Computer Vision

(Segmentation)

TASK (4)

Supervised by:
Dr. Mohamed Badawi
Eng. Peter Salah
Eng. Laila Abbas

Name	Section	Bn
Sama Mostafa	1	44
Misara Ahmed	2	43
Yousr Asharf	2	54
Rahma Abdelkader	1	31
Youssef Essam	2	57

Segmentation:

Image segmentation is the process of dividing an image into multiple homogeneous regions based on their Similarities, differences or even characteristics such as color, texture, shape, or brightness, in order to simplify or change the representation of an image into something more meaningful and easier model for analysis. Here, each pixel is labeled, where the pixels belonging to the same category have a common label assigned to them.

Segmentation is a fundamental task in various applications such as

- object recognition and tracking
- Medical imaging and Tumor detection
- Robotics

General concepts:

Local and global thresholding are two commonly used methods in image processing for segmenting an image into foreground and background regions based on pixel intensity values.

Local thresholding, also known as adaptive thresholding, involves computing a threshold value for each pixel based on the local image properties in a local neighborhood around that pixel. The threshold value is typically calculated as a function of the mean or median pixel intensity values within the neighborhood. This method is useful for images with non-uniform illumination or for objects with varying intensity levels.

Global thresholding, on the other hand, involves computing a single threshold value for the entire image based on the global properties of the image. This method is useful for images with a uniform background and well-defined foreground objects with consistent intensity values.

In summary, local thresholding calculates a threshold value for each pixel based on its local neighborhood, while global thresholding calculates a single threshold value for the entire image.

Segmentation techniques:

1. Thresholding:

1.1. Optimal Thresholding

The Optimal Thresholding algorithm is an image processing technique that automatically selects the best threshold value to separate an image into two classes, such as foreground and background. It is a widely used technique for image segmentation in computer vision and image processing.

Algorithm steps:

1. **Compute a histogram of the image intensities:** The algorithm starts by computing a histogram of the image intensities.

2. **Compute the cumulative distribution function (CDF) of the histogram:** The CDF is a function that gives the probability that a pixel has an intensity less than or equal to a certain value.

3. **Compute the mean intensity value of the entire image:** The mean intensity value of the image is computed as the weighted sum of the intensity values, where the weights are the probabilities from the normalized histogram.

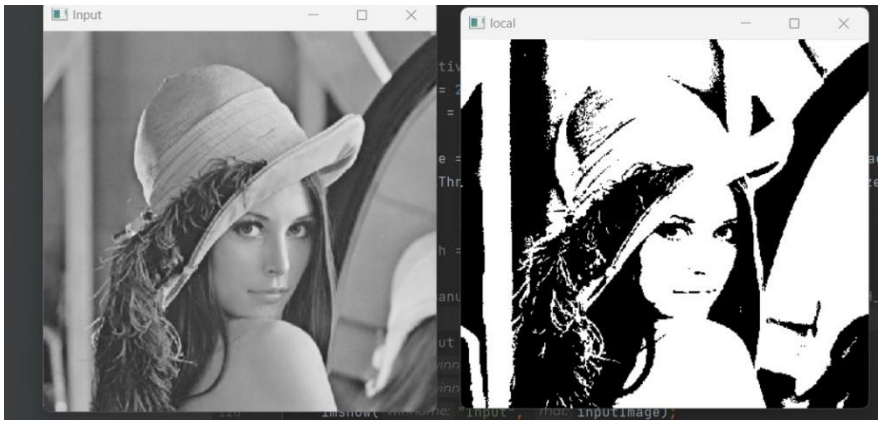
4. **Compute the between-class variance for each possible threshold value:** The algorithm computes the between-class variance for each possible threshold value. The between-class variance is defined as the weighted sum of the variances of the two classes (foreground and background), where the weights are the probabilities of the classes.

