CENG466-Fundamentals of Image Processing Report IV

Sena Nur Şengül
Computer Engineering
Middle East Technical University
Ankara, Turkey
e231049@metu.edu.tr

I. Introduction

In this assignment, the goal was to learn about basic techniques for manipulating and analyzing images using morphological image processing and image segmentation algorithms. These techniques allow us to modify and analyze the shape and structure of objects in an image and to separate different objects or regions within an image. By understanding and applying these techniques, we can better understand and analyze images and extract useful information from them.I used skimage,opencv,matplotlib,sklearn libraries.

II. OBJECT COUNTING

A. Introduction

Object counting is the process of determining the number of objects present in an image or video. .
First part of assignment, I am required to implement object counting algorithm on A1.png, A2.png,A3.png with only morphological techniques. Our pseudo-code of my implementation 1)Convert the input image to grayscale
2)Threshold the image to create a black and white image
3)Perform morphological opening on the image to remove small noise
4)Perform morphological closing on the image to fill in small holes
5)Count connected component with cv2.connectedComponent
6)Perform morphological dilation on the image to make the objects more distinct
7)Find the contours in the image using the cv2.findContours function
8)Create a blank image with the same dimensions as the input image
9)Draw the contours onto the blank image using the cv2.drawContours function
10,9Draw the contours onto the original image using the cv2.drawContours function
11)Display the image with the contours drawn on it using the plt.imshow function
12)Print the number of objects in the image using the len function on the contours variable
13)Return the image with the contours drawn on it

B. Implementation

My implementation like this: First, the image is converted to grayscale using cv2.cvtColor. This is done because the thresholding operation works better with grayscale images as it is easier to distinguish between different shades of gray.

Next, the image is thresholded using cv2.threshold. This creates a black and white image where the objects of interest are white and the background is black. The cv2.THRESH_BINARY INV flag inverts the image so that the objects are black and the background is white. The cv2.THRESH OTSU flag is used to automatically calculate the threshold value based on the image's intensity values.

After thresholding, the image is passed through a morphological opening operation using cv2.morphologyEx. This removes small noise and isolated pixels from the image.

The image is then passed through a morphological closing operation using cv2.morphologyEx. This fills in small holes in the objects of interest.

The image is then dilated using cv2.dilate to make the objects more distinct.

Finally, the contours of the objects are found using cv2.findContours and the number of objects is counted. The contours are also drawn on the original image using cv2.drawContours and the resulting image is returned.

The purpose of the final line of code, plt.imshow(black_contours) and plt.show(), is to display the image with the contours drawn on it. The print statement outputs the number of objects detected in the image.

C. Results

I showed results two ways which is represent objects as a black on white background and draw boundaries on original image.I think my codes have problems :
$\longrightarrow$The threshold values for the threshold function may not be appropriate for the image, resulting in either too many or too few objects being detected.
$\longrightarrow$The kernel size for the morphological operations may not be appropriate for the size of the objects in the image, resulting in either too much or too little noise removal.
$\longrightarrow$The number of iterations for the morphological operations may not be sufficient to effectively remove noise or fill in small holes, resulting in inaccurate object counting.

The contours in the image may not be accurately detected, resulting in an incorrect count of objects.
The image with contours may not be displayed correctly due to an issue with the display method or the image file format
I don't want to apply different function to every image, It seems to me a correct solution way,we have three images but in those homeworks we aim our algorithm should apply all images.So, I have some issues with codes but generally it gives general boundaries of images.

Fig. 1. A1.png

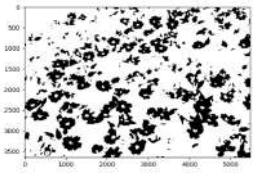

Fig. 2. Binary image

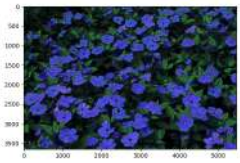

Fig. 3. Draw green line around objects

Fig. 4. A2.png

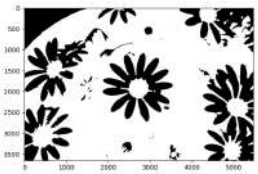

Fig. 5. Binary image

Fig. 6. Draw green line around objects

Fig. 7. A3.png

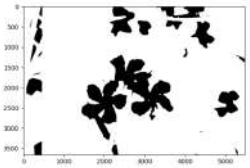

Fig. 8. Binary image

Fig. 9. Draw green line around objects

III. SEGMENTATION

A. Introduction

In this part, I am required to implement n-cut and meanshift segmentations algorithms to given images. Image segmentation is the process of dividing an image into multiple segments, or regions, each of which corresponds to a different object or background in the image. The goal of image segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

