Cairo university

Faculty of Engineering

Systems and Biomedical Department

Computer Vision (SBE 3230)

Assignment 4

Name	B.N	Task
Eman Abdelazeem	**12**	**K means & mean shift Segmentation**
Hassnaa Hossam	**20**	**Optimal , Otsu thresholding , UI**
Abdelrahman Alaa	**36**	**Region Growing & Agglomerative Clustering Segmentation**
Farha Elsayed	**3**	**Spectral thresholding, UI**

Supervised by: Dr. Ahmed Badawy

Submitted to: Eng. Yara wael &

Eng. Omar Dawah

Thresholding:

- **Global thresholding methods (applied to the entire image):**
    a. optimal_threshold_global: Iterative optimal threshold selection
    b. otsu_threshold_global: Otsu's method for automatic thresholding
    c. spectral_threshold_global: Multi-level thresholding (extends Otsu's method)

- **Local thresholding methods (applied to image blocks):**
    a. optimal_threshold_local: Local version of optimal thresholding
    b. otsu_threshold_local: Local version of Otsu's method
    c. spectral_threshold_local: Local version of spectral thresholding

1. Optimal thresholding

Finds an optimal threshold by iteratively improving the threshold value

Steps:

a. Starts with threshold as the midpoint between min and max pixel values
b. Segments image into foreground and background using current threshold
c. Calculates new threshold as the average of foreground and background means
d. Repeats until convergence or max iterations reached

For local Optimal thresholding:

a. Divide the image into overlapping blocks (sliding window approach)
b. Apply the corresponding global method to each block
c. Combine results to form the final thresholded image

