Banaras Hindu University
Institute of Science
Department of Computer Science

Assignment Number: 01

Subject: “Image Processing”

Submitted To:
Dr.Ankita Vaish
Department of Computer Science

Submitted By:
Sagar Timalsena

(24419CMP025)

Master of Science (Computer Science)

Academic Year:

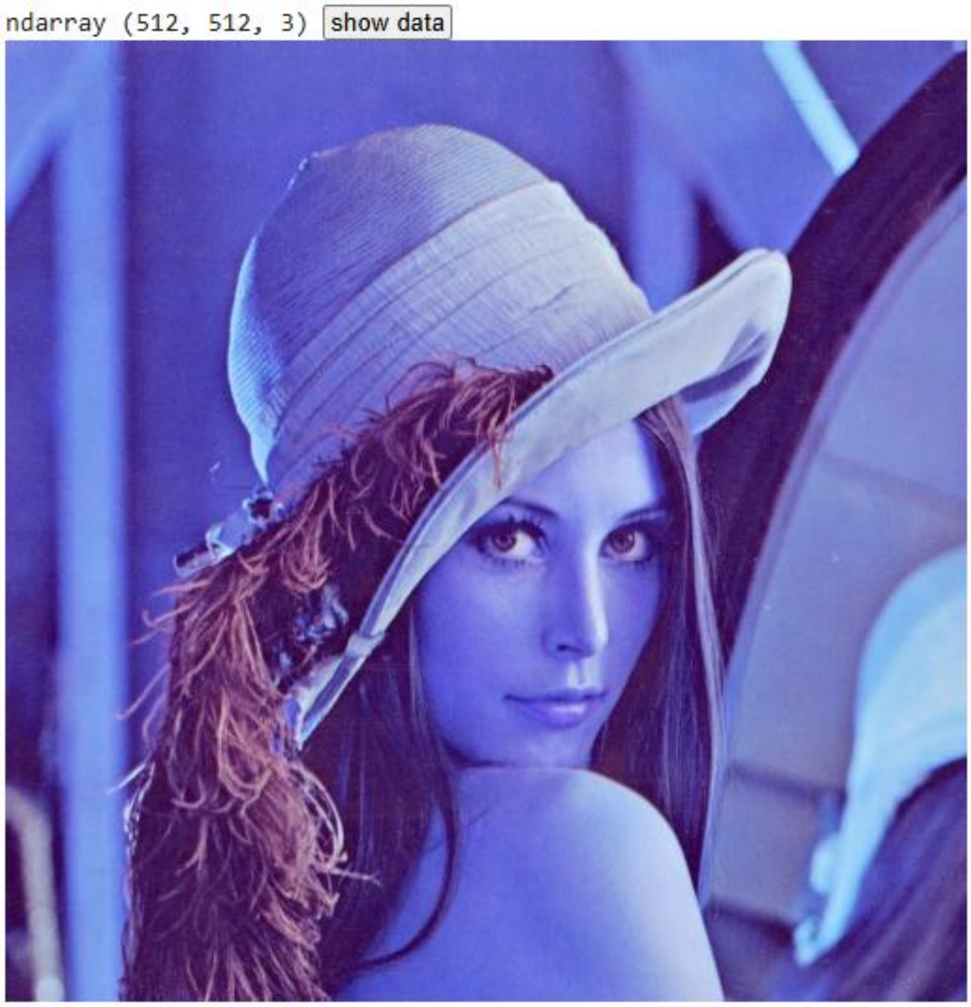
2024-2025

1. Load a color image.

Code:

```
        image = cv2.imread('Lena.png')
        image
```

Output:



