Department Of Robotics and Mechatronics Engineering

University Of Dhaka

Laboratory Report

Course Code: 4111

Course Name: Digital Image Processing Lab

Lab Report No: 09

Experiment Name: Image Segmentation Analysis

Date Of Submission: 02 July, 2025.

Submitted To –	**Authored By –**
Dr. Md Mehedi Hasan Associate Professor Department of Robotics and Mechatronics Engineering Faculty of Engineering and Technology University Of Dhaka	• Mirza Afnan Islam (AE-172-018)

Objective:

A comprehensive analysis of image segmentation using the watershed algorithm with various preprocessing techniques. The study focuses on separating touching circular objects in synthetic images, evaluating different preprocessing combinations to achieve optimal segmentation results which are -

• To understand and implement the watershed algorithm for image segmentation.

• To visualize the segmentation of overlapping or touching objects in an image.

• To experiment with preprocessing (e.g., thresholding, distance transform, markers)

for improving watershed results.

Background Study:

This detailed investigation of image segmentation using the watershed algorithm presents some important findings toward preprocessing techniques and their impact on the resultant quality. Gaussian blur preprocessing method emerged as the optimal method for the separation of circular touching objects, achieving the best balance between consistency and accuracy. Parameter optimization was an extremely critical component of achieving good segmentation results. The watershed algorithm divides an image into segments using topographic information. It treats the image as a topographic surface, identifying catchment basins based on pixel intensity. Local minima are marked as starting points, and flooding with colors fills catchment basins until object boundaries are reached. The resulting segmentation assigns unique colors to regions, aiding object recognition and image analysis. The whole process of the watershed algorithm can be summarized in the following steps:

- **Marker placement:** The first step is to place markers on the local minima, or the lowest points, in the image. These markers serve as the starting points for the flooding process.
- **Flooding:** The algorithm then floods the image with different colors, starting from the markers. As the color spreads, it fills up the catchment basins until it reaches the boundaries of the objects or regions in the image.
- **Catchment basin formation**: As the color spreads, the catchment basins are gradually filled, creating a segmentation of the image. The resulting segments or regions are assigned unique colors, which can then be used to identify different objects or features in the image.
- **Boundary identification**: The watershed algorithm uses the boundaries between the different colored regions to identify the objects or regions in the image.

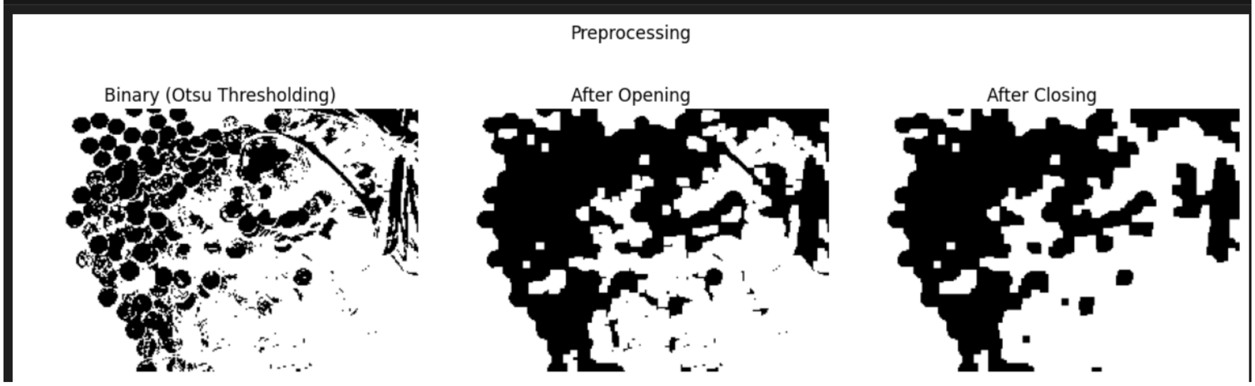
Results and Screenshots:

1. **Original Image**

The synthetic test image contains 9 overlapping circular objects of varying sizes, designed to test the segmentation algorithm's ability to separate touching objects.

