Computer Vision

“Task 4”
Image Segmentation Studio

Introduction

In this report, a Variety of algorithms implemented in C++ using Qt6 Creator will be explained. The algorithms are Optimal Thresholding - Otsu Thresholding - Spectral Thresholding - Mapping RGB to LUV - k-means Segmentation - Region Growing Segmentation - Agglomerative and Mean Shift segmentation.

Team Members:

Magdy Nasr - Ahmed Emad - Youssef Kadry - Mohab Ghobashy - Mohammed Mostafa

Team No. 3

Tab 1 - Segmentation

Page Content

- k-means Segmentation
- Region Growing Segmentation
- Agglomerative Segmentation
- Mean shift Segmentation

K-means Segmentation Algorithm:

1) Initialize the number of clusters K and the maximum number of iterations.
2) Randomly select K data points from the input data as the initial centroids.
3) Assign each data point to the closest centroid using the Euclidean distance as the similarity metric.
4) Calculate the mean of all data points assigned to each centroid, and update the centroid's position as the new mean.
5) Repeat steps 3 and 4 until the maximum number of iterations is reached or the centroids' positions do not change significantly between iterations.
6) Return the final cluster assignments and centroid positions.

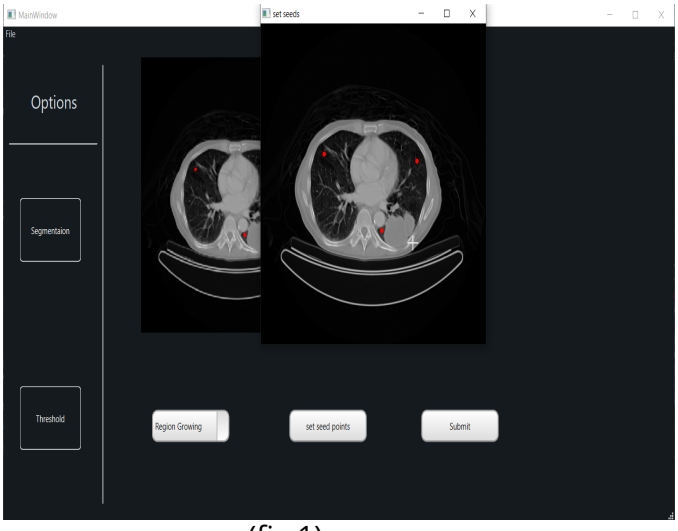
Example:

When using k-means segmentation with k=5, the output image will have 5 distinct segments, each represented by a unique color.

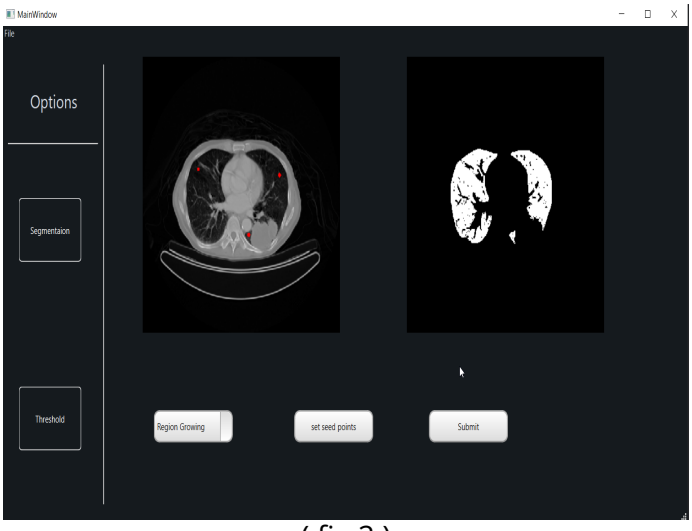
Region Growing Segmentation Algorithm:

1) Initialize a boolean matrix to keep track of visited pixels in the image.
2) Initialize a list of seed points that represent the starting points for the region growing process.
3) While the list of seed points is not empty, take the last point from the list.
4) Retrieve the pixel value of this point from the input image.
5) Mark this point as visited in the boolean matrix.
6) Check each neighboring pixel of this point within a specified window.
7) For each unvisited neighboring pixel that is within the specified tolerance range of this point's pixel value, mark it as visited in the boolean matrix and add it to the list of seed points.
8) Repeat steps 3-7 until the list of seed points is empty.
9) Return the boolean matrix, which contains the segmented region.

Example:



(fig.1)



(fig.2)

(fig.1) We select the seed points on the input image .

fig.2) Comparing input to output result .