IV. MEAN-SHIFT SEGMENTATIONS

A. Introduction

Mean shift segmentation is a technique used in image processing to segment an image into distinct regions or clusters based on color or intensity similarity. It involves iteratively shifting the mean value of each cluster towards the highest density of pixels within the cluster, until convergence is reached. The resulting clusters are considered to be the segments of the image. This method is often used for real-time image segmentation because it is relatively fast and simple to implement.

B. Implementation

My code performs mean shift segmentation on an image img and produces a segmentation map and a boundary overlay of the resulting segments.

The mean shift segmentation is performed using the pyrMeanShiftFiltering function from the OpenCV library, which uses the mean shift algorithm to cluster pixels in the image based on their color and spatial proximity. The function takes two parameters: sp (spatial radius) and sr (color radius), which control the size of the clusters and the range of colors that are considered to be in the same cluster.

2)The resulting segmented image is then compared to the original image pixel by pixel. For each pixel, if the pixel is not in the same cluster as the original image, it is colored black in the segmentation map. Otherwise, it is colored white.

3)The boundary overlay is produced by applying the Canny edge detection algorithm to the segmented image using the Canny function from OpenCV. This generates a black and white image where the white pixels represent the edges between different segments in the image.
4)I applied this algorithm for tree relationship structure :
⟶The function first initializes an empty list called tree structure which will be used to store the relationship structure of the tree. It then iterates over each pixel in the image, checking if it is a branch (black pixel) or leaf (white pixel). If the pixel is a branch, the function initializes an empty list called branch structure to store the relationship structure for the current branch.

⟶ Next, the function checks if there are any leaves surrounding the current branch. If there is a leaf to the left, right, above, or below the branch, the function adds the coordinates of that leaf to the branch structure list. Once it has checked all the surrounding pixels, the function adds the branch_structure list for the current branch to the tree_structure list. Finally, the function returns the tree structure list, which should contain the relationship structure of all the branches in the tree.

⟶ Finally, the code displays the original image, the segmentation map, the boundary overlay, and the output of the tree relationshipstructure function in a figure with four subplots.
5)Lastly I wroted algorithm for performing region adjacency graph (RAG) analysis on an image. It first uses the slic segmentation algorithm to divide the image into regions, and then creates a graph representing the relationships between these regions based on their mean color values. The resulting graph is then plotted using two different visualization techniques.

