Lab 4 Manual: Advanced Morphological Operations

📌 **Objective**: Understand and implement advanced morphological operations:

- Opening and Closing
- Morphological Gradient
- Top-Hat Transformation
- Black-Hat Transformation

📝 **Learning Outcomes**

By the end of this lab, students will:
✓ Understand the role of advanced morphological operations in image processing.
✓ Implement morphological transformations in OpenCV.
✓ Analyze the effects of these operations on different images.

🔹 **Step 1: Import Required Libraries**

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

Explanation:

- cv2 → OpenCV for image processing.
- numpy → Used for creating kernel matrices.
- matplotlib.pyplot → Used for visualizing images.

🔹 **Step 2: Load Image and Convert to Grayscale**

```
# Load image
image_path = "/path/to/your/image.jpg"  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


# Display the image
plt.imshow(image, cmap='gray')
plt.title("Original Grayscale Image")
plt.axis('off')
plt.show()
```

Explanation:

1. Reads the image in **grayscale** (morphological operations work on intensity values).

2. Displays the grayscale image.

✅ **Task for Students**:

- Change the image path and observe the results on different images.

🔹 **Step 3: Apply Morphological Operations**

Morphological operations are techniques that process images based on their shapes. They are used for tasks such as noise removal, image enhancement, and feature extraction. The two fundamental operations covered in this lab are:

1. **Opening** (Erosion followed by Dilation) - Removes small white noise (salt noise).
2. **Closing** (Dilation followed by Erosion) - Removes small black noise (pepper noise).

Both operations use a **structuring element (kernel)** to define the transformation's shape and size.

```
# Load the image
image_path = "path_to_your_image.jpg"  # Update this with your image path
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

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

# Apply Opening (Erosion followed by Dilation)
opening = cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)

# Apply Closing (Dilation followed by Erosion)
closing = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kernel)

# Display images
fig, ax = plt.subplots(1, 3, figsize=(15, 5))
ax[0].imshow(image, cmap='gray')
ax[0].set_title("Original Image")
ax[0].axis('off')

ax[1].imshow(opening, cmap='gray')
ax[1].set_title("After Opening (Erosion → Dilation)")
ax[1].axis('off')
```

```
ax[2].imshow(closing, cmap='gray')
ax[2].set_title("After Closing (Dilation → Erosion)")
ax[2].axis('off')

plt.show()
```

Student Tasks:

1. Try changing the kernel size to (3,3) or (7,7) and observe the effect on noise removal.
2. Experiment with different grayscale images containing noise and analyze how opening and closing affect them.
3. Use a different kernel shape (e.g., elliptical kernel using cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5))).

Python Code for Opening & Closing

🔹 **Step 4: Apply Morphological Gradient**

```
# Define a 5x5 kernel
kernel = np.ones((5,5), np.uint8)

# Apply morphological gradient
gradient = cv2.morphologyEx(image, cv2.MORPH_GRADIENT, kernel)

# Display images
plt.figure(figsize=(10,5))
plt.subplot(1,2,1), plt.imshow(image, cmap='gray'), plt.title("Original Image")
plt.subplot(1,2,2), plt.imshow(gradient, cmap='gray'), plt.title("Morphological Gradient")
plt.show()
```

Explanation:

- **Kernel (5x5)**: Defines the neighborhood for morphological operations.
- **Morphological Gradient** = **Dilation - Erosion** → Highlights object edges.
- **Use Case**: Helps detect object boundaries.

✅ **Task for Students**:

- Change the **kernel size** (e.g., (3,3), (7,7)) and observe how the results differ.

🔹 **Step 5: Apply Top-Hat Transformation**

```
# Apply top-hat transformation
top_hat = cv2.morphologyEx(image, cv2.MORPH_TOPHAT, kernel)

# Display results
plt.figure(figsize=(10,5))
plt.subplot(1,2,1), plt.imshow(image, cmap='gray'), plt.title("Original Image")
plt.subplot(1,2,2), plt.imshow(top_hat, cmap='gray'), plt.title("Top-Hat Transformation")
plt.show()
```

Explanation:

- **Top-Hat Transformation**: Extracts **small bright details** from the image.
- **How it works?**
    - First, the image undergoes **opening** (erosion → dilation).
    - The result is **subtracted from the original image**, leaving bright details.

✅ **Task for Students**:

- Apply **Top-Hat** on an image with **text** or **medical images** and analyze the result.

🔹 **Step 6: Apply Black-Hat Transformation**

```
# Apply black-hat transformation
black_hat = cv2.morphologyEx(image, cv2.MORPH_BLACKHAT, kernel)

# Display results
plt.figure(figsize=(10,5))
plt.subplot(1,2,1), plt.imshow(image, cmap='gray'), plt.title("Original Image")
plt.subplot(1,2,2), plt.imshow(black_hat, cmap='gray'), plt.title("Black-Hat Transformation")
plt.show()
```

Explanation:

- **Black-Hat Transformation**: Extracts **dark features** from the image.
- **How it works?**
    - First, the image undergoes **closing** (dilation → erosion).
    - The result is **subtracted from the original image**, leaving dark features.

✅ **Task for Students**:

- Try different **kernel sizes** and observe the effect.
- Test on an image with **shadows** or **handwritten text**.

🔎 **Conclusion**

✓ **Morphological Gradient** → Highlights object edges.
✓ **Top-Hat** → Extracts small bright features from the background.
✓ **Black-Hat** → Extracts small dark features from the background.

✅ **Final Task for Students**:

- Write a brief report explaining the differences between **Top-Hat, Black-Hat, and Gradient** transformations using different images.