OpenCV Course - Full Tutorial with Python (Notes)

-It is a computer vision library.
-To install write this command in your terminal : pip install  opencv-contrib-python
-Next package is you need the install is caer, to install: pip install caer(to speed up your workflow)

Reading Images and Videos

- import cv2 as cv
cv.imread(‘documentpath’): This method takes in a path to an image and returns that image as a matrix of pixels.

cv.imshow(‘name of the window’, ‘actual matrix of pixels to display’):it will display the image as a new window.It takes 2 parameters.

cv.waitKey(0):Basically a keyboard binding function, it waits for specific delay.(time in millisecond)

OpenCv does not have an inbuilt way of dealing with images that are far greater than your computer screen.There are ways to mitigate this issue.

Reading Videos

capture=cv.VideoCapture(): This method takes integer arguments like 0,1,2etc, or a path to a video file.0=our webcam be referenced by zero.1=reference the first camera connected.2= reference second camera
.read(): this method reads in this video frame by frame.
cv.imshow(‘)

Code for Reading Videos:

```
import cv2 as cv

img = cv.imread('../Resources/Photos/cats.jpg')
cv.imshow('Cats', img)

cv.waitKey(0)

# Reading Videos
capture = cv.VideoCapture('../Resources/Videos/dog.mp4')

while True:
    isTrue, frame = capture.read()
```

```
    # if cv.waitKey(20) & 0xFF==ord('d'):
    # This is the preferred way - if `isTrue` is false (the frame could
    # not be read, or we're at the end of the video), we immediately
    # break from the loop.
    if isTrue:
        cv.imshow('Video', frame)
        if cv.waitKey(20) & 0xFF==ord('d'):
            break
    else:
        break

capture.release()
cv.destroyAllWindows()
```

Resizing and Rescaling(ölçeklendirme)

To rescale a video frame or an image,we can create a function:

```
def rescaleFrame(frame, scale=0,75):
    #For Images,Videos and Live Videos
    width= int(frame.shape[1] *scale ): width of the image
    height=int(frame.shape[1] *scale ): height of the image
    dimensions=(width,height)

    return cv.resize(frame,dimension,interpolation=cv.INTER_AREA)

def changeRes(width,height):
 #it is only for live videos
    capture.set(3,width)
    capture.set(4,height)                (It is a function that changes the size of the photo)
```

DRAWING SHAPES AND PUTTING Text

To create a blank image:
you need to import numpy first

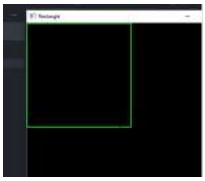
```
  blank= np.zeros((500,500),dtype='uint8')
```

To create a  square:

```
   blank[200:300,300:400]=0,0,255
```

   it will create a red square in window

```
cv.rectangle(img,pt1,pt2,color,thickness=None,lineType=None,shift=None)
example: cv.rectangle(blank,(0,0) ,(250,250),(0,255,0),thickness=2)
```



cv.rectangle(blank,(0,0) ,(250,250),(0,255,0),thickness=2)
Filling the image to certain color: cv.rectangle(blank,(0,0) ,(250,250),(0,255,0),thickness=cv.FILLED)
or you can attend -1 value to fill the shape with color.

```
cv.rectangle(blank, (0,0), (blank.shape[1]//2, blank.shape[0]//2), (0,255,0),
```

Draw a circle:
cv.circle(img,center radius ,color,thickness=None,lineType=None,shift=None)

Draw a line:
cv.line(img,pt1,pt2,color,thickness,lineType,shift)
Example:

```
cv.line(blank, (0,0), (blank.shape[1]//2, blank.shape[0]//2), (255,255,255),
thickness=3)
cv.imshow('Line', blank)
```

```
cv.line(blank, (100,250), (300,400), (255,255,255), thickness=3)
```

cv.putText(img,text,org,fontFace,fontScale,color,thickness,lineType,bottomLeftOrigin)
cv.FONT_HERSHEY_TRIPLEX:example of font type
Example:

```
cv.putText(blank, 'Hello', (225,225), cv.FONT_HERSHEY_TRIPLEX, 1.0, (0,255,0), 2)
cv.imshow('Text', blank)
```