C. Results

Fig. 10. B1.jpg

Fig. 11. B2.jpg

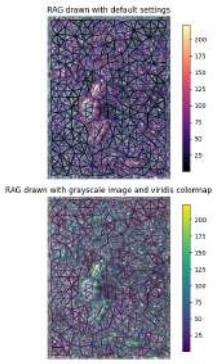

Fig. 12. ragB1.png

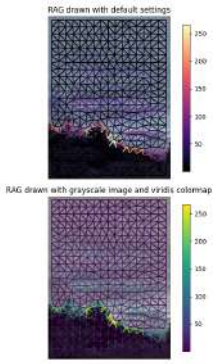

Fig. 13. ragB2.png

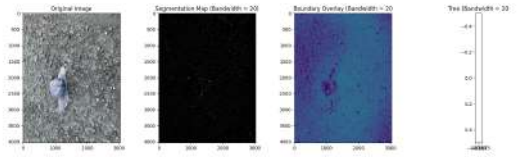

Fig. 14. B1 algorithm meanshift 15.png

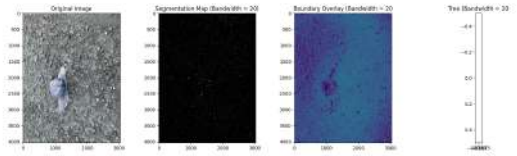

Fig. 15. B1 algorithm meanshift 20.png

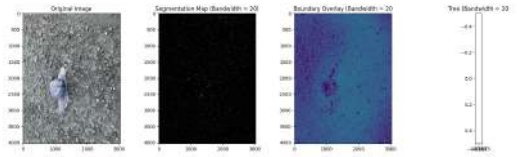

Fig. 16. B1 algorithm meanshift 25.png

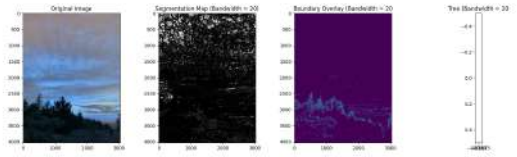

Fig. 17. B2 algorithm meanshift 15.png

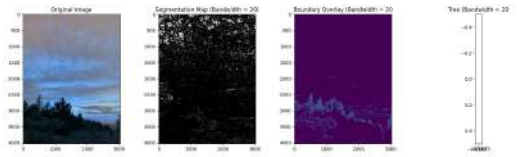

Fig. 18. B2 algorithm meanshift 20.png

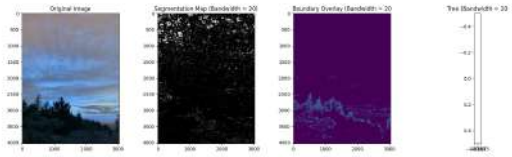

Fig. 19. B2 algorithm meanshift 25.png

Fig. 20. B3.jpg

Fig. 21. B4.jpg

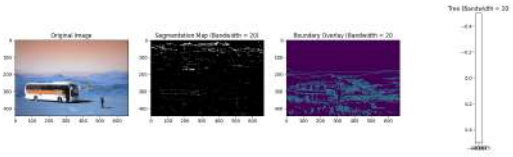

Fig. 22. B3 algorithm meanshift 15.png

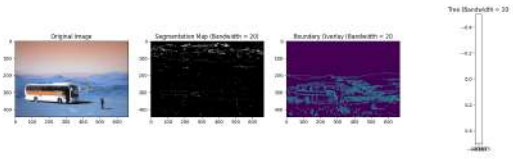

Fig. 23. B3 algorithm meanshift 20.png

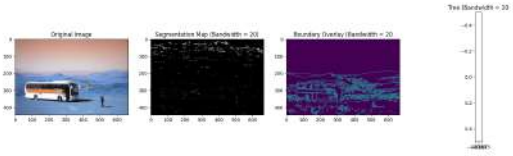

Fig. 24. B3 algorithm meanshift 25.png

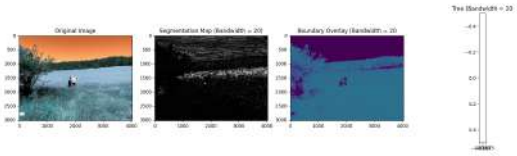

Fig. 25. B4 algorithm meanshift 15.png

I took parameter as a spatial radius in mean shifting algorithm.I chosed 15,20,25. I observed larger spatial radius

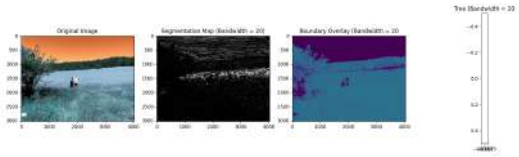

Fig. 26. B4 algorithm meanshift 20.png

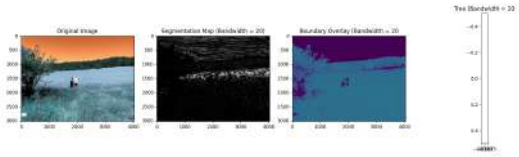

Fig. 27. B4 algorithm meanshift 25.png

will result in a coarser segmentation, while a smaller spatial radius will result in a finer segmentation with more detail. In this assignment, my computer had some issue with installing sklearn so I did homework with my friend's computer.Compiling my code last very long and Tree Structure graph does not work. Region Adjacency Graph is not suitable for our homework,maybe.Also I could not put RAG to others, it give error so I put seperately.

V. N-CUT SEGMENTATIONS

A. Introduction

N-cut segmentations are a method of dividing an image into multiple segments or regions using graph theory. The goal is to minimize the number of edges that must be cut in order to separate the image into distinct segments. This is achieved by finding a set of cuts in the graph that minimize the overall value of the N-cut, which is defined as the sum of the weights of the edges that are cut. N-cut segmentations are often used in image processing and computer vision applications, such as object recognition and image segmentation.

