Faculty of Engineering
Cairo University

Cairo University

Computer Vision

Thresholding & Segmentation

Team 19

Name	Section	B.N
Mohamed Alaa Ali	2	19
Mohamed Ibrahim Ismail	2	9
Mohamed El-sayed Ali	2	10
Mohamed El-sayed Eid	2	11

Introduction:

This task explores Thresholding & Segmentation techniques. For the Thresholding part, Spectral Thresholding, Otsu Thresholding, and Optimal Thresholding for both Local & Global Thresholding will be implemented. For the Segmentation part K-means clustering, Agglomerative clustering, Region growing, and Mean shift clustering.

1. Thresholding:

1.1 Optimal thresholding:

- **Algorithm**

The "**optimal_threshold"** function takes the following parameters:

1. initial_threshold: The initial threshold value. Default is 125.
2. max_iter: The maximum number of iterations for the algorithm. Default is 100.
3. min_diff: The tolerance for convergence. Default is 1e-5.

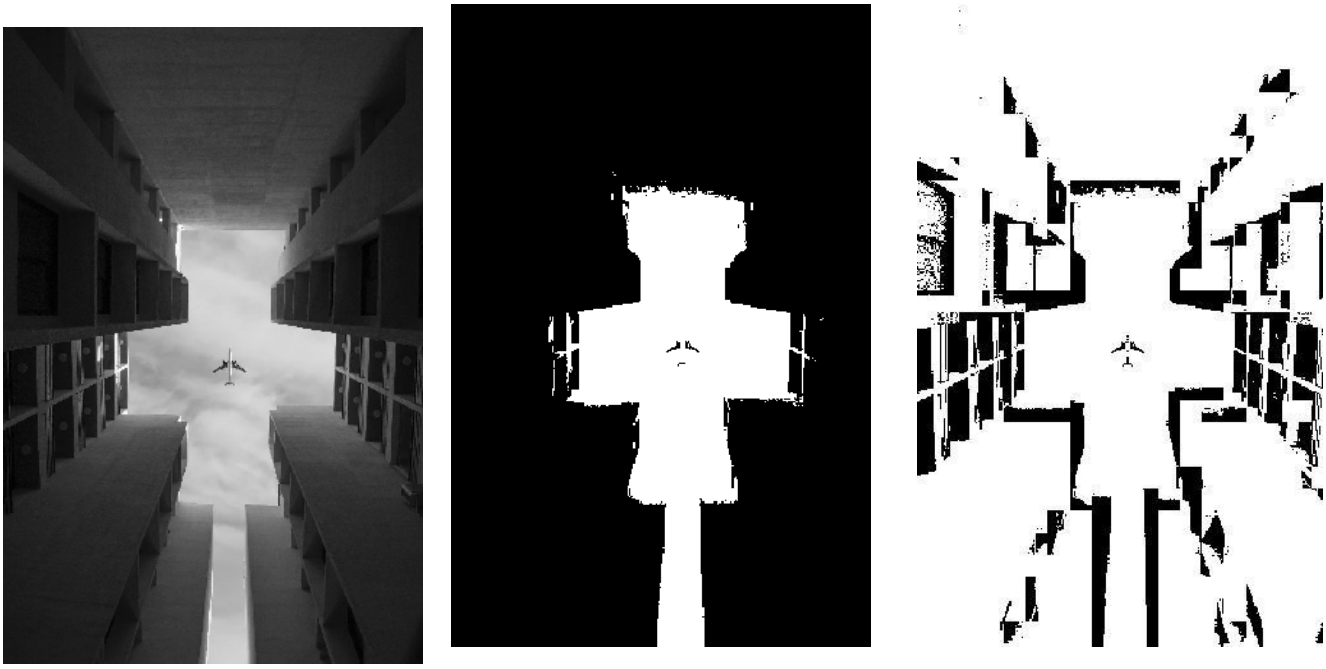
The iterative optimal thresholding algorithm starts with initializing the threshold at 125. It then iteratively calculates the mean intensity values for both the foreground and background regions, adjusting the threshold to the average of these means. This iterative process persists until either the difference in threshold values between iterations falls below the convergence threshold or the maximum iteration limit is attained. Throughout each iteration, the algorithm dynamically refines the threshold to optimize the segmentation process, ensuring accurate discrimination between foreground and background elements.