Agglomerative Segmentation Algorithm:

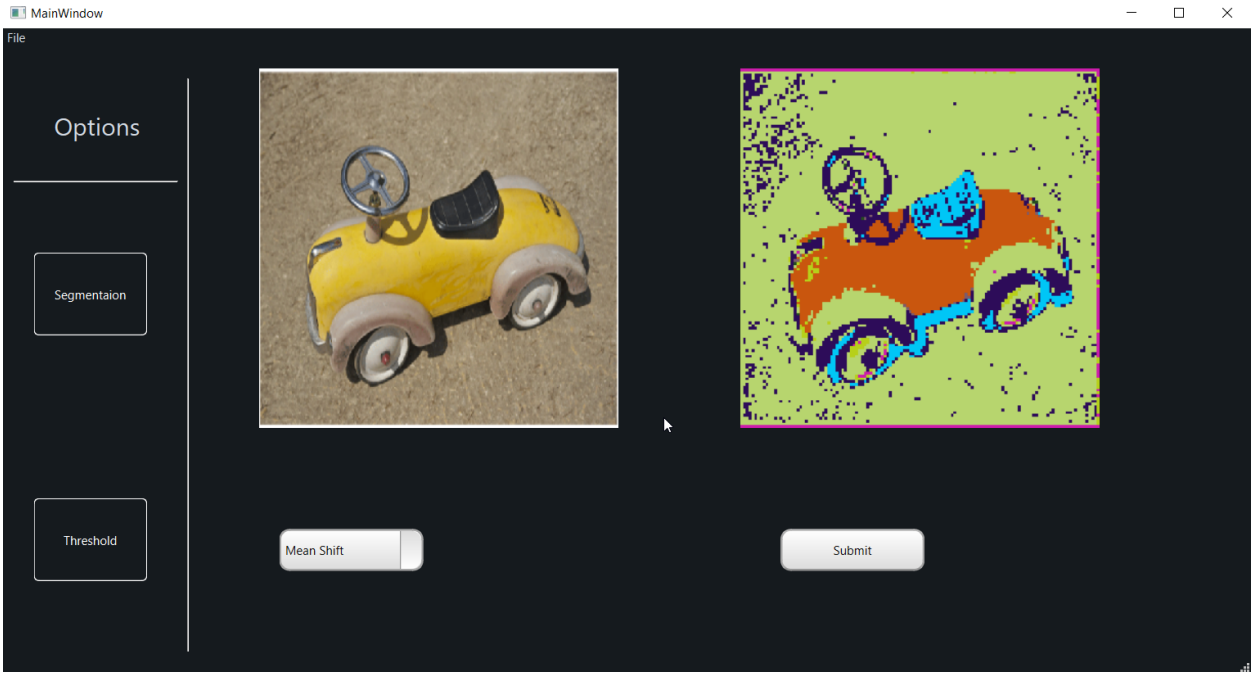
1) Initialize clusters with each pixel of the picture as a single cluster.
2) Calculate Euclidean distance between all clusters.
3) Find the pair of clusters with the minimum distance.
4) Merge the previous 2 clusters together and take the mean value of them.
5) Calculate distances again and repeat the previous step until the desired number of clusters is reached.
6) Apply Segmentation on the picture based on the obtained clusters.

Example:

Mean Shift Segmentation Algorithm:

1) Convert the input image from BGR color space to LAB color space.
2) Create an empty list of pixels to track and an empty list of segments.
3) Iterate over each pixel in the LAB image and add it to the list of pixels to track.
4) While there are still unassigned pixels:
    A) Initialize the mean vector and the total weight to zero.
    B) Find all pixels within the bandwidth.
    C) For each neighbor, calculate its weight based on its distance from the mean vector.
    D) Calculate the mean vector of the neighbors based on their weights.
    E) Assign all neighboring pixels within the bandwidth to the same segment.
    F) Remove the assigned pixels from the list of pixels to track.
    G) Add the segment to the list of segments.
5) Create a new image with the same dimensions as the input image.
6) Assign a random color to each segment.
7) Return the segmented image.