5 essential functions:
blur=cv.GaussianBlur(img,ksize,sigmaX,dst,sigmaY,borderType)
Example;
kernel size provides the blur of the photo.

```
blur = cv.GaussianBlur(img, (3,3), cv.BORDER_DEFAULT)
```

Edge Cascade:
canny=cv.Canny(image,threshold,threshold2,edges,apertureSize)

```
canny = cv.Canny(img, 125, 175)
cv.imshow('')
```

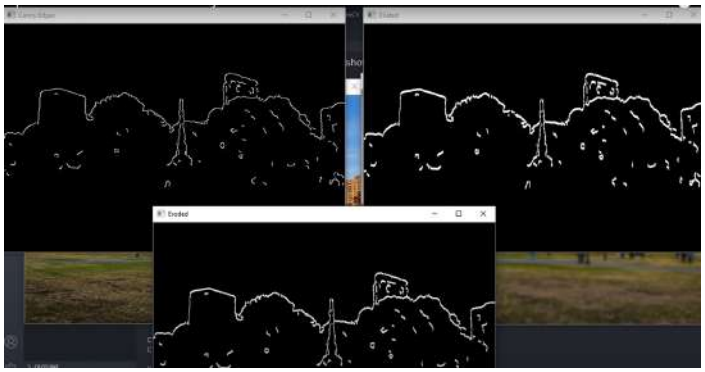
Example:

How to dilate(genişletmek) an image using the structural element?

Dilating the image:
dilated=cv.dilate(src,kernel,dst,anchor,iterations,borderType,borderValue)

To eroded:
eroded=cv.erode(src,kernel,dst,anchor,iterations,borderType,borderValue)
Example:

```
# Eroding
eroded = cv.erode(dilated, (3,3), iterations=1)
```



Resize:cv.resize(src,dsize,dst,fx,fy,interpolation)
Cropping:  img[50:200,200:400]

IMAGE TRANSFORMATIONS

-Translation:Using translation, you can shift an image up,down,left,right or with any combination of above.

```
# Translation
def translate(img, x, y):
    transMat = np.float32([[1,0,x],[0,1,y]])
    dimensions = (img.shape[1], img.shape[0])
    return cv.warpAffine(img, transMat, dimensions)

# -x --> Left
# -y --> Up
# x --> Right
# y --> Down
```

-Rotation:

```python
# Rotation
def rotate(img, angle, rotPoint=None):
    (height,width) = img.shape[:2]

    if rotPoint is None:
        rotPoint = (width//2,height//2)

    rotMat = cv.getRotationMatrix2D(rotPoint, angle, 1.0)
    dimensions = (width,height)

    return cv.warpAffine(img, rotMat, dimensions)
```

-Flipping:
flip=cv.flip(src,flipCode,dst)
Flipcode flips a 2D array around vertical,horizontal or both.
if you want to flip vertically and horizontal use -1 in flipCode

Contour Detection:

Contours and edges are different things.Contours are useful tools when you get into shape analysis and object detection.

gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)
(to get a gray image)

Contours Methods:
contours,hierarchies= findContours(image,mode,method,contours,hierarchy)
threshold(eşik)ıt will try to binarize that image.
ret,thresh=cv.threshold(src,thresh,maxval,type,dst)
cv.THRESH_BINARY it will binarize (type)

drawContours(image,contours,contourIdx,color,thickness,lineType,hierarchy,maxLevel,offset)

COLOR SPACES
-OpenCV is the default way of reading and images is BGR.

#BGR to GrayScale:

```python
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Gray' gray)
```

cvtColor(src,code,dst,dstCn) method syntax

#BGR to HSV:

```python
hsv = cv.cvtColor(img, cv.COLOR_BGR2HSV)
cv.imshow('HSV', hsv)
```

#BGR to L*a*b:lab=cvtColor(img,cv.COLOR_BGR2LAB)


#BGR to RGB

```
rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
cv.imshow('RGB' ,b)
```

```
# HSV to BGR
lab_bgr = cv.cvtColor(lab, cv.COLOR_LAB2BGR)
cv.imshow('LAB --> BGR', lab_bgr)
```

COLOR CHANNELS(How to split and merge color channels)

- We can split photo tho 3 different color channels(red,green,blue)

b,g,r=cv.split(m,[,mv]) m:input multi-channel array mv:output vector of arrays

merged=cv.merge(mv,dst) ( `cv2.merge` takes single channel images and combines them to make a multi-channel image)

example: cv.merge([b,g,r])

!!! Lighter portions represent a high distribution.