- **Global VS Local Thresholding**

1. Global thresholding involves determining a single threshold value for the entire image, which is then used to separate foreground objects from the background. This method is straightforward and computationally efficient, making it suitable for images with uniform illumination and consistent foreground-background contrast.

2. Optimal local thresholding calculates a threshold value for each pixel based on its local neighborhood, allowing for more adaptive segmentation in regions with varying illumination or contrast. By adjusting the threshold dynamically according to local properties.

Sample Results



Original Image

Image after Optimal Global thresholding

Image after Optimal Local thresholding

Original Image

Image with Optimal Local Thresholding (window size = 10)

Image with Optimal Local Thresholding (window size = 4)

1.2. Otsu thresholding:

Global Otsu's Thresholding

- Convert to Grayscale
- Compute the histogram of pixel intensities for the grayscale image.
- Initialize Variables: Sets up counters and accumulators for the total sum of pixel intensities and their weights (histogram counts).
- Calculate the total number of pixels and the sum of all weighted pixel intensities.
- For each possible threshold value, the pixels are divided into two groups (below and above the threshold).
- update and calculate the weights and the cumulative sums for these groups.
- Calculate the between-class variance for each threshold.
- Find the threshold that maximizes the between-class variance.
- Convert the grayscale image into a binary image where pixels below the threshold are set to 0 and those above are set to 255.

Local Otsu's Thresholding

- convert the image to grayscale if it is not already.
- Divide the Image: The image is divided into smaller blocks or regions, as defined by a set block size (e.g., 10x10 pixels).
- Apply Otsu's Threshold to each block
- Compile the Results: After processing each block individually, the results are compiled into a single image, combining all thresholded blocks to form the final locally thresholded image.

1.3 Spectral Thresholding:

Input:

- Input image: The grayscale image on which spectral thresholding will be applied.

- Parameters:

- **block_size**: Size of the local region for local spectral thresholding.
- **bandwidth**: Bandwidth parameter for local spectral thresholding.

Processes:

1. Global Spectral Thresholding:

- Convert the input image to grayscale.
- Compute the histogram of pixel intensities
- Compute the cumulative distribution function (CDF) of the histogram.
- Determine thresholds for low and high intensity levels:
- **threshold_low**: Intensity level separating the lower third of the CDF.
- `**threshold_high**: Intensity level separating the upper third of the CDF.
- Generate a new image (**global_image**) based on these thresholds:
- Pixels with intensity <= **threshold_low** are set to 0.

- Pixels with intensity > **threshold_low** and <= **threshold_high** are set to 127.
- Pixels with intensity > **threshold_high** are set to 255.

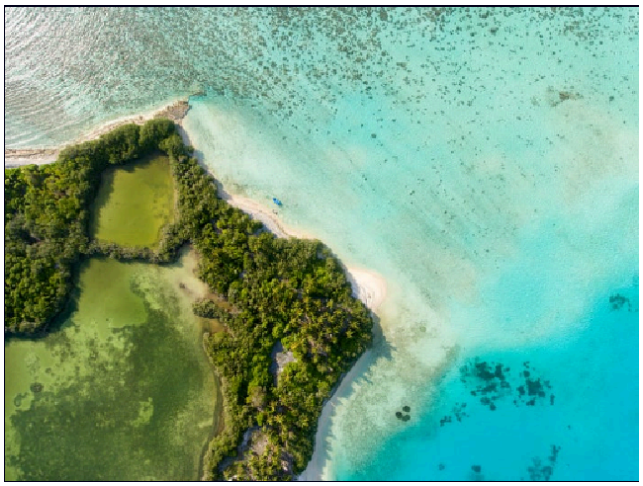
2. Local Spectral Thresholding:

