**Bitwise operations on image:**

AND operation:

cv2.bitwise\_and(image1,image2) function is used to take the AND operation between two images. It has two arguments which denotes to two images whose AND has to be taken.

The return type of this function is an image after performing the operation.

Truth table of AND operation:

|  |  |  |
| --- | --- | --- |
| A | B | A&B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Python code:**

#import openCV library

import cv2

#import numpy

import numpy as np

img1 = cv2.imread('test.jpg',1);

img2 = cv2.rectangle(np.zeros((183,275,3), np.uint8),(0,0),(275,183),(255,0,55),-1);

#bitwise AND operation of two images

bitAND = cv2.bitwise\_and(img1,img2);

cv2.imshow('image',bitAND);

#waitKey to wait for 10 sec

cv2.waitKey(10000);

#close all windows

cv2.destroyAllWindows();

Output:



OR operation:

cv2.bitwise\_or(image1,image2) function is used to take the OR operation between two images. It has two arguments which denotes to two images whose AND has to be taken.

The return type of this function is an image after performing the operation.

Truth table of AND operation:

|  |  |  |
| --- | --- | --- |
| A | B | A|B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Python code:**

#import openCV library

import cv2

#import numpy

import numpy as np

img1 = cv2.imread('test.jpg',1);

img2 = cv2.rectangle(np.zeros((183,275,3), np.uint8),(0,0),(275,183),(255,0,55),-1);

#bitwise OR operation of two images

bitOR = cv2.bitwise\_or(img1,img2);

cv2.imshow('image',bitOR);