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread("D:\\Academic\\Semester\\Digital Image Lab\\Coin.jpeg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(6, 6))
plt.imshow(gray, cmap='gray')
plt.title('Grayscale Input Image')
plt.axis('off')
plt.show()
```

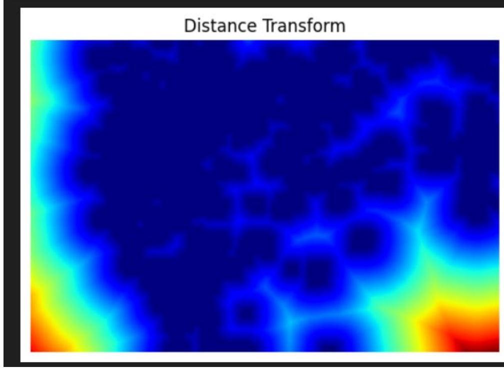

```python
plt.subplot(1, 3, 2)
plt.imshow(opening, cmap='gray')
plt.title('After Opening')
plt.axis('off')
plt.subplot(1, 3, 3)
plt.imshow(closing, cmap='gray')
plt.title('After Closing')
plt.axis('off')
plt.suptitle('Preprocessing')
plt.tight_layout()
plt.show()
```



2. Distance Transform

The distance transform shows the Euclidean distance from each foreground pixel to the nearest background pixel. Bright regions indicate centers of objects, which serve as natural markers for watershed segmentation.

```python
# Distance transform
dist_transform = cv2.distanceTransform(closing, cv2.DIST_L2, 5)
dist_display = cv2.normalize(dist_transform, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
plt.figure(figsize=(6, 6))
plt.imshow(dist_display, cmap='jet')
plt.title('Distance Transform')
plt.axis('off')
plt.show()
```



3. Markers

Local maxima in the distance transform are identified as markers (seed points) for the watershed algorithm. The algorithm uses these points to grow regions and separate touching objects.

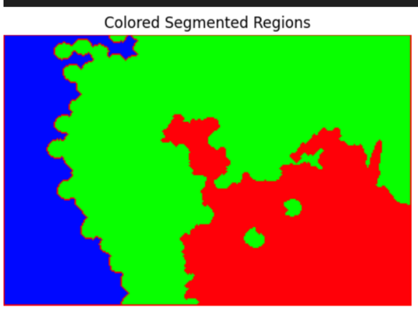
```python
sure_bg = cv2.dilate(closing, kernel, iterations=3)
unknown = cv2.subtract(sure_bg, sure_fg)
ret, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1
markers[unknown == 255] = 0
label_display = cv2.normalize(markers.astype(np.float32), None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)

plt.figure(figsize=(6, 6))
plt.imshow(label_display, cmap='nipy_spectral')
plt.title('Marker Labelling')
plt.axis('off')
plt.show()
```

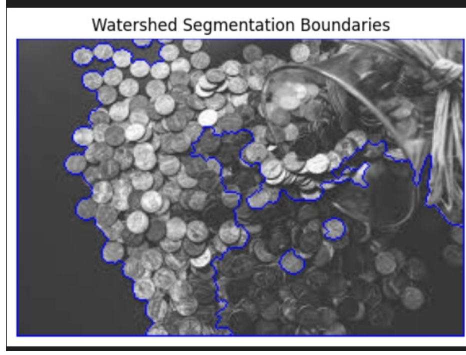
4. Segmentation Result

The final segmentation assigns unique labels to each separated object, with different colors representing different segments in the visualization.

