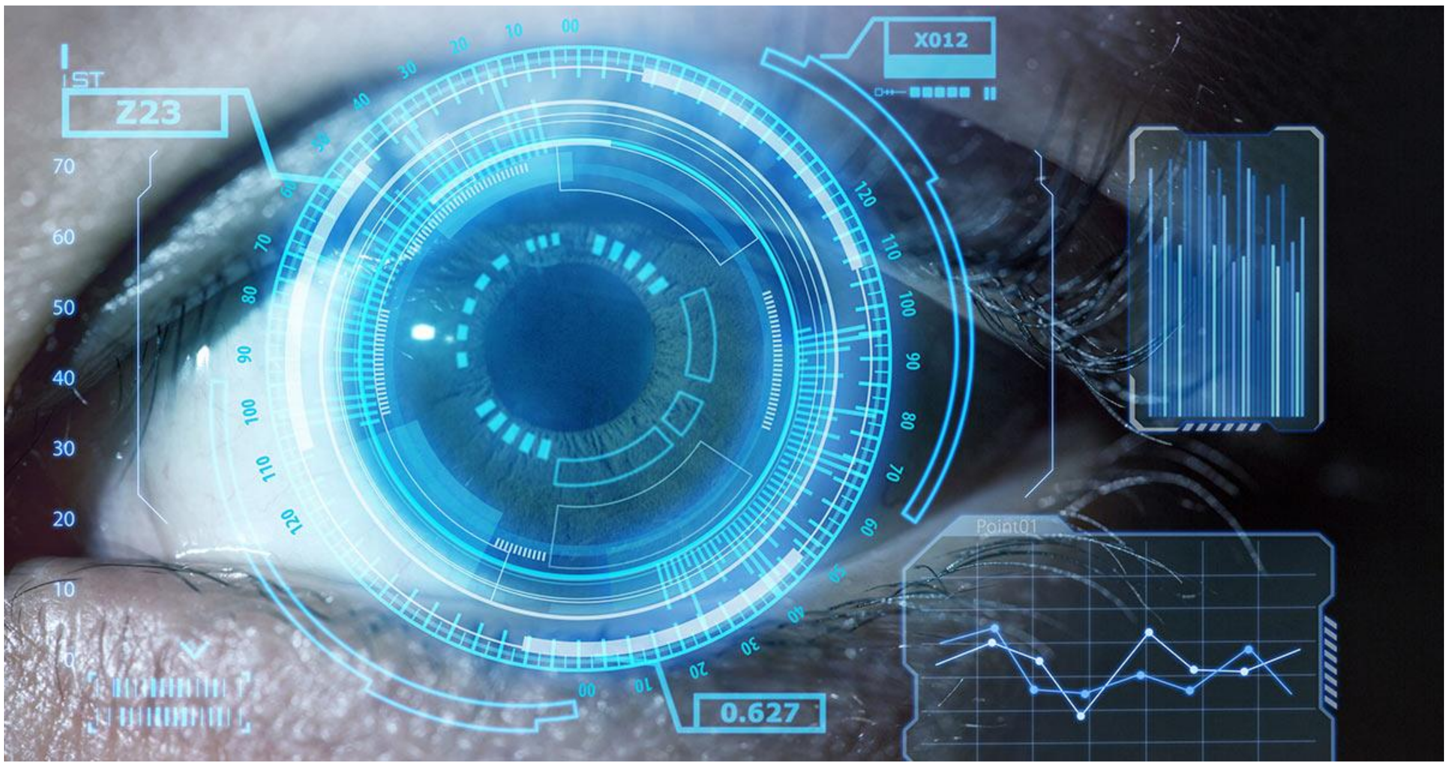
“COMPUTER VISION”

ENG: AHMED MUBARAK

01020451375



SESSION NO.“4”

- OPENCV LIBRARY
    1. Mouse
    2. Image tranformation
    3. Image arithmetic
    4. Bitwise operations
    5. Masking