5. **Select the threshold value that maximizes the between-class variance:** The threshold value that maximizes the between-class variance is selected as the optimal threshold value for segmenting the image. All pixels with intensities greater than or equal to the threshold value are classified as foreground pixels, and all pixels with intensities less than the threshold value are classified as background pixels.

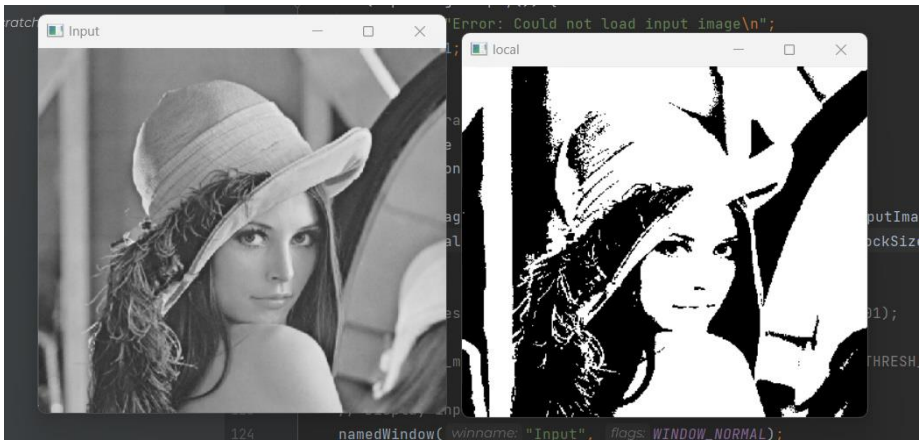
Output examples:

- Global optimal thresholding:

- Local optimal thresholding : (at blocksize =21)



- Local optimal thresholding : (at blocksize =91)



1.2. Otsu Thresholding

Otsu thresholding is an image processing technique that segments an image into foreground and background regions automatically. It calculates a threshold value that separates the pixels into two groups based on their intensities. The algorithm maximizes the between-class variance of the pixel intensities to find the threshold value, which is the value that best separates the foreground and background regions.

Algorithm steps:

1. **Compute a histogram of the image intensities**: The algorithm begins by computing a histogram of the image intensities. The histogram counts the number of pixels in the image at each intensity level.

2. **Normalize the histogram:** The histogram is normalized so that it represents a probability density function. This step ensures that the sum of the probabilities equals one.

3. **Compute the cumulative distribution function:** The cumulative distribution function (CDF) of the probability density function is computed. The CDF is a function that gives the probability that a pixel has an intensity less than or equal to a certain value.

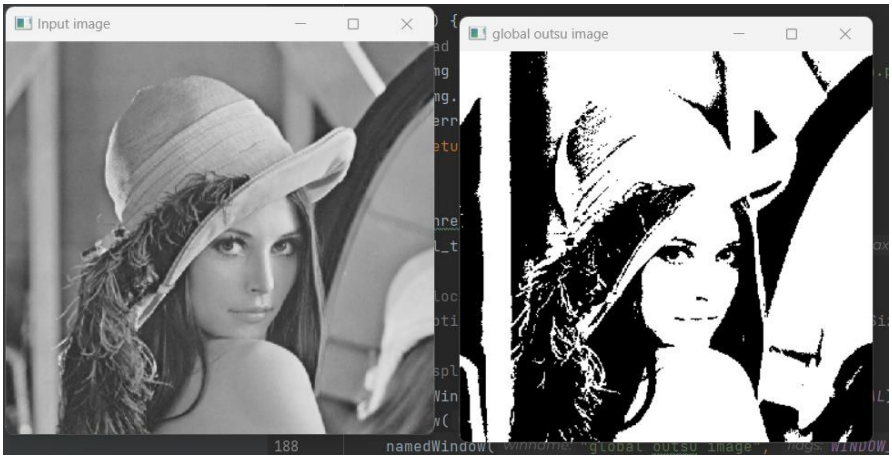
4. **Compute the mean intensity value of the entire image:** The mean intensity value of the image is computed as the weighted sum of the intensity values, where the weights are the probabilities from the normalized histogram.