B. Implementation

1)Uses the slic function from the segmentation module to perform SLIC (Simple Linear Iterative Clustering) segmentation on the image, with a desired number of segments equal to 250, compactness equal to 10, and sigma equal to 1.

2)Creates a subplot with the original image using the imshow function from matplotlib.pyplot.

3)Creates a subplot with a segmentation map of the image, using the label2rgb function from the skimage.color module to convert the segments into a color map.

4)Creates a subplot with a boundary overlay of the image, using the mark boundaries function from the skimage.segmentation module.

5)Calls the tree relationship structure function on the image and displays the output in a fourth subplot as mean shift segmentation algorithm.

6)Calls the find rag function on the image as mean shift segmentation algorithm.

C. Results

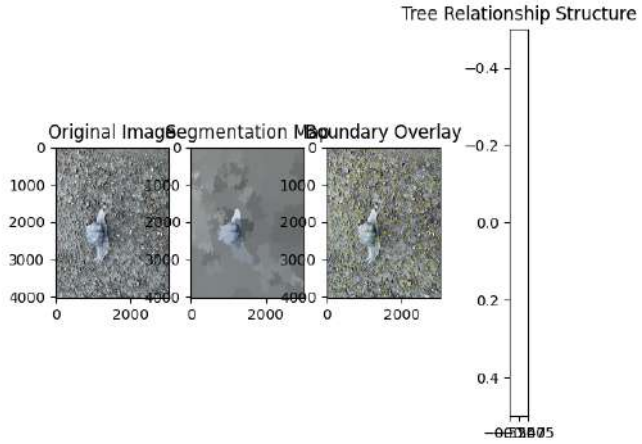

Fig. 28. B1 algorithm ncut 1.png

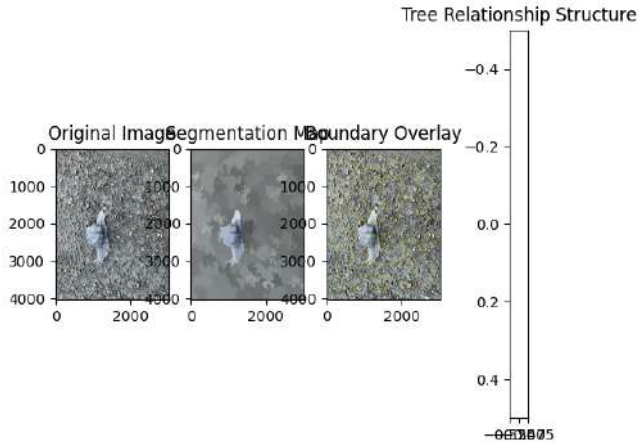

Fig. 29. B1 algorithm ncut 2.png

As before , tree structure does not work and RAG is same. Parameter is sigma in my implementation.The relationship between cut segmentation and sigma in the SLIC function is that sigma controls the compactness of the cut segments or superpixels. A larger sigma value results in more compact segments, while a smaller sigma value results in more dispersed segments.

