আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম

الجامعة الإسلامية العالمية شيتاغونغ

International Islamic University Chittagong

Lab 11

Lab Report

Department: Computer Science and Engineering.

Course Code: CSE-4876

Course Title: Pattern Recognition & Image Processing

Submitted By:

Name: Farida Nusrat

ID: C201242

Semester: 8th

Section: 8AF

Submitted To:

Mr. Mohammad Mahadi Hasaan

Associate Professor,

Department of CSE , IIUC.

Date of Submission: 23 June, 2024

Lab No: 11

Lab Title: Morphological Image Processing

- Dilation
- Erosion
- Opening
- Closing
- Boundary Detection
- Region Filling
- Thinning
- Thickening
- Convex Hull
- Skeleton

Introduction: Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. These techniques are particularly effective for analyzing geometrical structures in binary and grayscale images. This field of image processing leverages mathematical morphology, which provides a quantitative description of the geometry of structures within an image. The core operations in morphological image processing include dilation, erosion, opening, and closing, each of which manipulates the shape and structure of objects within an image in distinct ways.

Dilation is a fundamental morphological operation that enlarges the boundaries of objects in an image. By adding pixels to the object boundaries, dilation effectively increases the size of the objects, making them more prominent. This operation is often used to fill in small holes and gaps in objects.

Erosion, in contrast, shrinks the boundaries of objects. By removing pixels from the edges, erosion reduces the size of objects, which can help in eliminating small noise and detaching objects that are close to each other.

Opening is a combination of erosion followed by dilation. This operation is useful for removing small objects and noise from an image while preserving the shape and size of larger objects. It is particularly effective for separating objects that are close to each other.

Closing, the inverse of opening, involves dilation followed by erosion. It is used to close small holes and gaps within objects, thereby smoothing their boundaries and connecting nearby objects.

Beyond these basic operations, morphological image processing encompasses several advanced techniques. **Boundary detection** identifies the contours of objects, which is crucial for shape analysis and object recognition. **Region filling** fills in the interior of detected objects, which is useful for analyzing the entirety of object structures.

Thinning and **thickening** modify the thickness of objects' boundaries to create skeletonized or enhanced versions of the structures within an image. Thinning reduces objects to their essential structure, often resembling a skeleton, while thickening broadens the features of interest.

Convex hull is another important morphological technique that constructs the smallest convex boundary that can enclose an object, which aids in shape analysis and object recognition.

Finally, **skeletonization** reduces objects to their simple, central lines or skeletons, which is useful for analyzing the topological properties of objects.

These morphological operations are indispensable tools in image processing and computer vision, enabling the effective analysis and manipulation of image structures for a wide range of applications, from medical imaging to object recognition and beyond. This report delves into the methodologies and applications of these techniques, providing a comprehensive understanding of their utility in the field of morphological image processing.

Methodology:

1. **Image Loading and Preprocessing**

- The original grayscale image was loaded using OpenCV.

- The image was then thresholded to create a binary image. This conversion was done using a threshold value of 50, where pixel values greater than 50 were set to 255 (white), and pixel values less than or equal to 50 were set to 0 (black).

```
img = cv2.imread("/content/input.png", 0)
_, thresh = cv2.threshold(img, 50, 255, cv2.THRESH_BINARY)
```

2. **Structuring Element**

- A structuring element (kernel) of size 3x3 was created. This kernel was used for the morphological operations.

```
kernel = np.ones((3, 3), np.uint8)
```

3. **Morphological Operations**

- **Dilation:** Applied to the binary image to enlarge the boundaries of the white regions.
  `dilation = cv2.dilate(thresh, kernel, iterations=1)`
- **Erosion:** Applied to shrink the boundaries of the white region.
  `erosion = cv2.erode(thresh, kernel, iterations=1)`
- **Opening:** A combination of erosion followed by dilation, used to remove small objects from the foreground.
  `opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)`
- **Closing:** A combination of dilation followed by erosion, used to close small holes within the foreground objects.
  `closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)`
- **Boundary Detection:** Detected by subtracting the eroded image from the original binary image.
  `boundary = cv2.subtract(thresh, erosion)`
- **Thinning:** Reduced the objects in the image to their skeletal form.
  `thinning = cv2.ximgproc.thinning(thresh)`
- **Thickening:** Achieved by dilating the inverse of the binary image.
  `thickening = cv2.bitwise_not(cv2.dilate(cv2.bitwise_not(thresh), kernel, iterations=1)`