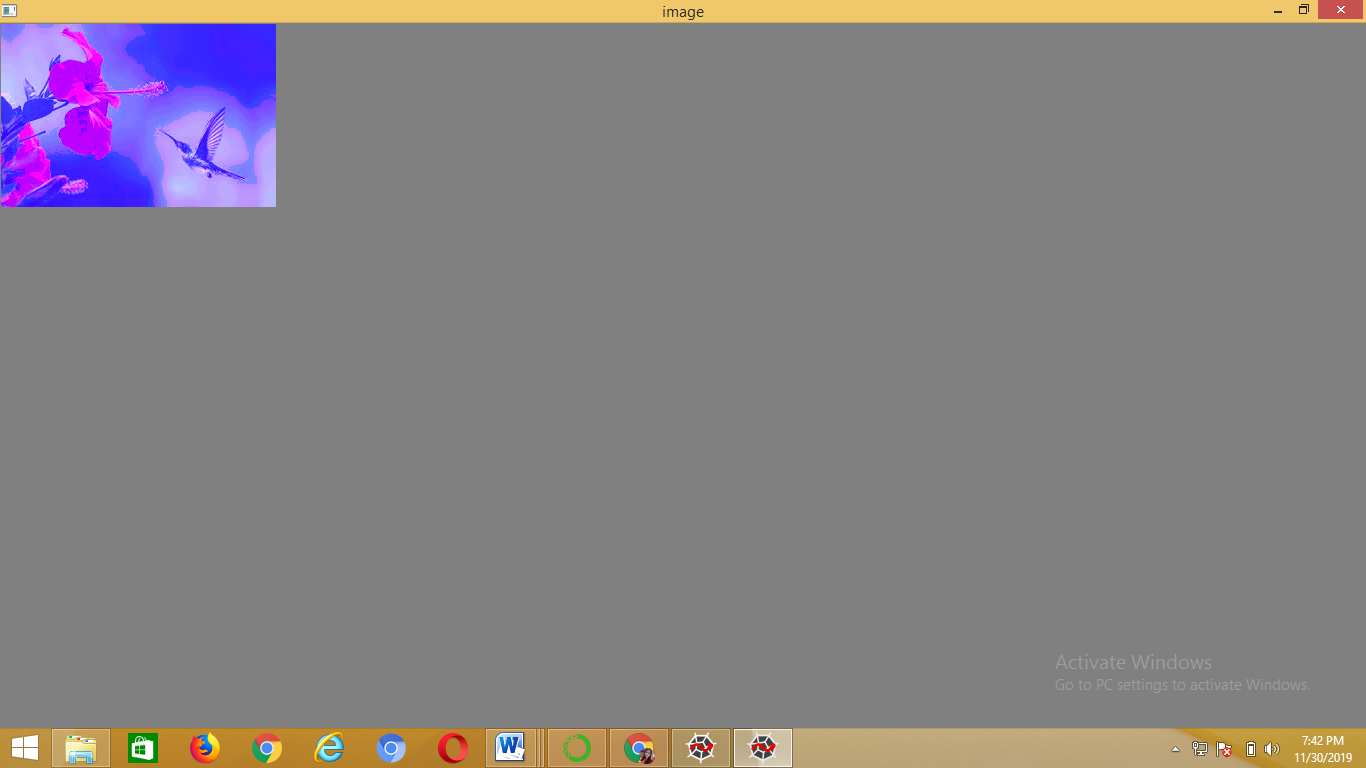
#waitKey to wait for 10 sec

cv2.waitKey(10000);

#close all windows

cv2.destroyAllWindows();

Output:



XOR operation:

cv2.bitwise\_xor(image1,image2) function is used to take the XOR operation between two images. It has two arguments which denotes to two images whose XOR has to be taken.

The return type of this function is an image after performing the operation.

Truth table of AND operation:

|  |  |  |
| --- | --- | --- |
| A | B | A^B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Python code:**

#import openCV library

import cv2

#import numpy

import numpy as np

img1 = cv2.imread('test.jpg',1);

img2 = cv2.rectangle(np.zeros((183,275,3), np.uint8),(0,0),(275,183),(255,0,55),-1);

#bitwise XOR operation of two images

bitXOR = cv2.bitwise\_xor(img1,img2);

cv2.imshow('image',bitXOR);

#waitKey to wait for 10 sec

cv2.waitKey(10000);

#close all windows

cv2.destroyAllWindows();

Output:



NOT operation:

cv2.bitwise\_not(image) function is used to take the NOT operation of an images. It has only one argument which denotes to image whose NOT has to be taken.

The return type of this function is an image after performing the operation.

Truth table of NOT operation:

|  |  |
| --- | --- |
| A | A’ |
| 0 | 1 |
| 1 | 0 |

**Python code:**

#import openCV library

import cv2

#import numpy

import numpy as np

img1 = cv2.imread('test.jpg',1);

#bitwise NOT operation of an image

bitNOT = cv2.bitwise\_not(img1);

cv2.imshow('image',bitNOT);