Smoothing and Blurring the Images

-There are many blurring techniques.

( Kernel size is basically the number of rows and the number of columns.)

Blur is applied to the middle pixel as a result of pixels around it.This middle pixel will essentially compute the pixel intensity of the middle pixel of the true center as the average of the surrounding pixel intensities.

#Averaging

average=cv.blur(src,ksize ,dst,anchor,borderType)

#Gaussian Blur (Instead of computing the average of all this running pixel intensity,each running pixel is given a particular weight)

- gauss=cv.GaussianBlur(src,ksize,sigmaX,dst,sigmaY,borderType)

#Median Blur (Instead of finding the average of surrounding pixels,it finds the median of surrounding pixels) - medianblur=cv.medianBlur(src,ksize,dst)

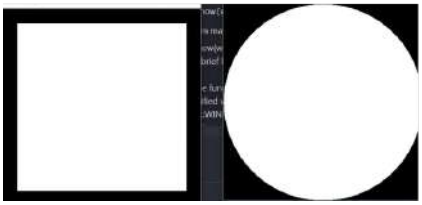
#Bilateral Blurring(applies blurring but retains(hold) the edges in the images)
-bilateral=cv.bilateral(src,d,sigmaColor,sigmaSpace,dst,borderType)

src − A Mat object representing the source (input image) for this operation.
dst − A Mat object representing the destination (output image) for this operation.
d − A variable of the type integer representing the diameter of the pixel neighborhood.
sigmaColor − A variable of the type integer representing the filter sigma in the color space.
sigmaSpace − A variable of the type integer representing the filter sigma in the coordinate space.
borderType − An integer object representing the type of the border used.

BITWISE OPERATIONS

--and,or,xor
-don't forget import the numpy

```
rectangle = cv.rectangle(blank.copy(), (30,30), (370,370), 255, -1)
circle = cv.circle(blank.copy(), (200,200), 200, 255, -1)
```



#bitwise and

bitwise_and ( InputArray src1,
InputArray src2,
OutputArray dst,
InputArray mask = noArray()
)

```
# bitwise OR --> non-intersecting and intersecting regions
bitwise_or = cv.bitwise_or(rectangle, circle)
cv.imshow('Bitwise OR', bitwise_or)

# bitwise XOR --> non-intersecting regions
bitwise_xor = cv.bitwise_xor(rectangle, circle)
cv.imshow('Bitwise XOR', bitwise_xor)
```

MASKING

-masked=cv.bitwise_and(src1,src2,dst)

```
masked = cv.bitwise_and(img,img,mask=mask)
cv.imshow('Masked Image', masked)
```

```
mask = cv.rectangle(blank, (img.shape[1]//2,img.shape[0]//2), , 100, 255, -1)
```

COMPUTING HISTOGRAMS

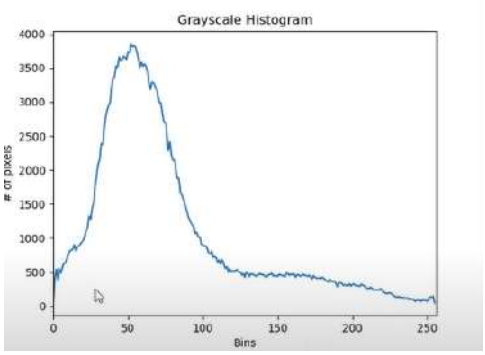
-calcHist(images,channels,mask,histSize,ranges,hist,accumulate)

histSize is basically the number of bins that we want to use for computing the histogram.

#GRayscale histogram example:

```
import cv2 as cv
import matplotlib.pyplot as plt

img = cv.imread('Photos/cats.jpg')
cv.imshow('Cats', img)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Gray', gray)

# GRayscale histogram
gray_hist = cv.calcHist([gray], [0], None, [256], [0,256] )

plt.figure()
plt.title('Grayscale Histogram')
plt.xlabel('Bins')
plt.ylabel('# of pixels')
plt.plot(gray_hist)
plt.xlim([0,256])

cv.waitKey(0)
```

Output:




Thresholding

- cv.threshold(src,thresh,maxval,type,dst) ımage compares each pixel value to this threshold value.

