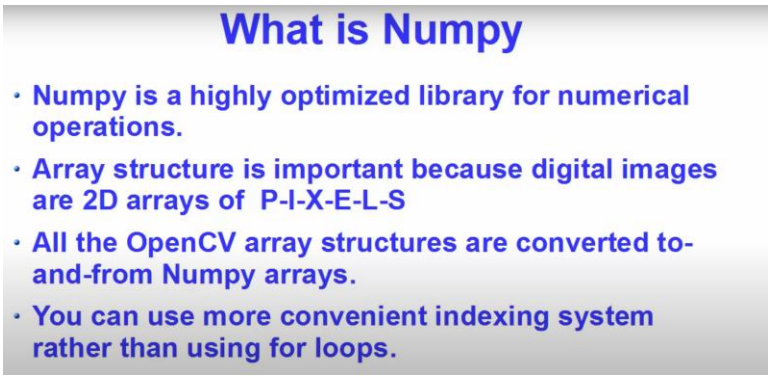
Computer Vision:

OPENCV – open source computer vision library

Computer vision is the way of teaching intelligence to machines and making them see things just like humans.

In a simplest form computer allows to see and process visual data just like humans

Example: self driving car – detecting the lanes, it detects the speed sign boards and many more.



Read an image:

Grey Scale image:

```
import cv2

img = cv2.imread('lena.jpg', 0)

print(img)

cv2.imshow('image', img)
cv2.waitKey(5000)
cv2.destroyAllWindows()
```

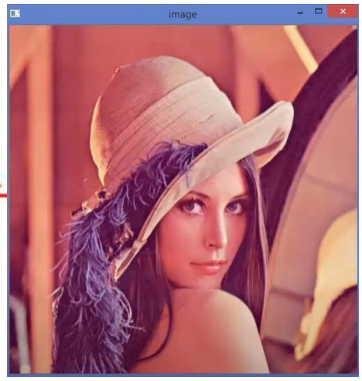
The window will close after 5 seconds

```
import cv2

img = cv2.imread('lena.jpg', 0)

print(img)

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

In this case the window will not close if we set the value of waitkey is zero.

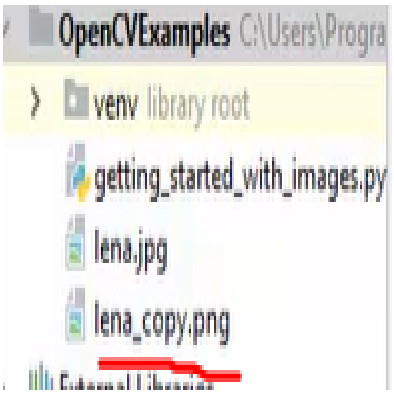
Colored image:

```
import cv2

img = cv2.imread('lena.jpg', 1)

print(img)

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



Write Function:

```
import cv2

img = cv2.imread('lena.jpg', -1)

print(img)

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

cv2.imwrite('lena_copy.png', img)
```



Modifications:

```
import cv2

img = cv2.imread('lena.jpg', -1)
cv2.imshow('image', img)
k = cv2.waitKey(0)

if k == 27:
    cv2.destroyAllWindows()
elif k == ord('s'):
    cv2.imwrite('lena_copy.png', img)
    cv2.destroyAllWindows()
```

In this code, I have mentioned k == 27 which means the escape button has the value of 27. If the user press escape button it close the window. If the user press the s button then it creates or writes the file with the file name that we had given previously.

Another method: To destroy the window

```
import cv2

img = cv2.imread('lena.jpg', -1)
cv2.imshow('image', img)
k = cv2.waitKey(0) & 0xFF

if k == 27:
    cv2.destroyAllWindows()
elif k == ord('s'):
    cv2.imwrite('lena_copy.png', img)
    cv2.destroyAllWindows()
```

Camera:

RGB:

```
import cv2

cap = cv2.VideoCapture(0);

while(True):
    ret, frame = cap.read()

    cv2.imshow('frame', frame)

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

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

Grey scale:

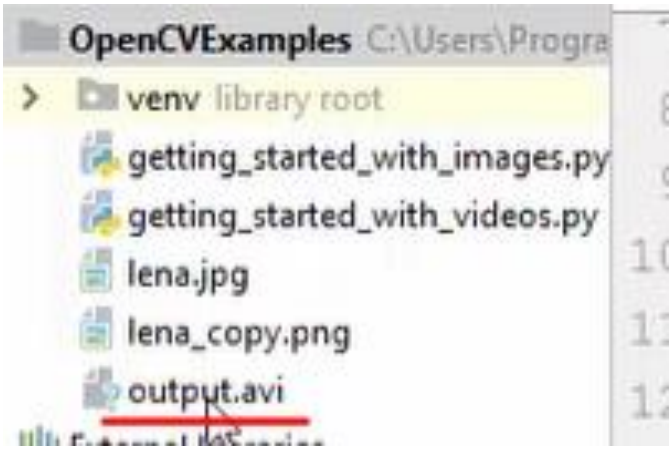
```
import cv2

cap = cv2.VideoCapture(0);

while(True):
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('frame', gray)

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

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

Video file:

```
cap = cv2.VideoCapture('name.avi');
```

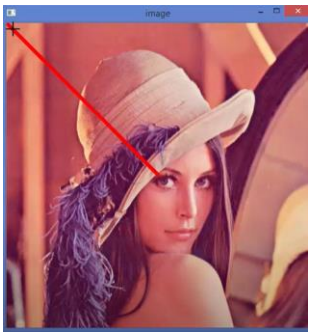
Video writer:

```python
import cv2

cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640,480))

print(cap.isOpened())
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:
        print(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        print(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

        out.write(frame)

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        cv2.imshow('frame', gray)

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

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



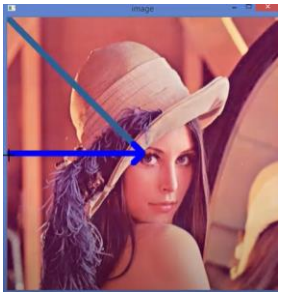
Draw geometric shapes:

```
import numpy as np
import cv2

img = cv2.imread('lena.jpg', 1)

img = cv2.line(img, (0,0), (255,255), (0, 0, 255), 5)

cv2.imshow('image', img)

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



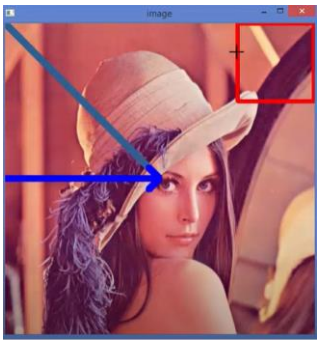
Arrowed line:

```
import cv2

img = cv2.imread('lena.jpg', 1)

img = cv2.line(img, (0,0), (255,255), (147, 96, 44), 10) # 44, 96, 147
img = cv2.arrowedLine(img, (0,255), (255,255), (255, 0, 0), 10)

cv2.imshow('image', img)

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



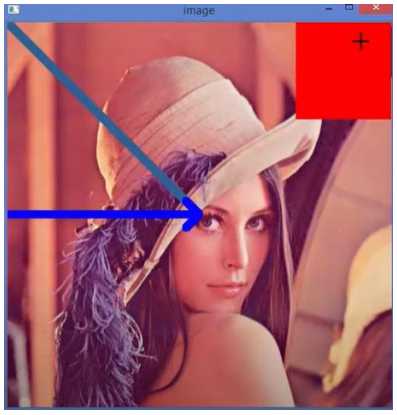
Rectangle:

```
import cv2

img = cv2.imread('lena.jpg', 1)

img = cv2.line(img, (0,0), (255,255), (147, 96, 44), 10) # 44,
img = cv2.arrowedLine(img, (0,255), (255,255), (255, 0, 0), 10)

img = cv2.rectangle(img, (384, 0), (510, 128), (0, 0, 255), 5)

cv2.imshow('image', img)

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



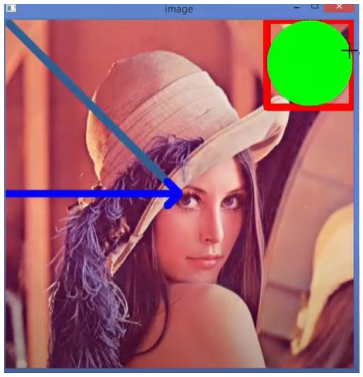
Fill color within the rectangle:

```
import cv2

img = cv2.imread('lena.jpg', 1)

img = cv2.line(img, (0,0), (255,255), (147, 96, 44), 10) # 44, 96, 147
img = cv2.arrowedLine(img, (0,255), (255,255), (255, 0, 0), 10)

img = cv2.rectangle(img, (384, 0), (510, 128), (0, 0, 255), -1)

cv2.imshow('image', img)

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



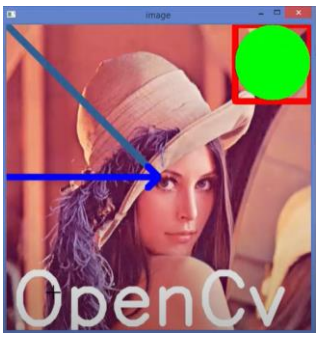
Circle:

```
import cv2

img = cv2.imread('lena.jpg', 1)

img = cv2.line(img, (0,0), (255,255), (147, 96, 44), 10) # 44,
img = cv2.arrowedLine(img, (0,255), (255,255), (255, 0, 0), 10)

img = cv2.rectangle(img, (384, 0), (510, 128), (0, 0, 255), 10)
img = cv2.circle(img, (447, 63), 63, (0, 255, 0), -1)

cv2.imshow('image', img)

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



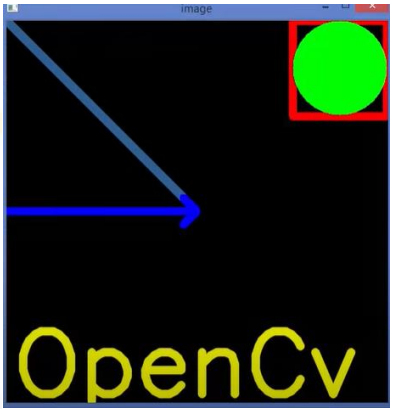
Text:

```
import cv2

img = cv2.imread('lena.jpg', 1)

img = cv2.line(img, (0,0), (255,255), (147, 96, 44), 10) # 44, 96, 147
img = cv2.arrowedLine(img, (0,255), (255,255), (255, 0, 0), 10)

img = cv2.rectangle(img, (384, 0), (510, 128), (0, 0, 255), 10)
img = cv2.circle(img, (447, 63), 63, (0, 255, 0), -1)
font = cv2.FONT_HERSHEY_SIMPLEX
img = cv2.putText(img, 'OpenCv', (10, 500), font, 4, (255, 255, 255), 10, cv2.LINE_AA)

cv2.imshow('image', img)

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



Create our own window:

```
import numpy as np
import cv2

#img = cv2.imread('lena.jpg', 1)
img = np.zeros([512, 512, 3], np.uint8)

img = cv2.line(img, (0,0), (255,255), (147, 96, 44), 10) # 44, 96, 147
img = cv2.arrowedLine(img, (0,255), (255,255), (255, 0, 0), 10)

img = cv2.rectangle(img, (384, 0), (510, 128), (0, 0, 255), 10)
img = cv2.circle(img, (447, 63), 63, (0, 255, 0), -1)
font = cv2.FONT_HERSHEY_SIMPLEX
img = cv2.putText(img, 'OpenCv', (10, 500), font, 4, (0, 255, 255), 10, cv2.LINE_AA)

cv2.imshow('image', img)

cv2.waitKey(0)
```



Setting camera parameters:

```
import cv2
cap = cv2.VideoCapture(0)
print(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
print(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

cap.set(3, 1208)
cap.set(4, 720)

print(cap.get(3))
print(cap.get(4))
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        cv2.imshow('frame', gray)

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

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

Show width and height on the videos:

```python
import cv2
cap = cv2.VideoCapture(0)
print(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
print(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
cap.set(3, 3000)
cap.set(4, 3000)
print(cap.get(3))
print(cap.get(4))
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:

        font = cv2.FONT_HERSHEY_SIMPLEX
        text = 'Width: '+ str(cap.get(3)) + ' Height:' + str(cap.get(4))
        frame = cv2.putText(frame, text, (10, 50), font, 1,
                            (0, 255, 255), 2, cv2.LINE_AA)
        cv2.imshow('frame', frame)

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

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

Date time

```python
import cv2
import datetime
cap = cv2.VideoCapture(0)
print(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
print(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
#cap.set(3, 3000)
#cap.set(4, 3000)
#print(cap.get(3))
#print(cap.get(4))
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:

        font = cv2.FONT_HERSHEY_SIMPLEX
        text = 'Width: '+ str(cap.get(3)) + ' Height:' + str(cap.get(4))
        datet = str(datetime.datetime.now())
        frame = cv2.putText(frame, datet, (10, 50), font, 1,
                            (0, 255, 255), 2, cv2.LINE_AA)
        cv2.imshow('frame', frame)