5. **Iterate through all possible threshold values:** The algorithm iterates through all possible threshold values and computes the between-class variance for each threshold value. The between-class variance is defined as the weighted sum of the variances of the two classes (foreground and background), where the weights are the probabilities of the classes.

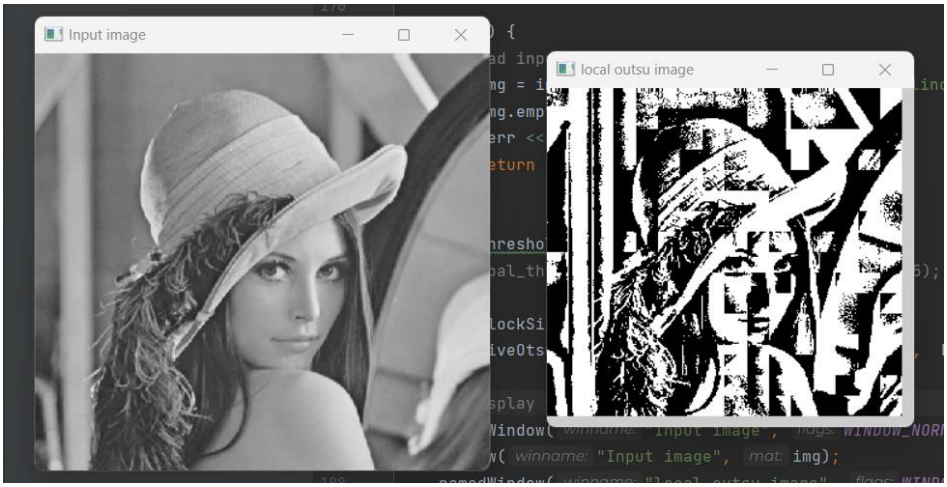
6. **Select the threshold value that maximizes the between-class variance:** The threshold value that maximizes the between-class variance is selected as the threshold value for segmenting the image. All pixels with intensities greater than or equal to the threshold value are classified as foreground pixels, and all pixels with intensities less than the threshold value are classified as background pixels.

Output examples:

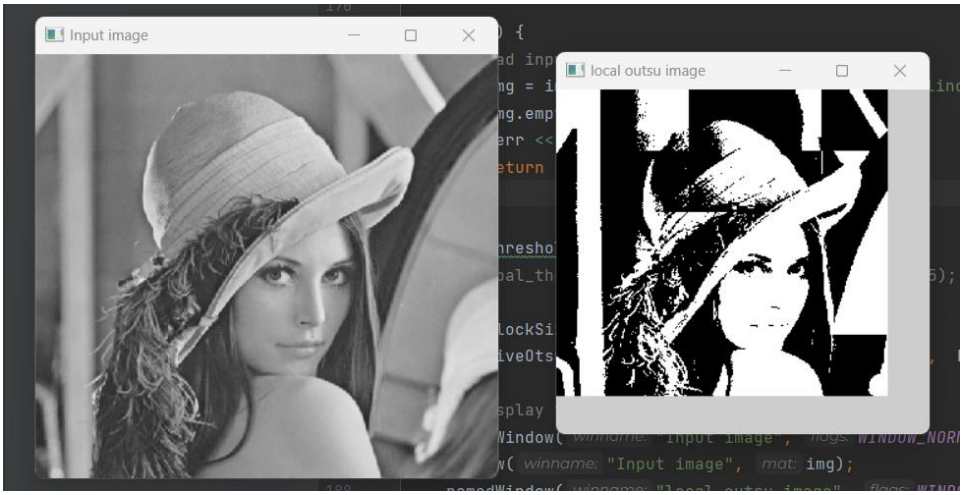
- Global Otsu thresholding :



- Local Otsu thresholding : (at blocksize =31)



- Local Otsu thresholding : (at blocksize =91)



1.3. Spectral Thresholding

Spectral thresholding is a technique used in hyperspectral image processing for image segmentation. It involves selecting a threshold value to separate an image into two or more classes based on the spectral signatures of the image pixels. It is a simple and efficient technique for hyperspectral image segmentation, especially when the spectral bands that contain the most discriminative information are known. However, the selection of the threshold values can be subjective, and the technique may not be suitable for images with high noise levels or complex spatial patterns.