Example:



Tab 2 - Thresholding

Page Content

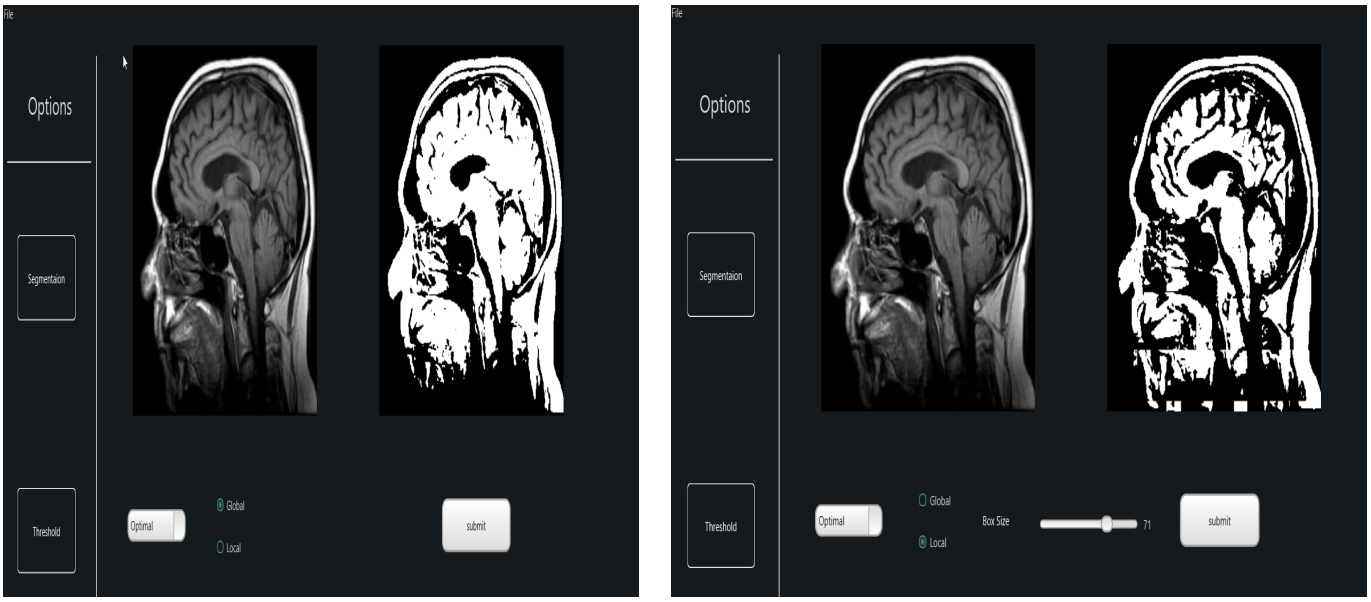
- Optimal Thresholding.
- Otsu Thresholding.
- Spectral Thresholding.

Optimal Thresholding Algorithm:

1. Convert the grayscale image to a histogram: A histogram is a plot of the frequency distribution of pixel intensity values in the image. This step involves counting the number of pixels at each intensity value and plotting them.
2. Calculate the total number of pixels in the image: This is the sum of the pixel count in the histogram.
3. Calculate the cumulative sum of the histogram: This is the sum of the pixel count for each intensity value up to that point.
4. Calculate the cumulative mean intensity values: This is the sum of the intensity values for each intensity value up to that point divided by the cumulative sum.
5. Calculate the global mean intensity value: This is the sum of the intensity values of all the pixels in the image divided by the total number of pixels.
6. Calculate the between-class variance for each possible threshold: This is a measure of the separation between the two classes of pixels (black and white) that would result from using a particular threshold value. It is calculated using the formula:
7. (global_mean * cumulative_sum[i] - cumulative_mean[i])^2 / (cumulative_sum[i] * (1 - cumulative_sum[i]))

8. Choose the threshold that maximizes the between-class variance: The optimal threshold is the one that maximizes the between-class variance value calculated in step 6.
9. Convert the grayscale image to a binary image: This involves assigning a black or white pixel value to each pixel in the image based on whether its intensity value is below or above the optimal threshold value, respectively.
10. Output the binary image

Example:

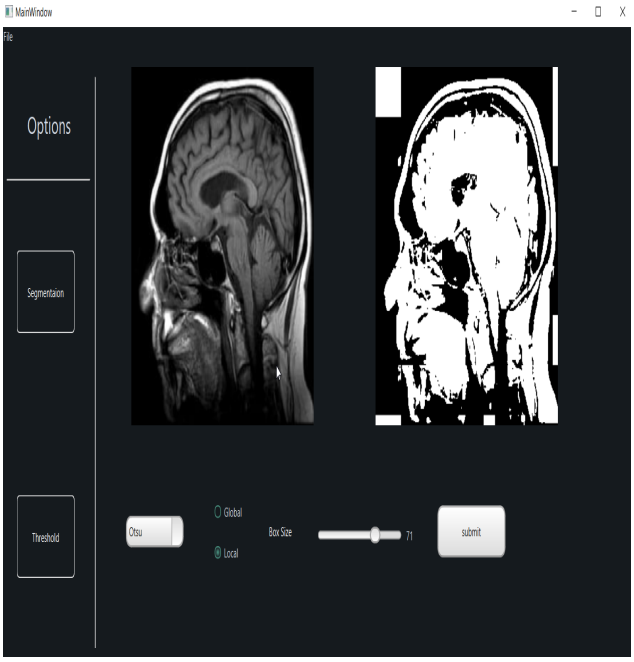


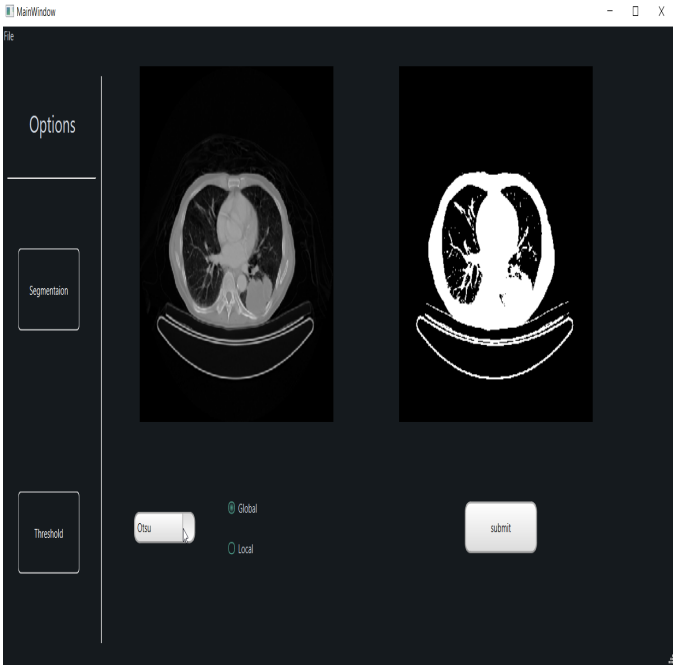
After applying optimal thresholding to an image, the resulting binary image highlights the important features of the original image by separating the foreground from the background. The optimal thresholding algorithm selects the threshold value that maximizes the separation between the two classes of pixels, making it a useful technique for enhancing image contrast and improving image segmentation. The resulting binary image can be used for further analysis, such as object recognition, tracking, and classification. Overall, optimal thresholding is a powerful technique for image processing that can help extract meaningful information from images.

Otsu Thresholding Algorithm:

1. Compute the histogram of the grayscale image: The histogram shows the frequency distribution of pixel intensities in the image.
2. Compute the total number of pixels in the image.
3. Compute the normalized histogram by dividing each pixel count by the total number of pixels.
4. Compute the cumulative sum of the normalized histogram and the cumulative sum of normalized intensities.
5. Compute the mean intensity value of the image.
6. Iterate over all possible threshold values and compute the between-class variance for each threshold value:
   a. Compute the weight and mean intensity value of the foreground and background pixels for the given threshold value.
   b. Compute the between-class variance using the formula:
   ((total_mean * weight_foreground - mean_foreground)^2) / (weight_foreground * (1 - weight_foreground)) +
7. ((total_mean * weight_background - mean_background)^2) / (weight_background * (1 - weight_background))
   where total_mean is the mean intensity value of the entire image.
8. Select the threshold value that maximizes the between-class variance computed in step 6.
9. Convert the grayscale image to a binary image by assigning each pixel a value of 0 (black) if its intensity value is less than the optimal threshold value or a value of 1 (white) if its intensity value is greater than or equal to the optimal threshold value.
10. Output the binary image.