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

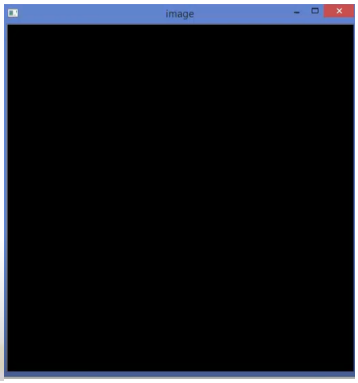
Handle Mouse events:

List of events

```
import cv2

events = [i for i in dir(cv2) if 'EVENT' in  i]
print(events)
```

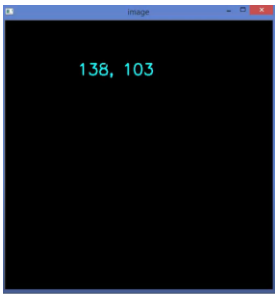
```
C:\Users\ProgrammingKnowledge\PycharmProjects\OpenCVExamples\venv\Scripts\python.exe C:/Users/ProgrammingKno
['EVENT_FLAG_ALTKEY', 'EVENT_FLAG_CTRLKEY', 'EVENT_FLAG_LBUTTON', 'EVENT_FLAG_MBUTTON', 'EVENT_FLAG_RBUTTON'
```

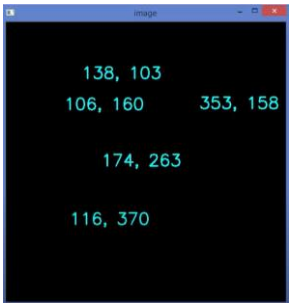
```
import numpy as np
import cv2

#events = [i for i in dir(cv2) if 'EVENT' in  i]
#print(events)

def click_event(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        print(x,', ',y)
        font = cv2.FONT_HERSHEY_SIMPLEX
        strXY = str(x) + ', '+ str(y)
        cv2.putText(img, strXY, (x, y), font, 1, (255, 255, 0), 2)
        cv2.imshow('image', img)

img = np.zeros((512, 512, 3), np.uint8)
cv2.imshow('image', img)

cv2.setMouseCallback('image', click_event)

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



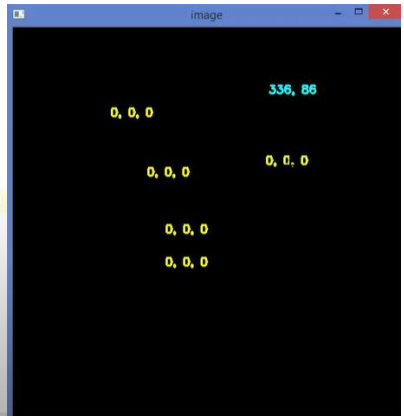
If I click anywhere on the screen then it will display the coordinates of the position

Where I clicked on the black screen.



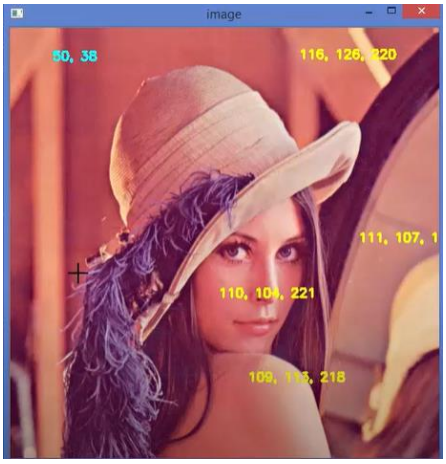


```python
def click_event(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        print(x,', ',y)
        font = cv2.FONT_HERSHEY_SIMPLEX
        strXY = str(x) + ', '+ str(y)
        cv2.putText(img, strXY, (x, y), font, .5, (255, 255, 0), 2)
        cv2.imshow('image', img)
    if event == cv2.EVENT_RBUTTONDOWN:
        blue = img[y, x, 0]
        green = img[y, x, 1]
        red = img[y, x, 2]
        font = cv2.FONT_HERSHEY_SIMPLEX
        strBGR = str(blue) + ', '+ str(green)+ ', '+ str(red)
        cv2.putText(img, strBGR, (x, y), font, .5, (0, 255, 255), 2)
        cv2.imshow('image', img)

img = np.zeros((512, 512, 3), np.uint8)
cv2.imshow('image', img)

cv2.setMouseCallback('image', click_event)

cv2.waitKey(0)
```



Now change the black image into some BGR image by opening some of the file.

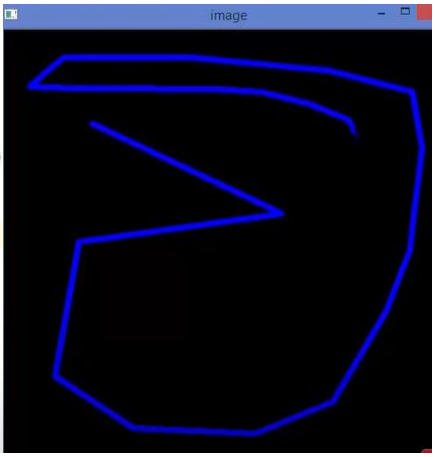
```python
#img = np.zeros((512, 512, 3), np.uint8)
img = cv2.imread('lena.jpg')
cv2.imshow('image', img)
```



More mouse click events:

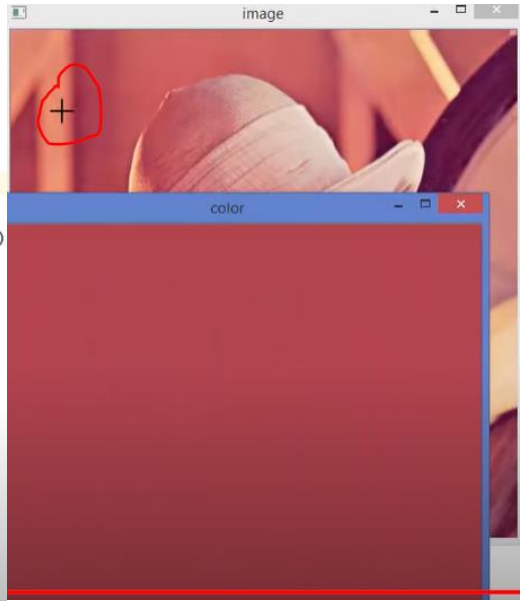
Drawing and connecting the points using line:

```python
import numpy as np
import cv2

def click_event(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(img, (x, y), 3, (0, 0, 255), -1)
        points.append((x, y))
        if len(points) >=2:
            cv2.line(img, points[-1], points[-2], (255, 0, 0), 5)
        cv2.imshow('image', img)

img = np.zeros((512, 512, 3), np.uint8)
#img = cv2.imread('lena.jpg')
cv2.imshow('image', img)
points = []
cv2.setMouseCallback('image', click_event)

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



Read an image and do click events(Color picker):

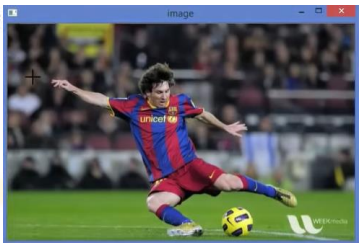
```python
import numpy as np
import cv2

def click_event(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        blue = img[x, y, 0]
        green = img[x, y, 1]
        red = img[x, y , 2]
        cv2.circle(img, (x, y), 3, (0, 0, 255), -1)
        mycolorImage = np.zeros((512, 512, 3), np.uint8)

        mycolorImage[:] = [blue, green, red]

        cv2.imshow('color', mycolorImage)

#img = np.zeros((512, 512, 3), np.uint8)
img = cv2.imread('lena.jpg')
cv2.imshow('image', img)
points = []
cv2.setMouseCallback('image', click_event)

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



Basic arithmetic operations using opencv:

Shape, size, dtype:

```
import cv2

img = cv2.imread('messi5.jpg')

print(img.shape)# returns a tuple of number of rows, columns, and channels
print(img.size)# returns Total number of pixels is accessed
print(img.dtype)# returns Image datatype is obtained
b,g,r = cv2.split(img)
img = cv2.merge((b,g,r))

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




```
C:\Users\ProgrammingKn
(342, 548, 3)
562248
uint8
```

ROI – Region of Interest:

```
import cv2

img = cv2.imread('messi5.jpg')

print(img.shape)# returns a tuple of number of rows, columns, and channel
print(img.size)# returns Total number of pixels is accessed
print(img.dtype)# returns Image datatype is obtained
b,g,r = cv2.split(img)
img = cv2.merge((b,g,r))

ball = img[280:340, 330:390]
img[273:333, 100:160] = ball

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

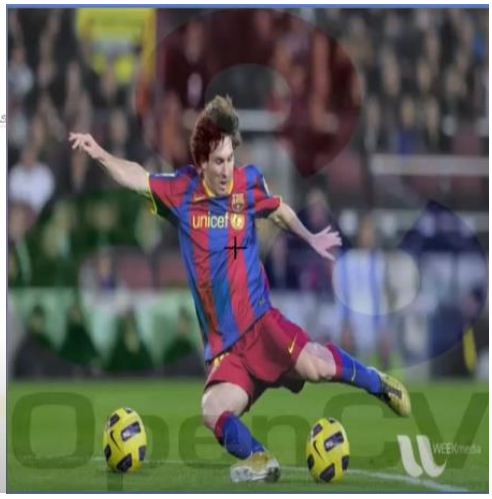
Add two images:

Make sure: both images should contain same size, so we need to resize the image

```
import cv2

img = cv2.imread('messi5.jpg')
img2 = cv2.imread('opencv-logo.png')

print(img.shape)# returns a tuple of n
print(img.size)# returns Total number
print(img.dtype)# returns Image datat
b,g,r = cv2.split(img)
img = cv2.merge((b,g,r))

ball = img[280:340, 330:390]
img[273:333, 100:160] = ball

img = cv2.resize(img, (512, 512))
img2 = cv2.resize(img2, (512, 512))

dst = cv2.add(img, img2);

cv2.imshow('image', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Add weightage to the image:

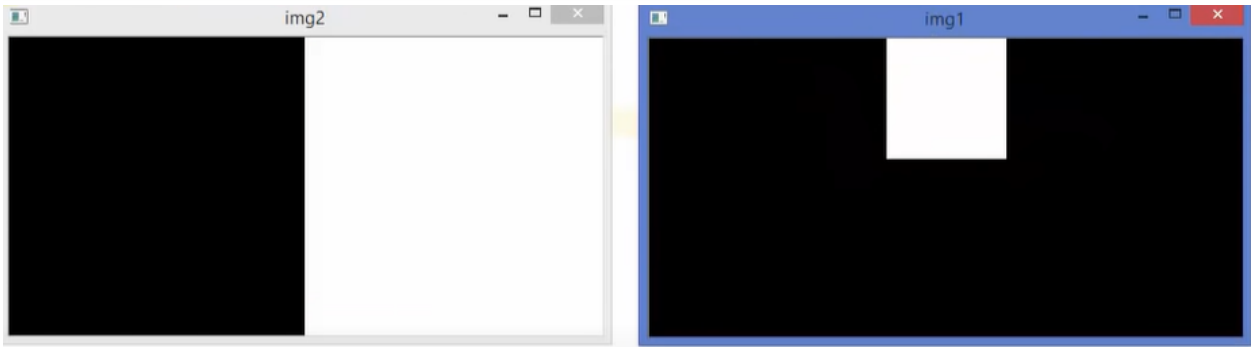
```
import cv2

img = cv2.imread('messi5.jpg')
img2 = cv2.imread('opencv-logo.png')

print(img.shape)# returns a tuple of number of rows, columns, and channels
print(img.size)# returns Total number of pixels is accessed
print(img.dtype)# returns Image datatype is obtained
b,g,r = cv2.split(img)
img = cv2.merge((b,g,r))

ball = img[280:340, 330:390]
img[273:333, 100:160] = ball

img = cv2.resize(img, (512, 512))
img2 = cv2.resize(img2, (512, 512))

#dst = cv2.add(img, img2);
dst = cv2.addWeighted(img, .9, img2, .1, 0);
cv2.imshow('image', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Bitwise operations on images:

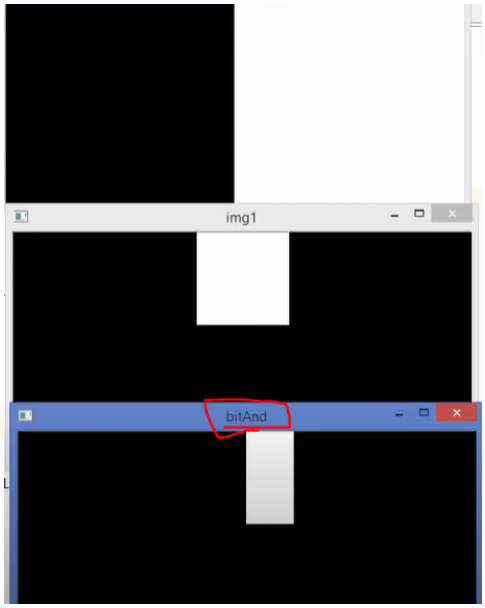
In this example, I have created the image with same size of the image 1.

```python
import cv2
import numpy as np

img1 = np.zeros((250, 500, 3), np.uint8)
img1 = cv2.rectangle(img1,(200, 0), (300, 100), (255, 255, 255), -1)
img2 = cv2.imread("image_1.png")

cv2.imshow("img1", img1)
cv2.imshow("img2", img2)

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



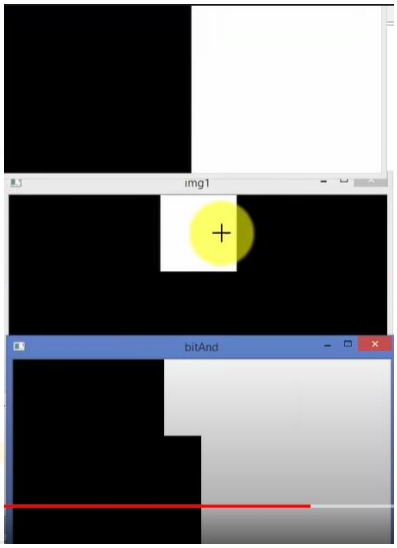
Bitwise and:

```python
import cv2
import numpy as np

img1 = np.zeros((250, 500, 3), np.uint8)
img1 = cv2.rectangle(img1,(200, 0), (300, 100), (255, 255, 255), -1)
img2 = cv2.imread("image_1.png")

bitAnd = cv2.bitwise_and(img2, img1)

cv2.imshow("img1", img1)
cv2.imshow("img2", img2)
cv2.imshow('bitAnd', bitAnd)

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



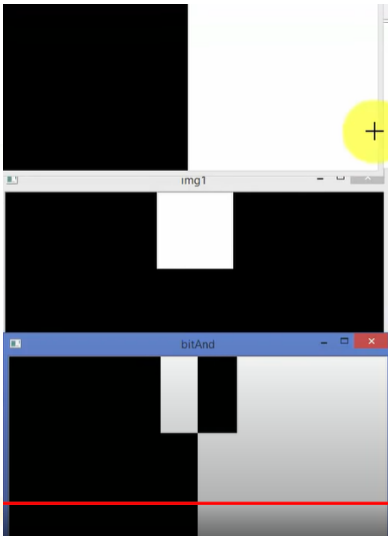
Bitwise or:

```python
import cv2
import numpy as np

img1 = np.zeros((250, 500, 3), np.uint8)
img1 = cv2.rectangle(img1,(200, 0), (300, 100), (255, 255, 255), -1)
img2 = cv2.imread("image_1.png")

#bitAnd = cv2.bitwise_and(img2, img1)
bitOr = cv2.bitwise_or(img2, img1)

cv2.imshow("img1", img1)
cv2.imshow("img2", img2)
#cv2.imshow('bitAnd', bitAnd)
cv2.imshow('bitAnd', bitOr)

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



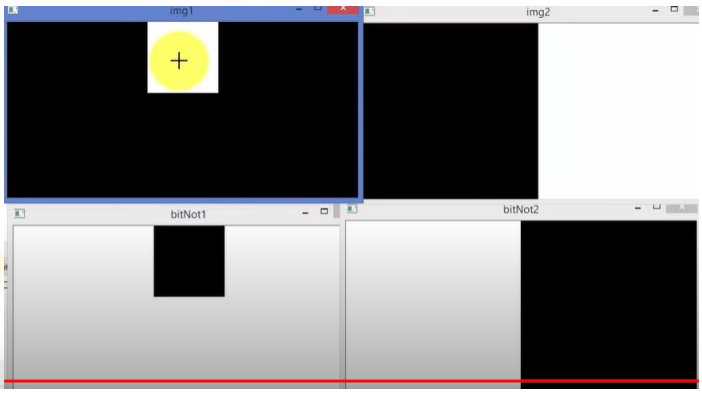
Bit XOR

```python
import cv2
import numpy as np

img1 = np.zeros((250, 500, 3), np.uint8)
img1 = cv2.rectangle(img1,(200, 0), (300, 100), (255, 255, 255), -1)
img2 = cv2.imread("image_1.png")

#bitAnd = cv2.bitwise_and(img2, img1)
#bitOr = cv2.bitwise_or(img2, img1)
bitXor = cv2.bitwise_xor(img1, img2)

cv2.imshow("img1", img1)
cv2.imshow("img2", img2)
#cv2.imshow('bitAnd', bitAnd)
#cv2.imshow('bitAnd', bitOr)
cv2.imshow('bitAnd', bitXor)
```



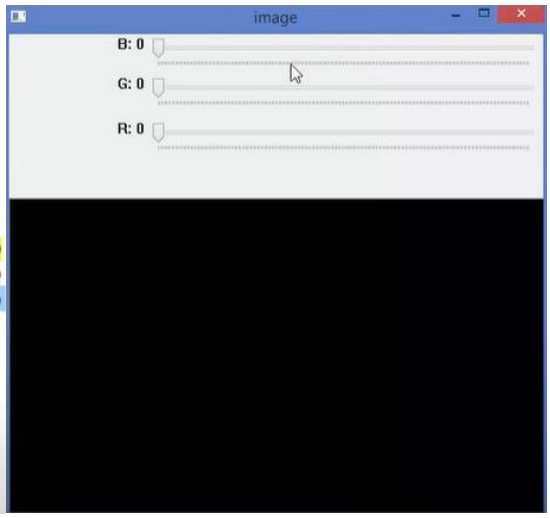
Bitwise Not:

```python
bitNot1 = cv2.bitwise_not(img1)
bitNot2 = cv2.bitwise_not(img2)

cv2.imshow("img1", img1)
cv2.imshow("img2", img2)
#cv2.imshow('bitAnd', bitAnd)
#cv2.imshow('bitOr', bitOr)
#cv2.imshow('bitXor', bitXor)
cv2.imshow('bitNot1', bitNot1)
cv2.imshow('bitNot2', bitNot2)
```

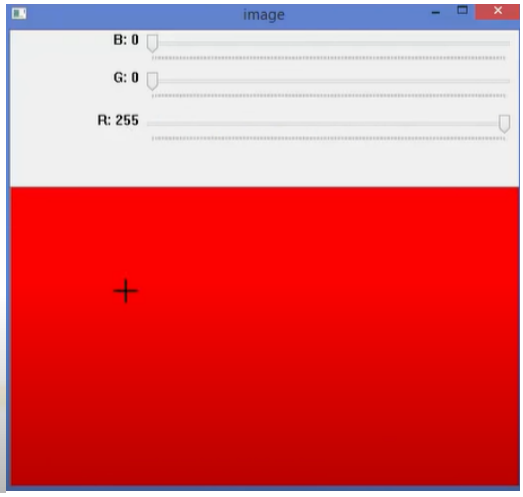


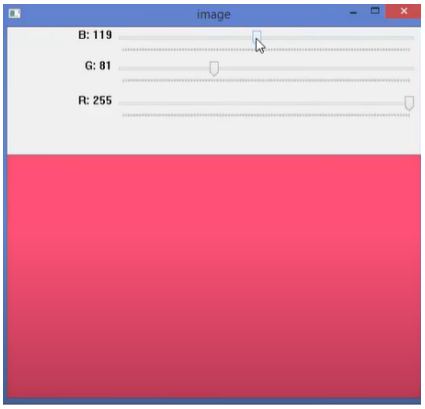
Trackbars:

```python
import numpy as np
import cv2 as cv

def nothing(x):
    print(x)

# Create a black image, a window
img = np.zeros((300,512,3), np.uint8)
cv.namedWindow('image')

cv.createTrackbar('B', 'image', 0, 255, nothing)
cv.createTrackbar('G', 'image', 0, 255, nothing)
cv.createTrackbar('R', 'image', 0, 255, nothing)

while(1):
    cv.imshow('image',img)
    k = cv.waitKey(1) & 0xFF
    if k == 27:
        break
cv.destroyAllWindows()
```



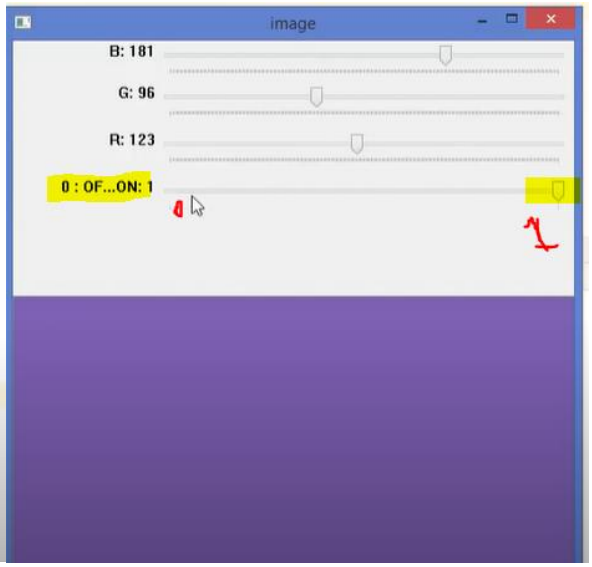

```python
# Create a black image, a window
img = np.zeros((300,512,3), np.uint8)
cv.namedWindow('image')

cv.createTrackbar('B', 'image', 0, 255, nothing)
cv.createTrackbar('G', 'image', 0, 255, nothing)
cv.createTrackbar('R', 'image', 0, 255, nothing)

while(1):
    cv.imshow('image',img)
    k = cv.waitKey(1) & 0xFF
    if k == 27:
        break

    b = cv.getTrackbarPos('B', 'image')
    g = cv.getTrackbarPos('G', 'image')
    r = cv.getTrackbarPos('R', 'image')

    img[:] = [b, g, r]

cv.destroyAllWindows()
```





Switch in the trackbar:

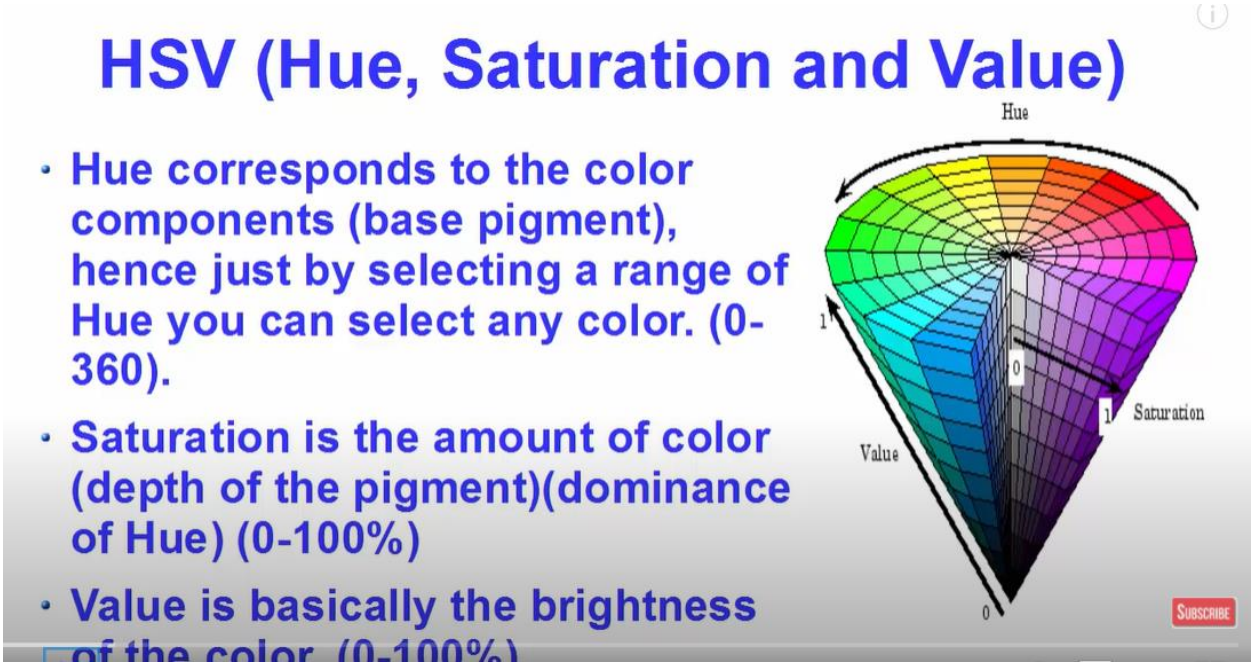
```
cv.createTrackbar('G', 'image', 0, 255, nothing)
cv.createTrackbar('R', 'image', 0, 255, nothing)

switch = '0 : OFF\n 1 : ON'
cv.createTrackbar(switch, 'image', 0, 1, nothing)

while(1):
    cv.imshow('image',img)
    k = cv.waitKey(1) & 0xFF
    if k == 27:
        break

    b = cv.getTrackbarPos('B', 'image')
    g = cv.getTrackbarPos('G', 'image')
    r = cv.getTrackbarPos('R', 'image')
    s = cv.getTrackbarPos(switch, 'image')

    if s == 0:
        img[:] = 0
    else:
        img[:] = [b, g, r]

cv.destroyAllWindows()
```



Another example:

```
import numpy as np
import cv2 as cv

def nothing(x):
    print(x)

# Create a black image, a window
cv.namedWindow('image')

cv.createTrackbar('CP', 'image', 10, 400, nothing)

switch = 'color/gray'
cv.createTrackbar(switch, 'image', 0, 1, nothing)

while(1):
    img = cv.imread('lena.jpg')
    pos = cv.getTrackbarPos('CP', 'image')
    font = cv.FONT_HERSHEY_SIMPLEX
    cv.putText(img, str(pos), (50, 150), font, 6, (0, 0, 255), 10)

    k = cv.waitKey(1) & 0xFF
    if k == 27:
        break

    s = cv.getTrackbarPos(switch, 'image')

    if s == 0:
       pass
    else:
       img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    img = cv.imshow('image',img)

cv.destroyAllWindows()
```

HSV:

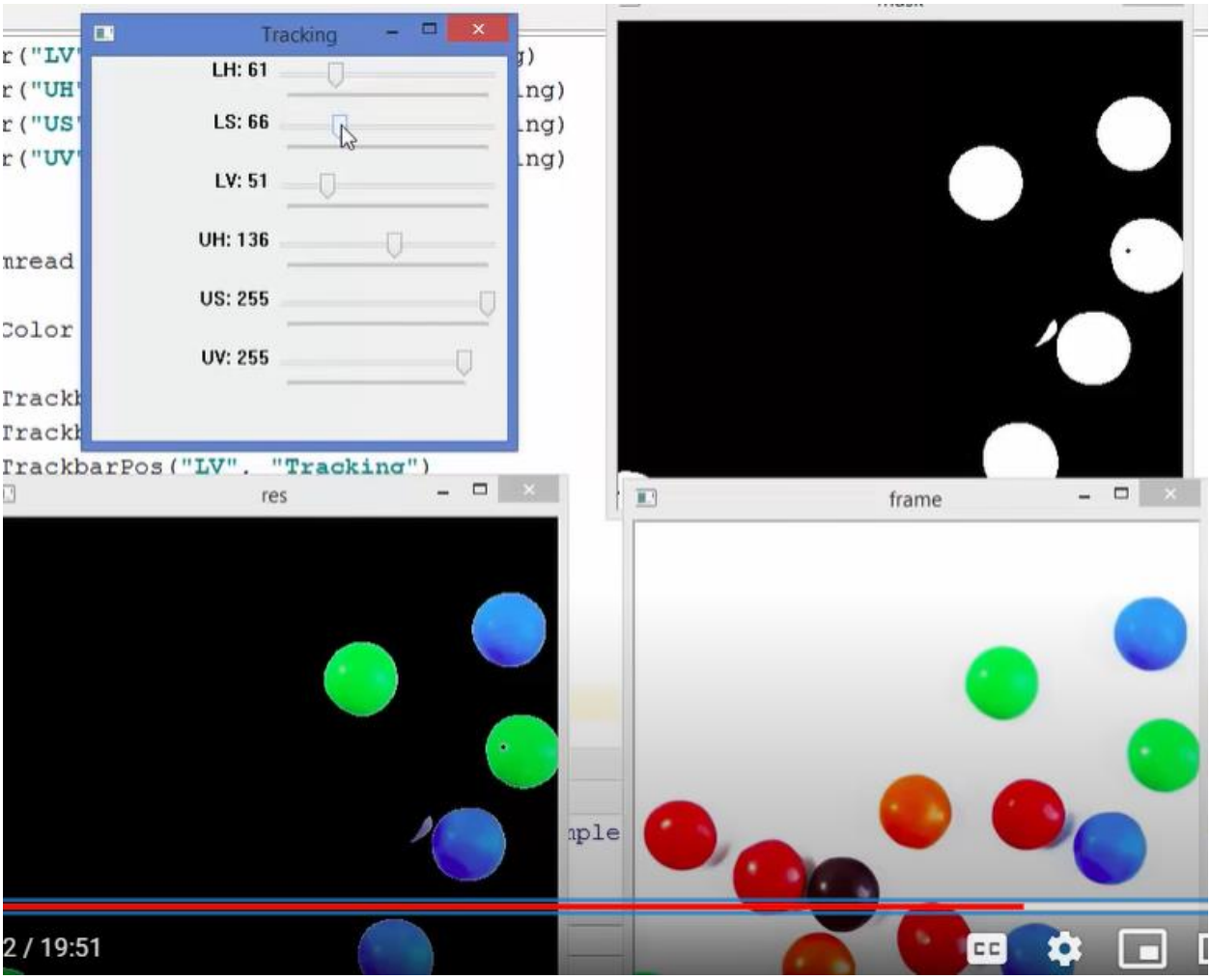


Object detection and tracking using HSV color space

opencv_python_object_detection.py

```
import
cv2
        import numpy as np

        def nothing(x):
            pass

        cv2.namedWindow("Tracking")
        cv2.createTrackbar("LH", "Tracking", 0, 255, nothing)
        cv2.createTrackbar("LS", "Tracking", 0, 255, nothing)
        cv2.createTrackbar("LV", "Tracking", 0, 255, nothing)
        cv2.createTrackbar("UH", "Tracking", 255, 255, nothing)
        cv2.createTrackbar("US", "Tracking", 255, 255, nothing)
        cv2.createTrackbar("UV", "Tracking", 255, 255, nothing)

        while True:
            frame = cv2.imread('smarties.png')

            hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

            l_h = cv2.getTrackbarPos("LH", "Tracking")
```

```
            l_s = cv2.getTrackbarPos("LS", "Tracking")
            l_v = cv2.getTrackbarPos("LV", "Tracking")

            u_h = cv2.getTrackbarPos("UH", "Tracking")
            u_s = cv2.getTrackbarPos("US", "Tracking")
            u_v = cv2.getTrackbarPos("UV", "Tracking")

            l_b = np.array([l_h, l_s, l_v])
            u_b = np.array([u_h, u_s, u_v])

            mask = cv2.inRange(hsv, l_b, u_b)

            res = cv2.bitwise_and(frame, frame, mask=mask)

            cv2.imshow("frame", frame)
            cv2.imshow("mask", mask)
            cv2.imshow("res", res)

            key = cv2.waitKey(1)
            if key == 27:
                Break

        cv2.destroyAllWindows()
```



opencv_python_object_detection_video.py

```
import cv2
import numpy as np

def nothing(x):
    pass

cap = cv2.VideoCapture(0);

cv2.namedWindow("Tracking")
cv2.createTrackbar("LH", "Tracking", 0, 255, nothing)
cv2.createTrackbar("LS", "Tracking", 0, 255, nothing)
cv2.createTrackbar("LV", "Tracking", 0, 255, nothing)
cv2.createTrackbar("UH", "Tracking", 255, 255, nothing)
```

```
cv2.createTrackbar("US", "Tracking", 255, 255, nothing)
cv2.createTrackbar("UV", "Tracking", 255, 255, nothing)

while True:
    #frame = cv2.imread('smarties.png')
    _, frame = cap.read()

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    l_h = cv2.getTrackbarPos("LH", "Tracking")
    l_s = cv2.getTrackbarPos("LS", "Tracking")
    l_v = cv2.getTrackbarPos("LV", "Tracking")

    u_h = cv2.getTrackbarPos("UH", "Tracking")
    u_s = cv2.getTrackbarPos("US", "Tracking")
    u_v = cv2.getTrackbarPos("UV", "Tracking")

    l_b = np.array([l_h, l_s, l_v])
    u_b = np.array([u_h, u_s, u_v])

    mask = cv2.inRange(hsv, l_b, u_b)

    res = cv2.bitwise_and(frame, frame, mask=mask)

    cv2.imshow("frame", frame)
    cv2.imshow("mask", mask)
    cv2.imshow("res", res)

    key = cv2.waitKey(1)
    if key == 27:
        break

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

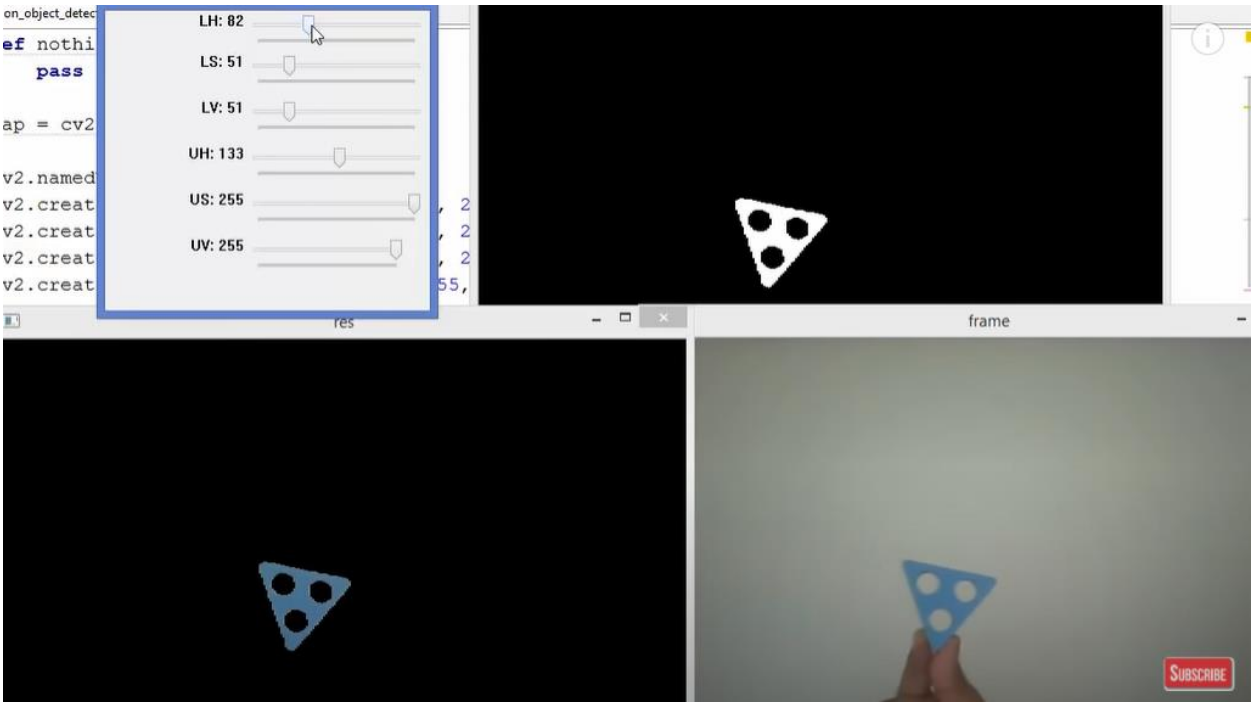
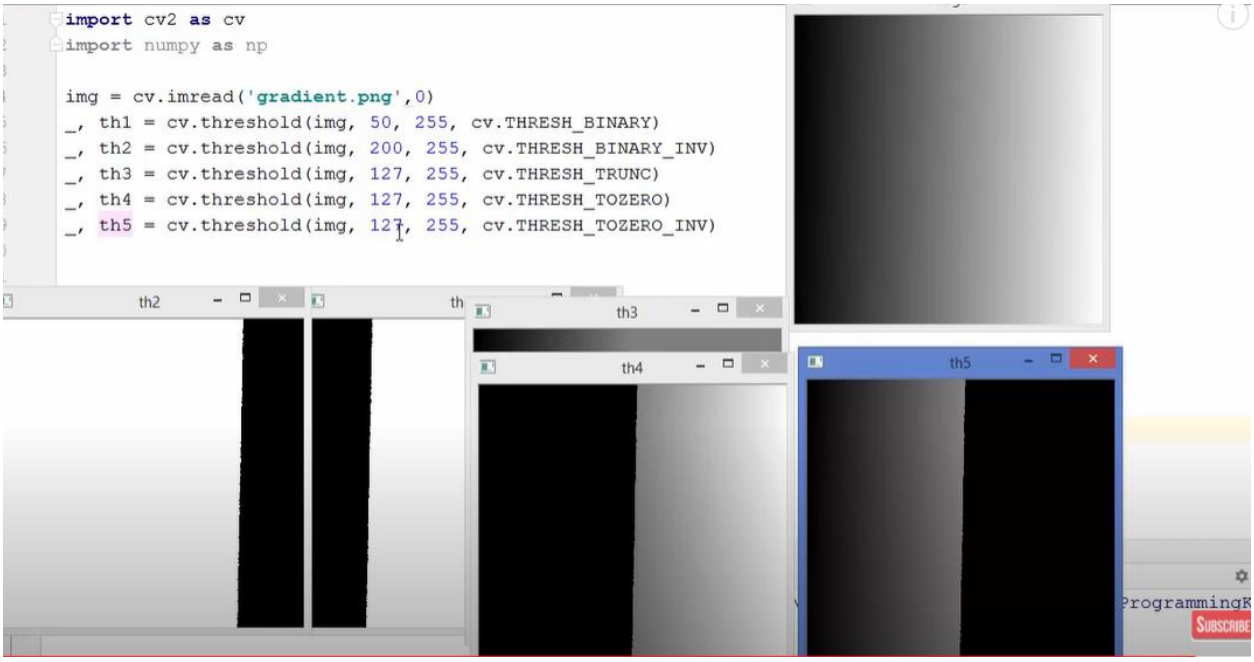


Image Thresholding:

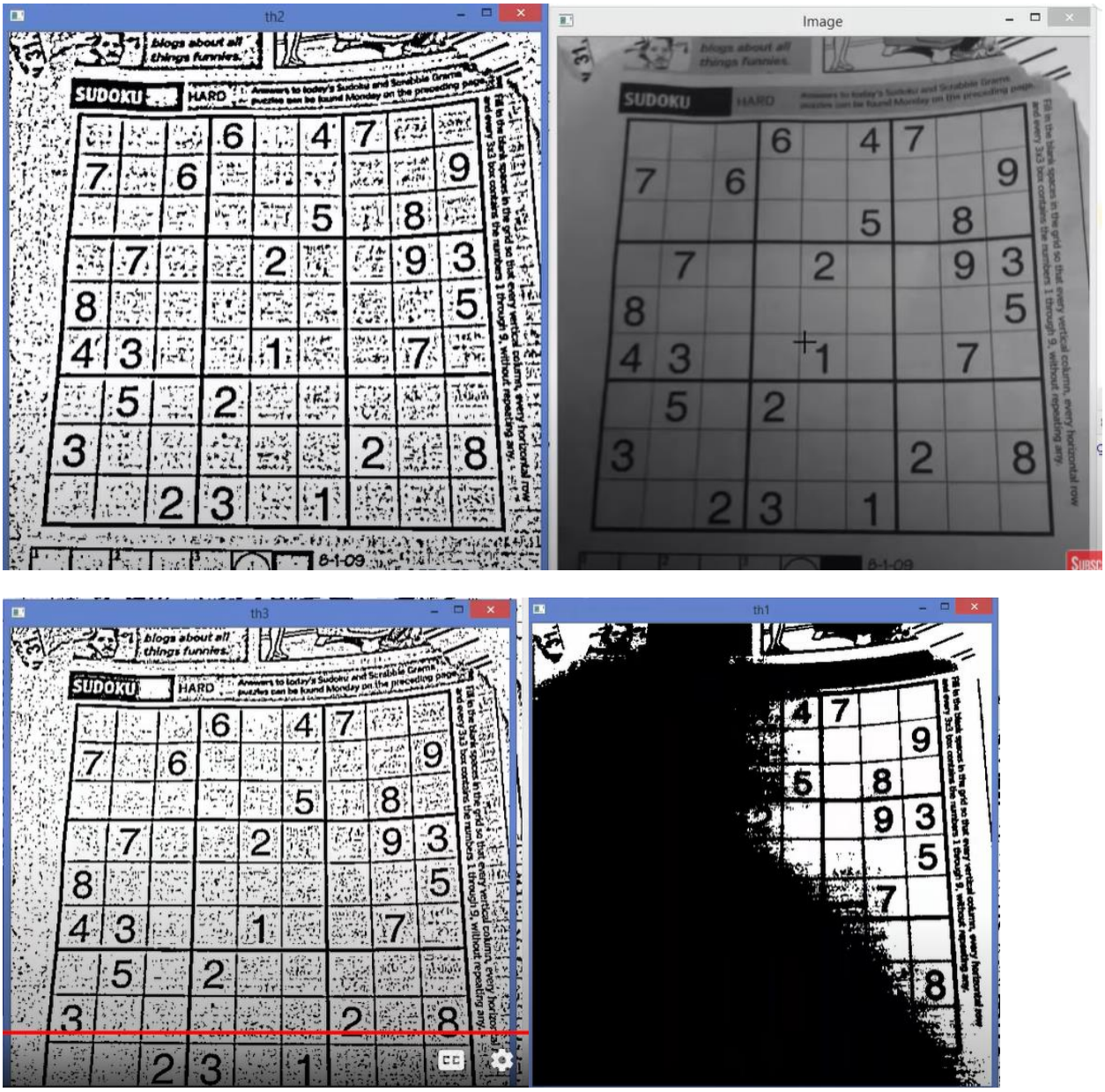
```
import cv2 as cv
import numpy as np

img = cv.imread('gradient.png',0)
_, th1 = cv.threshold(img, 50, 255, cv.THRESH_BINARY)
_, th2 = cv.threshold(img, 200, 255, cv.THRESH_BINARY_INV)
_, th3 = cv.threshold(img, 127, 255, cv.THRESH_TRUNC)
_, th4 = cv.threshold(img, 127, 255, cv.THRESH_TOZERO)
_, th5 = cv.threshold(img, 127, 255, cv.THRESH_TOZERO_INV)


cv.imshow("Image", img)
cv.imshow("th1", th1)
cv.imshow("th2", th2)
cv.imshow("th3", th3)
cv.imshow("th4", th4)
cv.imshow("th5", th5)

cv.waitKey(0)
```

```
        cv.destroyAllWindows()
```



Adaptive Thresholding:

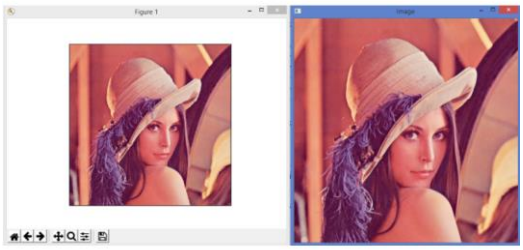
```
import
cv2 as
cv
        import numpy as np

        img = cv.imread('sudoku.png',0)
        _, th1 = cv.threshold(img, 127, 255, cv.THRESH_BINARY)
        th2 = cv.adaptiveThreshold(img, 255, cv.ADAPTIVE_THRESH_MEAN_C, cv.THRESH_BINARY,
        11, 2);
        th3 = cv.adaptiveThreshold(img, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv.THRESH_BINARY, 11, 2);

        cv.imshow("Image", img)
        cv.imshow("THRESH_BINARY", th1)
        cv.imshow("ADAPTIVE_THRESH_MEAN_C", th2)
        cv.imshow("ADAPTIVE_THRESH_GAUSSIAN_C", th3)

        cv.waitKey(0)
        cv.destroyAllWindows()
```



Matplotlib-opencv

```python
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('lena.jpg', -1)
cv2.imshow('image', img)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img)
plt.xticks([]), plt.yticks([])
plt.show()

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

Result:

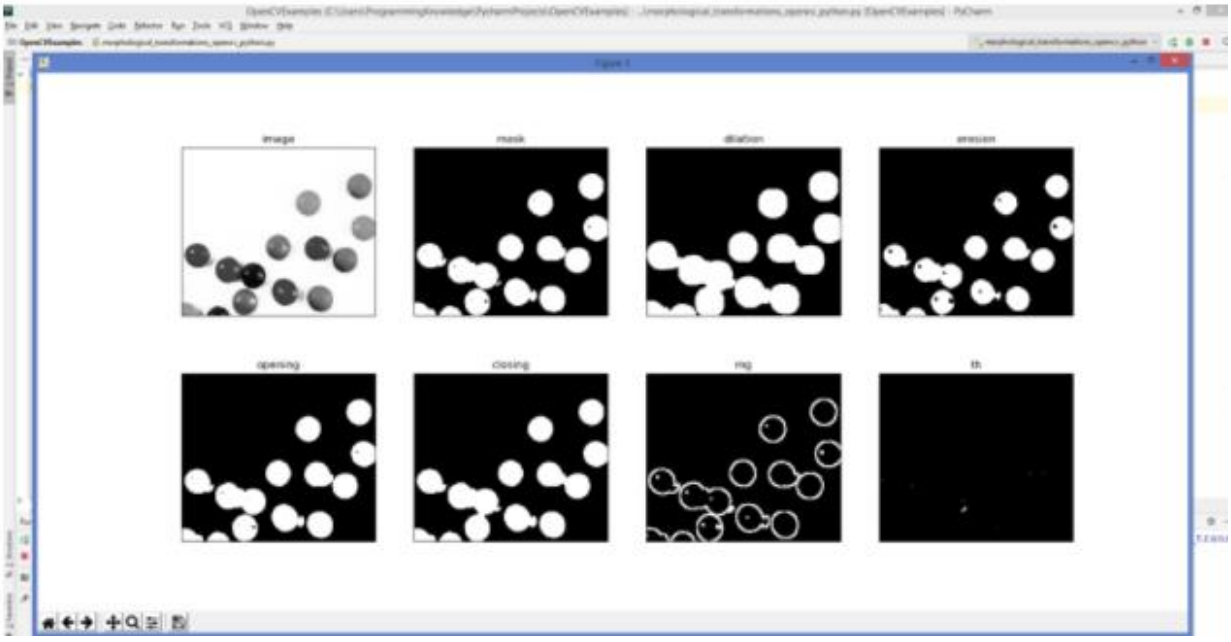


Morphological Transformations:

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

img = cv2.imread('smarties.png', cv2.IMREAD_GRAYSCALE)
_, mask = cv2.threshold(img, 220, 255, cv2.THRESH_BINARY_INV)

kernal = np.ones((5,5), np.uint8)

dilation = cv2.dilate(mask, kernal, iterations=2)
erosion = cv2.erode(mask, kernal, iterations=1)
opening = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernal)
closing = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernal)
mg = cv2.morphologyEx(mask, cv2.MORPH_GRADIENT, kernal)
th = cv2.morphologyEx(mask, cv2.MORPH_TOPHAT, kernal)

titles = ['image', 'mask', 'dilation', 'erosion', 'opening', 'closing', 'mg', 'th']
images = [img, mask, dilation, erosion, opening, closing, mg, th]

for i in range(8):
    plt.subplot(2, 4, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

Source Image:

OutPut:



Smoothing and Blurring images:

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

img = cv2.imread('lena.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

kernel = np.ones((5, 5), np.float32)/25
dst = cv2.filter2D(img, -1, kernel)
blur = cv2.blur(img, (5, 5))
gblur = cv2.GaussianBlur(img, (5, 5), 0)
median = cv2.medianBlur(img, 5)
bilateralFilter = cv2.bilateralFilter(img, 9, 75, 75)

titles = ['image', '2D Convolution', 'blur', 'GaussianBlur', 'median', 'bilateralFilter']
images = [img, dst, blur, gblur, median, bilateralFilter]

for i in range(6):
    plt.subplot(2, 3, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

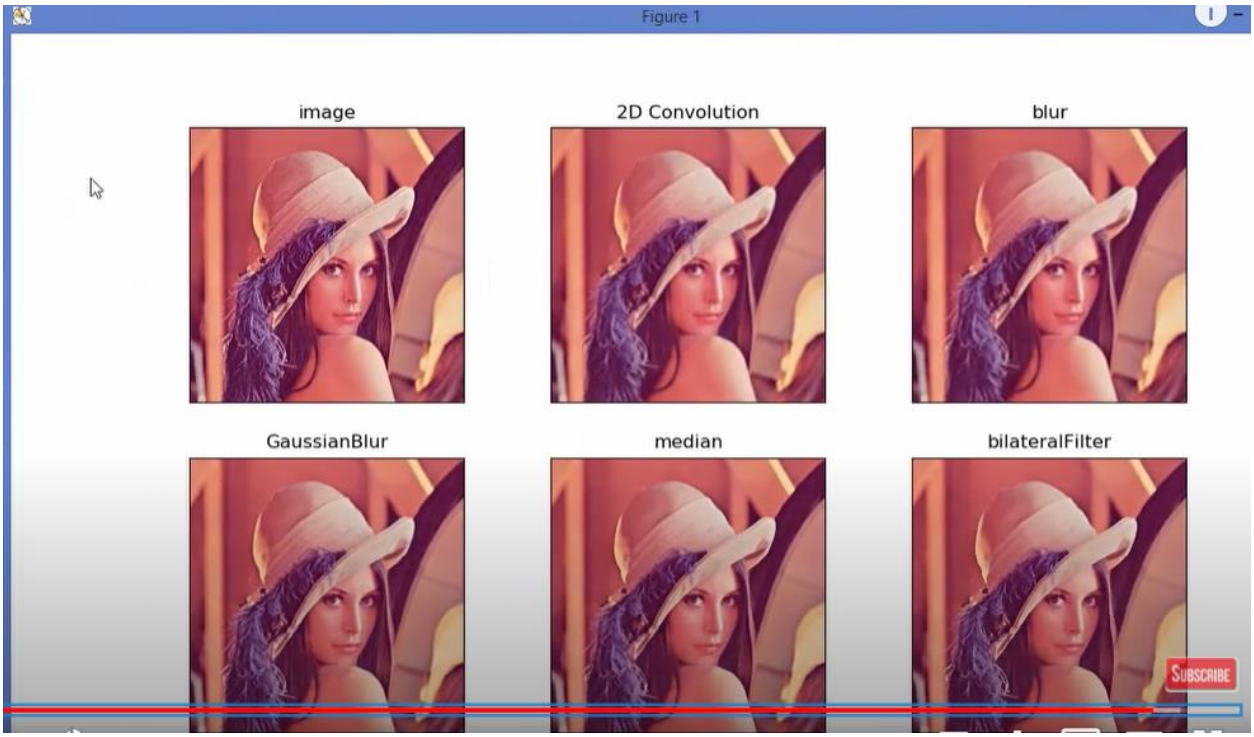
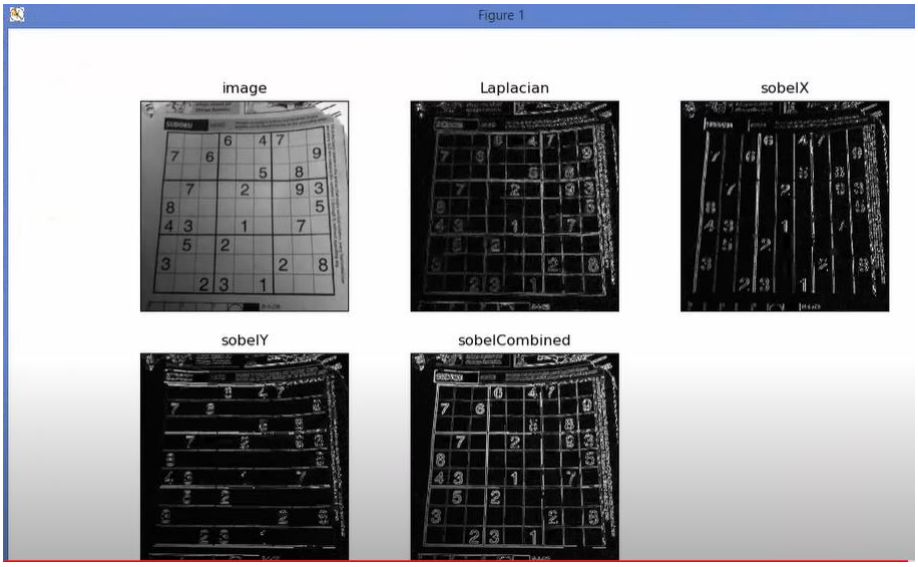


Image gradients and edge detection:

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

img = cv2.imread("sudoku.png", cv2.IMREAD_GRAYSCALE)
lap = cv2.Laplacian(img, cv2.CV_64F, ksize=3)
lap = np.uint8(np.absolute(lap))
sobelX = cv2.Sobel(img, cv2.CV_64F, 1, 0)
sobelY = cv2.Sobel(img, cv2.CV_64F, 0, 1)

sobelX = np.uint8(np.absolute(sobelX))
sobelY = np.uint8(np.absolute(sobelY))

sobelCombined = cv2.bitwise_or(sobelX, sobelY)

titles = ['image', 'Laplacian', 'sobelX', 'sobelY', 'sobelCombined']
images = [img, lap, sobelX, sobelY, sobelCombined]
for i in range(5):
    plt.subplot(2, 3, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```



Canny edge detection


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

img = cv2.imread("lena.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
canny = cv2.Canny(img, 100, 200)

titles = ['image', 'canny']
images = [img, canny]
for i in range(2):
    plt.subplot(1, 2, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```


```
import cv2
import numpy as np
```

```
from matplotlib import pyplot as plt

img = cv2.imread("messi5.jpg", cv2.IMREAD_GRAYSCALE)
lap = cv2.Laplacian(img, cv2.CV_64F, ksize=3)
lap = np.uint8(np.absolute(lap))
sobelX = cv2.Sobel(img, cv2.CV_64F, 1, 0)
sobelY = cv2.Sobel(img, cv2.CV_64F, 0, 1)
edges = cv2.Canny(img,100,200)

sobelX = np.uint8(np.absolute(sobelX))
sobelY = np.uint8(np.absolute(sobelY))

sobelCombined = cv2.bitwise_or(sobelX, sobelY)

titles = ['image', 'Laplacian', 'sobelX', 'sobelY', 'sobelCombined', 'Canny']
images = [img, lap, sobelX, sobelY, sobelCombined, edges]
for i in range(6):
    plt.subplot(2, 3, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

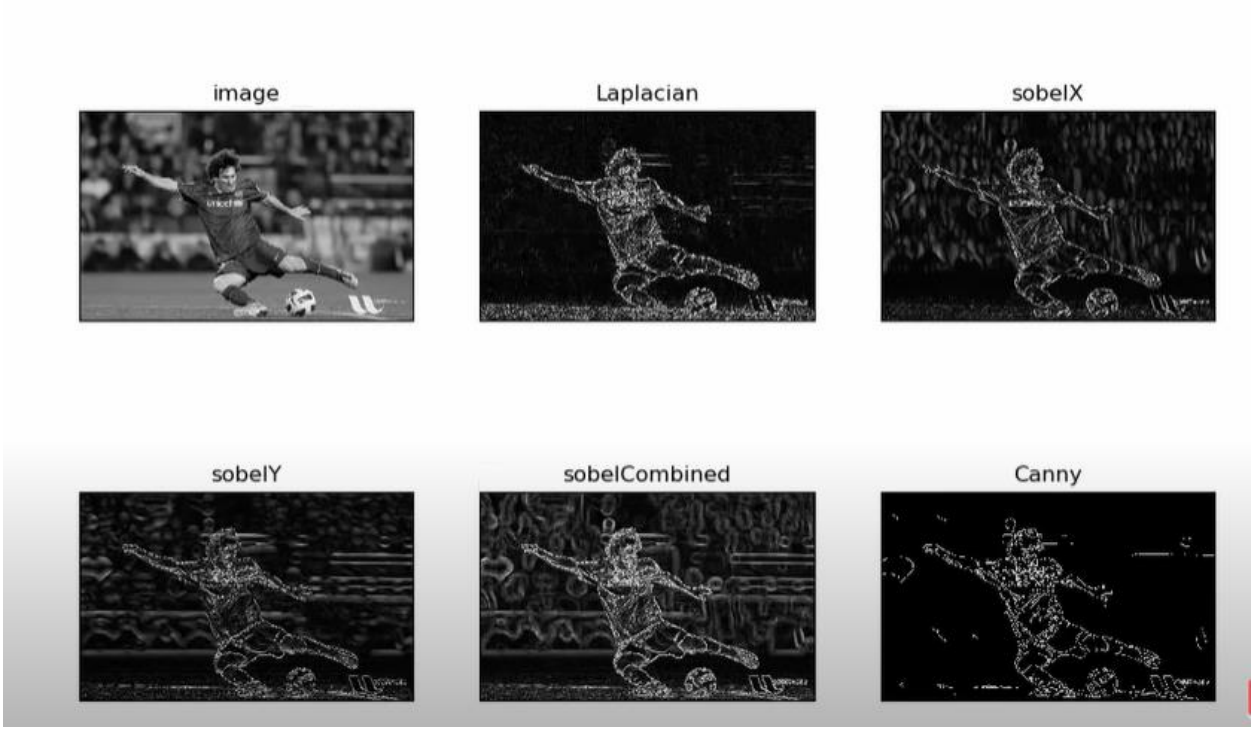
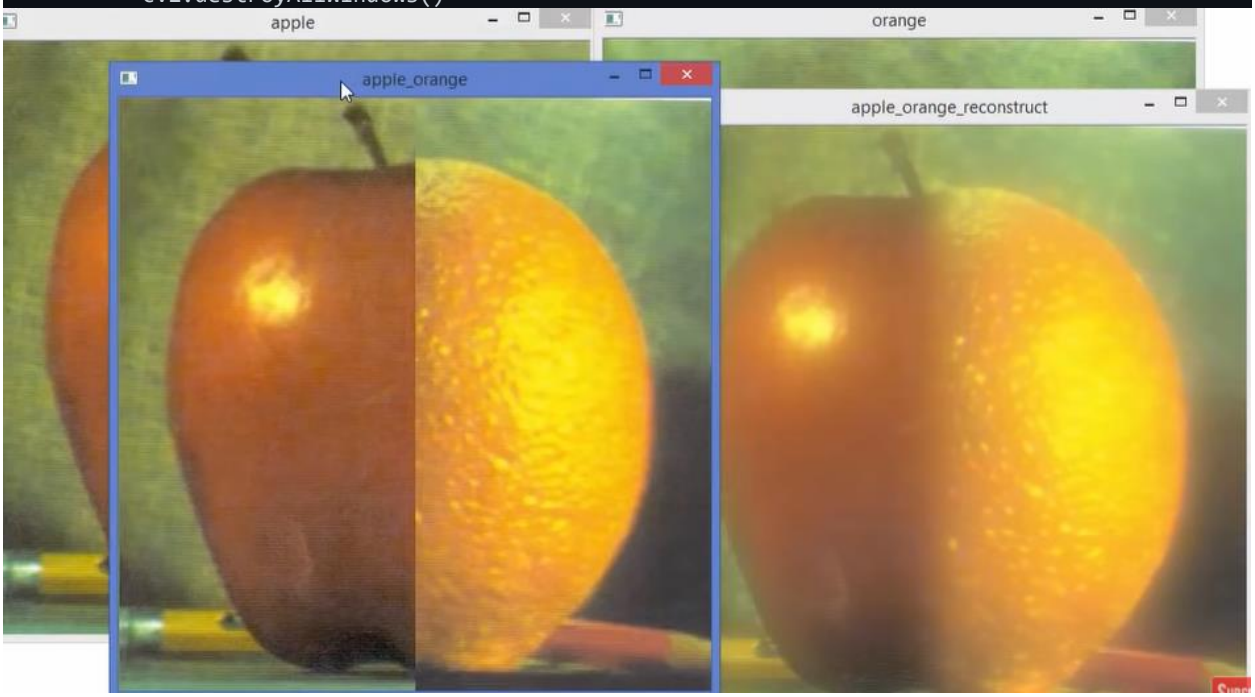


Image Pyramids

Image Blending:

```
import cv2
import numpy as np
apple = cv2.imread('apple.jpg')
orange = cv2.imread('orange.jpg')
print(apple.shape)
print(orange.shape)
apple_orange = np.hstack((apple[:, :256], orange[:, 256:]))

# generate Gaussian pyramid for apple
apple_copy = apple.copy()
gp_apple = [apple_copy]
for i in range(6):
    apple_copy = cv2.pyrDown(apple_copy)
    gp_apple.append(apple_copy)


# generate Gaussian pyramid for orange
orange_copy = orange.copy()
gp_orange = [orange_copy]
for i in range(6):
    orange_copy = cv2.pyrDown(orange_copy)
    gp_orange.append(orange_copy)

# generate Laplacian Pyramid for apple
apple_copy = gp_apple[5]
lp_apple = [apple_copy]
for i in range(5, 0, -1):
    gaussian_expanded = cv2.pyrUp(gp_apple[i])
    laplacian = cv2.subtract(gp_apple[i-1], gaussian_expanded)
    lp_apple.append(laplacian)

# generate Laplacian Pyramid for orange
orange_copy = gp_orange[5]
lp_orange = [orange_copy]
for i in range(5, 0, -1):
    gaussian_expanded = cv2.pyrUp(gp_orange[i])
    laplacian = cv2.subtract(gp_orange[i-1], gaussian_expanded)
    lp_orange.append(laplacian)

# Now add left and right halves of images in each level
```

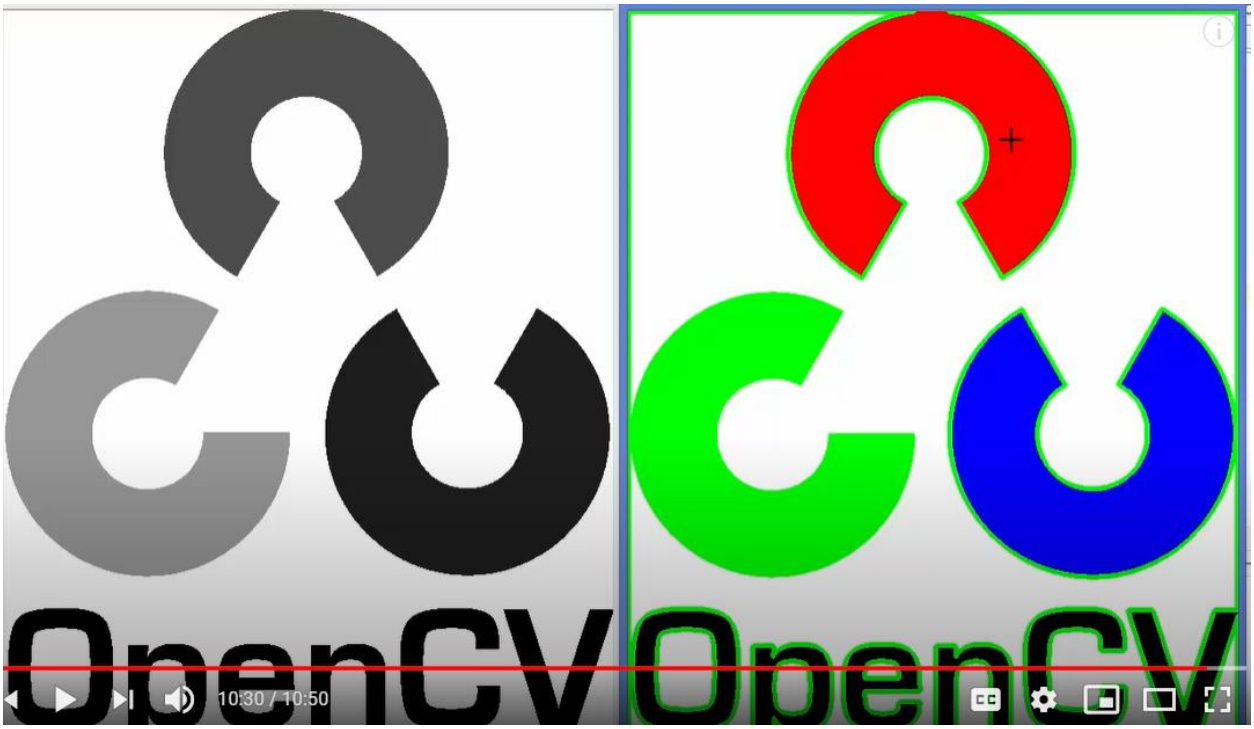
```
apple_orange_pyramid = []
n = 0
for apple_lap, orange_lap in zip(lp_apple, lp_orange):
    n += 1
    cols, rows, ch = apple_lap.shape
    laplacian = np.hstack((apple_lap[:, 0:int(cols/2)], orange_lap[:,
int(cols/2):]))
    apple_orange_pyramid.append(laplacian)
# now reconstruct
apple_orange_reconstruct = apple_orange_pyramid[0]
for i in range(1, 6):
    apple_orange_reconstruct = cv2.pyrUp(apple_orange_reconstruct)
    apple_orange_reconstruct = cv2.add(apple_orange_pyramid[i],
apple_orange_reconstruct)

