****

**Lab Manual**

**Computer Engineering – Artificial Intelligence**

**B. Tech. Year – II, 5th Semester, Academic Year (2023)**

**Subject Code: 01AI0504**

*Subject Name: Digital Image Processing*

*Name: Basid Al Siddik Shammo*

*Enrollment Number: 92100151052*

**Aim-** Display of an Image, Negative of an Image (Binary & Gray Scale)

**Description-** 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.

**Task- I**

**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)

**Images used as Input:**

**Input Image1:**

****

**Input Image2:**

****

**Code-**

import cv2

import numpy as np

image1 = cv2.imread('input1.jpg')

image2 = cv2.imread('input2.jpg')

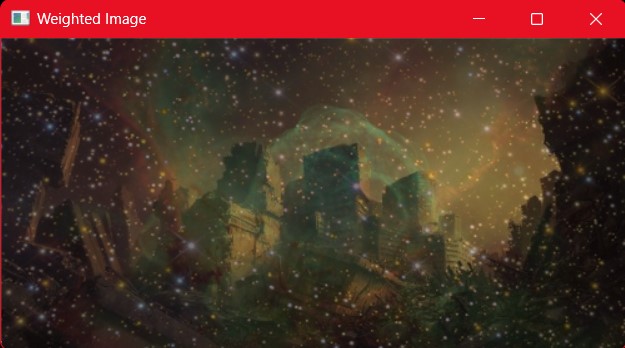
weightedSum = cv2.addWeighted(image1, 0.5, image2, 0.4, 0)

cv2.imshow('Weighted Image', weightedSum)

if cv2.waitKey(0) & 0xff == 27:

cv2.destroyAllWindows()

**Result-**

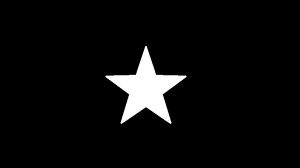
****

**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.

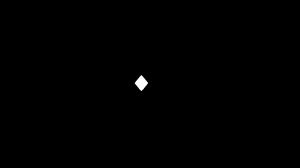
**Syntax:** cv2.subtract(src1, src2)

**Images used as Input:**

**Input Image3:**

****

**Input Image4:**

****

**Code-**

import cv2

import numpy as np

image1 = cv2.imread('input3.jpg')

image2 = cv2.imread('input4.jpg')

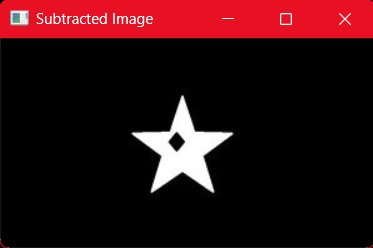
sub = cv2.subtract(image1, image2)

cv2.imshow('Subtracted Image', sub)

if cv2.waitKey(0) & 0xff == 27:

    cv2.destroyAllWindows()

**Result-**



**Task- II**

**Bitwise AND Operation on Image:** Bit-wise conjunction of input array elements.

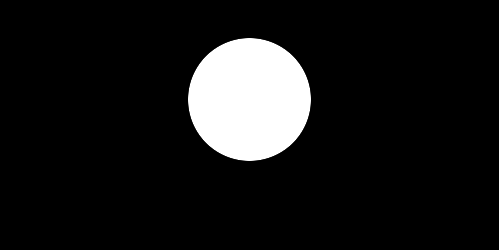
**Syntax:** cv2.bitwise\_and (source1, source2, destination, mask)

**Images used as Input:**

**Input Image5:**

****

**Input Image6:**

****

**Code-**

import cv2

import numpy as np

img1 = cv2.imread('input5.png')

img2 = cv2.imread('input6.png')

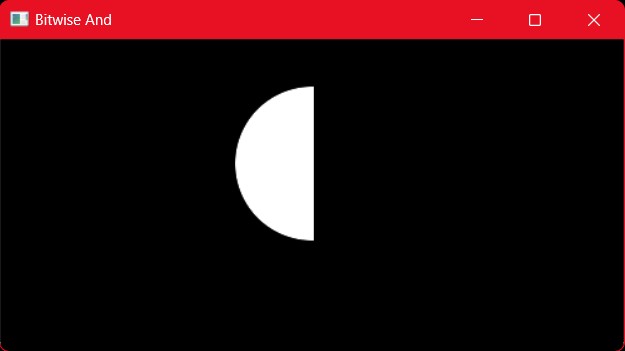
dest\_and = cv2.bitwise\_and(img2, img1, mask = None)

cv2.imshow('Bitwise And', dest\_and)

if cv2.waitKey(0) & 0xff == 27:

    cv2.destroyAllWindows()

**Result-**



**Bitwise OR Operation on Image:** Bit-wise disjunction of input array elements.

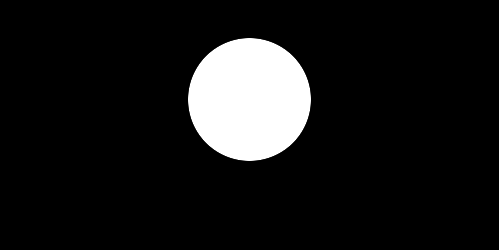
**Syntax:** cv2.bitwise\_or (source1, source2, destination, mask)

**Images used as Input:**

**Input Image7:**

****

**Input Image8:**

****

**Code-**

import cv2

import numpy as np

img1 = cv2.imread('input7.png')

img2 = cv2.imread('input8.png')