Algorithm steps:

1. **Compute the spectral signature for each pixel in the image:** The spectral signature is a vector that represents the intensity values of the pixel in each spectral band.

2. **Compute the mean and standard deviation of each spectral band:** The mean and standard deviation of the pixel intensities in each spectral band are computed.

3. **Select the bands to use for segmentation:** The user can select the spectral bands to use for segmentation based on the application requirements.

4. **Compute the threshold values for each selected band:** The threshold value for each selected spectral band is computed as a function of the mean and standard deviation of the pixel intensities in that band.

5. **Classify the pixels based on the threshold values:** Each pixel in the image is classified into one of the classes based on its spectral signature and the computed threshold values

6. **Refine the segmentation:** The segmentation can be refined by post-processing techniques such as morphological operations or edge detection.

Output examples:

2. k-means Clustering:

The k-means clustering algorithm partitions input data into k clusters represented by centroids. The algorithm randomly selects k data points as initial centroids, assigns each data point to the nearest centroid, and computes the new centroid as the mean of all data points in the cluster. The algorithm repeats these two steps until convergence or the maximum number of iterations is reached. In image segmentation, the input image is converted to a matrix of pixel values, and the

algorithm initializes centroids by randomly selecting k pixels. It assigns each pixel to the nearest centroid and computes the new centroid of each cluster. The algorithm repeats these steps until convergence or the maximum number of iterations is reached.

Algorithm steps:

1. **Initialize k centroids:** Randomly select k data points as initial centroids.

2. **Assign each data point to the nearest centroid:** Calculate the distance between each data point and each centroid. Assign each data point to the nearest centroid.

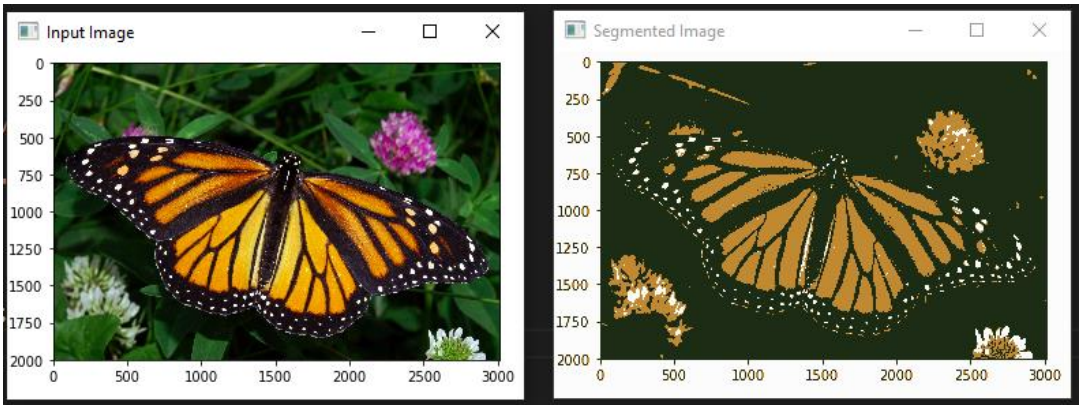
3. **Compute the new centroid of each cluster:** Calculate the mean of all data points assigned to each centroid. The new centroid becomes the center of the cluster.

4. **Repeat steps 2 and 3 until convergence:** Assign each data point to the nearest centroid and compute the new centroid of each cluster. Repeat these steps until the centroids no longer change significantly or until the maximum number of iterations is reached.

