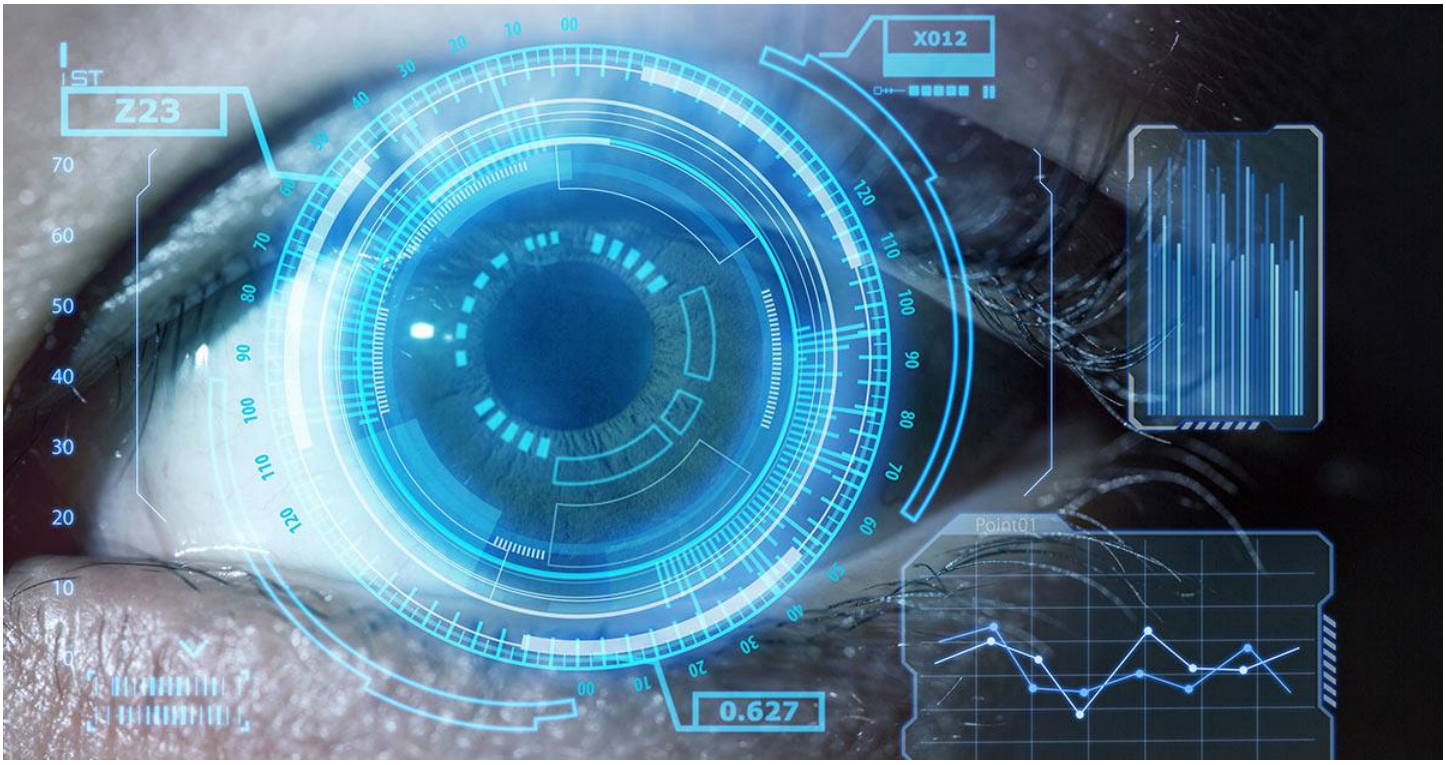
“COMPUTER VISION”

ENG: AHMED MUBARAK

01020451375



SESSION NO.“8”