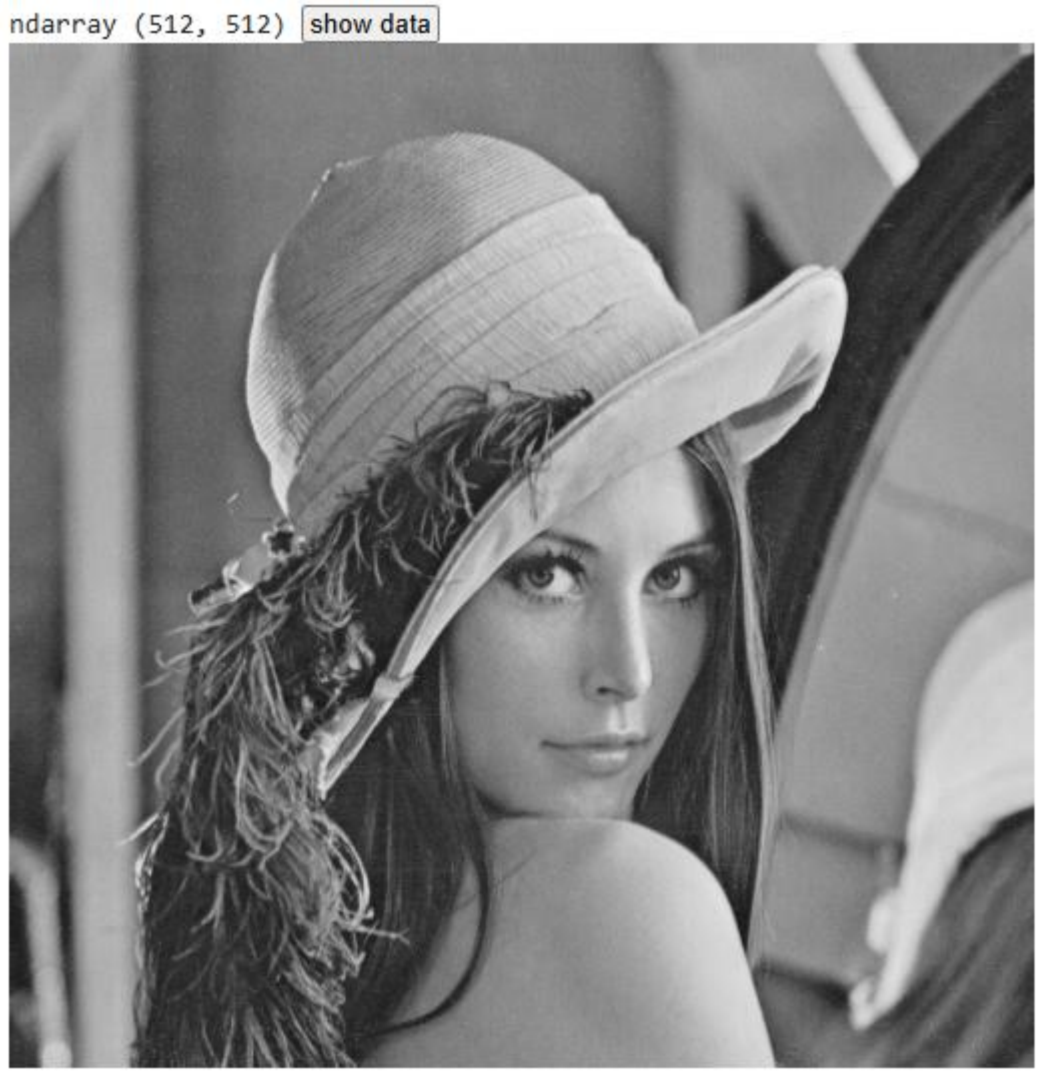
```
ndarray (512, 512, 3)
```

- Convert it into grayscale.

Code:

```
gray_image = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
gray_image
```

Output:



```
ndarray (512, 512)
```

- Resize it into different dimensions (e.g. 128 *128 and 64*64).