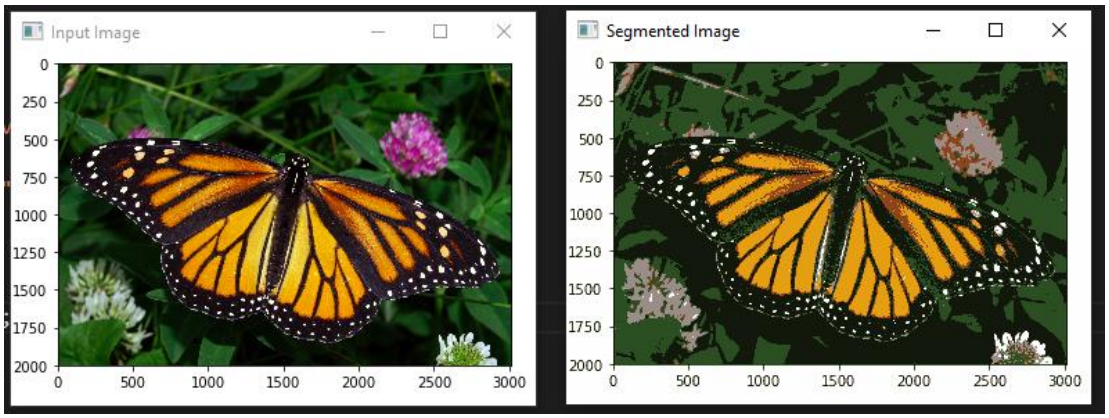
5. **Output the clusters:** Once the algorithm converges, the final k clusters are formed, each represented by its centroid.

Output examples:

- Number of clusters = 3



- Number of clusters = 6



3. Region Growing Segmentation

Region Growing Segmentation is an algorithm for image segmentation that groups pixels with similar properties (e.g., color, intensity, texture) into regions or segments. The algorithm starts with a seed pixel, which serves as the initial region, and then adds adjacent pixels to the region based on some similarity criteria. The process continues until no more pixels can be added to the region.

Algorithm steps:

1. **Choose a seed pixel:** The algorithm starts with a seed pixel that belongs to the region of interest. The seed pixel can be chosen randomly or based on prior knowledge of the region.

2. **Define a similarity criterion:** A similarity criterion is defined to determine which pixels should be added to the region. This criterion is based on some image properties, such as color, intensity, texture, or gradient.

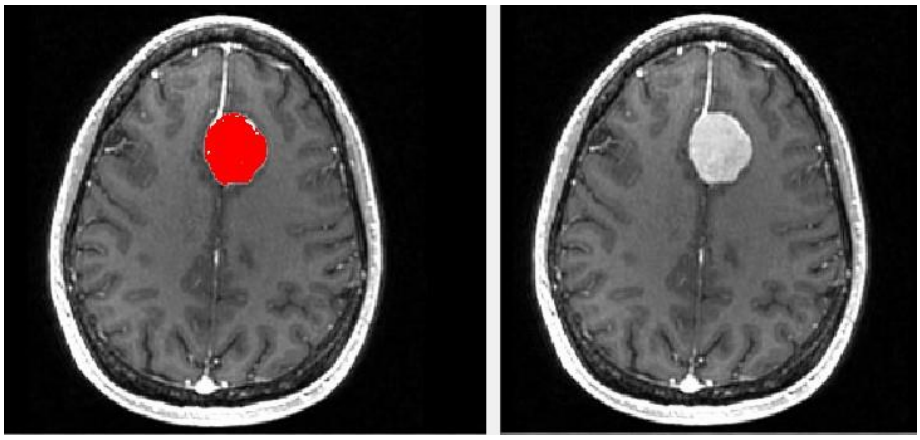
3. **Find adjacent pixels:** The algorithm searches for adjacent pixels to the seed pixel that meet the similarity criterion. Adjacent pixels are typically defined as pixels that share a common boundary or are within a certain distance of the seed pixel.

