# Arithmetic Operations on Images using OpenCV | Set-1 (Addition and Subtraction)

Arithmetic Operations like Addition, Subtraction, and Bitwise Operations(AND, OR, NOT, XOR) can be applied to the input images. These operations can be helpful in enhancing the properties of the input images. The Image arithmetics are important for analyzing the input image properties. The operated images can be further used as an enhanced input image, and many more operations can be applied for clarifying, thresholding, dilating etc of the image. 

### Addition of Image:

We can add two images by using function **cv2.add()**. This directly adds up image pixels in the two images. 

**Syntax:** cv2.add(img1, img2)

But adding the pixels is not an ideal situation. So, we use cv2.addweighted(). Remember, both images should be of equal size and depth. 

***Syntax****: cv2.addWeighted(img1, wt1, img2, wt2, gammaValue)****Parameters****:****img1****: First Input Image array(Single-channel, 8-bit or floating-point)****wt1****: Weight of the first input image elements to be applied to the final image****img2****: Second Input Image array(Single-channel, 8-bit or floating-point)****wt2****: Weight of the second input image elements to be applied to the final image****gammaValue****: Measurement of light*

**Images used as Input:**  
Input Image1:



Input Image2:



Below is the code: 

* Python3

|  |
| --- |
| # Python program to illustrate  # arithmetic operation of  # addition of two images    # organizing imports  import cv2  import numpy as np    # path to input images are specified and  # images are loaded with imread command  image1 = cv2.imread('input1.jpg')  image2 = cv2.imread('input2.jpg')    # cv2.addWeighted is applied over the  # image inputs with applied parameters  weightedSum = cv2.addWeighted(image1, 0.5, image2, 0.4, 0)    # the window showing output image  # with the weighted sum  cv2.imshow('Weighted Image', weightedSum)    # De-allocate any associated memory usage  if cv2.waitKey(0) & 0xff == 27:      cv2.destroyAllWindows() |

**Output:**

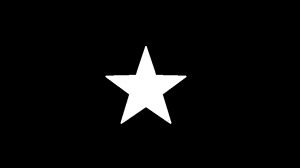


### Subtraction of Image:

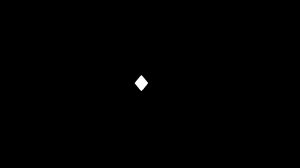
Just like addition, we can subtract the pixel values in two images and merge them with the help of cv2.subtract(). The images should be of equal size and depth. 

**Syntax:**  cv2.subtract(src1, src2)

**Images used as Input:**  
Input Image1:



Input Image2:



Below is the code:

* Python3

|  |
| --- |
| # Python program to illustrate  # arithmetic operation of  # subtraction of pixels of two images    # organizing imports  import cv2  import numpy as np    # path to input images are specified and  # images are loaded with imread command  image1 = cv2.imread('input1.jpg')  image2 = cv2.imread('input2.jpg')    # cv2.subtract is applied over the  # image inputs with applied parameters  sub = cv2.subtract(image1, image2)    # the window showing output image  # with the subtracted image  cv2.imshow('Subtracted Image', sub)    # De-allocate any associated memory usage  if cv2.waitKey(0) & 0xff == 27:      cv2.destroyAllWindows() |

**Output:**

