Lab2

April 24, 2022

1 Lab 2

1.0.1 Submitted By: Manav Doda

1.0.2 Roll No.: 195057

1.1 Importing Necessary Modules

[1]:
```
import numpy as np
import cv2
from PIL import Image
import matplotlib.pyplot as plt
```

1.2 Objective 1

1.2.1 To understand and implement the following task in MATLAB:

A: Image Addition

[2]:
```
img1 = cv2.imread('chessboard.png', cv2.COLOR_BGR2GRAY)
img2 = cv2.imread('chessboardMirror.png', cv2.COLOR_BGR2GRAY)
print('Image 1')
plt.imshow(img1)
plt.show()
print('Image 2')
plt.imshow(img2)
plt.show()
```

```
Image 1
```

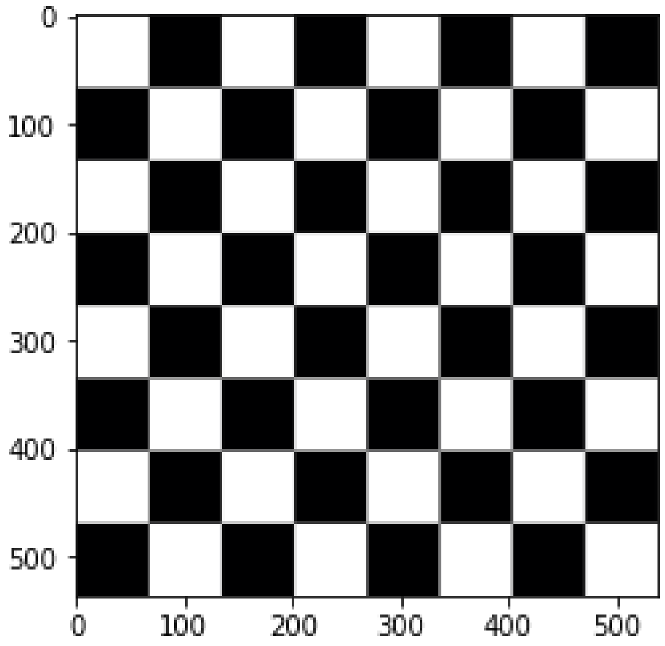
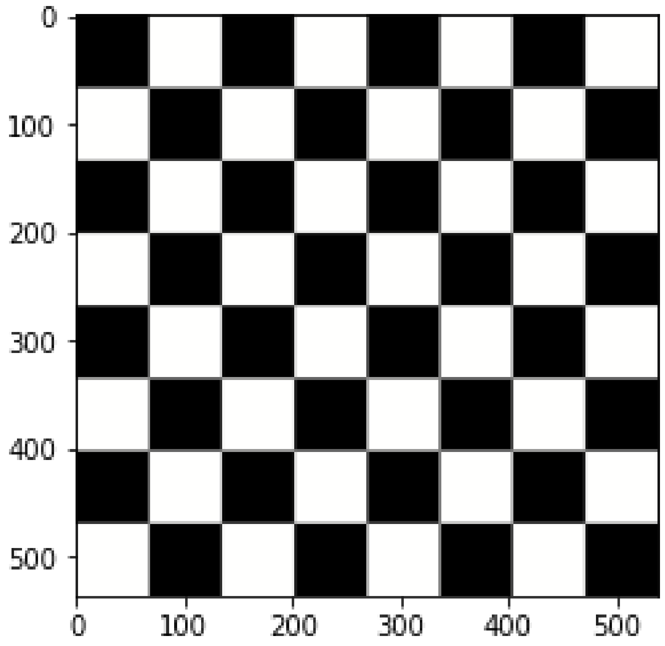
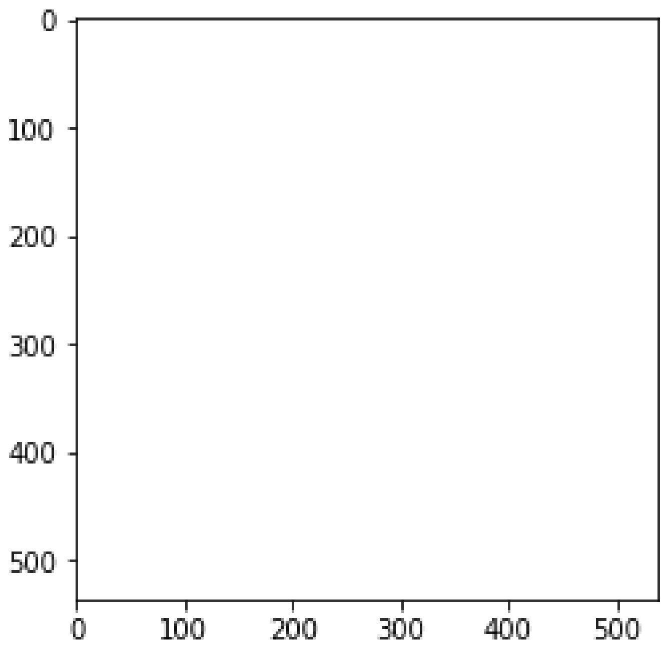