4. **Add adjacent pixels to the region:** The algorithm adds adjacent pixels that meet the similarity criterion to the region. The newly added pixels become seed pixels for the next iteration.

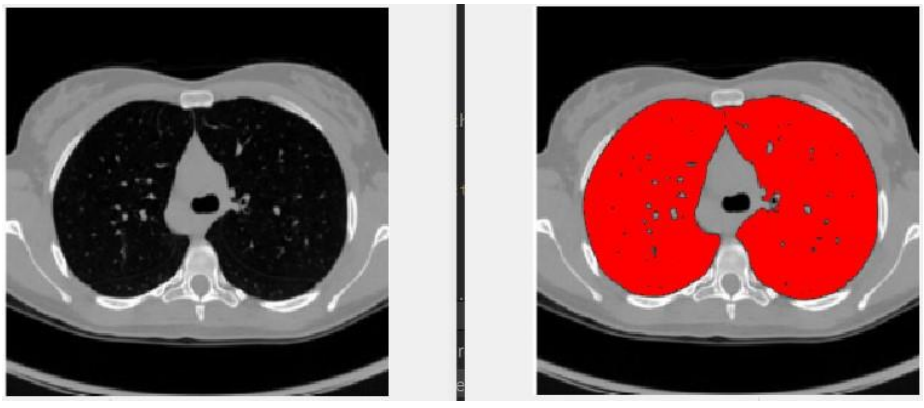
5. **Repeat steps 3 and 4:** The algorithm continues to search for adjacent pixels and adds them to the region until no more pixels can be added. This is typically determined by a stopping criterion, such as reaching a maximum region size or when the similarity criterion is no longer met.
6. **Output the segmented image:** Once the region is complete, the algorithm assigns a unique label to each pixel in the region. The algorithm can be applied to other regions of interest in the image, resulting in multiple segmented regions.

Output examples:

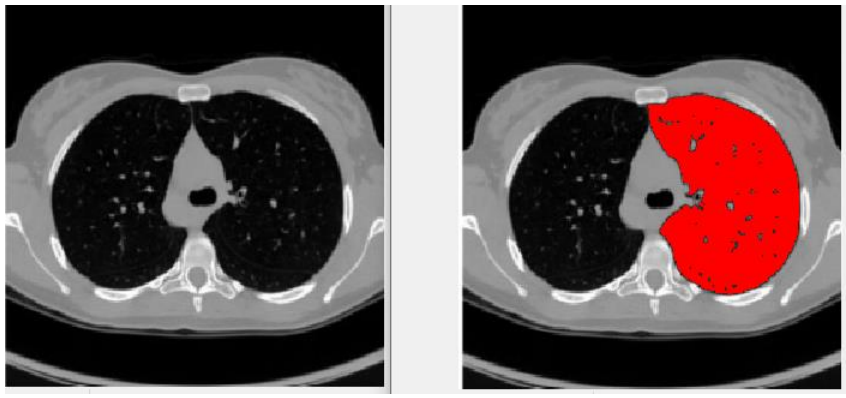
- Grayscale regoin growing at threshold 40 and seedpoint(169, 101)
-



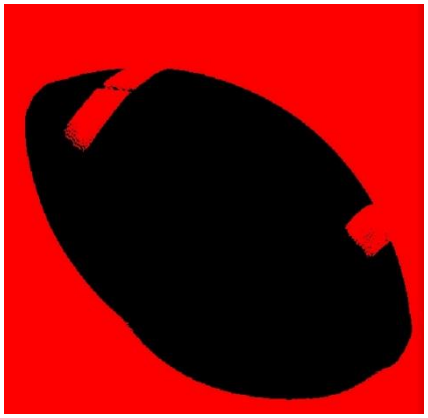
- Grayscale regoin growing at threshold 30 and seedpoint(200,133)