- OPENCV LIBRARY
  1. Thresholding
  2. Find edges
  3. Extract coins

ENG.AHMED MUBARAK

01020451375

1. Threeholding

```python
import cv2
import numpy as np

# pip install mahotas
# conda install -c conda-forge mahotas
import mahotas

Win_NAME = "Thresholding"
cv2.namedWindow(Win_NAME)
TB_T = "T"
TB_INVERSE = "Inverse"
TB_Blur = "Use Blur"
TB_Mean = "Mean"
TB_C = "C"
TB_SAVE = "Save"
TB_Block_Size = "Block Size"


def do_nothing(x):
    pass


image = cv2.cvtColor(cv2.imread("images/page.png"), cv2.COLOR_BGR2GRAY)

# image = cv2.cvtColor(cv2.imread("images/number.png"), cv2.COLOR_BGR2GRAY)
# image = np.hstack((image, image, image, image, image))

blurred = cv2.GaussianBlur(image, (17, 17), 0)
# blurred = cv2.bilateralFilter(image, 10, 31, 31)

# cv2.imshow("Original ---> Blurred", np.hstack((image, blurred)))

#!-------------------------------------------------------------------------!#


#?################################
###? Simple Thresolding ###
#?################################

def global_thresholding(T, inverse=False, use_blur=False):
    if inverse:
        if use_blur:
            _, b_inv_img = cv2.threshold(
                blurred, T, 255, cv2.THRESH_BINARY_INV)
            return b_inv_img
            # cv2.imshow("Blur INV Global", b_inv_img)
        else:
```

```
            _, t_inv_img = cv2.threshold(image, T, 255, cv2.THRESH_BINARY_INV)
            return t_inv_img
            # cv2.imshow("Blur INV Global", t_inv_img)

    else:
        if use_blur:
            _, b_img = cv2.threshold(blurred, T, 255, cv2.THRESH_BINARY)
            return b_img
            # cv2.imshow("Blur Global", t_img)
        else:
            _, t_img = cv2.threshold(image, T, 255, cv2.THRESH_BINARY)
            return t_img
            # cv2.imshow("Global", t_img)

    # cv2.waitKey(0)


cv2.createTrackbar(TB_T, Win_NAME, 120, 255, do_nothing)
cv2.createTrackbar(TB_INVERSE, Win_NAME, 0, 1, do_nothing)
cv2.createTrackbar(TB_Blur, Win_NAME, 0, 1, do_nothing)
cv2.createTrackbar(TB_SAVE, Win_NAME, 0, 1, do_nothing)


#* Examine Global THRESHOLDING #
while(True):
    T = cv2.getTrackbarPos(TB_T, Win_NAME)
    inv = cv2.getTrackbarPos(TB_INVERSE, Win_NAME)
    blur = cv2.getTrackbarPos(TB_Blur, Win_NAME)
    threshold_image = global_thresholding(T, bool(inv), bool(blur))

    save = cv2.getTrackbarPos(TB_SAVE, Win_NAME)
    if save:
        cv2.imwrite("images/modified.png", threshold_image)
        break

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

    cv2.imshow(Win_NAME, threshold_image)

#!------------------------------------------------------------------------!#


###? otsu and riddler-calvard ###

#* OSTU #
T = mahotas.thresholding.otsu(image)
print("Otsu's threshold: {}".format(T))

copy1 = image.copy()
```

```
copy1[copy1 > T] = 255
copy1[copy1 < 255] = 0
# copy1 = cv2.bitwise_not(copy1)  # cv2.THRESH_BINARY_INV
cv2.imshow("Otsu", copy1)
cv2.waitKey(0)

* Riddler-Calvard #
T = mahotas.thresholding.rc(image)
print("Riddler-Calvard: {}".format(T))

copy2 = image.copy()
copy2[copy2 > T] = 255
copy2[copy2 < T] = 0
# copy2 = cv2.bitwise_not(copy2)
cv2.imshow("Riddler-Calvard", copy2)
cv2.waitKey(0)

#!-------------------------------------------------------------------------!#


#?################################
##? Adaptive Thresolding ###
#?################################


def adaptive_thresholding(use_blur=False, mean=True, inverse=False, blockSize=11, C=5):
    if mean:
        if inverse:
            if use_blur:
                b_atm_inv = cv2.adaptiveThreshold(blurred,
                                                  255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                                  cv2.THRESH_BINARY_INV, blockSize, C)

                return b_atm_inv

            else:
                atm_inv = cv2.adaptiveThreshold(image,
                                                255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                                cv2.THRESH_BINARY_INV, blockSize, C)
                return atm_inv

        else:
            if use_blur:
                b_atm = cv2.adaptiveThreshold(blurred,
                                              255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                              cv2.THRESH_BINARY, blockSize, C)

                return b_atm

            else:
```

```
                atm = cv2.adaptiveThreshold(image,
                                            255, cv2.ADAPTIVE_THRESH_MEAN_C,
                                            cv2.THRESH_BINARY, blockSize, C)

                return atm

    else:
        if inverse:
            if use_blur:
                b_atg_inv = cv2.adaptiveThreshold(blurred,
                                                  255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                                  cv2.THRESH_BINARY_INV, blockSize, C)

                return b_atg_inv

            else:
                atg_inv = cv2.adaptiveThreshold(image,
                                                255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                                cv2.THRESH_BINARY_INV, blockSize, C)

                return atg_inv

        else:
            if use_blur:
                b_atg = cv2.adaptiveThreshold(blurred,
                                              255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                              cv2.THRESH_BINARY, blockSize, C)
                return b_atg

            else:
                atg = cv2.adaptiveThreshold(blurred,
                                            255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                            cv2.THRESH_BINARY, blockSize, C)
                return atg


cv2.createTrackbar(TB_Block_Size, Win_NAME, 11, 50, do_nothing)
cv2.setTrackbarMin(TB_Block_Size, Win_NAME, 3)

cv2.createTrackbar(TB_C, Win_NAME, 5, 30, do_nothing)
cv2.createTrackbar(TB_Mean, Win_NAME, 0, 1, do_nothing)
cv2.createTrackbar(TB_INVERSE, Win_NAME, 0, 1, do_nothing)
cv2.createTrackbar(TB_Blur, Win_NAME, 0, 1, do_nothing)
cv2.createTrackbar(TB_SAVE, Win_NAME, 0, 1, do_nothing)

#* Examine ADAPTIVE THRESHOLDING  #
while(True):
    inv = cv2.getTrackbarPos(TB_INVERSE, Win_NAME)
    blur = cv2.getTrackbarPos(TB_Blur, Win_NAME)
    mean = cv2.getTrackbarPos(TB_Mean, Win_NAME)
```

```
    c = cv2.getTrackbarPos(TB_C, Win_NAME)

    size = cv2.getTrackbarPos(TB_Block_Size, Win_NAME)
    if size % 2 == 0:
        size += 1

    threshold_image = adaptive_thresholding(mean=bool(mean), inverse=bool(inv),
                                            use_blur=bool(blur), blockSize=size, C=c)

    save = cv2.getTrackbarPos(TB_SAVE, Win_NAME)
    if save:
        cv2.imwrite("images/modified.png", threshold_image)
        break

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

    cv2.imshow(Win_NAME, threshold_image)
```