Example:

```
# Simple Thresholding
threshold, thresh = cv.threshold(gray, 150, 255, cv.THRESH_BINARY )
cv.imshow('Simple Thresholded', thresh)
```

To inverse:

```
threshold, thresh_inv = cv.threshold(gray, 150, 255, cv.THRESH_BINARY_INV )
cv.imshow('Simple Thresholded Inverse', thresh_inv)
```

Adaptive Threshold(tells machine which method to use when computing the optimal threshold value)

-cv.adaptiveThreshold(src,maxValue,adaptiveMethod,thresholdType,clockSize,dst)

All simple thresholding types are:

- **cv.THRESH_BINARY**
- **cv.THRESH_BINARY_INV**
- **cv.THRESH_TRUNC**
- **cv.THRESH_TOZERO**
- **cv.THRESH_TOZERO_INV**

EDGE DETECTION

-Laplacian
lap=cv.Laplacian(src,dst(destination),ddepth,ksize,scale,delta)
(source=Image Filtering)

| | |
|---|---|
| **src** | Source image. |
| **dst** | Destination image of the same size and the same number of channels as src . |
| **ddepth** | Desired depth of the destination image, see **combinations**. |
| **ksize** | Aperture size used to compute the second-derivative filters. See **getDerivKernels** fo details. The size must be positive and odd. |
| **scale** | Optional scale factor for the computed Laplacian values. By default, no scaling is applied. See **getDerivKernels** for details. |
| **delta** | Optional delta value that is added to the results prior to storing them in dst . |

The Laplacian edge detector uses only one kernel.It calculates second order derivatives in a single pass.
Example:

```
# Laplacian
lap = cv.Laplacian(gray, cv.CV_64F)
lap = np.uint8(np.absolute(lap))
cv.imshow('Laplacian', lap)
```

-Sobel

Using the sobel operation, you can detect the edges of an image in both horizontal and vertical directions. You can apply sobel operation on an image using the method sobel().
Following is the syntax of this method −
Sobel(src, dst, ddepth, dx, dy)

This method accepts the following parameters −

src − An object of the class Mat representing the source (input) image.dst − An object of the class Mat representing the destination (output) image.
ddepth − An integer variable representing the depth of the image (-1)
dx − An integer variable representing the x-derivative. (0 or 1)
dy − An integer variable representing the y-derivative. (0 or 1)

The sobel is one of the most commonly used edge detectors. It is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations.

Face Detection

-Detects the presence of a face in an image,while face recognition involves identifying whose face it is.
-First go to github and paste:haarascade_frontalface_default..xml to your workspace
To read this file in that code:

```
haar_cascade = cv.CascadeClassifier('haar_face.xml')
```

.detectMultiScale(image,scaleFactor,minNeighbors,flags,minSize,maxSize)

To detect a face example:(ıt will show the face in a square)

```
haar_cascade = cv.CascadeClassifier('haar_face.xml')

faces_rect = haar_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=3)

print(f'Number of faces found = {len(faces_rect)}')

for (x,y,w,h) in faces_rect:
    cv.rectangle(img, (x,y), (x+w,y+h), (0,255,0), thickness=2)

cv.imshow('Detected Faces')
```

.LBPHFaceRecognizer_create()

This method in OpenCV is used to create a face recognition object based on the Local Binary Pattern Histogram (LBPH) algorithm. It's particularly useful for face recognition applications. Below are the parameters explained in English:

Parameters:
radius (default=1):
Integer value representing the radius used for building the Local Binary Pattern. It defines the radius around the central pixel where the circular pattern is calculated.
neighbors (default=8):
Integer value specifying the number of sample points to be considered in the circular pattern around each pixel.
grid_x (default=8):

Integer value defining the number of cells in the horizontal direction. The face region is divided into a grid of cells, and LBP histograms are calculated for each cell.
grid_y (default=8):
Integer value defining the number of cells in the vertical direction.
threshold (default=50.0):

Double value representing the threshold used in the prediction phase. If the confidence score for a match is below this threshold, the prediction is considered unreliable.
histogram_bins (default=8):

Integer value determining the number of bins to be used in the histograms representing the Local Binary Patterns.