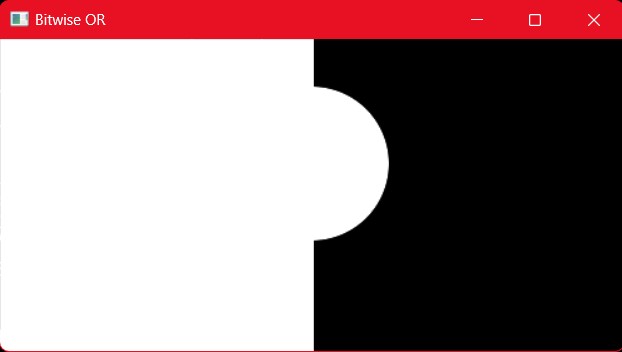
dest\_or = cv2.bitwise\_or(img2, img1, mask = None)

cv2.imshow('Bitwise OR', dest\_or)

if cv2.waitKey(0) & 0xff == 27:

    cv2.destroyAllWindows()

**Result-**



**Bitwise XOR Operation on Image:** Bit-wise exclusive OR operation of input array elements.

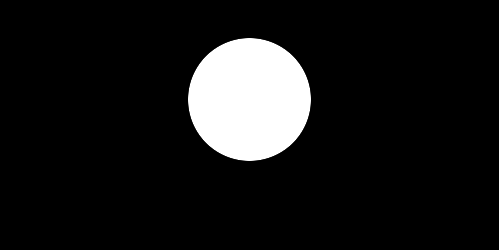
**Syntax:** cv2.bitwise\_xor (source1, source2, destination, mask)

**Images used as Input:**

**Input Image9:**

****

**Input Image10:**

****

**Code-**

import cv2

import numpy as np

img1 = cv2.imread('input9.png')

img2 = cv2.imread('input10.png')

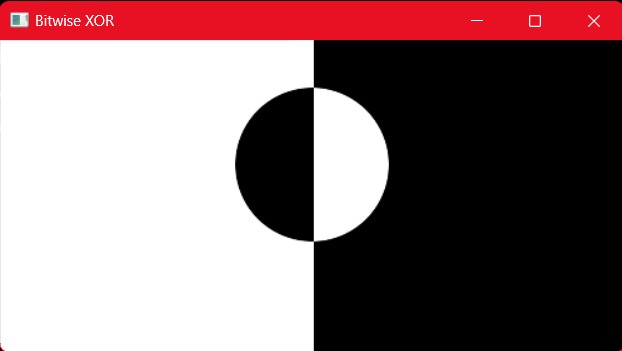
dest\_xor = cv2.bitwise\_xor(img2, img1, mask = None)

cv2.imshow('Bitwise XOR', dest\_xor)

if cv2.waitKey(0) & 0xff == 27:

    cv2.destroyAllWindows()

**Result-**



**Bitwise NOT Operation on Image:** Inversion of input array elements.

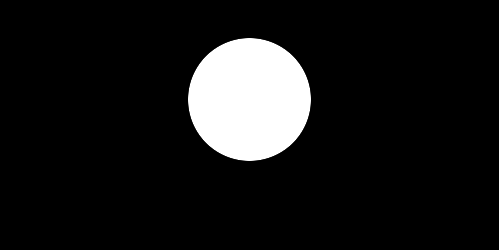
**Syntax:** cv2.bitwise\_not(source, destination, mask)

**Images used as Input:**

**Input Image9:**

****

**Input Image10:**

****

**Code-**

import cv2

import numpy as np

img1 = cv2.imread('input11.png')

img2 = cv2.imread('input12.png')

dest\_not1 = cv2.bitwise\_not(img1, mask = None)

dest\_not2 = cv2.bitwise\_not(img2, mask = None)

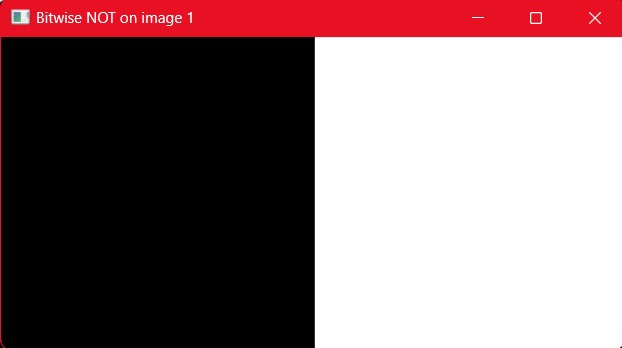
cv2.imshow('Bitwise NOT on image 1', dest\_not1)

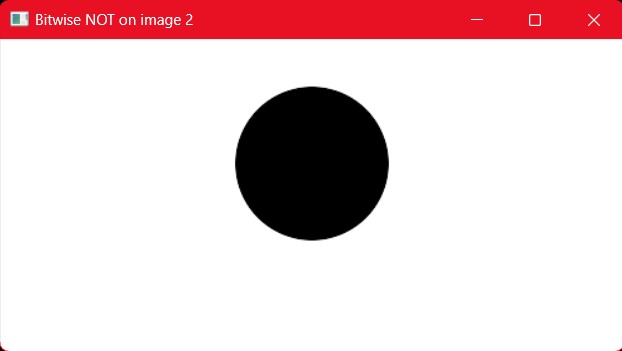
cv2.imshow('Bitwise NOT on image 2', dest\_not2)

if cv2.waitKey(0) & 0xff == 27:

    cv2.destroyAllWindows()

**Result-**

****



**Conclusion-**