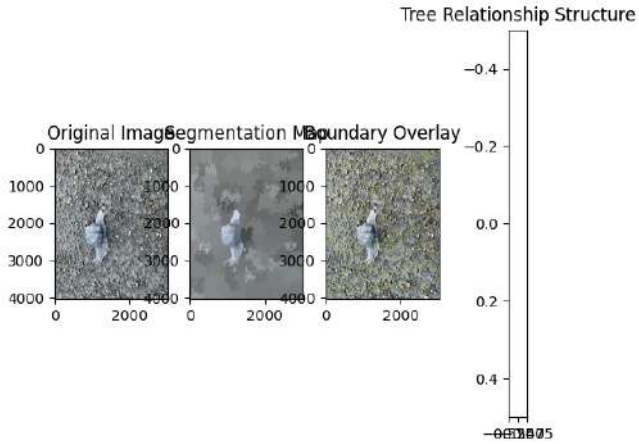

Fig. 30. B1 algorithm ncut 3.png

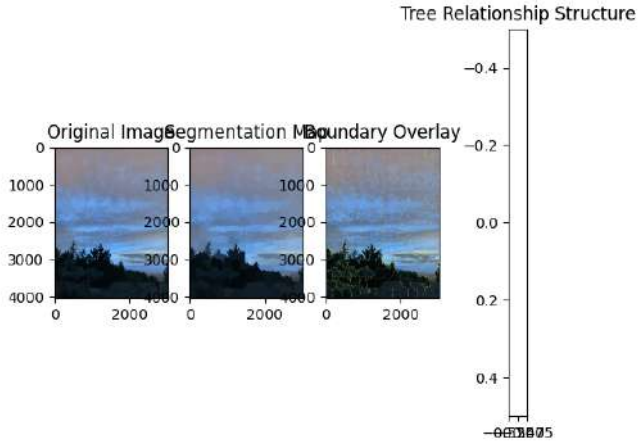

Fig. 31. B2 algorithm ncut 1.png

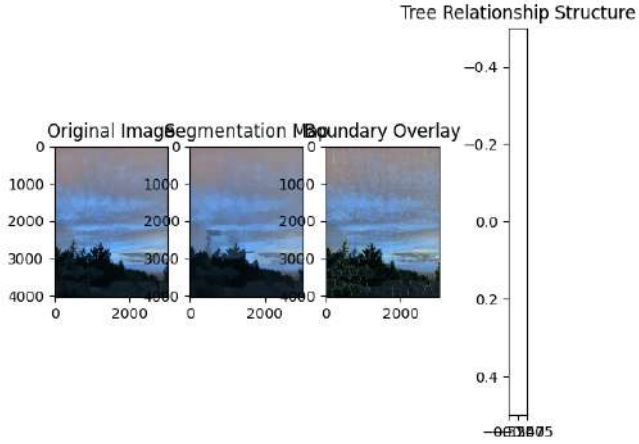

Fig. 32. B2 algorithm ncut 2.png

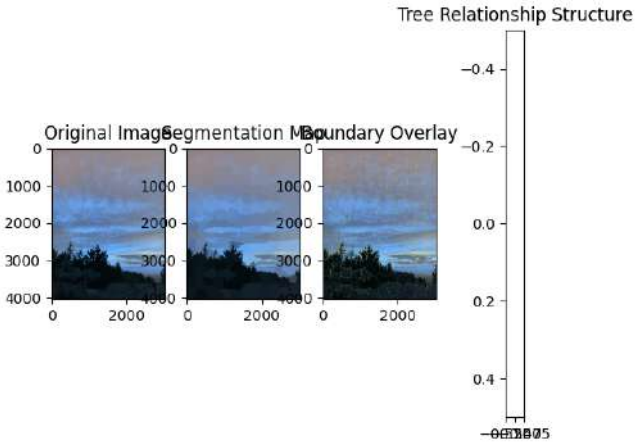

Fig. 33. B2 algorithm ncut 3.png

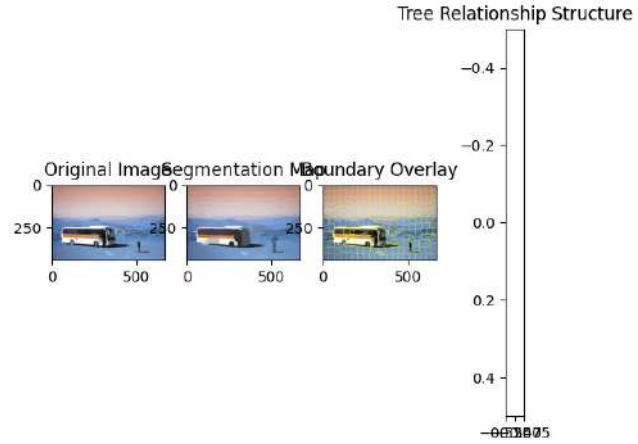

Fig. 34. B3 algorithm ncut 1.png