ENG.AHMED MUBARAK
01020451375

1. Mouse

```
import cv2

WINDOW_NAME = "image"
CROP_WINDOW_NAME = "cropped"

cv2.namedWindow(WINDOW_NAME)
image = cv2.imread('images/khwarizmy.jpg')

POINTS = []
clone = image.copy()


def mouse_callback(event, x, y, flags, param):
"""
يتم مناداة الدالة كل مرة تتحرك فيا الفأرة
او يتم الضغط على اى زر فيها

Arguments:
event : حالة الفأرة
x : الاحداثى الأفقى
y : الاحداثى الرأسى
flags : معلومات اضافية
param : اعدادات او قيم تريد استخدامها
"""

global clone

# * تم الضغط على زر الفأرة الأيسر
if event == cv2.EVENT_LBUTTONDOWN:
cv2.destroyWindow(CROP_WINDOW_NAME)
clone = image.copy()
POINTS.append((x, y))
# cv2.circle(clone, (x, y), 30, param['green'], -1)

# * تم رفع الاصبع عن زر الفأرة الأيسر
if event == cv2.EVENT_LBUTTONUP:
POINTS.append((x, y))


colors = {'red': (0, 0, 255), 'green': (0, 255, 0)}

cv2.setMouseCallback(WINDOW_NAME, mouse_callback, param=colors)

while True:

cv2.imshow(WINDOW_NAME, clone)
```

```
if len(POINTS) == 2:
cv2.rectangle(clone, *POINTS, (0, 0, 255), 3)

# * جعل قيم السينات مع بعض
# * وجعل قيم الصادات مع بعض
xs, ys = zip(*POINTS)

cv2.imshow(CROP_WINDOW_NAME,
#!  لان لو حاولت ان تكون المستطيل من اليمن لليسار
#! أو من اسفل لاعلى سيحدث خطأ لان النهاية اقل من البدية
#! فى قيم السينات او الصادات او كليهما
clone[min(ys):max(ys), min(xs):max(xs)]
#    clone[POINTS[0][1]:POINTS[-1][1],
#          POINTS[0][0]:POINTS[-1][0]]
)

POINTS = []

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

cv2.waitKey(0)
```

2. Image Transformation

```
import numpy as np
import cv2

image = cv2.imread('images/khwarizmy-small.jpg')
print(image.shape)
cv2.imshow("The Image", image)

#? translation #
M = np.float32([[1, 0, 50], [0, 1, 100]])
shifted = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))

cv2.imshow('[{},{}] Shifted Image'.format(50, 100), shifted)
cv2.waitKey(0)
cv2.destroyAllWindows()


def translate(image, x, y, show=False):
    """
        Our translation matrix M is defined as a floating point array
        this is important because OpenCV expects this matrix
        to be of floating point type.

        The first row of the matrix is [1, 0, x],
        where x is the number of pixels we will shift
        the image left or right. Negative values of x will shift the
        image to the left and positive values will shift the image to
        the right.

        Then, we define the second row of the matrix as [0, 1, y],
        where y is the number of pixels we will shift the image up
        or down. Negative value of y will shift the image up and
        positive values will shift the image down


        Arguments:
            image {np.array} -- the image to translate
            x {int} -- the horizontal shift
            y {int} -- the vertical shift

        Keyword Arguments:
            show {bool} -- show the shiftted image or not (default: {False})

        Returns:
            [np.array] -- the shiftted image
    """
    M = np.float32([[1, 0, x], [0, 1, y]])
    shifted = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))

    if show:
        cv2.imshow('[{},{}] Shifted Image'.format(x, y), shifted)
        cv2.waitKey(0)

    return shifted

# ? rotatation
center = map(lambda x: x // 2, image.shape[1::-1])  # (h, w, channels)
M = cv2.getRotationMatrix2D(tuple(center), -45, float(1))

rotated = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))

cv2.imshow('{} Degree Rotated Image'.format(45), rotated)
cv2.waitKey(0)


def rotate(image, angle, scale, show=False):
    """
        Rotate the image in anti-clockwise in a given angle

        Arguments:
            image {np.array} -- original image
            angle {[int]} -- rotation angle
            scale {[float]} -- the scale of the rotated image
        Keyword Arguments:
            show {bool} -- show the rotated image or not (default: {False})
```

```python
        Returns:
            [np.array] -- the rotated image
    """

    center = map(lambda x: x // 2, image.shape[1::-1])  # (h, w, channels)
    M = cv2.getRotationMatrix2D(tuple(center), angle, float(scale))

    rotated = cv2.warpAffine(image, M, (image.shape[1], image.shape[0]))

    if show:
        cv2.imshow('{} Degree Rotated Image'.format(angle), rotated)
        cv2.waitKey(0)

    return rotated


#? Resizing #
new_width = 700
r = new_width / image.shape[1]
height = int(r * image.shape[0])

resized = cv2.resize(image, (new_width, height), interpolation=cv2.INTER_AREA)
# cv2.INTER_LINEAR
# cv2.INTER_CUBIC
# cv2.INTER_NEAREST

cv2.imshow('Image Resized By {0:.2f}'.format(r), resized)
cv2.waitKey(0)
cv2.destroyAllWindows()


def resize(image, width=None, height=None, show=False):
    """
        Arguments:
            image {np.array} -- original image
            width {[int]} -- the new width of the resized image
            height {[int]} -- the new height of the resized image

        Keyword Arguments:
            show {bool} -- show the resized image or not (default: {False})

        Returns:
            [np.array] -- the resized image
    """
    if width is None and height is None:
        return image

    if width is None:
        r = height / image.shape[0]
        width = int(r * image.shape[1])
    elif height is None:
        r = width / image.shape[1]
        height = int(r * image.shape[0])

    resized = cv2.resize(image, (width, height), interpolation=cv2.INTER_AREA)

    if show:
        cv2.imshow('Image Resized By {0:.2f}'.format(r), resized)
        cv2.waitKey(0)

    return resized


#? Flipping #
cv2.imshow("Original", image)
flipped = cv2.flip(image, 1)
cv2.imshow("Flipped Horizontally", flipped)

flipped = cv2.flip(image, 0)
cv2.imshow("Flipped Vertically", flipped)

flipped = cv2.flip(image, -1)
cv2.imshow("Flipped Horizontally & Vertically", flipped)
cv2.waitKey(0)


#! Cropping#
# ? See the Previous code.
```