4. **Image Resizing**

- All images were resized to a common size of 400x400 pixels to ensure uniform display and comparison.

```
common_size = (400, 400)
original_resized = resize_image(img, common_size)
dilation_resized = resize_image(dilation, common_size)
erosion_resized = resize_image(erosion, common_size)
opening_resized = resize_image(opening, common_size)
closing_resized = resize_image(closing, common_size)
boundary_resized = resize_image(boundary, common_size)
thinning_resized = resize_image(thinning, common_size)
thickening_resized = resize_image(thickening, common_size)
```

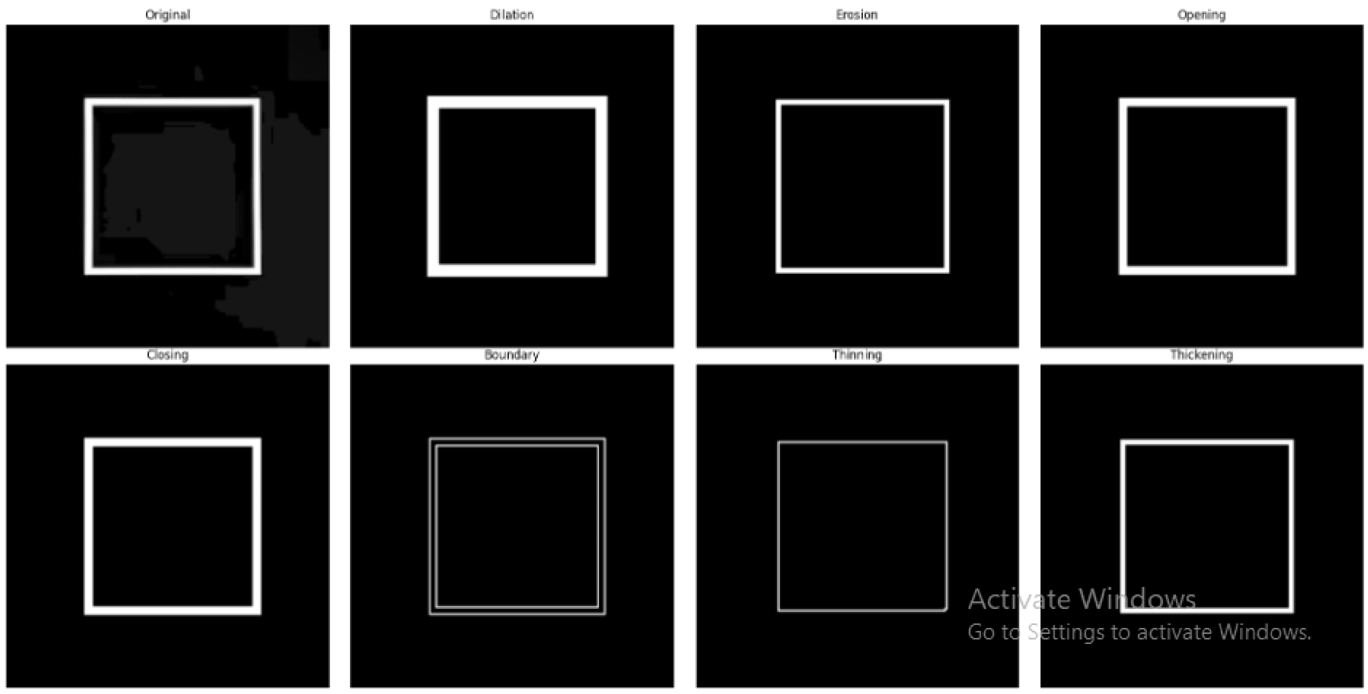
5. **Displaying Results**

