**Banaras Hindu University**  
**Institute of Science**  
**Department of Computer Science**



**Subject: “Image Processing”**

**Submitted To:**  
**Dr.Ankita Vaish**  
Department of Computer Science

**Submitted By:**  
Sagar Timalsena

(24419CMP025)

**Academic Year:**

2024-2025

**Banaras Hindu University**  
**Institute of Science**  
**Department of Computer Science**



**Subject: “Image Processing”**

**Submitted To:**  
**Dr.Ankita Vaish**  
Department of Computer Science

**Submitted By:**  
Vikas Yadav

(24419MSC020)

**Academic Year:**

2024-2025

1. Load a color image.

Code:

|  |
| --- |
| *image = cv2.imread('Lena.png')*  *image* |

Output:



* Convert it into grayscale.

**Code:**

|  |
| --- |
| *gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)*  *gray\_image* |

**Output:**



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

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.

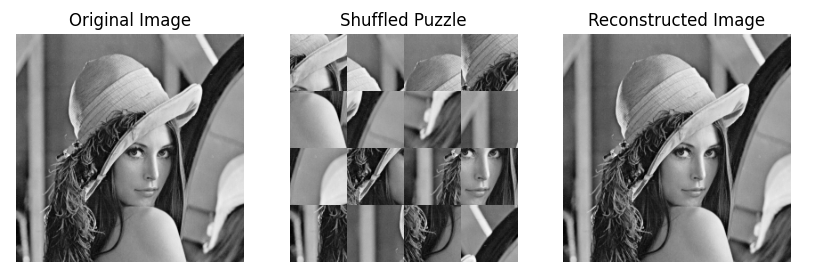
|  |
| --- |
| *original\_image = np.zeros\_like(resized\_image)*  *inverse\_permutation = np.argsort(shuffled\_indices)*  *for idx, (i, j) in zip(inverse\_permutation, indices):*  *original\_image[i:i+block\_size, j:j+block\_size] = shuffled\_blocks[idx]* |

vi. Store the permuted and reconstructed image.

**Code:**

|  |
| --- |
| *cv2.imwrite('shuffled\_puzzle.jpg', shuffled\_image)*  *cv2.imwrite('reconstructed\_image.jpg', original\_image)* |

**Final Output**

****