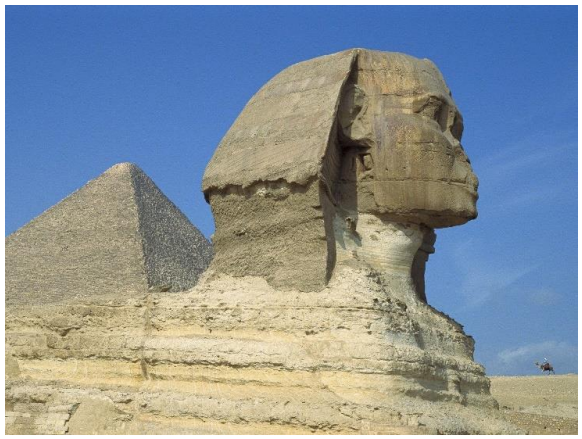


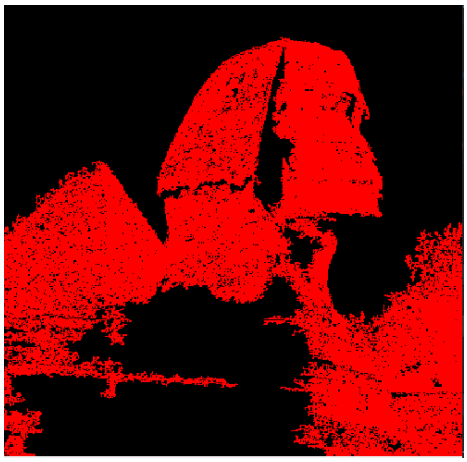
- Grayscale regoin growing at threshold 20 and seedpoint(200,133)



- Colored images







4. Agglomerative Clustering

Agglomerative clustering is a hierarchical clustering algorithm that starts with each data point as a separate cluster and then iteratively merges the closest clusters until a stopping criterion is met.

In the context of image segmentation, agglomerative clustering can be used to group pixels with similar characteristics into clusters, which can then be assigned to different objects in the image.

Algorithm steps:

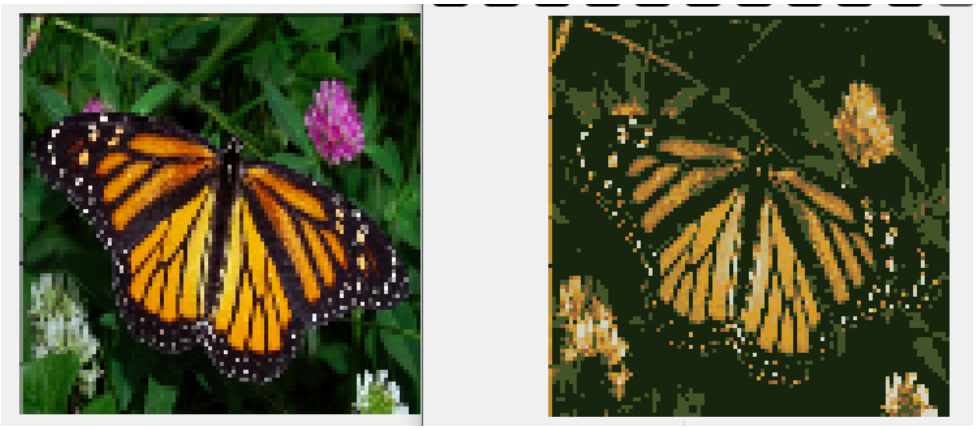
1. Convert the image to grayscale.
2. Represent each pixel as a data point in a high-dimensional space, where each dimension corresponds to a feature of the pixel (e.g., intensity, color, texture).
3. Initialize each pixel as a separate cluster.
4. Compute the pairwise distance between all clusters using a distance metric such as Euclidean distance or Manhattan distance.
5. Merge the two closest clusters into a single cluster.

6. Repeat steps 4-5 until the desired number of clusters is reached or until a stopping criterion is met (e.g., a minimum distance threshold).