0
100
200
300
400
500
0
100
200
300
400
500

Image 2


0
100
200
300
400
500
0
100
200
300
400
500

```
[4]: resAdd = cv2.imread('chessboard.png', cv2.COLOR_BGR2GRAY)
     for i in range(img1.shape[0]):
         for j in range(img1.shape[1]):
             resAdd[i][j] = img1[i][j]+img2[i][j]
     cv2.imwrite('resAdd.png', resAdd)
     print('Image after conversion')
     plt.imshow(resAdd)
     plt.show()
```

```
Image after conversion
```

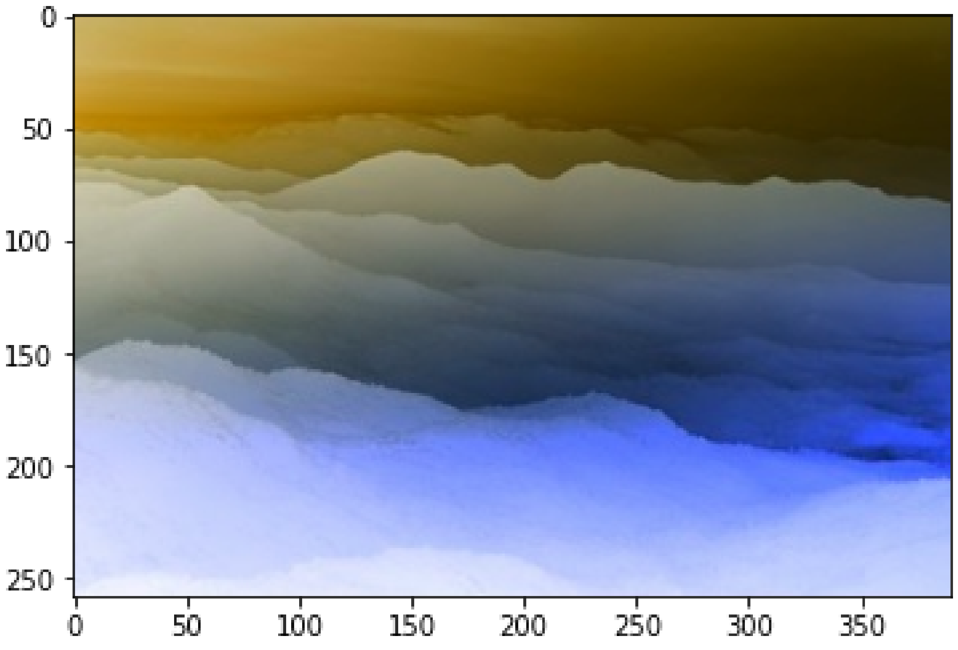


B: Image Negation

```
[5]: img = cv2.imread('testImage.jpeg')
     print('Original Image')
     plt.imshow(img)
     plt.show()
     print('Negative Image')
     negativeImg= 255-img
     plt.imshow(negativeImg)
     plt.show()
     cv2.imwrite('negativeTestImage.jpeg', negativeImg)
```

```
Original Image
```

```
Negative Image
```



```
[5]: True
```