Example:





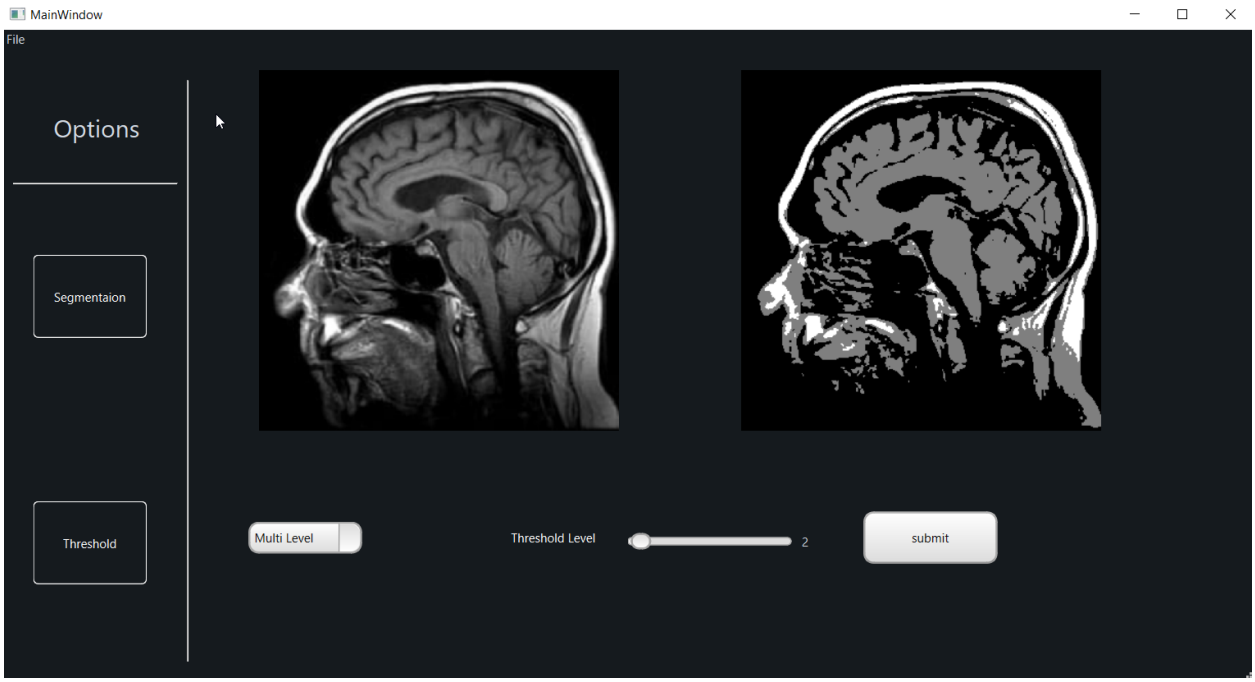
Applying Otsu thresholding to an image results in a binary image where the foreground and background pixels are separated based on an optimal threshold value that maximizes the between-class variance. Otsu thresholding is a powerful technique for image segmentation and can effectively separate objects from the background. The resulting binary image highlights the important features of the original image and can be used for further analysis, such as object recognition, tracking, and classification. Overall, Otsu thresholding is a useful technique for enhancing image contrast and extracting meaningful information from images.

Spectral Thresholding Algorithm:

1. Read the multiband image: A multiband image contains multiple spectral bands, each representing the same area of the scene but with different wavelengths.
2. Compute the covariance matrix of the image: This matrix contains the covariances between the different spectral bands and can be computed using the formula:
3. C = (1/N) * ((X - M) * (X - M)^T)
4. where X is an NxM matrix representing the image (N pixels and M spectral bands), M is a 1xM vector representing the mean of the image, and ^T denotes matrix transpose.

5. Compute the eigenvalues and eigenvectors of the covariance matrix: This can be done using standard linear algebra techniques. The eigenvectors represent the principal components of the image, and the eigenvalues represent the amount of variance explained by each principal component.
6. Compute the projection matrix: This matrix contains the eigenvectors corresponding to the largest eigenvalues and is used to project the multiband image onto a lower-dimensional space.
7. Project the multiband image onto a lower-dimensional space using the projection matrix: This results in a new image with fewer spectral bands.
8. Compute the histogram of the lower-dimensional image: This histogram shows the frequency distribution of pixel intensities in the new image.
9. Compute the optimal threshold value using a threshold selection method such as Otsu's thresholding algorithm.
10. Apply the threshold to the original multiband image: Pixels with values above the threshold are classified as one spectral class, and pixels with values below the threshold are classified as another spectral class.
11. Output the binary image.

Example:



After applying spectral thresholding to a multiband image, the resulting binary image highlights the spectral features that are most important for distinguishing between different classes of objects or materials. Spectral thresholding selects the optimal threshold value that maximizes the separability between different spectral classes, making it a powerful technique for image segmentation and object recognition. The resulting binary image can be used for further analysis, such as classification and mapping of land cover or crop health. Overall, spectral thresholding is a useful technique for extracting meaningful information from multispectral images and improving the accuracy of image classification and analysis.