A Python Program to load animage in grey format:

NB/Install a Package using pip

cv2.IMREAD_COLOR: It specifies to load a color image. Any transparency of image will beneglected. It is the default flag. Alternatively, we can pass integer value 1 for this flag.

cv2.IMREAD_GRAYSCALE: It specifies to load animage in grayscale mode. Alternatively, wecan pass integer value 0 for this flag.

cv2.IMREAD_UNCHANGED: It specifies to load animage as such including alpha channel.Alternatively, we can pass integer value - 1 for this flag.

EXAMPLE

```
import cv2
# path
path = r'C:\Users\WAINAINA\Desktop\VISION\
WAINAINA.jpg'
# Using cv2.imread() method
img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
# Displaying the image
cv2.imshow("My First Computer Vision Program", img)
cv2.waitKey(0)
```

Feature detection and matching:

Resize an RGB image

in the case of RGB image we have 3 planes i.e. Red, Green, Blue plane. So, for resizing RGB image we have to take each individual plane then perform resize operation on it and then merge it all.

Python program to resize an RGB image

```
# open-cv library is installed as cv2 in python
# import cv2 library into this program
import cv2
# import numpy library as np into this program
import numpy as np
# read an image using imread() function of cv2
# we have to pass only the path of the image
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
img = cv2.imread(path)
# displaying the image using imshow() function of cv2
# In this : 1st argument is name of the frame
# 2nd argument is the image matrix
cv2.imshow('original image',img)
# print shape of the image matrix
# using shape attribute
print("original image shape:",img.shape)
# assigning number of rows, coulmns and
# planes to the respective variables
row,col,plane = img.shape
# give value by which you want to resize an image
# here we want to resize an image as one half of the original image
x, y = 2, 2
# assign Blue plane of the BGR image
# to the blue_plane variable
blue_plane = img[:,:,0]
# assign Green plane of the BGR image
# to the green_plane variable
green_plane = img[:,:,1]
# assign Red plane of the BGR image
# to the red_plane variable
red_plane = img[:,:,2]
# we take one-half pixel of rows and columns from
# each plane respectively so that, it is one-half of image matrix.
# here we take alternate row,column pixel of blue plane.
resize_blue_plane = blue_plane[1::x,1::x]
# here we take alternate row,column pixel of green plane.
resize_green_plane = green_plane[1::x,1::x]
# here we take alternate row,column pixel of red plane.
resize_red_plane = red_plane[1::x,1::x]
# here image is of class 'uint8', the range of values
# that each colour component can have is [0 - 255]
# create a zero matrix of specified order of 3-dimension
resize_img = np.zeros((row//x, col//y, plane),np.uint8)
# assigning resized blue, green and red plane of image matrix to the
# corresponding blue, green, red plane of resize_img matrix variable.
resize_img[:,:,0] = resize_blue_plane
resize_img[:,:,1] = resize_green_plane
resize_img[:,:,2] = resize_red_plane
cv2.imshow('resize image',resize_img)
print("resize image shape:",resize_img.shape)# open-cv library is installed as
cv2 in python
cv2.waitKey(0)
```

The below code reads an input image translates it and shows it:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
```

```
# Using cv2.imread() method
img = cv2.imread(path)
# convert from BGR to RGB so we can plot using matplotlib
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# disable x & y axis
plt.axis('off')
# show the image
plt.imshow(img)
plt.show()
# get the image shape
rows, cols, dim = img.shape
# transformation matrix for translation
M = np.float32([[1, 0, 50],
[0, 1, 50],
[0, 0, 1]])
# apply a perspective transformation to the image
translated_img = cv2.warpPerspective(img, M, (cols, rows))
# disable x & y axis
plt.axis('off')
# show the resulting image
plt.imshow(translated_img)
plt.show()
# save the resulting image to disk
plt.imsave("city_translated.jpg", translated_img)
```

A **python Program to detect face from an imag**e:

```
import cv2
# Load the cascade
face_cascade =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml'
)
# Read the input image
path = r'C:\Users\WAINAINA\Desktop\VISION\G.jpg'
img = cv2.imread(path)
# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
# Draw rectangle around the faces
for (x, y, w, h) in faces:
cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255,0), 2)
# Display the output
cv2.imshow('img', img)
cv2.waitKey(0)
```

A **python Program to detect Number of faces from an image:**

```
import cv2
# Load the cascade
face_cascade =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml'
)
# Read the input image
path = r'C:\Users\WAINAINA\Desktop\VISION\GEO.jpg'
img = cv2.imread(path)
# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
```

```
print ("Found {0} faces!".format(len(faces)))
# Draw rectangle around the faces
for (x, y, w, h) in faces:
cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0,255), 2)
# Display the output
cv2.imshow('img', img)
cv2.waitKey(0)
```

A **python Program to detect eyes from an image:**

```
import cv2
# Load the cascade
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_eye.xml')
# Read the input image
path = r'C:\Users\WAINAINA\Desktop\VISION\GEO.jpg'
img = cv2.imread(path)
# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Detect eyes
eyes = eye_cascade.detectMultiScale(img, scaleFactor = 1.2, minNeighbors = 4)
# Draw rectangle around the faces
for (x,y,w,h) in eyes:
cv2.rectangle(img,(x,y),(x+w,y+h),(0, 255, 0),5)
# Display the output
cv2.imshow("Eyes Detected", img)
cv2.waitKey(0)
```

A **python Program to detect Number of eyes from an image:**

```
import cv2
# Load the cascade
```

```
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_eye.xml')

# Read the input image

path = r'C:\Users\WAINAINA\Desktop\VISION\G.jpg'

img = cv2.imread(path)

# Convert into grayscale

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

# Detect eyes

eyes = eye_cascade.detectMultiScale(img, scaleFactor = 1.2, minNeighbors = 4)

print ("Found {0} Eyes!".format(len(eyes)))

# Draw rectangle around the faces

for (x,y,w,h) in eyes:

cv2.rectangle(img,(x,y),(x+w,y+h),(0, 0, 255),5)

# Display the output

cv2.imshow("Eyes Detected", img)

cv2.waitKey(0)
```

IMAGE REFLECTOMETRY INVOLVING COLOR

```
import cv2

import numpy as np

# image path

path = r'C:\Users\WAINAINA\Desktop\VISION\wainaina.jpg'

# using imread()

input_image = cv2.imread(path)

if input_image is None:

print('Could not load image: ', input_image)

exit(0)

# Splitting image into RGB channels:

blue, green, red = cv2.split(input_image)

# We create a dummy 3D array
```

```
blue_channel = np.zeros(input_image.shape, input_image.dtype)
green_channel = np.zeros(input_image.shape, input_image.dtype)
red_channel = np.zeros(input_image.shape, input_image.dtype)
# We match each color channel to a 3D dimension:
# Blue Rendering : [blue; 0; 0]
# Green Rendering: [0; green; 0]
# Red Rendering: [0; 0; red]
cv2.mixChannels([blue, green, red], [blue_channel], [0,0])
cv2.mixChannels([blue, green, red], [green_channel], [1,1])
cv2.mixChannels([blue, green, red], [red_channel], [2,2])
# Display the three obtained images
cv2.imshow('Blue Channel', blue_channel)
cv2.imshow('Green Channel', green_channel)
cv2.imshow('Red Channel', red_channel)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Blurring an Image using a Custom 2D-Convolution Kernel

```
import cv2 as cv
import numpy as np
import cv2
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
image = cv.imread(path)
# Print error message if image is null
if image is None:
print('Could not read image')
# Apply identity kernel
kernel1 = np.array([[0, 0, 0],
[0, 1, 0],
[0, 0, 0]])
identity = cv2.filter2D(src=image, ddepth=-1, kernel=kernel1)
cv2.imshow('Original', image)
cv2.imshow('Identity', identity)
cv2.waitKey()
cv2.imwrite('identity.jpg', identity)
cv2.destroyAllWindows()
# Apply blurring kernel
kernel2 = np.ones((5, 5), np.float32) / 25
img = cv2.filter2D(src=image, ddepth=-1, kernel=kernel2)
cv2.imshow('Original', image)
cv2.imshow('Kernel Blur', img)
cv2.waitKey()
cv2.imwrite('blur_kernel.jpg', img)
cv2.destroyAllWindows()
```

<u>Sharpen Kernel/image</u>

```
_load the required packages
import cv2 as cv
import numpy as np
import cv2
# load the image into system memory
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
image = cv.imread(path)
# display the image to the screen
cv2.imshow('Original Image', image)
cv2.waitKey()
cv2.destroyAllWindows()
kernel = np.array([[0, -1, 0],
[-1, 5,-1],
[0, -1, 0]])
image_sharp = cv2.filter2D(src=image, ddepth=-1, kernel=kernel)
cv2.imshow('Sharpened Image', image_sharp)
cv2.waitKey()
cv2.destroyAllWindows()
```

The code given below demonstrates Gaussian Blur Filter:

```
# Importing the OpenCV and Numpy libraries
import cv2
import numpy as np
# Reading the image from the disk using cv2.imread() function
# Showing the original image using matplotlib library function plt.imshow()
path = r'C:\Users\WAINAINA\Desktop\VISION\wainaina.jpg'
img = cv2.imread(path)
cv2.imshow('Original image',img)
cv2.waitKey()
# Applying Gaussian Blur Filter using cv2.GaussianBlur() function
# src is the source of image(here, img)
# ksize is the size of kernel in the form A x B (here 3 x 3)
# sigmaX is standard deviation of X axis
# sigmaY is the standard deviation of Y axis
# Since sigmaX and sigmaY is 0, the standard deviation the size of kernel
gaussian_blur = cv2.GaussianBlur(src=img, ksize=(3,3),sigmaX=0, sigmaY=0)
# Showing the Gaussian blur image using matplotlib library function plt.imshow()
cv2.imshow('GAUSSIAN BLUR',gaussian_blur)
cv2.waitKey()
```

Example of Median Filter

```
# Importing the OpenCV and Numpy libraries
import cv2
import numpy as np
# Reading the image from the disk using cv2.imread() function
# Showing the original image using matplotlib library function plt.imshow()
path = r'C:\Users\WAINAINA\Desktop\VISION\wainaina.jpg'
img = cv2.imread(path)
cv2.imshow('Original image',img)
cv2.waitKey()
# Applying median Blur Filter using cv2.medianBlur() function
# src is the source of image(here, img)
# ksize is the size of kernel. Should have a positive odd value
median_blur = cv2.medianBlur(src=img, ksize=9)
# Showing the Median blur image using matplotlib library function plt.imshow()
cv2.imshow('MEDIAN BLUR',median_blur)
cv2.waitKey()
```

BILATERAL FILTER

```
# image path
path = r'C:\Users\WAINAINA\Desktop\VISION\wainaina.jpg'
# using imread()
img = cv2.imread(path)
dst = cv2.bilateralFilter(img, 5, 10, 10)
cv2.imshow('image', numpy.hstack((img, dst)))
cv2.waitKey(0);
cv2.destroyAllWindows();
cv2.waitKey(1)
```

Emboss:

Emboss effect is similar to Edge extraction but it is more like a 3D effect. An embossing filter will take an image and convert it into an embossed image. We basically take each pixel and replace it with a shadow or a highlight.

```
# Importing the OpenCV, Numpy and Matplotlib libraries
import cv2
import numpy as np
# Reading the image from the disk using cv2.imread() function
# Showing the original image using matplotlib library function plt.imshow()
path = r'C:\Users\WAINAINA\Desktop\VISION\wainaina.jpg'
img = cv2.imread(path)
cv2.imshow('Original image',img)
cv2.waitKey()
# Apply kernel for embossing
emboss_kernel = np.array([[1, 0, 0],
[0, 0, 0],
[0, 0, -1]])
# Embossed image is obtained using the variable emboss_img
# cv2.fliter2D() is the function used
# src is the source of image(here, img)
# ddepth is destination depth. -1 will mean output image will have same depth as
input image
# kernel is used for specifying the kernel operation (here, emboss_kernel)
emboss_img = cv2.filter2D(src=img, ddepth=-1, kernel=emboss_kernel)
# Showing the embossed image using matplotlib library function plt.imshow()
cv2.imshow('EMBOSS IMAGE',emboss_img)
cv2.waitKey()
```

Real-Time Blur of video in OpenCV Python:

```
import cv2
import numpy as np
cap = cv2.VideoCapture(0)
while(1):
_, frame = cap.read()
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
lower_red = np.array([30,150,50])
upper_red = np.array([255,255,180])
mask = cv2.inRange(hsv, lower_red, upper_red)
res = cv2.bitwise_and(frame,frame, mask= mask)
kernel = np.ones((15,15),np.float32)/225
blur = cv2.GaussianBlur(res,(15,15),0)
cv2.imshow('Original',frame)
cv2.imshow('Gaussian Blurring',blur)
k = cv2.waitKey(5) & 0xFF
if k == 27:
break
cv2.destroyAllWindows()
cap.release()
```