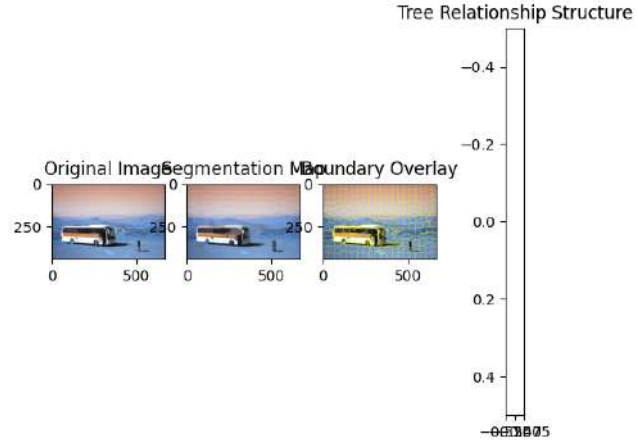

Fig. 35. B3 algorithm ncut 2.png

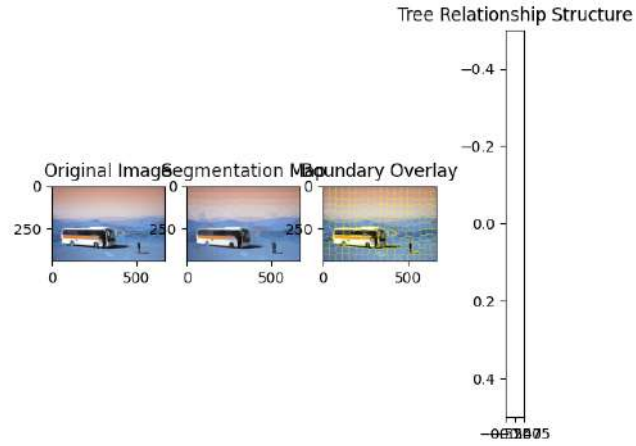

Fig. 36. B3 algorithm ncut 3.png

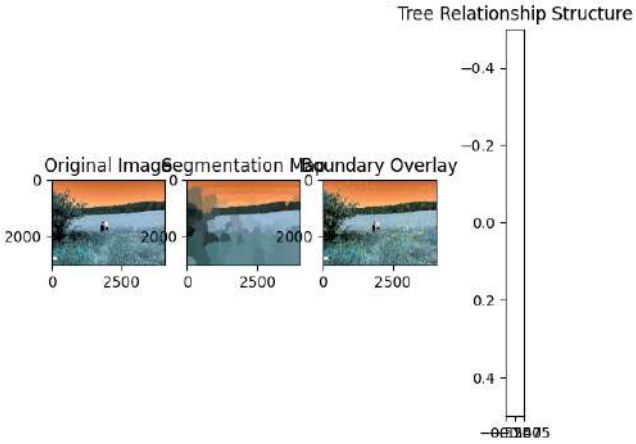

Fig. 37. B4 algorithm ncut 1.png

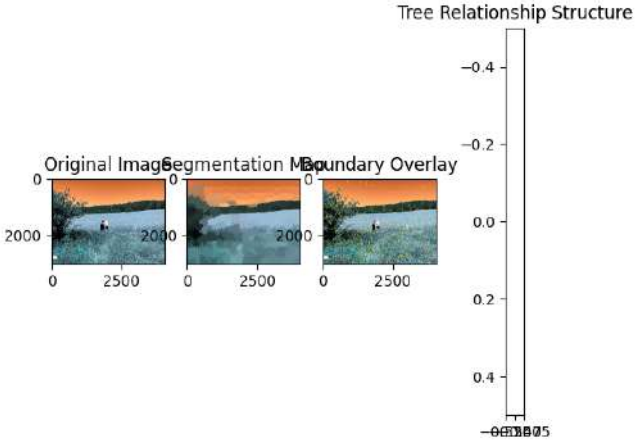

Fig. 38. B4 algorithm ncut 2.png

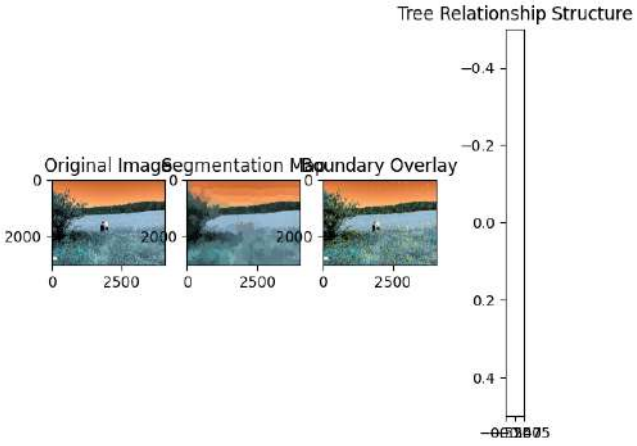

Fig. 39. B4 algorithm ncut 3.png