Morphological Image Processing

Morphological Image Processing involves manipulating the structure or shape of objects in an image using mathematical morphology. It operates on binary or grayscale images, typically using a structuring element (kernel) to probe and modify the image's geometry. Morphological operations are particularly effective for processing binary images, where objects are represented as white (foreground) and the background as black.

Key Operations:

1. Erosion: Shrinks or erodes the boundaries of foreground objects (white regions) by removing pixels on object edges.
2. Effect: Small objects or noise may disappear, and gaps within objects widen. Mathematical Operation: For a pixel to remain in the output, the structuring element must fully fit within the foreground at that position. Dilation: Expands or dilates the boundaries of foreground objects by adding pixels to object edges.
3. Effect: Fills small holes, connects disjointed parts, and thickens objects. Mathematical Operation: A pixel is set in the output if any part of the structuring element overlaps the foreground.
4. Opening: Erosion followed by dilation. Effect: Removes small objects/noise while preserving the shape and size of larger objects. Useful for noise reduction and separating touching objects.
5. Closing: Dilation followed by erosion. Effect: Fills small holes and gaps within objects while preserving their overall shape. Useful for connecting broken parts and smoothing object boundaries

Applications:

1. Noise Removal: Opening removes small noise speckles; closing fills small holes.
2. Object Separation: Opening can separate touching objects in binary images.
3. Shape Analysis: Used in medical imaging (e.g., cell counting), industrial inspection, and character recognition.
4. Edge Enhancement: Morphological gradients (difference between dilation and erosion) highlight edges.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

```
img = cv2.imread('/content/drive/MyDrive/Colab Notebooks/imageprocessingimages/5.jpg', 0)

# Ensure binary image (threshold if needed)
_, img = cv2.threshold(img, 128, 255, cv2.THRESH_BINARY)

# Define 5x5 structuring element
kernel = np.ones((5, 5), np.uint8)

# Apply morphological operations
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)
img_opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
img_closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
```

```
# Save output images
cv2.imwrite('Erosion.png', img_erosion)
cv2.imwrite('Dilation.png', img_dilation)
cv2.imwrite('Opening.png', img_opening)
cv2.imwrite('Closing.png', img_closing)
```

```
True
```

```
# Display images using Matplotlib
plt.figure(figsize=(15, 10))
plt.subplot(2, 3, 1)
plt.imshow(img, cmap='gray')
plt.title('Input')
plt.axis('off')

plt.subplot(2, 3, 2)
plt.imshow(img_erosion, cmap='gray')
plt.title('Erosion')
plt.axis('off')

plt.subplot(2, 3, 3)
plt.imshow(img_dilation, cmap='gray')
plt.title('Dilation')
plt.axis('off')

plt.subplot(2, 3, 4)
plt.imshow(img_opening, cmap='gray')
plt.title('Opening')
plt.axis('off')

plt.subplot(2, 3, 5)
plt.imshow(img_closing, cmap='gray')
plt.title('Closing')
plt.axis('off')
```

```
plt.show()
```