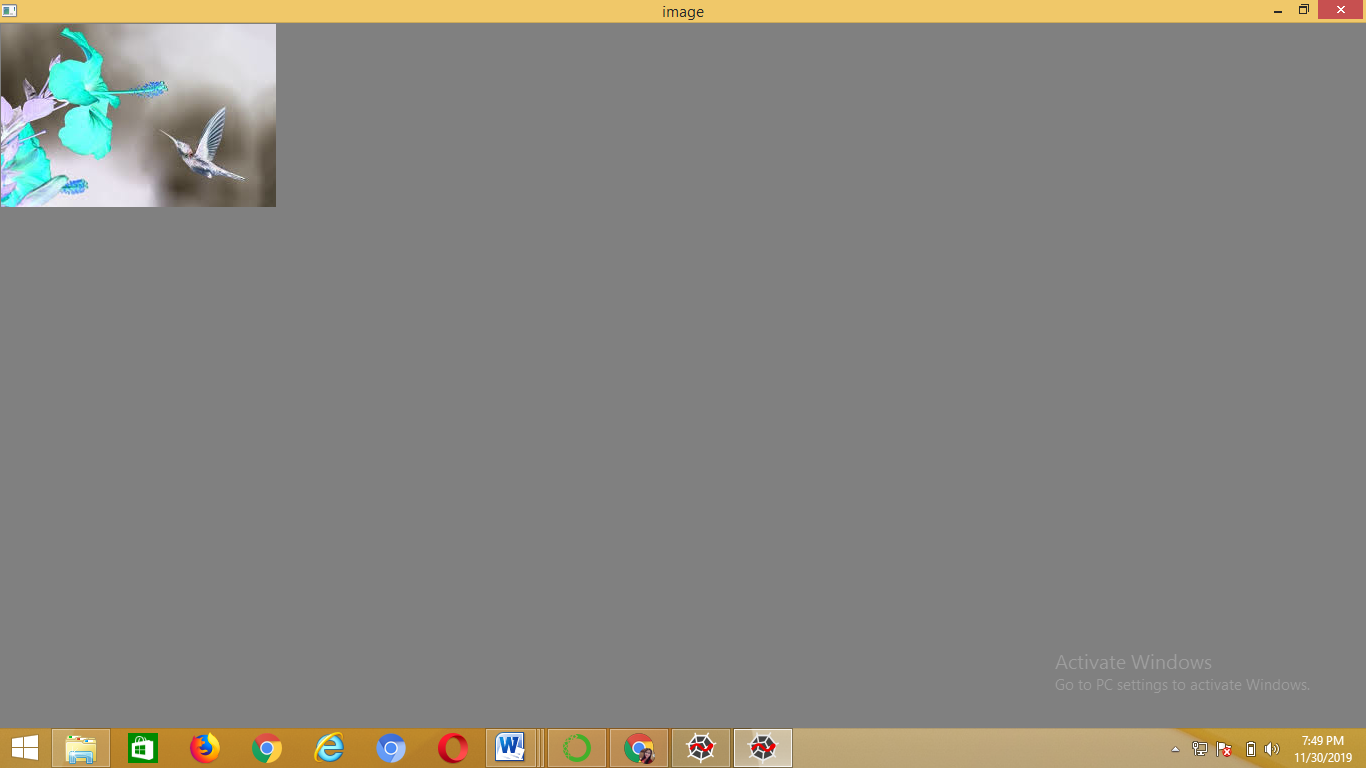
#waitKey to wait for 10 sec

cv2.waitKey(10000);

#close all windows

cv2.destroyAllWindows();

