# Introduction to OpenCV

It is a image processing package under Python.  It usually work with numpy.

An online video to explain https://www.youtube.com/watch?v=xjrykYpaBBM

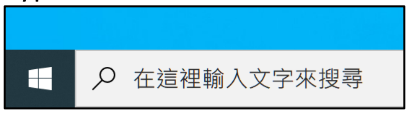
Pre-requistite
In order to run OpenCV, we must first have Python and OpenCV installed:

Installation of Python
Visit https://www.python.org/downloads/windows/
Download the most recent release version of Python and run the file.

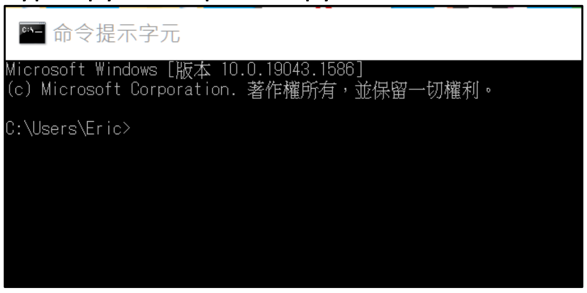
Installation of OpenCV
Visit https://sourceforge.net/projects/opencvlibrary/files/opencv-win/
Download the most recent release version of OpenCV and run the file.

Try the following programs by saving them into different files under the same name.  If everything is working correctly, then on Windows "CMD", type "python opencv?.py", (note: ? can be 1 to 10 as illustrated in the filenames OpenCV1 to OpenCv10 below.)

Type "CMD" in the box below.



Type "python opencv?.py".   ? stands for a number (1 to 10) as shown below.



```
Microsoft Windows [版本 10.0.19043.1586]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\Eric>
```

OpenCV1 – show the live video captured from web cam. Hit ‘q’ to quit.

```
import cv2

cap = cv2.VideoCapture(0)


while True:
    ret, frame = cap.read()
    if ret:
        frame = cv2.resize(frame, (0, 0), fx=1.2, fy=1.2)
        cv2.imshow('video', frame)
    else:
        break
    if cv2.waitKey(10) == ord('q'):
        break
```

OpenCV2 – display an image “sample.jpg”. img.shape[0] = heigh size, image.shape[1] = width size, image.shape[2] = number of channels, for RGB colour images = 3

```
import cv2
import numpy as np
import random

img = cv2.imread('sample.jpg')
# print(img.shape)

# img = np.empty((300, 300, 3), np.uint8)

# for row in range(300):
#     for col in range(img.shape[1]):
#         img[row][col] = [random.randint(0, 255), random.randint(0, 255),
random.randint(0, 255)]
newImg = img[100:160, 100:160]

cv2.imshow('img', img)
cv2.imshow('newImg', newImg)
cv2.waitKey(0)
```

OpenCV3 – show the image processes of Gaussian Blur, Canny, dilate and erode effects

```
import cv2
import numpy as np

kernel = np.ones((5, 5), np.uint8)
kernel1 = np.ones((5, 5), np.uint8)

img = cv2.imread('colorcolor.jpg')
img = cv2.resize(img, (0, 0), fx=0.5, fy=0.5)
```

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(img, (15, 15), 10)
canny = cv2.Canny(img, 200, 250)
dilate = cv2.dilate(canny, kernel, iterations=1)
erode = cv2.erode(dilate, kernel1, iterations=1)


# cv2.imshow('img', img)
# cv2.imshow('gray', gray)
# cv2.imshow('blur', blur)
cv2.imshow('canny', canny)
cv2.imshow('dilate', dilate)
cv2.imshow('erode', erode)
cv2.waitKey(0)
```

OpenCV4 – draw lines, rectangle, circle and text on image

```
import cv2
import numpy as np

img = np.zeros((600, 600, 3), np.uint8)
cv2.line(img, (0, 0), (img.shape[1], img.shape[0]), (0, 255, 0), 2)
cv2.rectangle(img, (0, 0), (400, 300), (0, 0, 255), cv2.FILLED)
cv2.circle(img, (300, 400), 30, (255, 0, 0), cv2.FILLED)
cv2.putText(img, '你好', (100, 500), cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255,
255), 2)