2. Find edges

```
import cv2
import numpy as np

img_path = "images/coins.jpg"
# img_path = "images/brain.jpg"
image = cv2.imread(img_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Original", image)


#? Laplacian #
lap = cv2.Laplacian(image, cv2.CV_64F)
lap = np.uint8(np.absolute(lap))
cv2.imshow("Laplacian", lap)
cv2.waitKey(0)

#? Sobel #
sobel_x = np.uint8(np.absolute(cv2.Sobel(image, cv2.CV_64F, 1, 0)))
sobel_y = np.uint8(np.absolute(cv2.Sobel(image, cv2.CV_64F, 0, 1)))
sobel_combined = cv2.bitwise_or(sobel_x, sobel_y)

cv2.imshow("Sobel X", sobel_x)
cv2.imshow("Sobel Y", sobel_y)
cv2.imshow("Sobel Combined", sobel_combined)
cv2.waitKey(0)

#? Canny Edge Detector #
```

```python
blur = cv2.GaussianBlur(image, (7, 7), 0)
canny = cv2.Canny(blur, 30, 160)
cv2.imshow("Canny", canny)
cv2.waitKey(0)

#? Canny With TrackBar #
WIN_NAME = "Detection"
TB_Kernel_Size = "Kernel Size"
cv2.namedWindow(WIN_NAME)
cv2.createTrackbar(TB_Kernel_Size, WIN_NAME, 11, 50, lambda x: x)
cv2.setTrackbarMin(TB_Kernel_Size, WIN_NAME, 3)

while(True):
    k_size = cv2.getTrackbarPos(TB_Kernel_Size, WIN_NAME)
    if k_size % 2 == 0:
        k_size += 1
        cv2.setTrackbarPos(TB_Kernel_Size, WIN_NAME, k_size)

    copy = image.copy()

    blurred = cv2.GaussianBlur(copy, (k_size, k_size), 0)
    edges = cv2.Canny(blurred, 30, 160)

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

    cv2.imshow(WIN_NAME, edges)
```

3. Extract coins

```python
import cv2
import numpy as np

img_path = "E:\Work\OpenCV\OpenCV-in-Arabic-for-Beginners-master\OpenCV-in-Arabic-for-Beginners-master\images/coins.jpg"
image = cv2.imread(img_path)
cv2.imshow("Original", image)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#! changing the blur kernel size affects the detection
#! like with >>>  coins.webp
blurred = cv2.GaussianBlur(gray, (15, 15), 0)
edged = cv2.Canny(blurred, 30, 180)
cv2.imshow("Edged", edged)

#? Finding Contours #
```

```python
# ! opencv  مكتبة لاصدار وفقا المخرجات قائمةفى contours ترتيب ال
if (cv2.__version__)[0] in '24':
    contours, _ = cv2.findContours(edged,
                                   cv2.RETR_EXTERNAL,
                                   cv2.CHAIN_APPROX_SIMPLE)
else:
    _, contours, _ = cv2.findContours(edged,
                                      cv2.RETR_EXTERNAL,
                                      cv2.CHAIN_APPROX_SIMPLE)
print('=' * 30)
print("{} objects found".format(len(contours)).title())
print('=' * 30)

cv2.drawContours(image, contours, -1, (0, 255, 0), 2)
cv2.imshow("Objects Found", image)

#? Extracting Objects One By One #

# * we used the mask in full size of the original image
# * so we can use x and y of the bounding triangele
# * and also so we can draw a circle
mask = np.zeros(image.shape[:2], np.uint8)

for i, c in enumerate(contours):
    # ? if taken with PhotoShop
    x, y, w, h = cv2.boundingRect(c)
    coin = image[y:y+h, x:x+w]

    # ? Original Coin
    (cX, cY), r = cv2.minEnclosingCircle(c)
    cv2.circle(mask, (int(cX), int(cY)), int(r), 255, -1)
    cv2.imshow('Mask', mask)

    coin_mask = mask[y:y+h, x:x+w]
    masked = cv2.bitwise_and(coin, coin, mask=coin_mask)
    cv2.imshow("Original & Masked", np.hstack((coin, masked)))
    print('Object #{}'.format(i+1))
    cv2.waitKey(0)
```

WITH MY BEST WISHES

ENG/AHMED MUBARAK 😉