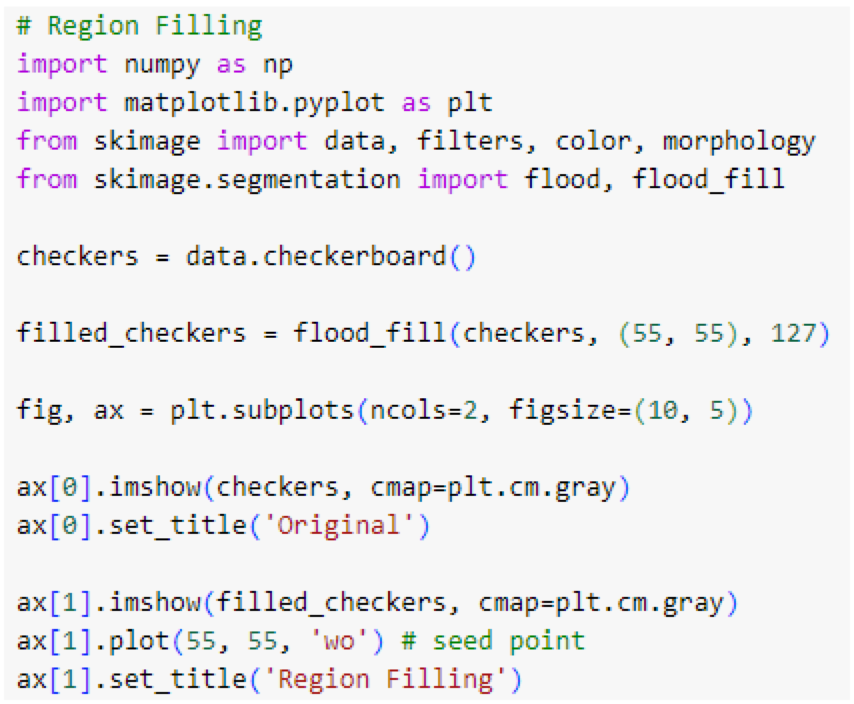
- The results of the morphological operations were displayed using Matplotlib. The images were arranged in a grid for easy comparison.

```
titles = ['Original', 'Dilation', 'Erosion', 'Opening', 'Closing', 'Boundary', 'Thinning', 'Thickening']
```

```
        images = [original_resized, dilation_resized, erosion_resized, opening_resized, closing_resized,
        boundary_resized, thinning_resized, thickening_resized]
        plt.figure(figsize=(20, 10))
        for i in range(len(images)):
        plt.subplot(2, 4, i+1)
         plt.imshow(images[i], cmap='gray')
        plt.title(titles[i])
        plt.axis('off')
plt.tight_layout()
plt.show()
```