Code

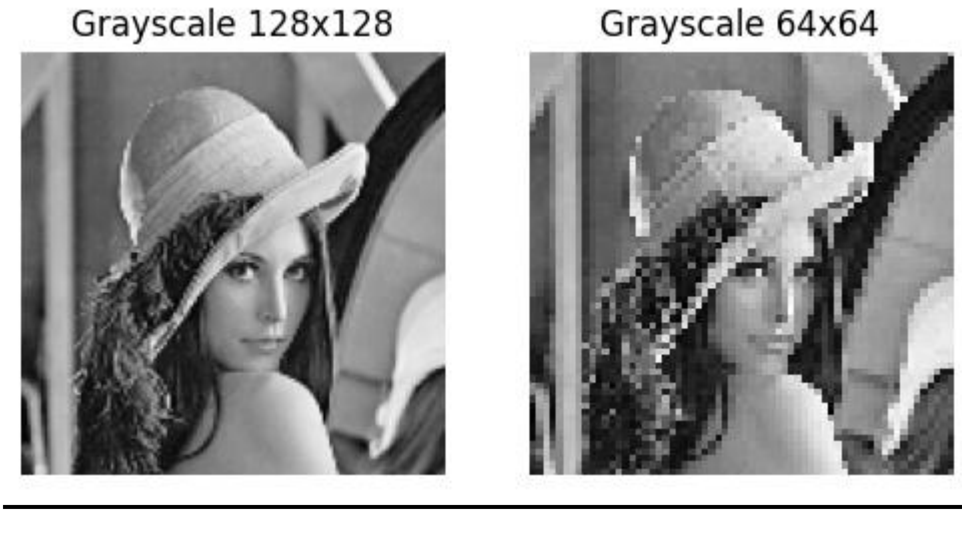
```
resized_128 = cv2.resize(gray_image, (128, 128))
resized_64 = cv2.resize(gray_image, (64, 64))
```

- Display it into subplots

Code

```
plt.figure(figsize=(6, 4))
plt.subplot(1, 2, 1)
plt.imshow(resized_128, cmap='gray')
plt.title("Grayscale 128x128")
plt.axis("off")
plt.subplot(1, 2, 2)
plt.imshow(resized_64, cmap='gray')
plt.title("Grayscale 64x64")
plt.axis("off")
plt.show()
```

Output



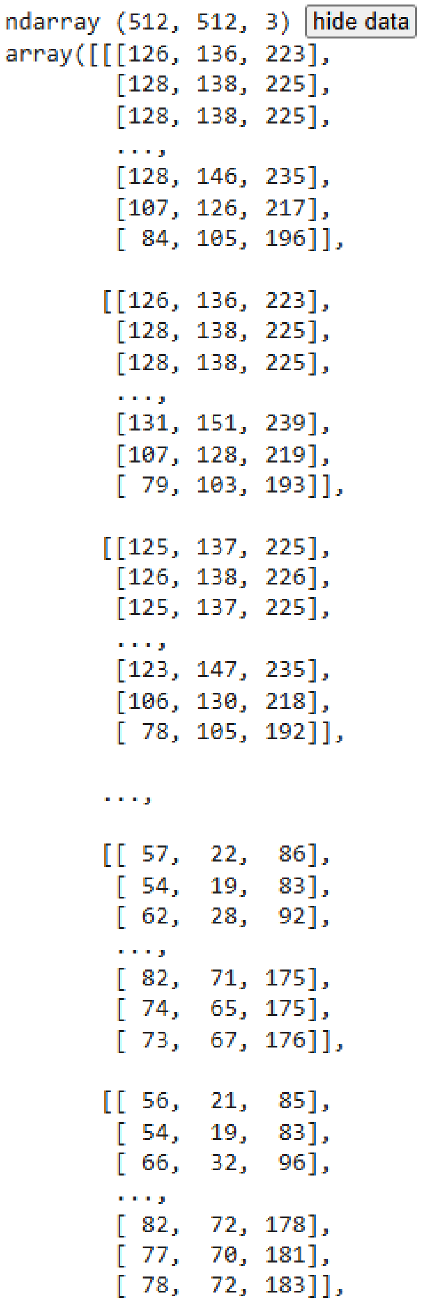
2.Create an image puzzle.