cv2.imshow("apple", apple)
cv2.imshow("orange", orange)
cv2.imshow("apple_orange", apple_orange)
cv2.imshow("apple_orange_reconstruct", apple_orange_reconstruct)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Find and draw contours

```
import
numpy
as np
```

```
import cv2

img = cv2.imread('baseball.png')
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(imgray, 127, 255, 0)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
print("Number of contours = " + str(len(contours)))
print(contours[0])

cv2.drawContours(img, contours, -1, (0, 255, 0), 3)
cv2.drawContours(imgray, contours, -1, (0, 255, 0), 3)

cv2.imshow('Image', img)
cv2.imshow('Image GRAY', imgray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

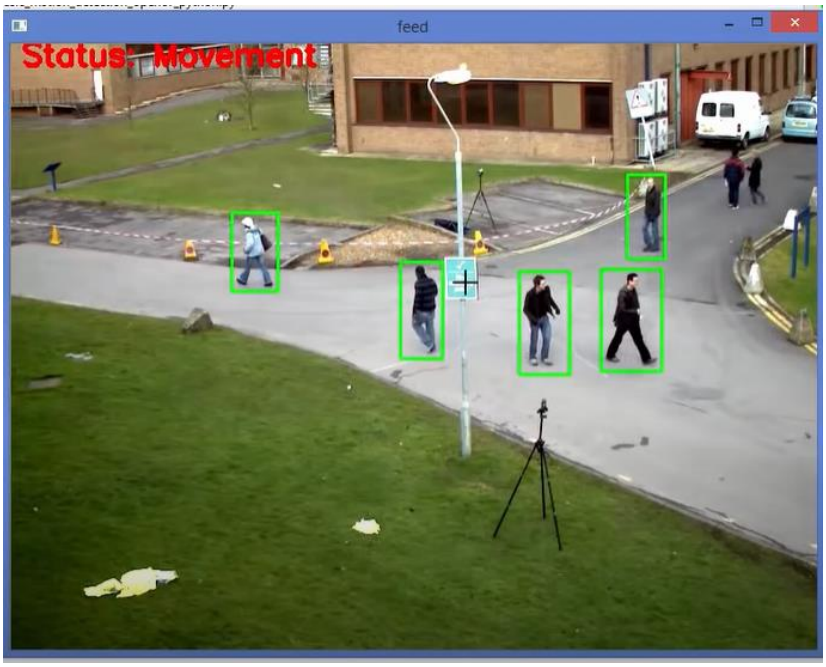


Motion Detection and Tracking

https://www.youtube.com/watch?v=MkcUgPhOlP8&list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K&index=28

```
import cv2
import numpy as np

cap = cv2.VideoCapture('vtest.avi')
frame_width = int( cap.get(cv2.CAP_PROP_FRAME_WIDTH))

frame_height =int( cap.get( cv2.CAP_PROP_FRAME_HEIGHT))

fourcc = cv2.VideoWriter_fourcc('X','V','I','D')

out = cv2.VideoWriter("output.avi", fourcc, 5.0, (1280,720))

ret, frame1 = cap.read()
ret, frame2 = cap.read()
print(frame1.shape)
while cap.isOpened():
    diff = cv2.absdiff(frame1, frame2)
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)

        if cv2.contourArea(contour) < 900:
            continue
        cv2.rectangle(frame1, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame1, "Status: {}".format('Movement'), (10, 20),
cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 3)
    #cv2.drawContours(frame1, contours, -1, (0, 255, 0), 2)

    image = cv2.resize(frame1, (1280,720))
    out.write(image)
```

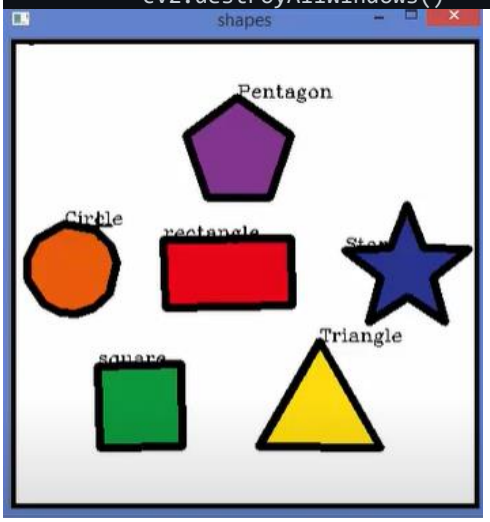
```
            cv2.imshow("feed", frame1)
            frame1 = frame2
            ret, frame2 = cap.read()

            if cv2.waitKey(40) == 27:
                break

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



Shape detection

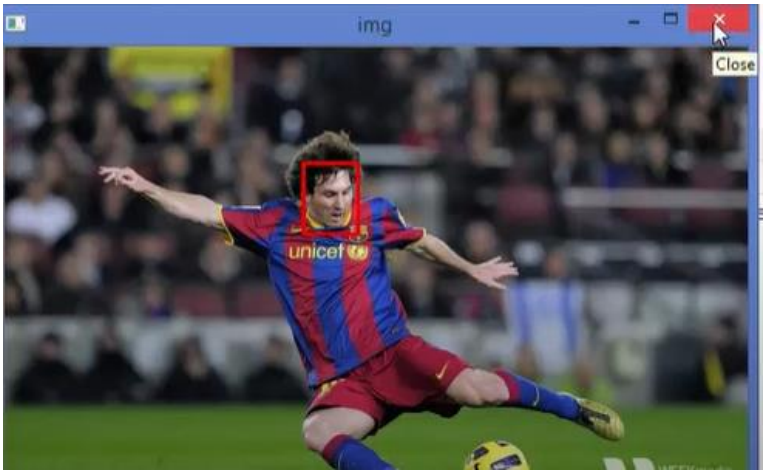
```
import
numpy
as np
        import cv2

        img = cv2.imread('shapes.jpg')
        imgGrey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        _, thrash = cv2.threshold(imgGrey, 240, 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(thrash, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

        cv2.imshow("img", img)
        for contour in contours:
```

```
        approx = cv2.approxPolyDP(contour, 0.01* cv2.arcLength(contour, True), True)
        cv2.drawContours(img, [approx], 0, (0, 0, 0), 5)
        x = approx.ravel()[0]
        y = approx.ravel()[1] - 5
        if len(approx) == 3:
            cv2.putText(img, "Triangle", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0,
    0))
        elif len(approx) == 4:
            x1 ,y1, w, h = cv2.boundingRect(approx)
            aspectRatio = float(w)/h
            print(aspectRatio)
            if aspectRatio >= 0.95 and aspectRatio <= 1.05:
              cv2.putText(img, "square", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0,
    0))
            else:
              cv2.putText(img, "rectangle", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0,
    0, 0))
        elif len(approx) == 5:
            cv2.putText(img, "Pentagon", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0,
    0))
        elif len(approx) == 10:
            cv2.putText(img, "Star", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0))
        else:
            cv2.putText(img, "Circle", (x, y), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0,
    0))


    cv2.imshow("shapes", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

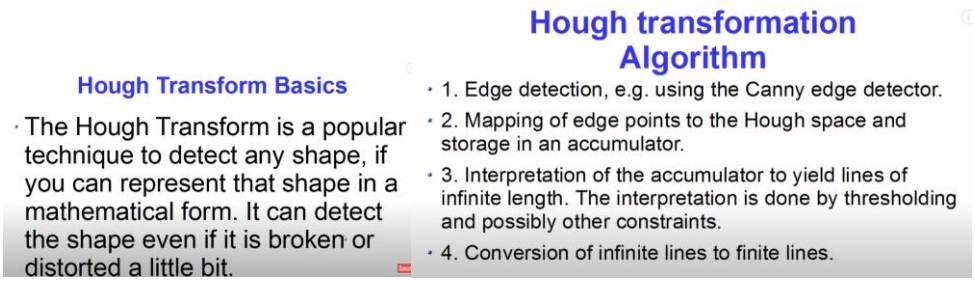


Template matching

```
import cv2
import numpy as np
img = cv2.imread("messi5.jpg")
grey_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
template = cv2.imread("messi_face.jpg", 0)
w, h = template.shape[::-1]

res = cv2.matchTemplate(grey_img, template, cv2.TM_CCORR_NORMED )
print(res)
threshold = 0.99;
loc = np.where(res >= threshold)
print(loc)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img, pt, (pt[0] + w, pt[1] + h), (0, 0, 255), 2)

cv2.imshow("img", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

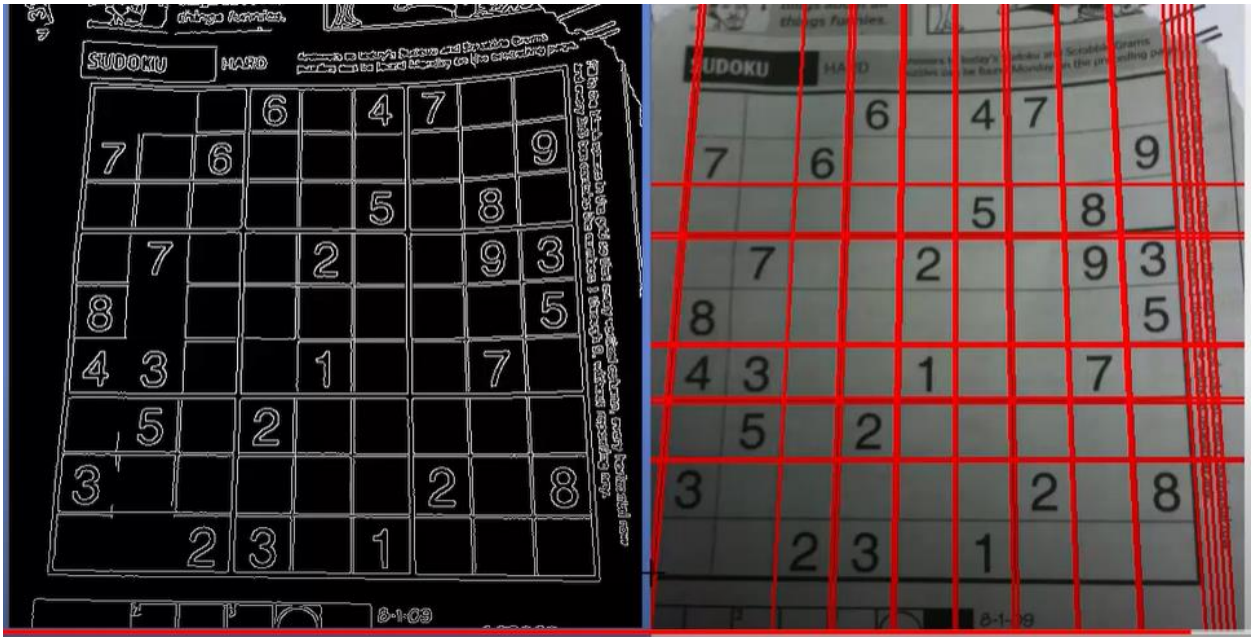


Hough Transform:



Hough line transform using Hough lines method in opencv

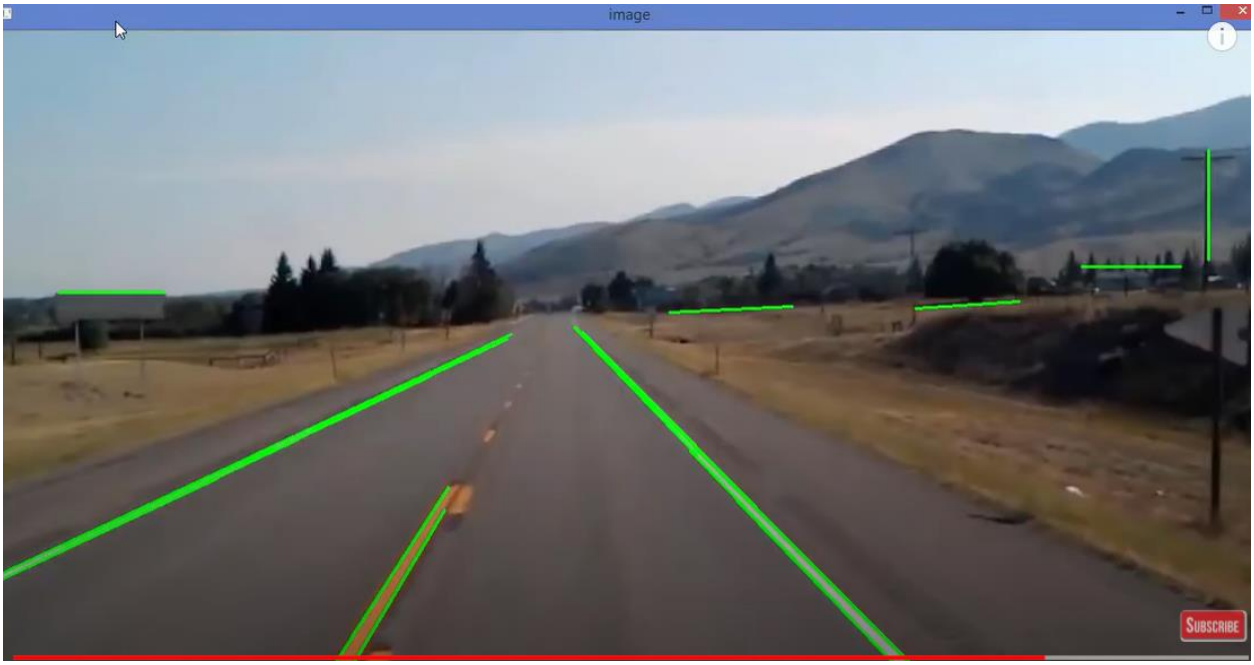
```
import cv2
import numpy as np

img = cv2.imread('sudoku.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150, apertureSize=3)
cv2.imshow('edges', edges)
lines = cv2.HoughLines(edges, 1, np.pi / 180, 200)

for line in lines:
    rho,theta = line[0]
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a * rho
    y0 = b * rho
    # x1 stores the rounded off value of (r * cos(theta) - 1000 * sin(theta))
    x1 = int(x0 + 1000 * (-b))
    # y1 stores the rounded off value of (r * sin(theta)+ 1000 * cos(theta))
    y1 = int(y0 + 1000 * (a))
    # x2 stores the rounded off value of (r * cos(theta)+ 1000 * sin(theta))
    x2 = int(x0 - 1000 * (-b))
    # y2 stores the rounded off value of (r * sin(theta)- 1000 * cos(theta))
    y2 = int(y0 - 1000 * (a))
    cv2.line(img, (x1, y1), (x2, y2), (0, 0, 255), 2)


cv2.imshow('image', img)
k = cv2.waitKey(0)
cv2.destroyAllWindows()
```



Hough line transform using probabilistic Hough lines method in opencv

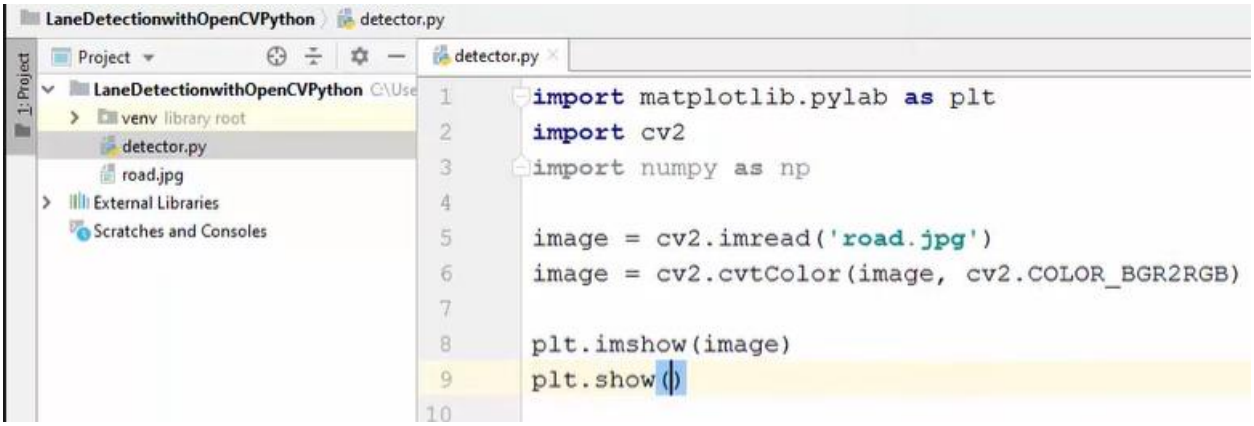
```
import
cv2
```

```
import numpy as np
img = cv2.imread('road.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,50,150,apertureSize = 3)
cv2.imshow('edges', edges)
lines = cv2.HoughLinesP(edges,1,np.pi/180,100,minLineLength=100,maxLineGap=10)
for line in lines:
    x1,y1,x2,y2 = line[0]
    cv2.line(img,(x1,y1),(x2,y2),(0,255,0),2)

cv2.imshow('image', img)
k = cv2.waitKey(0)
cv2.destroyAllWindows()
```



Lane Detection project :

Step 1: load the image



Step 2:

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

image = cv2.imread('road.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

print(image.shape)
height = image.shape[0]
width = image.shape[1]

plt.imshow(image)
plt.show()
```

```
C:\Users\Programm
(704, 1279, 3)
```

Region of interest

```
image = cv2.imread('road.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

print(image.shape)
height = image.shape[0]
width = image.shape[1]

region_of_interest_vertices = [
    (0, height),
    (width/2, height/2),
    width, height
]

plt.imshow(image)
plt.show()
```

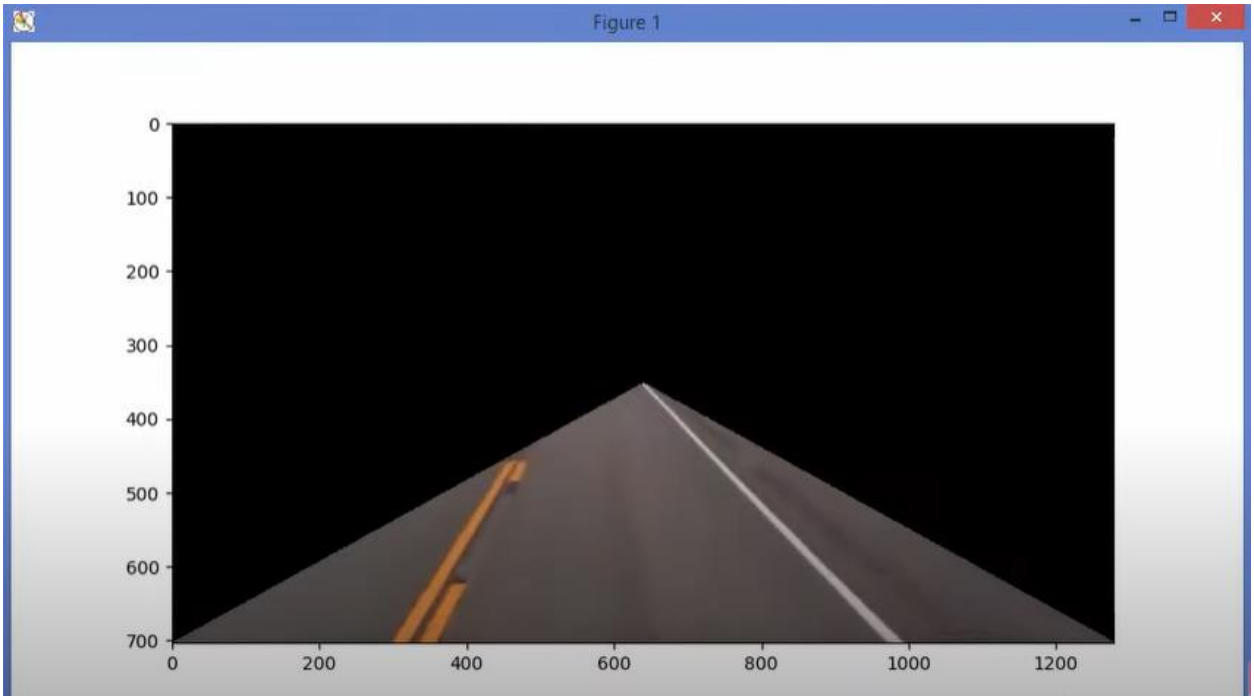
Apply region of interest on the image and remaining thing will be masked.

```
width = image.shape[1]

region_of_interest_vertices = [
    (0, height),
    (width/2, height/2),
    width, height
]

def region_of_interest(img, vertices):
    mask = np.zeros_like(img)
    channel_count = img.shape[2]
    match_mask_color = (255,) * channel_count
    cv2.fillPoly(mask, vertices, match_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image

cropped_image = region_of_interest(image,
                np.array([region_of_interest_vertices], np.int32))
plt.imshow(cropped_image)
plt.show()
```

Full code:

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

image = cv2.imread('road.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
print(image.shape)
height = image.shape[0]
width = image.shape[1]

region_of_interest_vertices = [
    (0, height),
    (width/2, height/2),
    (width, height)
]

def region_of_interest(img, vertices):
    mask = np.zeros_like(img)
    channel_count = img.shape[2]
    match_mask_color = (255,) * channel_count
    cv2.fillPoly(mask, vertices, match_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image

cropped_image = region_of_interest(image,
                np.array([region_of_interest_vertices], np.int32),)

plt.imshow(cropped_image)
plt.show()
```



Part 2

```
import
matplotlib.pylab
as plt
```

```
import cv2
import numpy as np

def region_of_interest(img, vertices):
    mask = np.zeros_like(img)
    #channel_count = img.shape[2]
    match_mask_color = 255
    cv2.fillPoly(mask, vertices, match_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image

def drow_the_lines(img, lines):
    img = np.copy(img)
    blank_image = np.zeros((img.shape[0], img.shape[1], 3),
dtype=np.uint8)

    for line in lines:
        for x1, y1, x2, y2 in line:
            cv2.line(blank_image, (x1,y1), (x2,y2), (0, 255, 0),
thickness=10)

    img = cv2.addWeighted(img, 0.8, blank_image, 1, 0.0)
    return img

image = cv2.imread('road.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
print(image.shape)
height = image.shape[0]
width = image.shape[1]
region_of_interest_vertices = [
    (0, height),
    (width/2, height/2),
    (width, height)
]
gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
canny_image = cv2.Canny(gray_image, 100, 200)
cropped_image = region_of_interest(canny_image,
                np.array([region_of_interest_vertices], np.int32),)
```

```
                    lines = cv2.HoughLinesP(cropped_image,
                                            rho=6,
                                            theta=np.pi/180,
                                            threshold=160,
                                            lines=np.array([]),
                                            minLineLength=40,
                                            maxLineGap=25)
                    image_with_lines = drow_the_lines(image, lines)
                    plt.imshow(image_with_lines)
                    plt.show()
```

Video File:

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

                    def region_of_interest(img, vertices):
```

```
    mask = np.zeros_like(img)
    #channel_count = img.shape[2]
    match_mask_color = 255
    cv2.fillPoly(mask, vertices, match_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image

def drow_the_lines(img, lines):
    img = np.copy(img)
    blank_image = np.zeros((img.shape[0], img.shape[1], 3),
dtype=np.uint8)

    for line in lines:
        for x1, y1, x2, y2 in line:
            cv2.line(blank_image, (x1,y1), (x2,y2), (0, 255, 0),
thickness=10)

    img = cv2.addWeighted(img, 0.8, blank_image, 1, 0.0)
    return img

# = cv2.imread('road.jpg')
#image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
def process(image):
    print(image.shape)
    height = image.shape[0]
    width = image.shape[1]
    region_of_interest_vertices = [
        (0, height),
        (width/2, height/2),
        (width, height)
    ]
    gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    canny_image = cv2.Canny(gray_image, 100, 120)
    cropped_image = region_of_interest(canny_image,
                    np.array([region_of_interest_vertices], np.int32),)
    lines = cv2.HoughLinesP(cropped_image,
                            rho=2,
                            theta=np.pi/180,
                            threshold=50,
                            lines=np.array([]),
                            minLineLength=40,
                            maxLineGap=100)
    image_with_lines = drow_the_lines(image, lines)
```

```
    return image_with_lines

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

while cap.isOpened():
    ret, frame = cap.read()
    frame = process(frame)
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

PyTesseract – OCR

```python
import cv2
import pytesseract


def ocr_core(img):
    text = pytesseract.image_to_string(img)
    return text


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


# get grayscale image
def get_grayscale(image):
    return cv.2.cvtColor(image, cv2.COLOR_BGR2GRAY)


# noise removal
def remove_noise(image):
    return cv2.medianBlur(image, 5)


# thresholding
def thresholding(image):
    return cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]


img = get_grayscale(img)
img = thresholding(img)
img = remove_noise(img)

print(ocr_core(img))
```

Image file:

Output:

```
~/Desktop/work/progknowledge/pytessearct_demo
❯ python ocr.py
A Python Approach to Character
Recognition
```