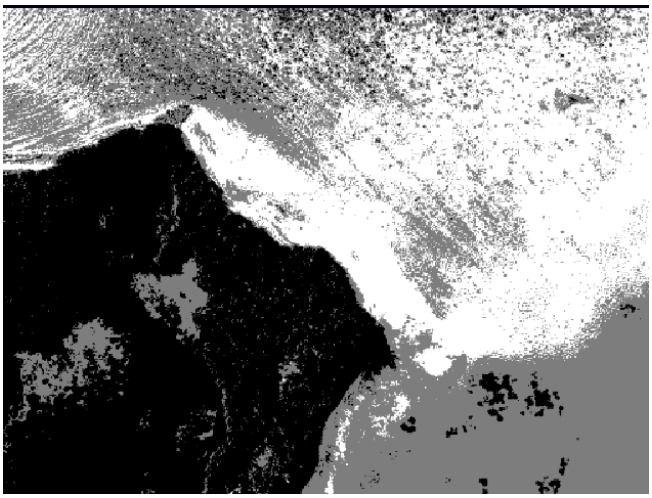
- Convert the input image to grayscale.
- Initialize an empty local image.
- Define a padding size based on the **block_size**.
- Iterate over the image pixels excluding the padded borders:
- Extract a local block centered at each pixel
- Compute the mean intensity of the block.
- Determine thresholds for low and high intensity levels based on the block mean and `bandwidth`.
- Assign intensity values to the corresponding pixel in the local image based on the thresholds:
    - If intensity <= **threshold_low**, set to 0.
    - If intensity > **threshold_low** and <= **threshold_high**, set to 127.
    - If intensity > **threshold_high**, set to 255.

Output:

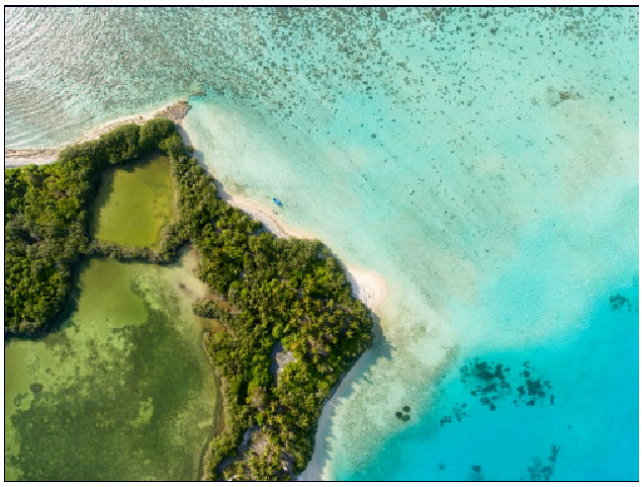
- Global Thresholded Image: The image resulting from spectral thresholding applied globally, where pixels are categorized into three intensity levels.
- Local Thresholded Image: The image resulting from spectral thresholding applied locally, where intensity levels are determined based on local regions.

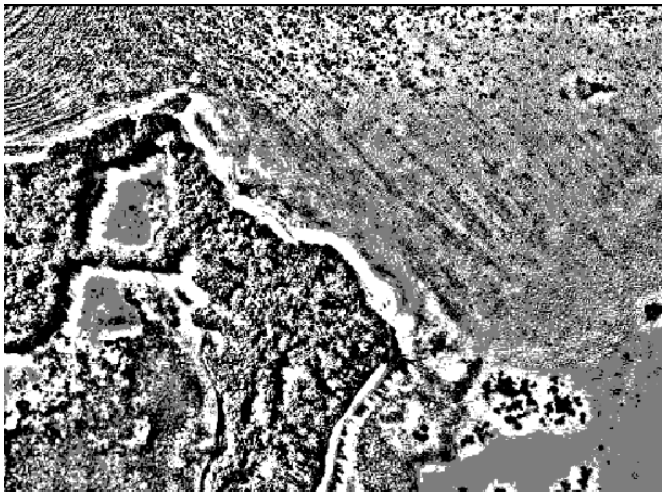
Global Spectral Thresholding example:





Local Spectral Thresholding example:





1. Segmentation:

2.1 K-Means Clustering:

Initialization:

- Define the number of clusters (K) and maximum iterations (max_iters),
- Randomly initialize K centroids.