Output examples:

- At 50 clusters:

- At 15 clusters:



- At 100 clusters:

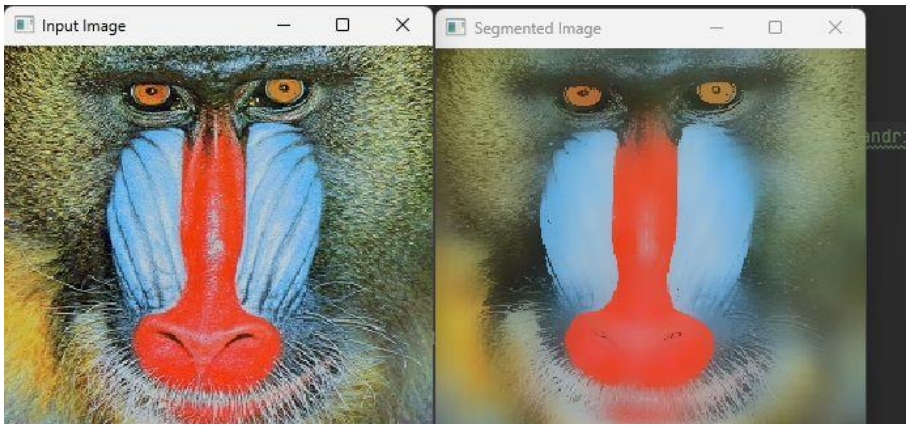
5. Mean Shift Segmentation

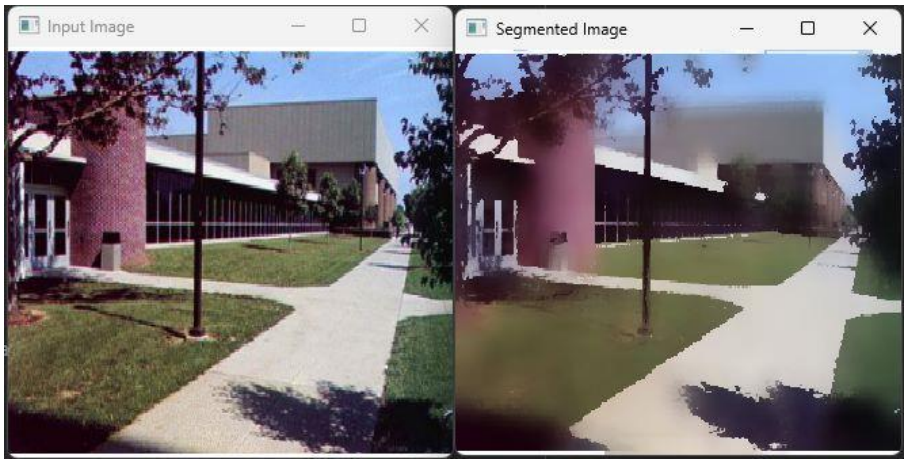
Image segmentation is the process of dividing an image into multiple homogeneous regions based on their Similarities, differences or even characteristics such as color, texture, shape, or brightness, in order to simplify or change the representation of an image into something more meaningful and easier model for analysis. Here, each pixel is labeled, where the pixels belonging to the same category have a common label assigned to them.

Algorithm steps:

1. Define a window around each pixel in the image. The size of the window is determined by a bandwidth parameter.
2. Compute the color similarity between the pixels within the window using a distance metric such as the Euclidean distance.
3. Compute the mean color of the pixels within the window.
4. Shift the window towards the direction of the maximum increase in the density of the data points. The direction of the shift is given by the gradient of the density function
5. Repeat steps 2 to 4 until convergence, i.e., until the window no longer moves.
6. Assign a unique label to each segment based on the pixels that converge to the same window.
7. Repeat steps 1 to 6 for all pixels in the image.
8. Output the final segmentation of the image.

Output examples:




Input Image
Segmented Image