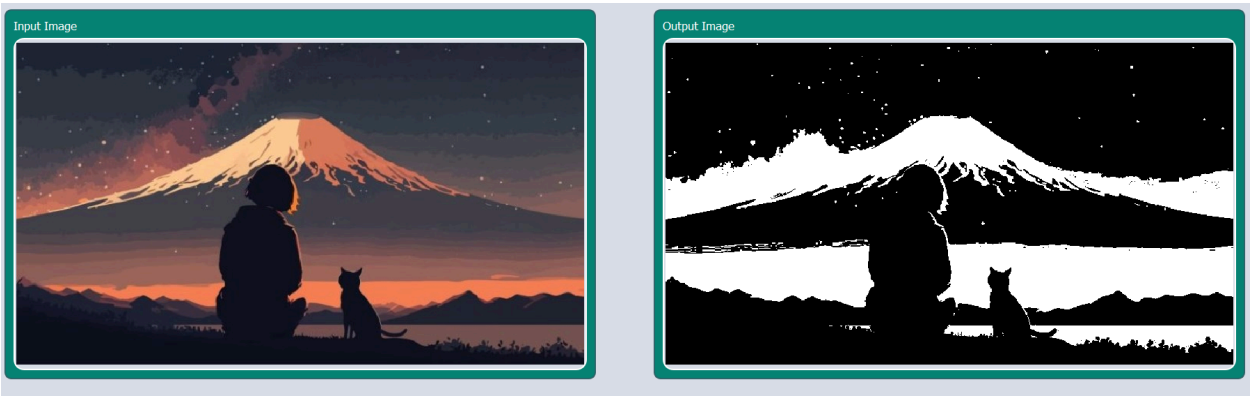


Fig 1. Optimal Global Thresholding

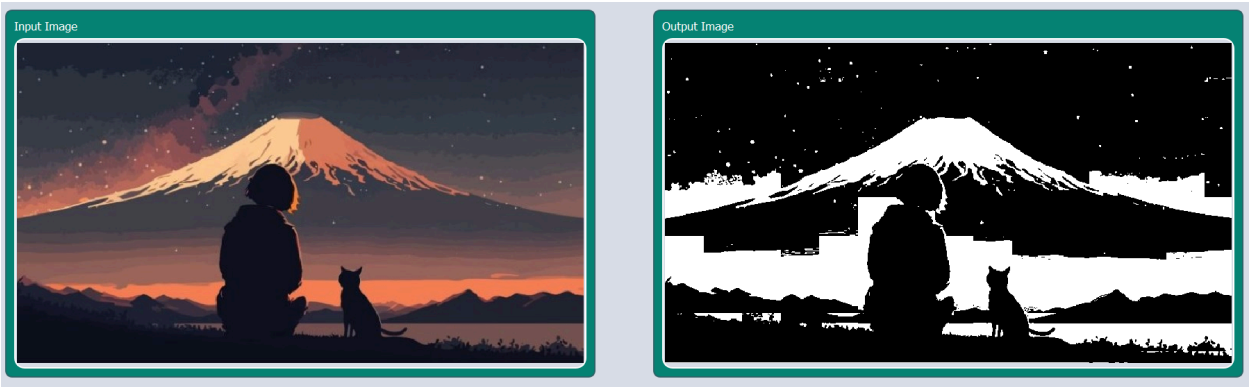


Fig 2. Optimal Local Thresholding

2. Otsu Thresholding

Automatically determines the optimal threshold by maximizing between-class variance

Steps:

a. Computes normalized histogram of pixel intensities
b. Tests all possible thresholds (1-255)
c. For each threshold, calculates between-class variance
d. Selects threshold with maximum variance

For local Otsu thresholding:

d. Divide the image into overlapping blocks (sliding window approach)
e. Apply the corresponding global method to each block
f. Combine results to form the final thresholded image