Cluster Assignment:

- For each data point, calculate its distance to all centroids.
- Assign each data point to the cluster represented by the nearest centroid.

Centroid Update:

- Recalculate centroids by computing the mean of all data points assigned to each cluster.
- Update the positions of centroids based on these means.

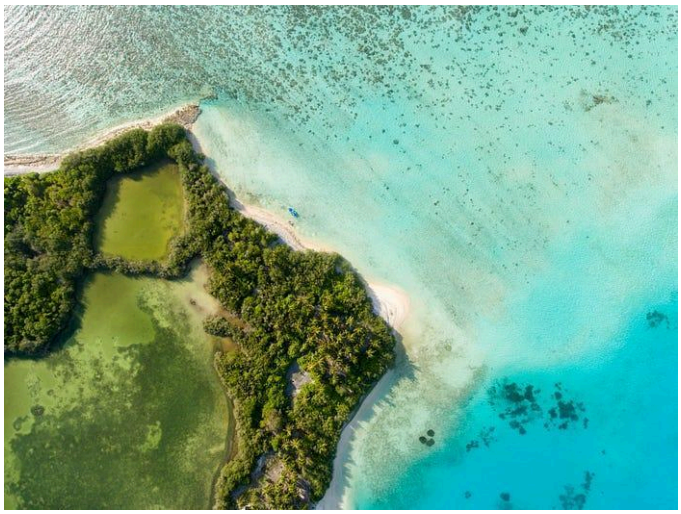
Convergence Check:

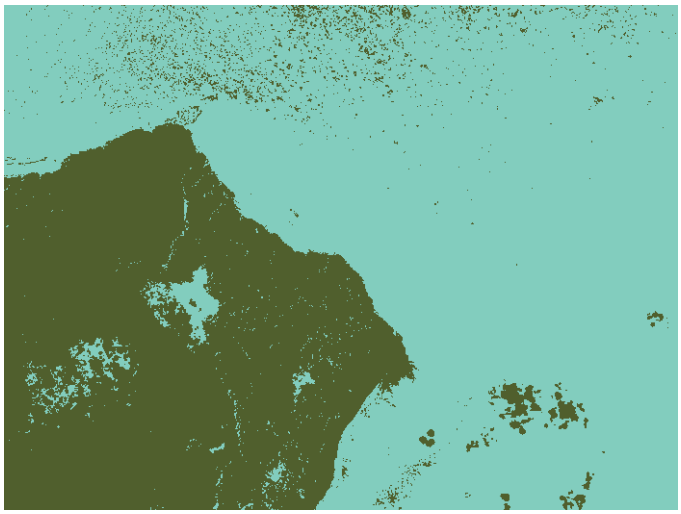
- Check if the centroids have converged (i.e., if the change in centroids is below a specified threshold) or if the maximum number of iterations has been reached.

Finalization:

- Once convergence is achieved:
- Assign cluster labels to all data points based on the final centroids.

- Each data point is labeled with the cluster index corresponding to its assigned centroid.





