Arithmetic and Bitwise Operations on Images

Addition and Subtraction of Images

- `dst=cv2.add(src1, src2[, dst[, mask[, dtype]]])`
- `dst=cv2.subtract(src1, src2[, dst[, mask[, dtype]]])`

src1 first input array or a scalar.

src2 second input array or a scalar.

dst output array that has the same size and number of channels as the input array(s); the depth is defined by dtype or src1/src2.

mask optional operation mask - 8-bit single channel array, that specifies elements of the output array to be changed.

dtype optional depth of the output array

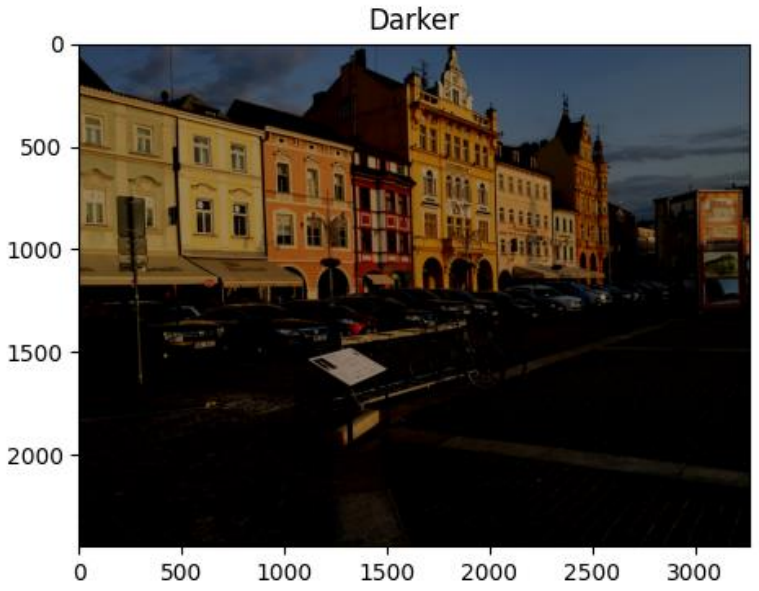
Adding/subtracting a constant value from each pixel

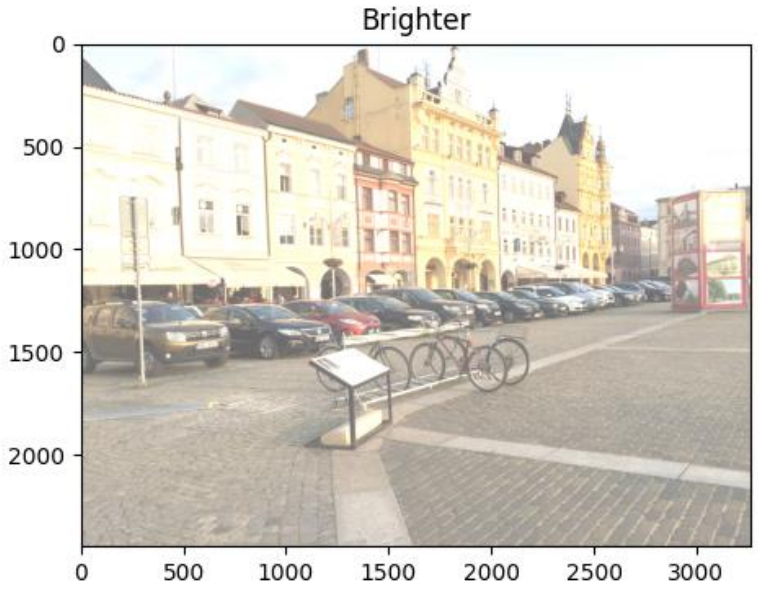
```
img = cv2.imread('FamilyTrip.jpg', cv2.IMREAD_COLOR)
# Create a matrix with constant intensity.
matrix = np.ones(img.shape, dtype = 'uint8') * 100

# Create brighter and darker images.
img_brighter = cv2.add(img, matrix)
img_darker   = cv2.subtract(img, matrix)

# Display the images
plt.figure(figsize = [18,5])
plt.subplot(131); plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB));   plt.title('Original')
plt.subplot(132); plt.imshow(cv2.cvtColor(img_darker, cv2.COLOR_BGR2RGB));   plt.title('Darker')
plt.subplot(133); plt.imshow(cv2.cvtColor(img_brighter, cv2.COLOR_BGR2RGB)); plt.title('Brighter')
plt.show()
```

Adding/subtracting a constant value from each pixel



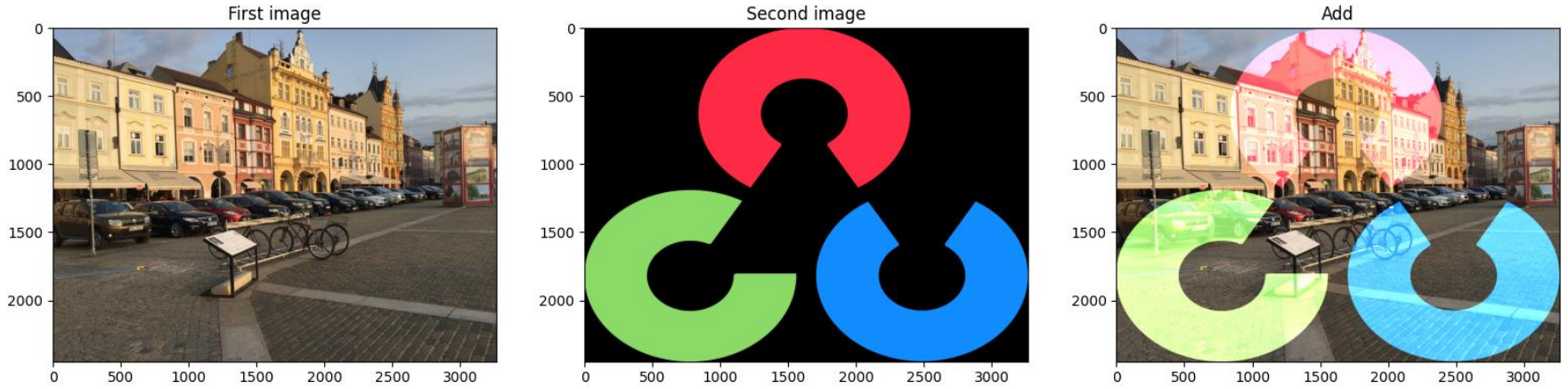


Adding/subtracting a constant value from each pixel

- OpenCV clips values outside the range of the min/max image values

```
>>> cv2.add(np.uint8([200]), np.uint8([100]))
array([[255]], dtype=uint8)
>>> cv2.subtract(np.uint8([30]),np.uint8([100]))
array([[0]], dtype=uint8)
```

Example




```
img = cv2.imread('FamilyTrip.jpg', cv2.IMREAD_COLOR)
opencv_logo = cv2.imread('opencv_logo.png', cv2.IMREAD_COLOR)
h,w = img.shape[0], img.shape[1]
opencv_logo = cv2.resize(opencv_logo, (w,h))
img_add = cv2.add(img, opencv_logo)

plt.figure()
plt.subplot(131); plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB));
plt.title('First image')
plt.subplot(132); plt.imshow(cv2.cvtColor(opencv_logo, cv2.COLOR_BGR2RGB));
plt.title('Second image')
plt.subplot(133); plt.imshow(cv2.cvtColor(img_add, cv2.COLOR_BGR2RGB));
plt.title('Add')
plt.show()
```

Bitwise operations on images: AND, NOT, OR, XOR

- AND and OR operators are applied to unsigned 8-bit integers

```
dst = cv2.bitwise_and( src1, src2[, dst[, mask]] )
dst = cv2.bitwise_or( src1, src2[, dst[, mask]] )
dst = cv2.bitwise_xor( src1, src2[, dst[, mask]] )
dst = cv2.bitwise_not( src[, dst[, mask]] )
```

src1 first input array or a scalar.

src2 second input array or a scalar.