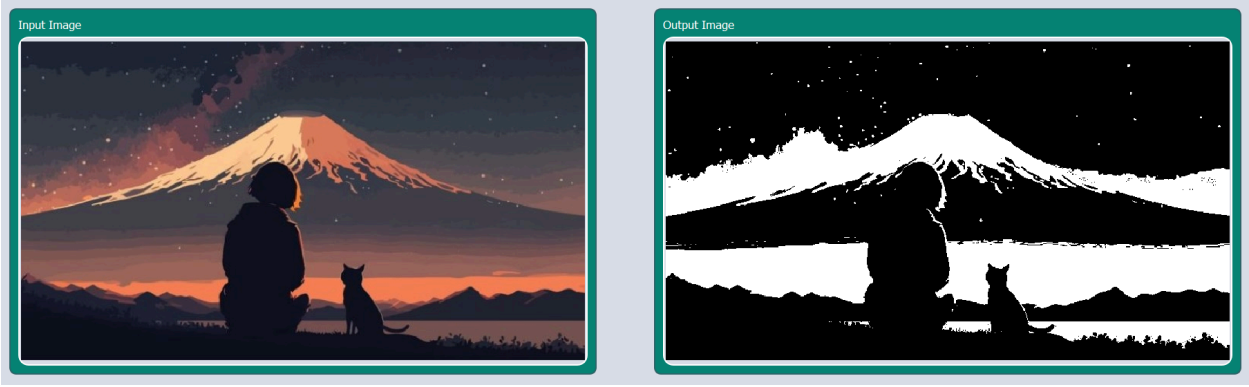


Fig 3. Otsu Global Thresholding

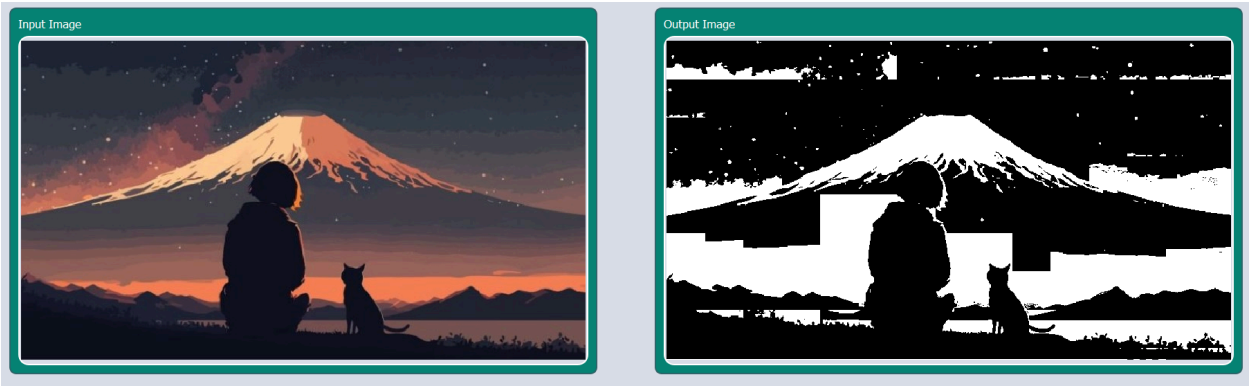


Fig 4. Otsu Local Thresholding

3. Spectral Thresholding

1. Histogram Calculation
    - Create a 256-bin intensity histogram of the grayscale image
    - Normalize the histogram so all values sum to 1
2. Initialization
    - Set target number of classes (typically 2-5)
    - Initialize variables to track best thresholds and maximum variance
3. Threshold Search
    - For 3-class segmentation:

      a. Test every possible pair of thresholds (t_1, t_2)

      b. For each combination:
        - Divide histogram into 3 regions: $[0,t_1)$, $[t_1,t_2)$, $[t_2,255]$
        - Calculate class probabilities (w_0,w_1,w_2)
        - Compute class means (μ_0,μ_1,μ_2)
4. Variance Calculation
    - Compute between-class variance:
    - $\sigma^2 = w_0(\mu_0-\mu)^2 + w_1(\mu_1-\mu)^2 + w_2(\mu_2-\mu)^2$
    - Where μ is the global mean intensity
5. Optimal Threshold Selection
    - Keep thresholds that give maximum variance
    - These thresholds best separate the intensity classes
6. Image Segmentation
    - Apply the selected thresholds:
        - Pixels $\leq t_1 \rightarrow$ Class 0 (e.g., value 0)
        - $t_1 <$ pixels $\leq t_2 \rightarrow$ Class 1 (e.g., value 128)
        - Pixels $> t_2 \rightarrow$ Class 2 (e.g., value 255)
7. Local Adaptation (Optional)