i. Load the image you want to use and convert it to a NumPy array

Code:

```
image = cv2.imread('Lena.png')
image
```

Output:



```
ndarray (512, 512, 3)
array([[[126, 136, 223],
        [128, 138, 225],
        [128, 138, 225],
        ...,
        [128, 146, 235],
        [107, 126, 217],
        [ 84, 105, 196]],

       [[126, 136, 223],
        [128, 138, 225],
        [128, 138, 225],
        ...,
        [131, 151, 239],
        [107, 128, 219],
        [ 79, 103, 193]],

       [[125, 137, 225],
        [126, 138, 226],
        [125, 137, 225],
        ...,
        [123, 147, 235],
        [106, 130, 218],
        [ 78, 105, 192]],

       ...,

       [[ 57,  22,  86],
        [ 54,  19,  83],
        [ 62,  28,  92],
        ...,
        [ 82,  71, 175],
        [ 74,  65, 175],
        [ 73,  67, 176]],

       [[ 56,  21,  85],
        [ 54,  19,  83],
        [ 66,  32,  96],
        ...,
        [ 82,  72, 178],
        [ 77,  70, 181],
        [ 78,  72, 183]],
```

ii. Divide the image into blocks of equal size. The size of the blocks will depend on how big you want your puzzle pieces to be.

Code:

```
block_size = 64  # Define block size
blocks = []
indices = []
for i in range(0, resized_image.shape[0], block_size):
   for j in range(0, resized_image.shape[1], block_size):
       blocks.append(resized_image[i:i+block_size,
j:j+block_size])
       indices.append((i, j))
```

iii. Shuffle the order of the blocks to create a puzzle. You can do this by randomly permuting the indices of the blocks.

Code:

```
shuffled_indices = np.random.permutation(len(blocks))
shuffled_blocks = [blocks[i] for i in shuffled_indices]
```

iv. Display the shuffled blocks as a puzzle by stitching them back together in their shuffled order.

Code:

```
shuffled_indices = np.random.permutation(len(blocks))
shuffled_blocks = [blocks[i] for i in shuffled_indices]
```

v. To reconstruct the original image, unshuffled the blocks by applying the inverse permutation to the shuffled blocks and stitching them back together in their original order.

Code:

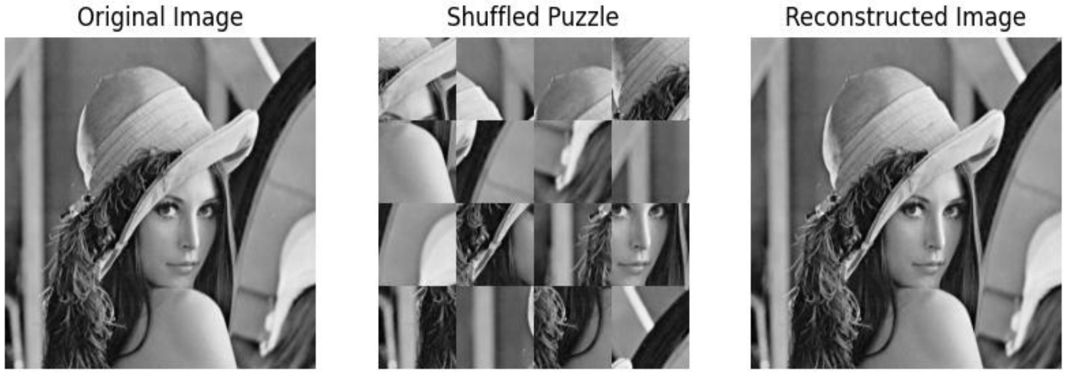
```
original_image = np.zeros_like(resized_image)
inverse_permutation = np.argsort(shuffled_indices)
for idx, (i, j) in zip(inverse_permutation, indices):
    original_image[i:i+block_size, j:j+block_size] = shuffled_blocks[idx]
```

vi. Store the permuted and reconstructed image.

Code:

```
cv2.imwrite('shuffled_puzzle.jpg', shuffled_image)
cv2.imwrite('reconstructed_image.jpg', original_image)
```

Final Output



Conclusion

This lab demonstrated image processing techniques using OpenCV. The process included grayscale conversion, resizing, and an image puzzle implementation using block manipulation. The experiment highlights the practical applications of OpenCV in image transformation and reconstruction.