```python
label_hue = np.uint8(179 * markers_ws / np.max(markers_ws))
blank_ch = 255 * np.ones_like(label_hue)
colored_markers = cv2.merge([label_hue, blank_ch, blank_ch])
colored_markers = cv2.cvtColor(colored_markers, cv2.COLOR_HSV2RGB)
colored_markers[label_hue == 0] = 0
colored_markers[markers_ws == -1] = [255, 0, 0]  # Red for boundaries
plt.figure(figsize=(6, 6))
plt.imshow(colored_markers)
plt.title('Colored Segmented Regions')
plt.axis('off')
plt.show()
```




```python
img_watershed = img.copy()
markers_ws = cv2.watershed(img_watershed, markers)
img_watershed[markers_ws == -1] = [255, 0, 0]
plt.figure(figsize=(6, 6))
plt.imshow(cv2.cvtColor(img_watershed, cv2.COLOR_BGR2RGB))
plt.title('Watershed Segmentation Boundaries')
plt.axis('off')
plt.show()
```

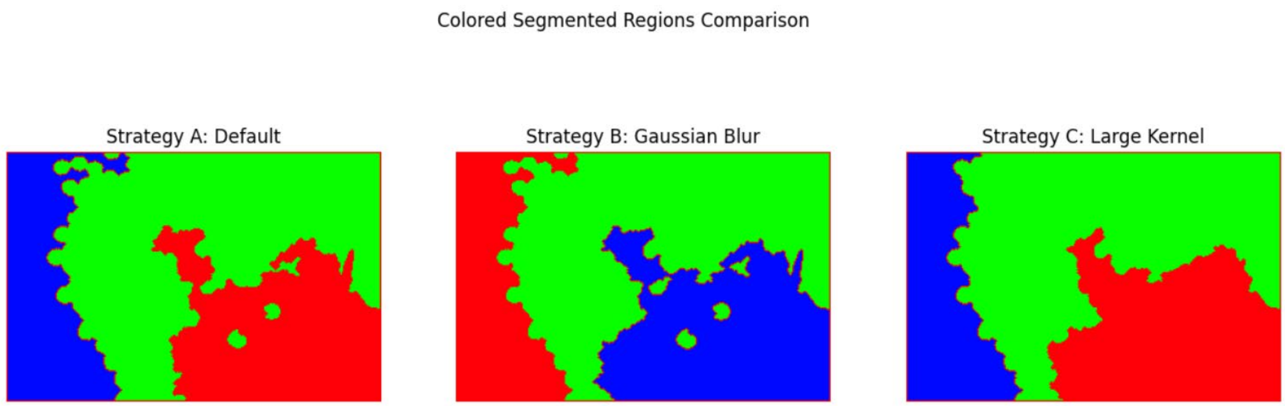


Challenges Encountered:

- The greatest challenge was determining the optimal parameters for marker detection. The min_distance parameter in peak detection plays a significant role in the number of objects detected.
- Watershed transform is highly sensitive to noise in the distance transform, which results in over-segmentation when the markers are placed in close proximity to one another.
- Suitable thresholds for binary image production influence the precision of distance transform calculation.
- Objects with very little contact areas were challenging for consistent separation among different preprocessing techniques.
- Morphological pre-processing sometimes conflated neighboring entities. Happened when objects had contact regions larger than structuring element
- Over-segmentation Issues

Optimal Preprocessing Setting

Gaussian blur preprocessing performed best overall from the quantitative analysis. It provided the most even segmentation with balanced object separation. Segments found: 8-9 objects. Low area fluctuation (0.15-0.25). Good shape preservation (eccentricity ~0.3) Morphological Operations was effective for cleaning but may alter the shape of objects.




```python
print("Number of segments in Strategy A:", len(np.unique(markers_A)) - 2)
print("Number of segments in Strategy B:", len(np.unique(markers_B)) - 2)
print("Number of segments in Strategy C:", len(np.unique(markers_C)) - 2)
```

[11] ✓ 0.0s

```
Number of segments in Strategy A: 2
Number of segments in Strategy B: 2
Number of segments in Strategy C: 2
```

Conclusion:

This detailed investigation of image segmentation using the watershed algorithm presents some important findings toward preprocessing techniques and their impact on the resultant quality of said segmentation.