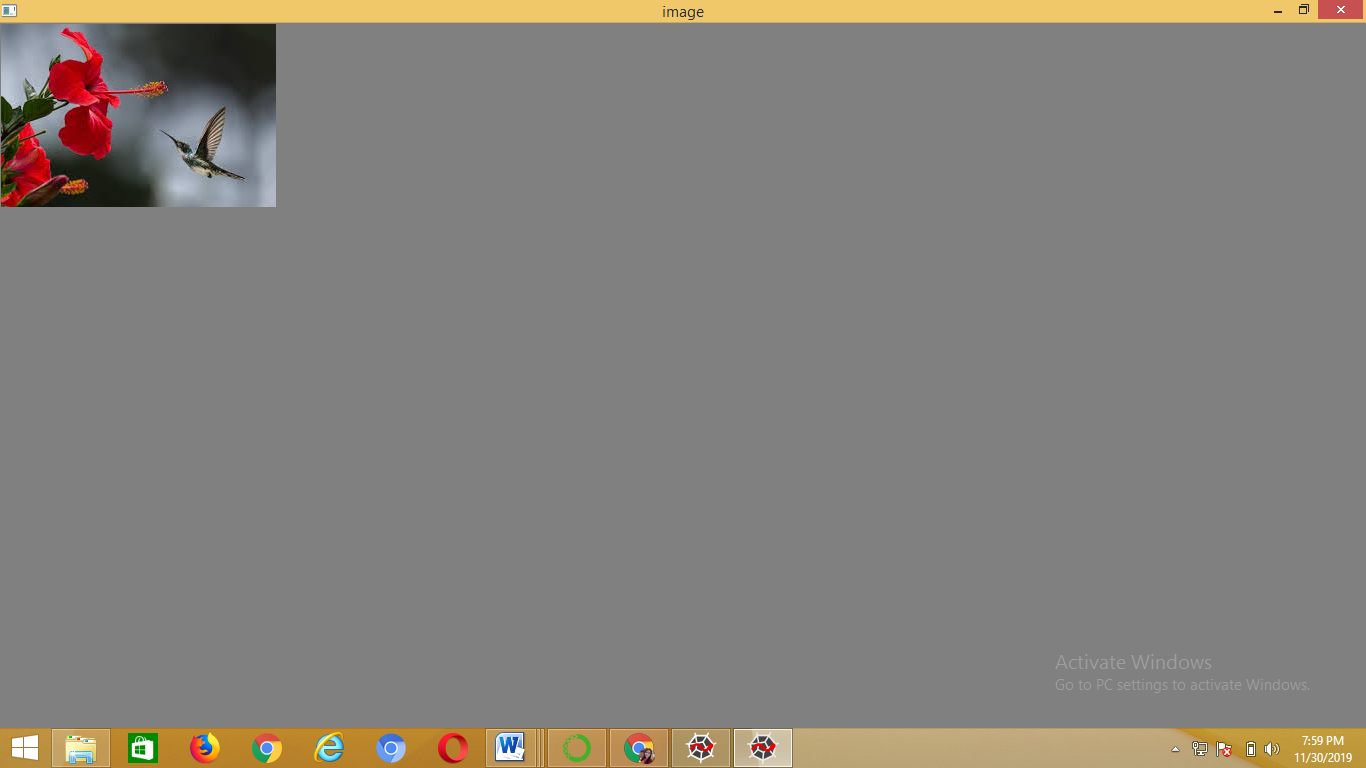
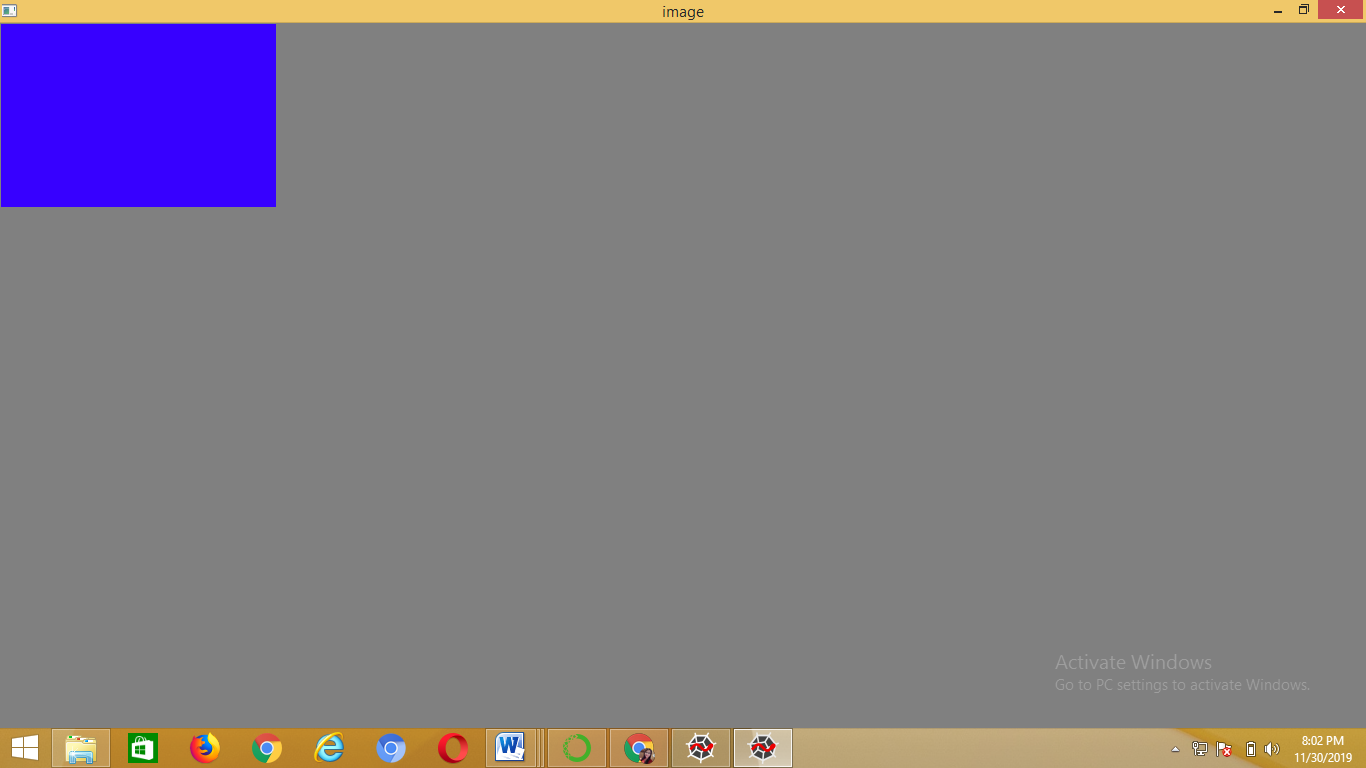
Output:



**Comparison:**

Original images:

Img1: Img2:

|  |  |  |  |
| --- | --- | --- | --- |
| AND | OR | XOR | NOT |
|  |  |  |  |