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

OpenCV5 – adjust hue (顏色色相), saturation (顏色飽和度), value (光暗度) of an image (refer to https://learn.leighcotnoir.com/artspeak/elements-color/hue-value-saturation/)

```
import cv2
import numpy as np

def empty(v):
    pass

img = cv2.imread('XiWinnie.jpg')
img = cv2.resize(img, (0, 0), fx=0.5, fy=0.5)

cv2.namedWindow('TrackBar')
cv2.resizeWindow('TrackBar', 640, 320)

cv2.createTrackbar('Hue Min', 'TrackBar', 0, 179, empty)
cv2.createTrackbar('Hue Max', 'TrackBar', 179, 179, empty)
cv2.createTrackbar('Sat Min', 'TrackBar', 0, 255, empty)
cv2.createTrackbar('Sat Max', 'TrackBar', 255, 255, empty)
cv2.createTrackbar('Val Min', 'TrackBar', 0, 255, empty)
```

```
cv2.createTrackbar('Val Max', 'TrackBar', 255, 255, empty)


hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
while True:
    h_min = cv2.getTrackbarPos('Hue Min', 'TrackBar')
    h_max = cv2.getTrackbarPos('Hue Max', 'TrackBar')
    s_min = cv2.getTrackbarPos('Sat Min', 'TrackBar')
    s_max = cv2.getTrackbarPos('Sat Max', 'TrackBar')
    v_min = cv2.getTrackbarPos('Val Min', 'TrackBar')
    v_max = cv2.getTrackbarPos('Val Max', 'TrackBar')
    print(h_min, h_max, s_min, s_max, v_min, v_max)

    lower = np.array([h_min, s_min, v_min])
    upper = np.array([h_max, s_max, v_max])

    mask = cv2.inRange(hsv, lower, upper)
    result = cv2.bitwise_and(img, img, mask=mask)

    cv2.imshow('img', img)
    cv2.imshow('hsv', hsv)
    cv2.imshow('mask', mask)
    cv2.imshow('reslut', result)
    cv2.waitKey(1)
```

OpenCV6 – get contour and hierarchy of an image (refer to https://docs.opencv.org/4.x/d9/d8b/tutorial_py_contours_hierarchy.html)

```
import cv2

img = cv2.imread('shape.jpg')
imgContour = img.copy()
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
canny = cv2.Canny(img, 150, 200)
contours, hierarchy = cv2.findContours(canny, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

for cnt in contours:
    cv2.drawContours(imgContour, cnt, -1, (255, 0, 0), 4)
    area = cv2.contourArea(cnt)
    if area > 500:
        # print(cv2.arcLength(cnt, True))
        peri = cv2.arcLength(cnt, True)
        vertices = cv2.approxPolyDP(cnt, peri * 0.02, True)
        corners = len(vertices)
        x, y, w, h = cv2.boundingRect(vertices)
        cv2.rectangle(imgContour, (x, y), (x+w, y+h), (0, 255, 0), 4)
        if corners == 3:
            cv2.putText(imgContour, 'triangle', (x, y-5), cv2.FONT_HERSHEY_SIMPLEX,
1, (0, 0, 255), 2)
        elif corners == 4:
            cv2.putText(imgContour, 'rectangle', (x, y-5),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        elif corners == 5:
            cv2.putText(imgContour, 'pentagon', (x, y-5),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
        elif corners >= 6:
            cv2.putText(imgContour, 'circle', (x, y-5), cv2.FONT_HERSHEY_SIMPLEX,
1, (0, 0, 255), 2)



cv2.imshow('img', img)
cv2.imshow('canny', canny)
cv2.imshow('imgContour', imgContour)
cv2.waitKey(0)
```

OpenCV7 – face detection

```
import cv2

img = cv2.imread('qq.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faceCascade = cv2.CascadeClassifier('face_detect.xml')
face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
faceRect = faceCascade.detectMultiScale(gray, 1.1, 5)
print(len(faceRect))

for (x, y, w, h) in faceRect:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

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

OpenCV8 – (refer to https://towardsdatascience.com/face-detection-in-10-lines-for-beginners-1787aa1d9127)

```
import cv2

face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
image = cv2.imread("qq.jpg")
image = cv2.resize(image, (800,533))
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces=face_cascade.detectMultiScale(gray_image,scaleFactor=1.10,minNeighbors=
5)
for x,y,w,h in faces:
    image=cv2.rectangle(image, (x,y), (x+w, y+h), (0, 255, 0),1)
    cv2.imshow("Face Detector", image)
    k=cv2.waitKey(2000)
cv2.destroyAllWindows()
```

OpenCV9 – detect faces and overlay glasses on a still image