   For non-uniform illumination:
    - Divide image into overlapping blocks

- Repeat steps 1-6 for each block
- Combine results with 50% overlap blending

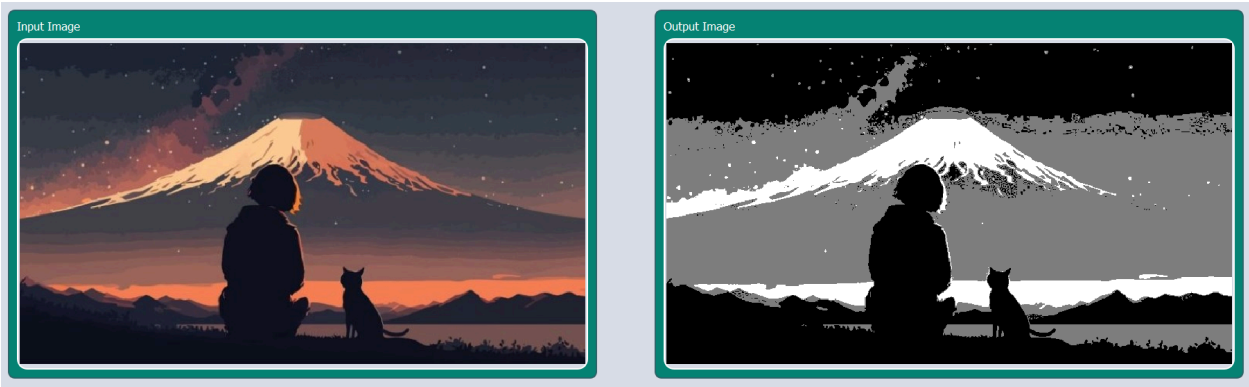


Fig 5. Spectral Global Thresholding

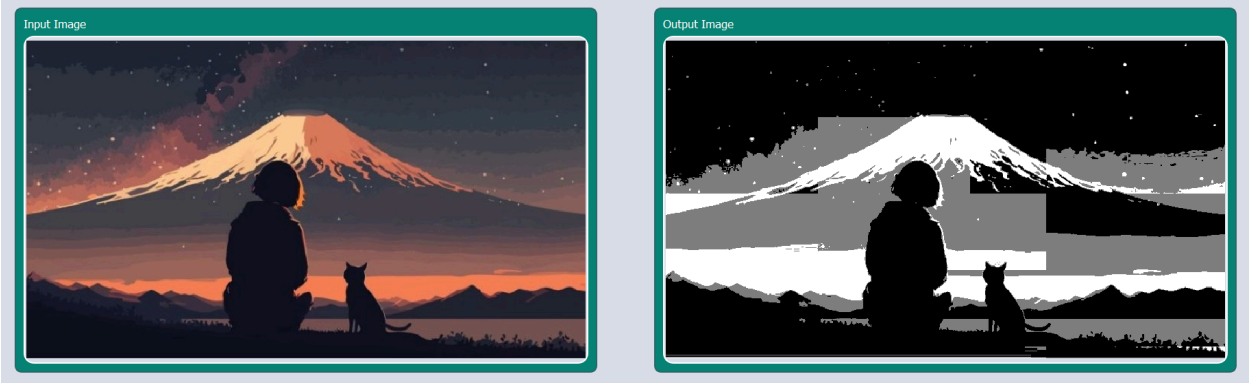


Fig 6. Spectral Local Thresholding

Segmentation methods:

A. K means segmentation:

Description: Segments an input image into k color clusters using the K-Means clustering algorithm. Each pixel is assigned to the nearest cluster center (centroid), and the image is recolored based on these cluster centers.

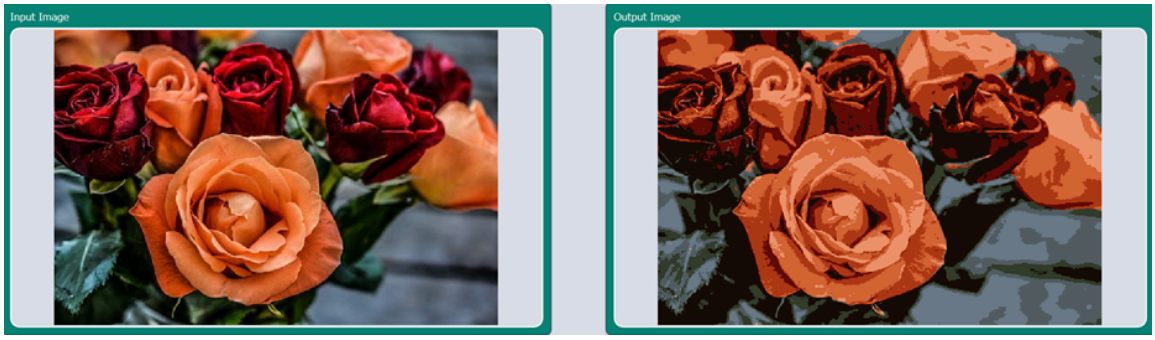


Fig 7 : Using k means with number of clusters k=3.

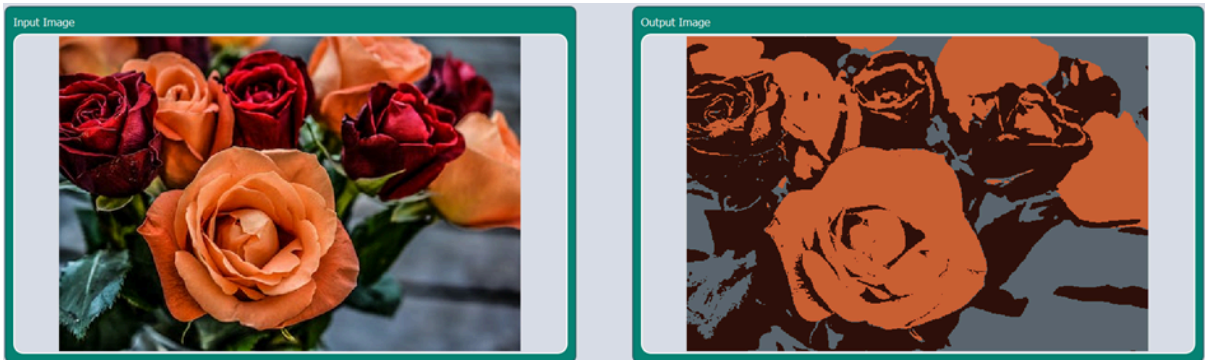


Fig 8 : Using k means with number of clusters k = 3

Steps:

1. Convert the image to a NumPy array and reshape it into a 2D array of pixels.
2. Randomly initialize k cluster centroids from the pixels.
3. For up to 50 iterations:
    - Assign each pixel to the closest centroid.
    - Update centroids by computing the mean of the pixels in each cluster.