3. Image arithmetic

```
import numpy as np
import argparse
import cv2

"""
How to solve this problem?
==========================
1- use if statement
2- use moduls %255
"""

"""
THE SOLUTION:
===============
1-NumPy will perform arithmetic and “wrap around”.

2-OpenCV will perform clipping stop if exceeded 255 or less than zero.
"""
```

```python
print("max of 255: {}".format(cv2.add(np.uint8([200]), np.uint8
                                       ([100]))))
print("min of 0: {}".format(cv2.subtract(np.uint8([50]), np.uint8
                                          ([100]))))

print("wrap around: {}".format(np.uint8([200]) + np.uint8([156])))

print("wrap around: {}".format(np.uint8([50]) - np.uint8([100])))


IMG_PATH = "images\khwarizmy.jpg"

image = cv2.imread(IMG_PATH)
cv2.imshow("Original", image)

M = np.ones(image.shape, dtype="uint8") * 100
added = cv2.add(image, M)
cv2.imshow("Added", added)

M = np.ones(image.shape, dtype="uint8") * 50
subtracted = cv2.subtract(image, M)
cv2.imshow("Subtracted", subtracted)

cv2.waitKey(0)
```

4. Bitwise operation

```python
import numpy as np
import cv2


square = np.zeros((300, 300), dtype="uint8")
color = 255  # ! Why I used only one integer instead of a tuple?
cv2.rectangle(square, (25, 25), (275, 275), color, -1)
cv2.imshow("Square", square)

circle = np.zeros((300, 300), dtype="uint8")
cv2.circle(circle, (150, 150), 150, 255, -1)
cv2.imshow("Circle", circle)
cv2.waitKey(0)

#? And #
bitwiseAnd = cv2.bitwise_and(square, circle)
cv2.imshow("AND", bitwiseAnd)
cv2.waitKey(0)

#? Or #
bitwiseOr = cv2.bitwise_or(square, circle)
cv2.imshow("OR", bitwiseOr)
```

```python
cv2.waitKey(0)

#? Xor #
bitwiseXor = cv2.bitwise_xor(square, circle)
cv2.imshow("XOR", bitwiseXor)
cv2.waitKey(0)

#? Truthy and Falsy Values #
v = 1
if v:
    print("it wasn't zero.".title())
else:
    print("it was zero.".title())

#? Not #
bitwiseNot = cv2.bitwise_not(circle)
cv2.imshow("NOT", bitwiseNot)
cv2.waitKey(0)
```

5. Masking

```python
import numpy as np
import cv2

image = cv2.imread("images/khwarizmy.jpg")
cv2.imshow("Original", image)

###? Masking ###
mask = np.zeros(image.shape[:2], dtype=np.uint8)
center = (image.shape[1] // 2, image.shape[0] // 2)

cv2.circle(mask, tuple(center), 200, 255, -1)
cv2.imshow("Mask", mask)

selected = cv2.bitwise_and(image, image, mask=mask)
cv2.imshow('With Mask', selected)
cv2.waitKey(0)
```

WITH MY BEST WISHES

ENG/AHMED MUBARAK 😉