```
#this is a demo to read still image and detect the faces, then overlay them with
glasses

import cv2
import numpy as np
import imageio

from scipy import ndimage  #for rotating image



def overlay_transparent(background, overlay, x, y):

    background_width = background.shape[1]
    background_height = background.shape[0]

    if x >= background_width or y >= background_height:
        return background

    h, w = overlay.shape[0], overlay.shape[1]

    if x + w > background_width:
        w = background_width - x
        overlay = overlay[:, :w]

    if y + h > background_height:
        h = background_height - y
        overlay = overlay[:h]

    if overlay.shape[2] < 4:
        overlay = np.concatenate(
            [
                overlay,
                np.ones((overlay.shape[0], overlay.shape[1], 1), dtype = overlay.dtype) *
255
            ],
            axis = 2,
        )

    overlay_image = overlay[..., :3]
    mask = overlay[..., 3:] / 255.0

    background[y:y+h, x:x+w] = (1.0 - mask) * background[y:y+h, x:x+w] + mask *
overlay_image

    return background


img = cv2.imread('qq.jpg')
```

```
#img = cv2.VideoCapture(0)

img_overlay = cv2.imread('glasses2.png')
bgr = img_overlay[:,:,:3] # Channels 0..2
gray = cv2.cvtColor(bgr, cv2.COLOR_BGR2GRAY)
bgr = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
alpha = img_overlay[:,:,0] # Channel 3
img_overlay = np.dstack([bgr, alpha]) # Add the alpha channel
aspect_ratio = img_overlay.shape[1] / img_overlay.shape[0] #width/height

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#faceCascade = cv2.CascadeClassifier('face_detect.xml')  //both can use
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
eyeCascade = cv2.CascadeClassifier("haarcascade_eye.xml")


faceRect = faceCascade.detectMultiScale(gray, 1.1, 5)
eyeRect = eyeCascade.detectMultiScale(gray, 1.1, 5)
print(len(faceRect))
print(len(eyeRect))

for (x, y, w, h) in faceRect:
    #cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

    down_width = w
    down_height = int(w / aspect_ratio)
    down_points = (down_width, down_height)
    resized_down = cv2.resize(img_overlay, down_points)
    #rotation angle in degree
    rotated = ndimage.rotate(resized_down,0 )
    overlay_transparent(img, rotated, x, y)

cv2.imshow('img', img)

cv2.waitKey(0)
```

OpenCV10 – detect faces on live camera and overly glasses

```
#this is a demo to capture live camera video and detect the faces, then overlay them
with glasses

import cv2
import numpy as np
import imageio

from scipy import ndimage  #for rotating image
```

```
def overlay_transparent(background, overlay, x, y):  #for overlaying transparent
image

    background_width = background.shape[1]
    background_height = background.shape[0]

    if x >= background_width or y >= background_height:
        return background

    h, w = overlay.shape[0], overlay.shape[1]

    if x + w > background_width:
        w = background_width - x
        overlay = overlay[:, :w]

    if y + h > background_height:
        h = background_height - y
        overlay = overlay[:h]

    if overlay.shape[2] < 4:
        overlay = np.concatenate(
            [
                overlay,
                np.ones((overlay.shape[0], overlay.shape[1], 1), dtype = overlay.dtype) *
255
            ],
            axis = 2,
        )

    overlay_image = overlay[..., :3]
    mask = overlay[..., 3:] / 255.0

    background[y:y+h, x:x+w] = (1.0 - mask) * background[y:y+h, x:x+w] + mask *
overlay_image

    return background




img_overlay = cv2.imread('glasses2.png')  #the overlay image
bgr = img_overlay[:,:,:3] # Channels 0..2  #to prepare for the transparency
gray = cv2.cvtColor(bgr, cv2.COLOR_BGR2GRAY)
bgr = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
alpha = bgr[:,:,2] # Channel
img_overlay = np.dstack([img_overlay, alpha]) # Add the alpha channel
aspect_ratio = img_overlay.shape[1] / img_overlay.shape[0] #width/height
```

```
#faceCascade = cv2.CascadeClassifier('face_detect.xml')  //both can use
faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
# eyeCascade = cv2.CascadeClassifier("haarcascade_eye.xml") #this is for eye
detection


cap = cv2.VideoCapture(0)
while 1:
        ret, img = cap.read()
        if ret:
            img = cv2.resize(img, (0, 0), fx=1.2, fy=1.2)



            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            #cv2.resizeWindow('img', 500,500)
            faceRect = faceCascade.detectMultiScale(gray, 1.1, 5)
            #eyeRect = eyeCascade.detectMultiScale(gray, 1.1, 5)
            print(len(faceRect))
            for (x, y, w, h) in faceRect:
                print(x, y)
                #cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

                down_width = w
                down_height = int(w / aspect_ratio)
                down_points = (down_width, down_height)
                resized_down = cv2.resize(img_overlay, down_points)  #resize the overlay
                #rotation angle in degree
                rotated = ndimage.rotate(resized_down,0 ) #can rotate the overlayed
image to a certain degree
                overlay_transparent(img, rotated, x, y)
                cv2.imshow('video', img)  #displayed the overlayed frame or image

        if cv2.waitKey(10) == ord('q'):  #press q to break
            break
```