**CV Practical No.: 6**

**Aim: Morphological Operations**

**New Concept:**

**i. np.ones:** it is a function from the NumPy library that creates an array of a specified shape, filled with ones. This is often used to create masks, kernels (for convolution), or structures that need a uniform value (in this case, 1) across all their elements. For example, in operations like dilation or erosion (in morphological operations), a kernel might be created using np.ones to define the neighborhood of pixels to be considered.

**ii. cv2.erode:** The cv2.erode function is part of OpenCV and performs a morphological operation called erosion. Erosion works by shrinking the white regions of an image, or in simpler terms, it erodes the boundaries of the foreground object in a binary image.

**iii. cv2.dilate:** The cv2.dilate function is another morphological operation in OpenCV, but instead of shrinking, it dilates or expands the white regions of an image. It increases the size of the foreground object. Dilation is useful for closing small gaps in an object, emphasizing the foreground, or connecting nearby objects. It's often applied after erosion to restore the original shape after removing small noise.

**iv. skeletonize:** Skeletonization is a process that reduces objects in a binary image to their skeletal form. It involves iteratively removing layers from the boundaries of the foreground object until it becomes a line (skeleton) that represents the structure of the object. Skeletonization is important in shape analysis, pattern recognition, and feature extraction. It helps in identifying key points and structures in the image, which can then be used for further analysis such as object recognition or classification.

**v. skimage.morphology:** it is a module from the scikit-image library that contains functions for performing morphological operations on images. These operations include functions like erosion, dilation, opening, closing, skeletonization, etc. This module is highly useful for binary image processing, such as removing noise, enhancing features, or performing shape analysis. Operations like erosion and dilation in skimage.morphology are typically used in tasks such as segmentation, noise reduction, and boundary extraction.

**Theory:**

Morphological operations are a set of image processing techniques used to process the structure or shape of objects within an image. These operations typically work on binary (black and white) images, although they can be adapted for grayscale images as well. They are based on the principles of set theory and are used to extract, enhance, or modify the shapes of objects in an image.

The main idea behind morphological operations is to probe the image with a small shape or structure called a structuring element (SE). The structuring element is typically a small matrix (e.g., a 3x3 square or disk) that is moved over the image, and the result is a transformation of the image based on the interaction between the image pixels and the structuring element.

**Morphological Operations:**

1. **Dilation**:
   * **Dilation** expands or grows the white regions (foreground) of a binary image. When the structuring element is applied to the image, any pixel in the image that overlaps with the structuring element will be set to the foreground value (typically white).
   * **Effect**: It makes the white objects in the image thicker or larger.
   * **Example**: If you have an image of a thin line, after dilation, the line will appear thicker.
2. **Erosion**:
   * **Erosion** is the opposite of dilation. It shrinks or erodes the white regions of a binary image. When the structuring element is applied, only those pixels that are entirely contained within the structuring element will remain in the foreground (usually white).
   * **Effect**: It makes the white objects in the image thinner or smaller.
   * **Example**: If you have a thick line in the image, erosion will make it thinner or even disappear if it’s too thin.
3. **Opening**:
   * **Opening** is a combination of erosion followed by dilation. The process starts by eroding the image, which removes small noise or objects from the foreground, and then dilating it to restore the size of the remaining structures.
   * **Effect**: It removes small objects or noise while preserving the overall structure of larger objects.
   * **Example**: If you have small noise (like tiny white spots) in a black background, opening can help remove those.
4. **Closing**:
   * **Closing** is the reverse of opening. It involves dilation followed by erosion. The dilation first fills small holes or gaps in the foreground, and then the erosion step removes any unwanted pixels that may have been added during dilation.
   * **Effect**: It closes small holes and gaps in the foreground objects while preserving the overall shape.
   * **Example**: If you have objects with small holes or gaps, closing will help to fill them in.

**Program:**

import cv2

import numpy as np

from skimage.morphology import skeletonize

import matplotlib.pyplot as plt

# Read the image

image = cv2.imread("doraemon.jpg")

rgb\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

gray\_image = cv2.imread("doraemon.jpg", cv2.IMREAD\_GRAYSCALE)

\_,binary\_image = cv2.threshold(rgb\_image, 127, 255, cv2.THRESH\_BINARY)

# Create a kernel

kernel = np.ones((5,5), np.uint8)

# Apply morphological operations

erosion = cv2.erode(rgb\_image, kernel, iterations=1)

dilation = cv2.dilate(rgb\_image, kernel, iterations=1)

opening = cv2.dilate(erosion, kernel, iterations=1)

closing = cv2.erode(dilation, kernel, iterations=1)

binary = binary\_image // 255 # Normalize to 0 and 1

skeleton = skeletonize(binary) # Apply thinning

# Convert back to uint8 (0 and 255)

skeleton = (skeleton \* 255).astype(np.uint8)

#Image1

plt.figure(figsize = (10,5))

plt.subplot(2,4,1)

plt.imshow(rgb\_image)

#plt.imshow(image, cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image2

plt.subplot(2,4,2)

#plt.imshow(gray\_image)

plt.imshow(gray\_image, cmap = 'gray')

plt.title('GrayScale Image')

plt.axis('off')

#Image3

plt.subplot(2,4,3)

plt.imshow(binary\_image)

#plt.imshow(binary\_image, cmap = 'gray')

plt.title('Binary Image')

plt.axis('off')

#Image4

plt.subplot(2,4,4)

plt.imshow(erosion)

#plt.imshow(erosion, cmap = 'gray')

plt.title('Eroded Image')

plt.axis('off')

#Image5

plt.subplot(2,4,5)

plt.imshow(dilation)

#plt.imshow(dilation, cmap = 'gray')

plt.title('Dilated Image')

plt.axis('off')

#Image6

plt.subplot(2,4,6)

plt.imshow(opening)

#plt.imshow(opening, cmap = 'gray')

plt.title('Opening Image')

plt.axis('off')

#Image7

plt.subplot(2,4,7)

plt.imshow(closing)

#plt.imshow(closing, cmap = 'gray')

plt.title('Closing Image')

plt.axis('off')

#Image8

plt.subplot(2,4,8)

plt.imshow(skeleton)

#plt.imshow(skeleton, cmap = 'gray')

plt.title('SKeleton Image')

plt.axis('off')

plt.show()

**Output:**

