Arithmetic Operations on Images

Goal

- Learn several arithmetic operations on images, like addition, subtraction, bitwise operations, and etc.
- Learn these functions: **cv.add()**, **cv.addWeighted()**, etc.

Image Addition

You can add two images with the OpenCV function, **cv.add()**, or simply by the numpy operation res = img1 + img2. Both images should be of same depth and type, or the second image can just be a scalar value.

Note

There is a difference between OpenCV addition and Numpy addition. OpenCV addition is a saturated operation while Numpy addition is a modulo operation.

For example, consider the below sample:

```
>>> x = np.uint8([250])
>>> y = np.uint8([10])

>>> print( cv.add(x,y) ) # 250+10 = 260 => 255
[[255]]

>>> print( x+y )          # 250+10 = 260 % 256 = 4
[4]
```

This will be more visible when you add two images. Stick with OpenCV functions, because they will provide a better result.

Image Blending

This is also image addition, but different weights are given to images in order to give a feeling of blending or transparency. Images are added as per the equation below:

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

By varying α from $0 \rightarrow 1$, you can perform a cool transition between one image to another.

Here I took two images to blend together. The first image is given a weight of 0.7 and the second image is given 0.3. **cv.addWeighted()** applies the following equation to the image:

$$dst = \alpha \cdot img1 + \beta \cdot img2 + \gamma$$

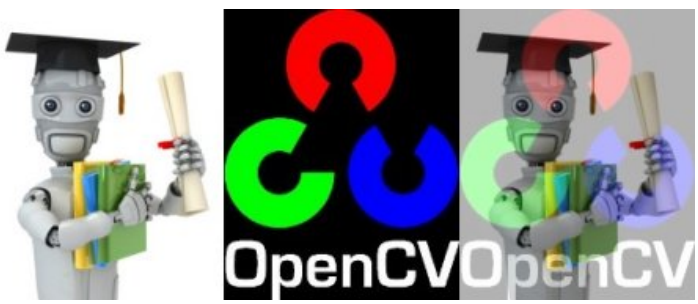
Here γ is taken as zero.

```
img1 = cv.imread('ml.png')
img2 = cv.imread('opencv-logo.png')

dst = cv.addWeighted(img1,0.7,img2,0.3,0)

cv.imshow('dst',dst)
cv.waitKey(0)
cv.destroyAllWindows()
```

Check the result below:



image

Bitwise Operations

This includes the bitwise AND, OR, NOT, and XOR operations. They will be highly useful while extracting any part of the image (as we will see in coming chapters), defining and working with non-rectangular ROI's, and etc. Below we will see an example of how to change a particular region of an image.

I want to put the OpenCV logo above an image. If I add two images, it will change the color. If I blend them, I get a transparent effect. But I want it to be opaque. If it was a rectangular region, I could use ROI as we did in the last chapter. But the OpenCV logo is a not a rectangular shape. So you can do it with bitwise operations as shown below:

```
# Load two images
img1 = cv.imread('messi5.jpg')
img2 = cv.imread('opencv-logo-white.png')

# I want to put logo on top-left corner, So I create a ROI
rows,cols,channels = img2.shape
roi = img1[0:rows, 0:cols]

# Now create a mask of logo and create its inverse mask also
img2gray = cv.cvtColor(img2,cv.COLOR_BGR2GRAY)
ret, mask = cv.threshold(img2gray, 10, 255, cv.THRESH_BINARY)
mask_inv = cv.bitwise_not(mask)

# Now black-out the area of logo in ROI
img1_bg = cv.bitwise_and(roi,roi,mask = mask_inv)

# Take only region of logo from logo image.
img2_fg = cv.bitwise_and(img2,img2,mask = mask)

# Put logo in ROI and modify the main image
dst = cv.add(img1_bg,img2_fg)
img1[0:rows, 0:cols ] = dst

cv.imshow('res',img1)
cv.waitKey(0)
cv.destroyAllWindows()
```

See the result below. Left image shows the mask we created. Right image shows the final result. For more understanding, display all the intermediate images in the above code, especially img1_bg and img2_fg.

image

Additional Resources

Exercises

1. Create a slide show of images in a folder with smooth transition between images using **cv.addWeighted** function