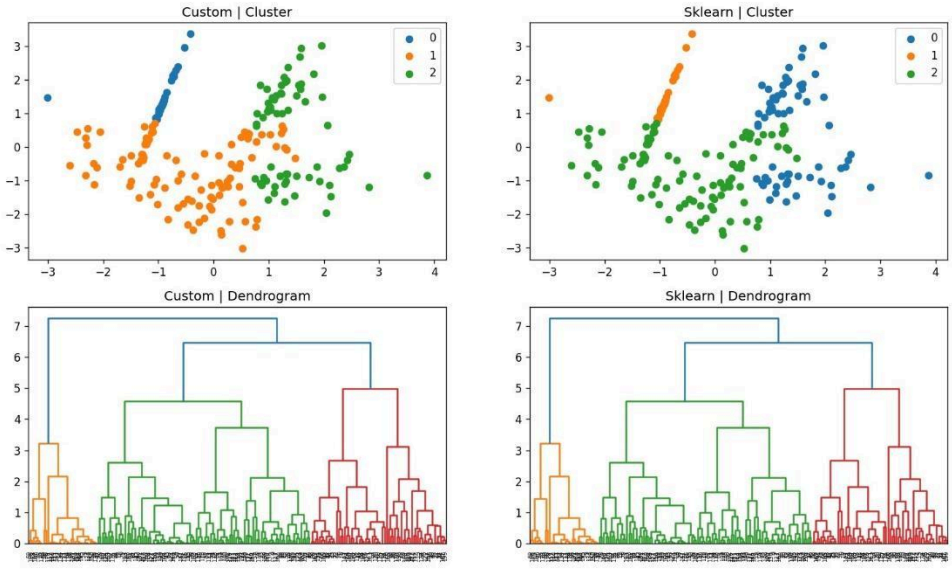
2.2 Agglomerative Clustering:

Algorithm

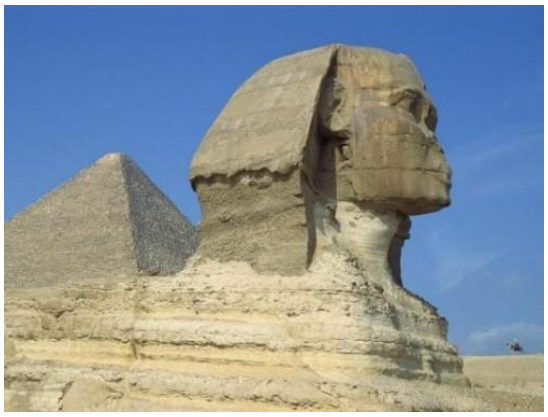
1. Initialize each data point as its own cluster.
2. Compute the pairwise distance matrix between all data points.
3. Initialize a list of clusters, initially containing all individual data points.
4. While the number of clusters is greater than 1:
    a. Find the closest pair of clusters based on a chosen linkage criterion.
    b. Merge the two closest clusters into a single cluster.
    c. Update the distance matrix to reflect the distance between the new cluster and all other clusters.
    d. Remove the merged clusters from the list of clusters.
    e. Add the newly merged cluster to the list of clusters.
5. Return the final list of clusters.

This process creates a hierarchical tree-like structure called a dendrogram, where the root represents a single cluster containing all data points, and each intermediate node represents a cluster formed by merging its children.

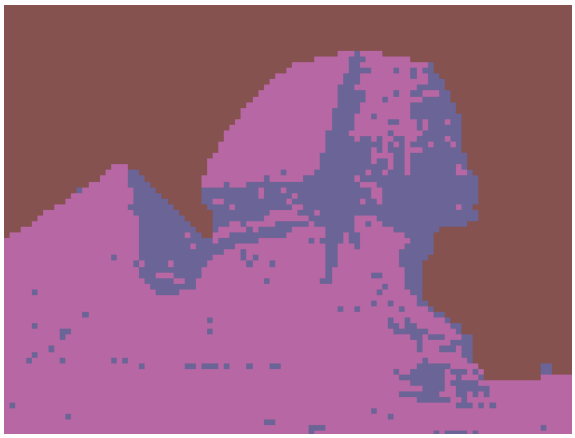
Comparison Between Built-in function in Sikit-learn and implemented function from scratch.