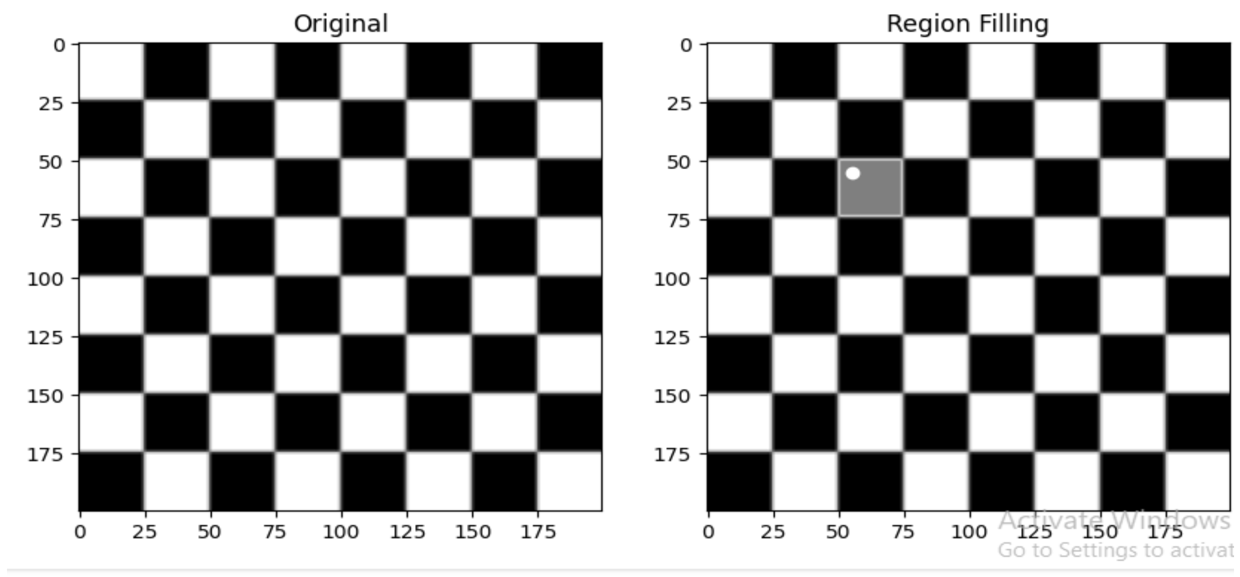
Result:





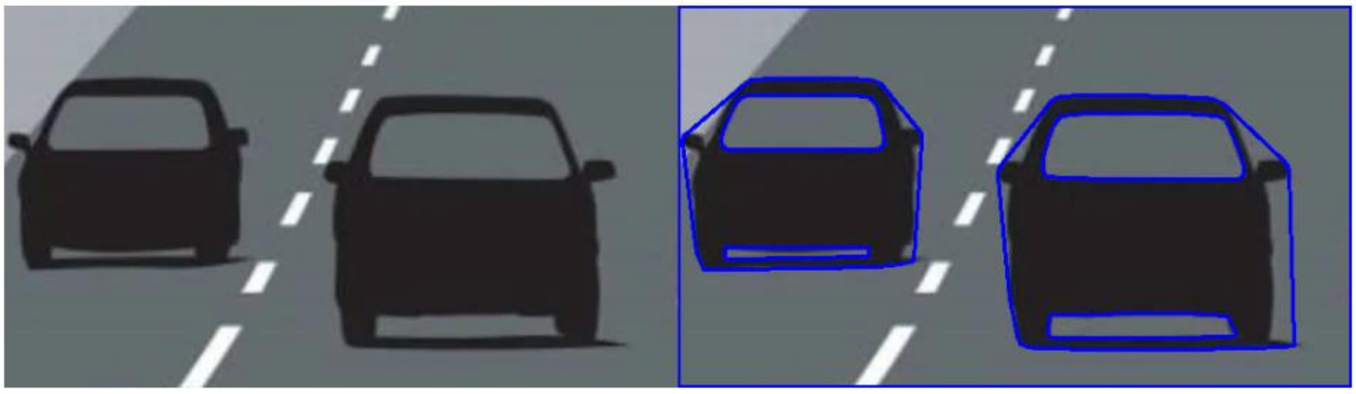
```
# Region Filling
import numpy as np
import matplotlib.pyplot as plt
from skimage import data, filters, color, morphology
from skimage.segmentation import flood, flood_fill

checkers = data.checkerboard()

filled_checkers = flood_fill(checkers, (55, 55), 127)

fig, ax = plt.subplots(ncols=2, figsize=(10, 5))

ax[0].imshow(checkers, cmap=plt.cm.gray)
ax[0].set_title('Original')

ax[1].imshow(filled_checkers, cmap=plt.cm.gray)
ax[1].plot(55, 55, 'wo') # seed point
ax[1].set_title('Region Filling')
```

Result:



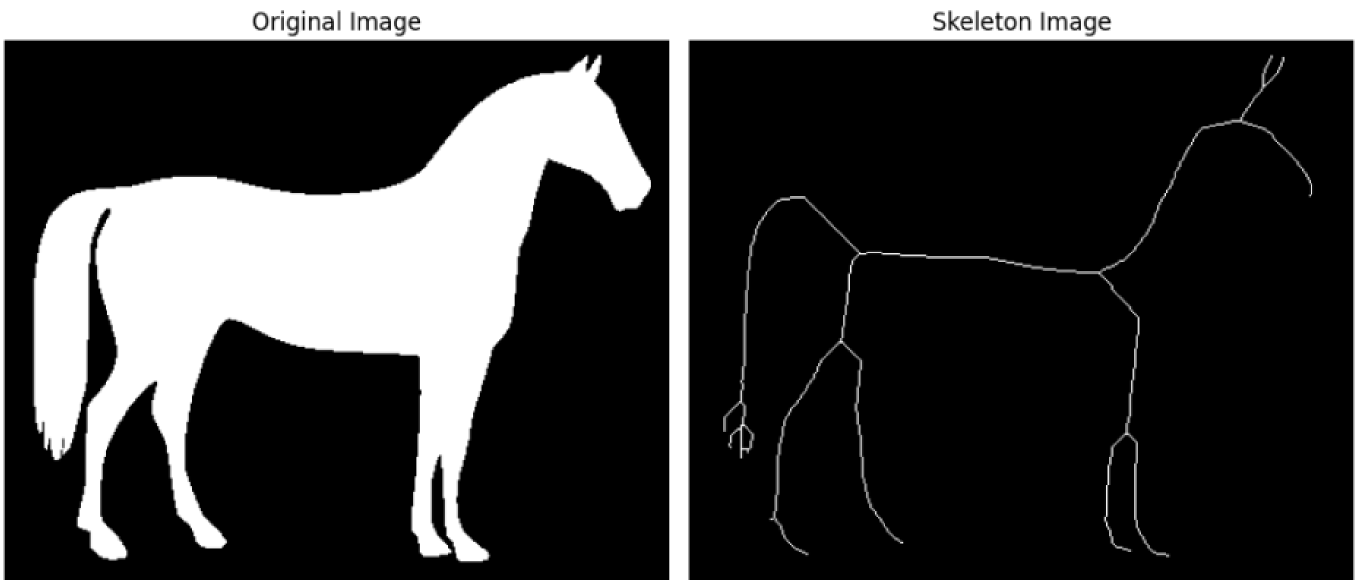
```
import cv2
from google.colab.patches import cv2_imshow

# Load the original image
img1 = cv2.imread('/content/car.webp')

# Convert it to greyscale
img_gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)

# Threshold the image
ret, thresh = cv2.threshold(img_gray, 50, 255, 0)

# Find the contours
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Create a copy of the original image to draw the convex hull
img_with_hull = img1.copy()

# For each contour, find the convex hull and draw it on the original image
for contour in contours:
    hull = cv2.convexHull(contour)
    cv2.drawContours(img_with_hull, [hull], -1, (255, 0, 0), 2)

# Display the original image and the image with convex hull side by side
combined_image = cv2.hconcat([img1, img_with_hull])
cv2_imshow(combined_image)
cv2.waitKey(0)
```

Result:



```python
import matplotlib.pyplot as plt
from skimage import data
from skimage.util import invert
from skimage.morphology import skeletonize

# Load and invert the image
image = invert(data.horse())

# Compute the skeleton
skeleton = skeletonize(image)

# Plotting
plt.figure(figsize=(10, 10))

# Original Image
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image, cmap=plt.cm.gray)
plt.axis('off')

# Skeleton Image
plt.subplot(1, 2, 2)
plt.title('Skeleton Image')
plt.imshow(skeleton, cmap=plt.cm.gray)
plt.axis('off')

plt.tight_layout()
plt.show()
```

Result:



Discussion: The various morphological operations explored in this lab are powerful tools for image processing. They allow for effective analysis and manipulation of image structures, facilitating tasks such as noise removal, shape analysis, object segmentation, and feature extraction. These techniques are widely used in fields like medical imaging, computer vision, and digital image analysis, providing essential functionalities for interpreting and modifying image data.