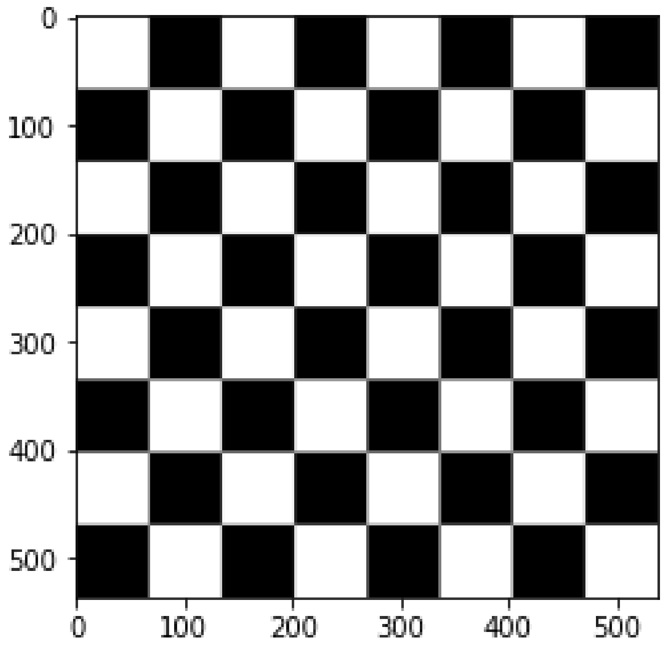
1.3 Objective 2

Logical operations such as

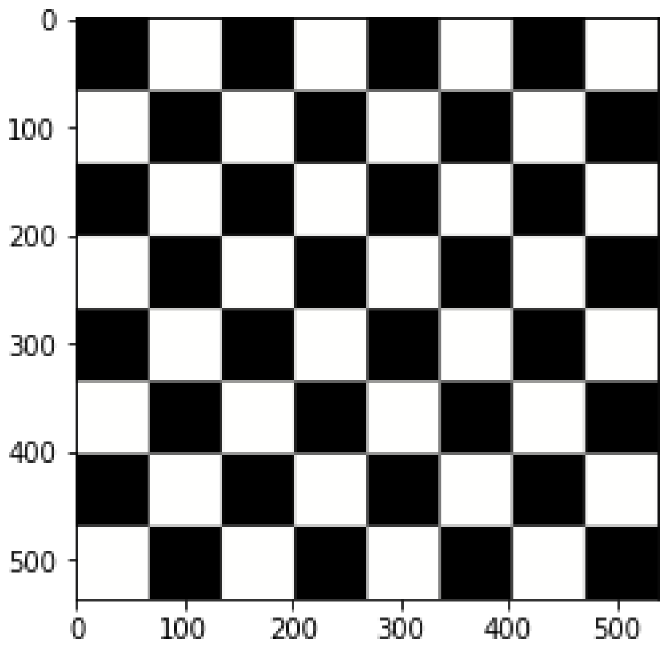
'NOT' operation

```
[7]: img = cv2.imread('chessboard.png')
print('Original Image')
plt.imshow(img)
plt.show()
print('Image after applying NOT Operation')
notImg= 255-img
plt.imshow(notImg)
plt.show()
cv2.imwrite('resultNot.png', notImg)
```

```
Original Image
```



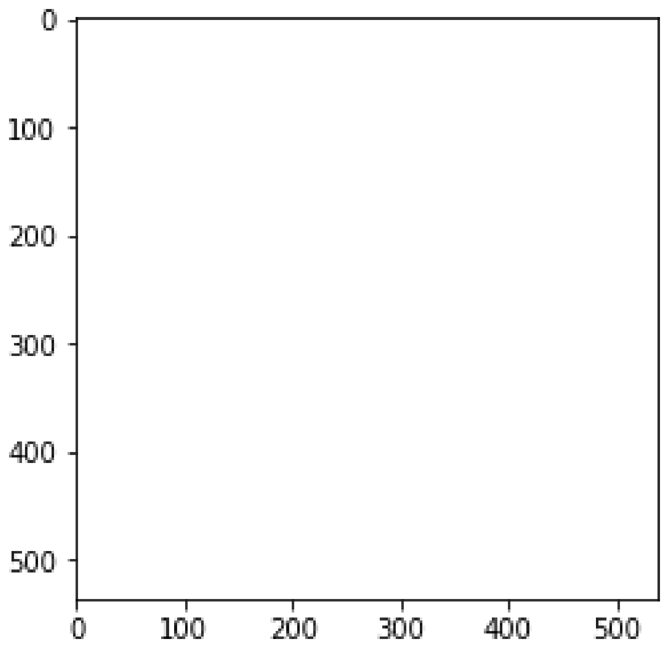
```
Image after applying NOT Operation
```