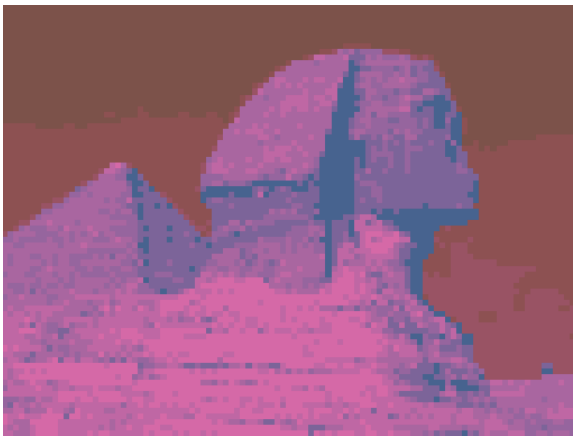
- **Sample Results**



Original Image

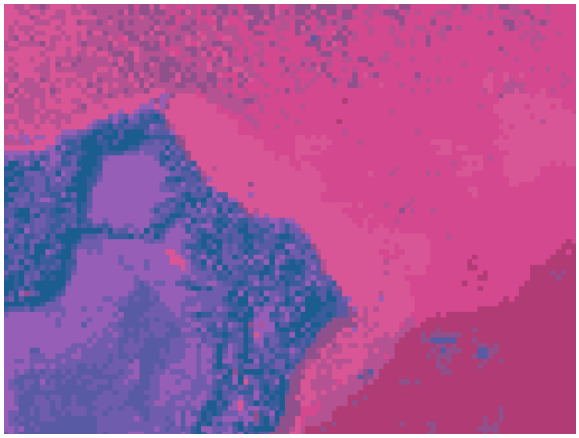


Segmented Image with
number of Clusters = 3



Segmented Image with
number of Clusters = 9

Original Image

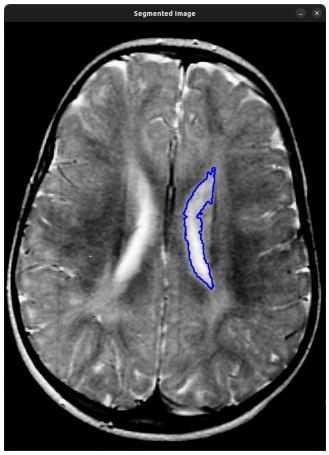


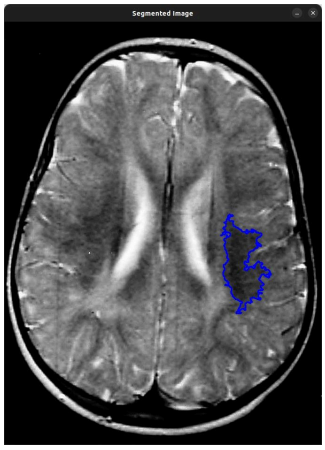
Segmented Image with number of Clusters = 10

2.3 Region growing:

- Converts the input image to grayscale.
- Initializes arrays to track visited pixels and the segmented image.
- Iterates over each seed point provided.
- For each seed point, starts a region growing process using a queue to manage the pixels to be processed.
- While the queue is not empty, a pixel is popped from the queue.
- Checks if the pixel is within the bounds of the image and has not been visited.
- If the pixel's intensity is within the threshold of the mean intensity of the region, mark it as part of the region and add its neighbors to the queue.

Results:





2.4 Mean Shift:

Processes:

1. Initialization:

- Initialize MeanShiftSegmentation object with the input image.
- Reshape the image into a 2D array of colors and positions.
- Select a random pixel as the initial mean.

2. Segmentation Process:

- Iterate until there are pixels remaining in the list:
  - Calculate distances between each pixel color and the current mean.
  - Group pixels whose distances are below a threshold.
  - Calculate the mean color and position of the grouped pixels.
  - If total distance to the new mean is below a threshold:
    - Update pixels in the cluster to the new mean color.
  - Otherwise:
    - Update the current mean to the new mean and continue iterating.

Output:

- Segmented Image: The image with pixels assigned colors representing the clusters they belong to, after Mean Shift segmentation.

Example: