OpenCV Basic Operation on Images

In this tutorial, we will learn the essential operations that are related to the images. We are going to discuss the following topics.

- Access pixel values and modify them
- Access Image Properties
- Setting Region of Image
- Splitting and merging images
- Change the image color

Accessing and Modifying pixel values

We can retrieve a pixel value by its row and column coordinates. It returns an array of blue, green, red values of the BGR image. It returns the corresponding intensity for the grayscale image. First, we need to load the BGR image.

```
import numpy as np
import cv2
img = cv2.imread('C:\Users\DEVANSH SHARMA\cat.jpeg',1)
pixel = img[100,100]
print(pixel)
```

Output:

```
[198 166 250]
```

Accessing Image Properties

It is better to know the size of the image to work with the image processing application. In OpenCV, images are generally stored in the Numpy ndarray. To get the image shape or size, use ndarray.shape to find the dimension of the image. Then, we can use the index position to get the height, width, and number of channels.

Consider the following example:

```
import cv2
# read image
img = cv2.imread('C:\Users\DEVANSH SHARMA\cat.jpeg',1)

# height, width, number of channels in image
height = img.shape[0]
width = img.shape[1]
channels = img.shape[2]
size1 = img.size

print('Image Dimension    : ',dimensions)
print('Image Height       : ',height)
print('Image Width        : ',width)
print('Number of Channels : ',channels)
print('Image Size         : ',size1)
```

Output:

```
Image Dimension    :  (4, 1, 3)
Image Height       :  4
Image Width        :  1
Number of Channels :  3
Image Size         :  12
```

Image ROI (Region of Interest)

Sometimes, we need to work with some areas of the image. As we discuss in the previous tutorial face detection is over the entire picture. When a face is obtained, we select only the face region and search for eyes inside it instead of searching the whole image. It enhances accuracy and performance because eyes are always on the face and don't need to search the entire image.

In the above image, if we need to select the ball. We only require selecting the ball region.

Splitting and Merging Image channels

An image's BGR channels can be split into their planes when needed. Then, the individual channels can be merged back together from the BGR image again. This can be done by following way:

```
b,g,r = cv2.split(img)
img = cv2.merge((b,g,r))
```

or

```
b = img[:,:,0]
```

> Note: The cv2.split() function is a slow function. Numpy indexing is quit efficient and it should be used if possible.

Making Borders for Images

OpenCV provides the **cv2.copyMakeBorder()** function to create a border around the image, something like a photo frame. The syntax of the function is given below.

```
cv2.copyMakeBorder(src,top,bottom,left,right,border type)
```

Parameters:

src = It donates input image.

top, bottom, left, right = It defines the border width in the number of pixels in the corresponding direction.

borderType = It defines what kind of border to be added. The border can be the following types.

value = Color of border if border type is cv.BORDER_CONSTANT

Consider the following example:

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
BLUE = [255,0,0]
img1 = cv.imread('C:\Users\DEVANSH SHARMA\flower.jpg',1)
replicate = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER_REPLICATE)
reflect = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER_REFLECT)
reflect101 = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER_REFLECT_101)
wrap = cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER_WRAP)
constant= cv.copyMakeBorder(img1,10,10,10,10,cv.BORDER_CONSTANT,value=BLUE)
plt.subplot(231),plt.imshow(img1,'gray'),plt.title('ORIGINAL')
plt.subplot(232),plt.imshow(replicate,'gray'),plt.title('REPLICATE')
plt.subplot(233),plt.imshow(reflect,'gray'),plt.title('REFLECT')
plt.subplot(234),plt.imshow(reflect101,'gray'),plt.title('REFLECT_101')
plt.subplot(235),plt.imshow(wrap,'gray'),plt.title('WRAP')
plt.subplot(236),plt.imshow(constant,'gray'),plt.title('CONSTANT')
plt.show()
```

Change in Image color

OpenCV cvtColor

The **cvtColor** is used to convert an image from one color space to another. The syntax is following:

```
cv2.cvtColor(src, dst, code)
```

Parameters:

src = It is used to input an image: 8-bit unsigned.

dst = It is used to display an image as output. The output image will be same size and depth as input image.

code = color space conversion code.

Consider the following example:

```
# importing cv2
import cv2

# path of the input image
path = ('C:\Users\DEVANSH SHARMA\cat.jpeg')

# Reading an image in default mode
src = cv2.imread(path)

# Window name in which image is displayed
window_name = 'Image'
# Using cv2.cvtColor() method
# Using cv2.COLOR_BGR2GRAY color space for convert BGR image to grayscale
# conversion code
image = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY )
# Displaying the image
cv2.imshow(window_name, image)
```

Output:

Next Topic OpenCV Resize Image

← prev | next →