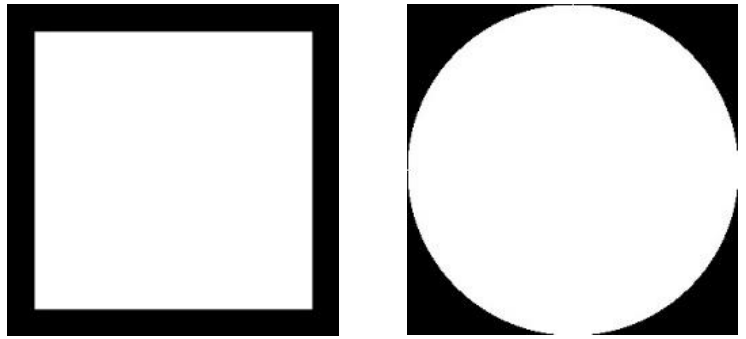
dst output array that has the same size and type as the input arrays.

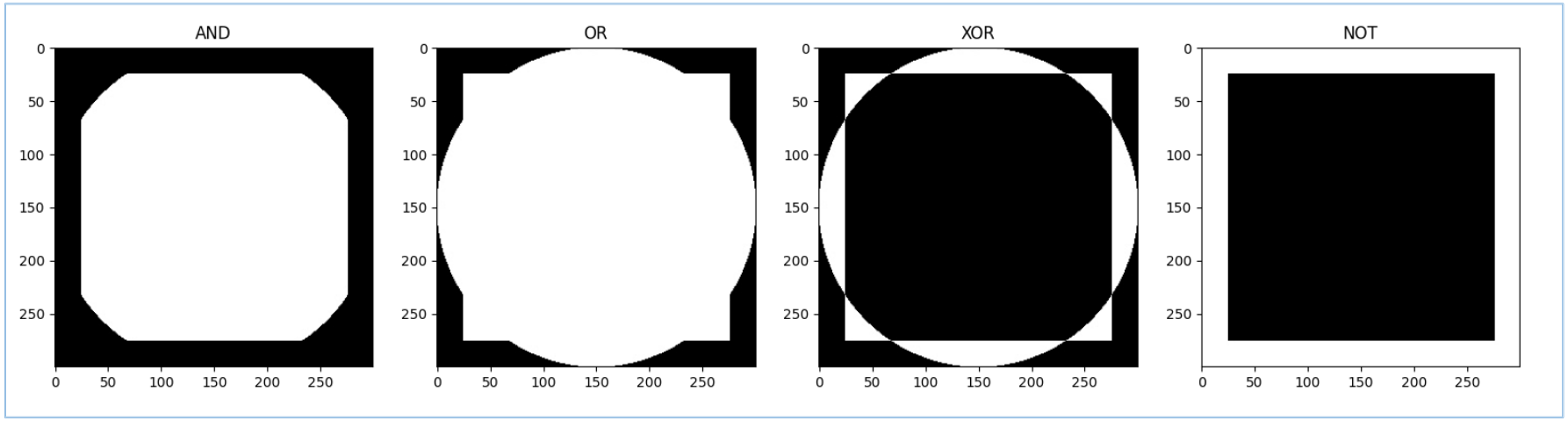
mask optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed.

Example

```
rectangle = cv2.imread('rectangle.jpg', cv2.IMREAD_GRAYSCALE)
circle = cv2.imread('circle.jpg', cv2.IMREAD_GRAYSCALE)

bitwiseAnd = cv2.bitwise_and(rectangle, circle)
bitwiseOr  = cv2.bitwise_or(rectangle, circle)
bitwiseXor = cv2.bitwise_xor(rectangle, circle)
bitwiseNot = cv2.bitwise_not(rectangle, rectangle)
```





Masking

```
img  = …
mask = …

result = cv2.bitwise_and(img, img, mask=mask)
```