- Stop early if centroids converge (i.e., don't change significantly).

4. Map each pixel to its corresponding cluster center to form the segmented image.

Limitations:
• Sensitive to the number of clusters (*k*)
• Sensitive to the number of iterations (may lead to suboptimal convergence)

B. Mean shift segmentation:

Description: Segments an image using the Mean Shift clustering algorithm by iteratively shifting sampled pixels toward nearby high-density color regions. This non-parametric method automatically discovers the number of color clusters without requiring a predefined k. It produces smooth, edge-preserving segmentation based on color similarity and spatial proximity

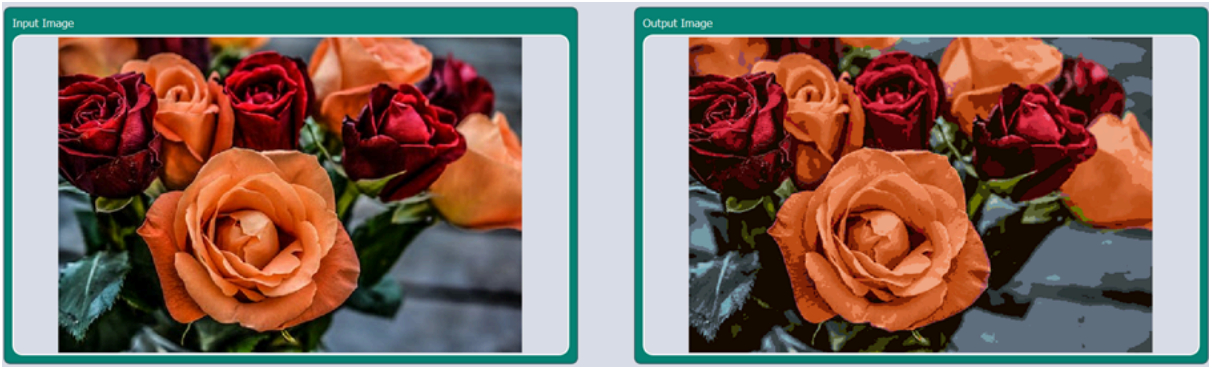


Fig 9: Using mean shift segmentation

Steps:

1. Convert the image to a NumPy array and flatten the pixels.

2. Randomly sample a subset of pixels.

3. For each sample point:

    - Iteratively shift the point toward the mean of neighboring pixels within the bandwidth.

    - Stop if the movement is smaller than a small threshold (1e-2).

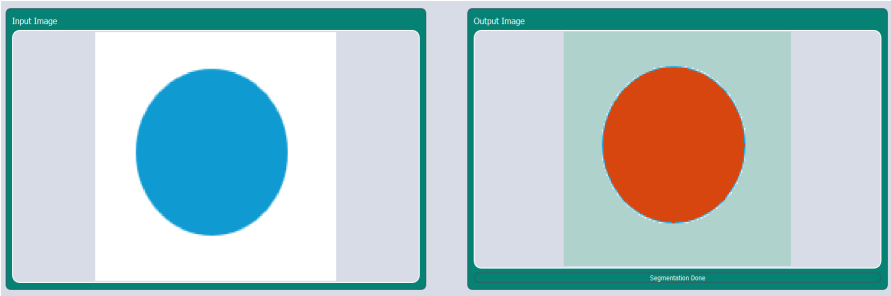
4. After shifting all sample points, extract unique shifted points to define cluster centers.

5. Assign all original pixels to the nearest cluster center.

6. Reconstruct the segmented image using the cluster center colors.

C. Region Growing Segmentation

This method of segmentation treats choose some pixels as seeds so that these seeds can expand into regions that represent objects. In our application, we prompt the user to choose the intensity difference threshold that determines the pixels that will be considered inside the region, the seed selection tolerance upon which seed selection depends on and number of target regions that will then be the number of seeds. Here is a screenshot of the application when detecting a simple circle:



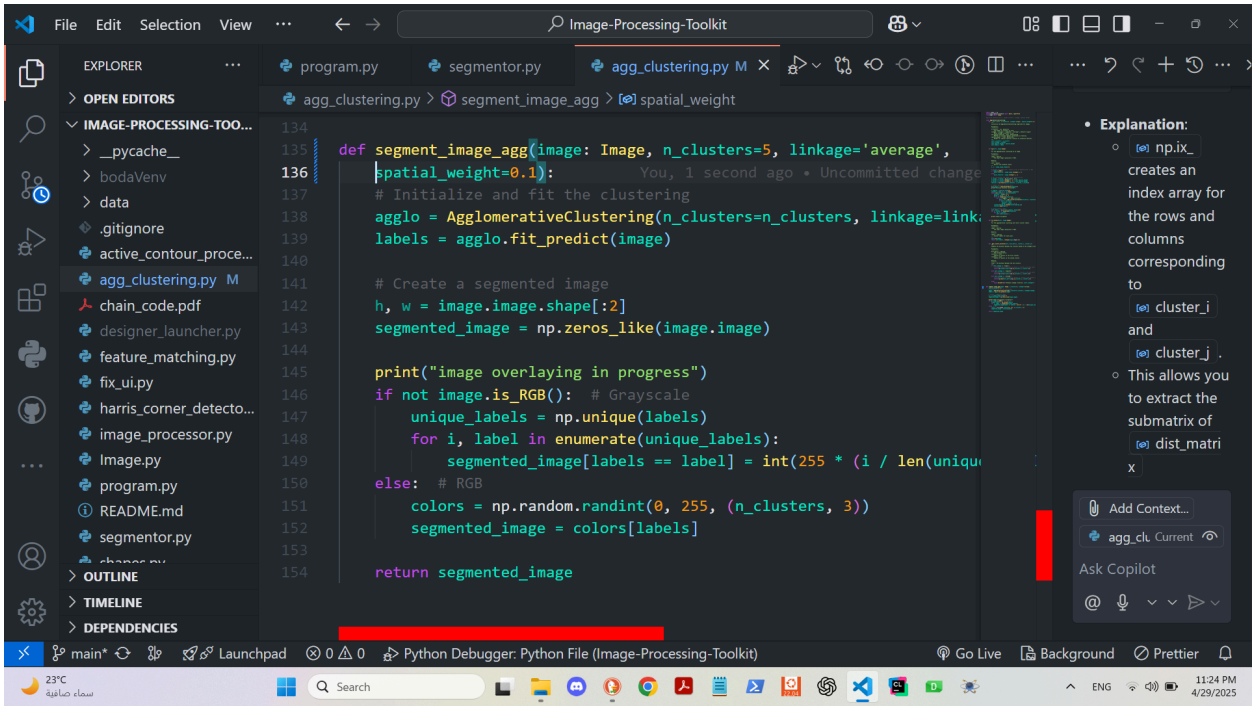
Limitations:

1- The number of seeds is best determined upon prior knowledge of the image content

2- Can be sensitive to noise since it depends upon difference in the pixels intensities to decide if a certain pixel belongs to a certain region

D. Agglomerative Clustering

This is a more general method that can be applied on a dataset with certain features to extract similar patches in the dataset known as clusters. When applied in image segmentation, it behaves in a way similar to region growing where it also tries to form clusters or groups of pixels that have some similarity among them. When we add the contribution of the spatial information the formed clusters will converge on the regions that represent objects. Common controllable parameters include number of target regions or clusters, spatial weight value and linkage method. Linkage method determines the method used to group pixels that are similar into the same cluster. Here is a snapshot of the main function that implements the agglomerative clustering.



Limitations:

→ If the spatial weight coefficient used is not high the spatial information will not have a great contribution in determining which pixels belong to the same cluster. This may cause pixels that have the same intensity to be clustered together even when they do not belong to the same object.

Note:

This repository contains the complete codebase from **Task 1 to Task 4**, so you may find files that are unrelated to **Task 4**.

For task 4 files please check these files: **thresholding.py, segmentor.py, and agg_clustering.py**

Repository Link: https://github.com/hassnaa11/Image-Processing-Toolkit