[7]: True

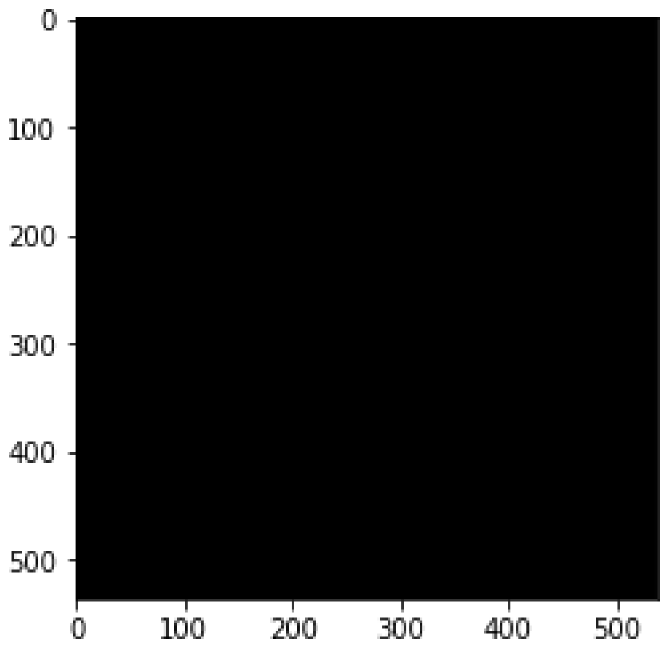
‘OR’ Operation

[8]:
```
img1=cv2.imread('chessboard.png', cv2.COLOR_BGR2GRAY)
img2=cv2.imread('chessboardMirror.png', cv2.COLOR_BGR2GRAY)
for i in range(img1.shape[0]):
    for j in range(img2.shape[0]):
        img1[i][j][0]=img1[i][j][0] | img2[i][j][0]
        img1[i][j][1]=img1[i][j][1] | img2[i][j][1]
        img1[i][j][2]=img1[i][j][2] | img2[i][j][2]
plt.imshow(img1)
plt.show()
```



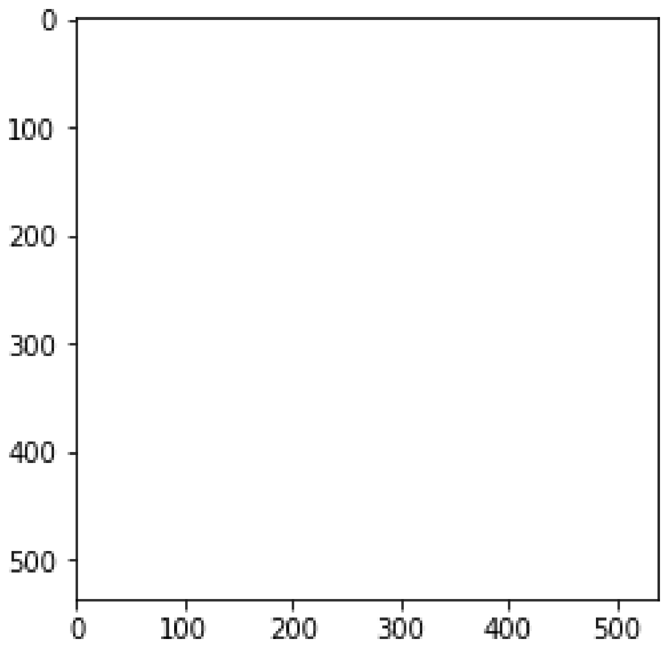
'AND' Operation

```
[9]: img1=cv2.imread('chessboard.png', cv2.COLOR_BGR2GRAY)
     img2=cv2.imread('chessboardMirror.png', cv2.COLOR_BGR2GRAY)
     for i in range(img1.shape[0]):
         for j in range(img2.shape[0]):
             img1[i][j][0]=img1[i][j][0] & img2[i][j][0]
             img1[i][j][1]=img1[i][j][1] & img2[i][j][1]
             img1[i][j][2]=img1[i][j][2] & img2[i][j][2]
     plt.imshow(img1)
     plt.show()
```



'XOR' operation.

[10]:
```
img1=cv2.imread('chessboard.png', cv2.COLOR_BGR2GRAY)
img2=cv2.imread('chessboardMirror.png', cv2.COLOR_BGR2GRAY)
for i in range(img1.shape[0]):
    for j in range(img2.shape[0]):
        img1[i][j][0]=img1[i][j][0] ^ img2[i][j][0]
        img1[i][j][1]=img1[i][j][1] ^ img2[i][j][1]
        img1[i][j][2]=img1[i][j][2] ^ img2[i][j][2]
plt.imshow(img1)
plt.show()
```


0
100
200
300
400
500
0
100
200
300
400
500

```
[ ]:
```