CENG466, Fall 2022, THE 4

1st Fırat Ağış
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
e2236867@ceng.metu.edu.tr

2nd Robin Koç
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
e2468718@ceng.metu.edu.tr

***Abstract*—**
***Index Terms*—**

I. Object Counting

Fig. 1. Original A1.png

Fig. 2. Original A2.png

Because mathematical morphology is limited to the gray-scale or binary images and our input images are 5472x3648 in dimensions, we start our process by first down-sampling our input images by a factor of 16, and then turned them to gray-scale by using their intensity value.

Then, for object counting, we used top hat operation on the gray-scale image to detect objects. We had a variety of options when it comes to the size of the structuring element. We tested a many sizes for the element, all being square in shape. Results of different structuring elements can be seen in row 1 of Figures 4, 5, and 6.

Fig. 3. Original A3.png

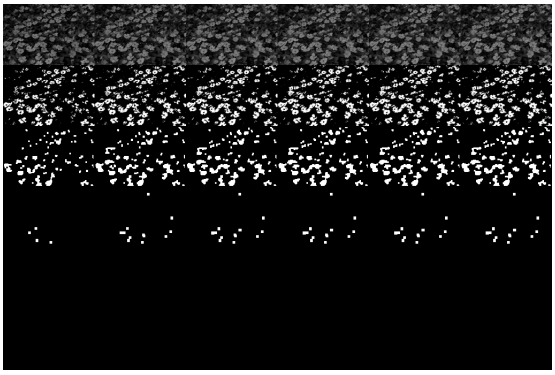

Fig. 4. Different size of structuring elements as columns, gray-scale output of top hat operation, binarized version of the operation result, and different sizes of structuring elements used for cleaning the noise by opening and closing operations as its rows

After using top-hat operation, we binarized the images, using $I = 100$ as the threshold intensity value. We arrived at this threshold by examining the intensity values in the results of the morphological operation. Binarized outputs for different structuring elements can also be seen in Figures row 2 of 4, 5, and 6. After binarization, we still had some amount of noise. To eliminate it, we decided to use opening and closing morphological operations on the binarized images. This once again introduced the design decision about the size of the structuring element. We once again tried a variety of options,

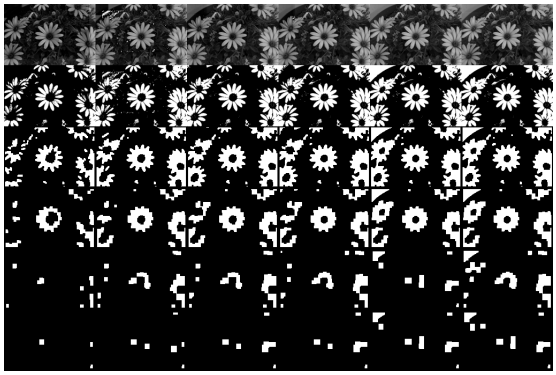

Fig. 5. Different size of structuring elements as columns, gray-scale output of top hat operation, binarized version of the operation result, and different sizes of structuring elements used for cleaning the noise by opening and closing operations as its rows

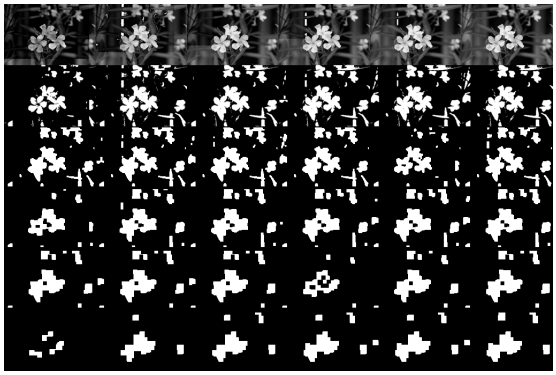

Fig. 6. Different size of structuring elements as columns, gray-scale output of top hat operation, binarized version of the operation result, and different sizes of structuring elements used for cleaning the noise by opening and closing operations as its rows

results of which can be seen in rows 3, 4, 5, and 6 of Figures 4, 5, and 6.

After comparing different options we arrived at the values which turned the images at Figures 1, 2, and 3 into Figures 7, 8, and 9 with the object counts 53, 7, an 6. We had configurations which had more accurate object counts but they didn't counted the actual flowers and counted some amount of noise to compensate for the flowers they didn't count. We chose the configurations that were the most successful in both the accurate count and the ones that counted the flowers, as best as we could.

II. Segmentation Algorithms

Image segmentation is a very important topic that has usages spanning where there is some context based application. We had two algorithms to use in this assignment. Namely mean shift algorithm and n-cut algorithm. In both usages of these algorithms we pre-processed the images before feeding them to the algorithms. We applied median blurring to get rid of some of the noise from the image. We thought median filter would be better than an averaging filter due to median filter not increasing the number of different colors by using the existing

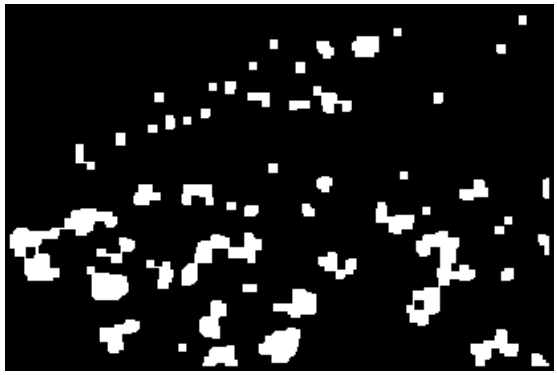

Fig. 7. Final A1.png

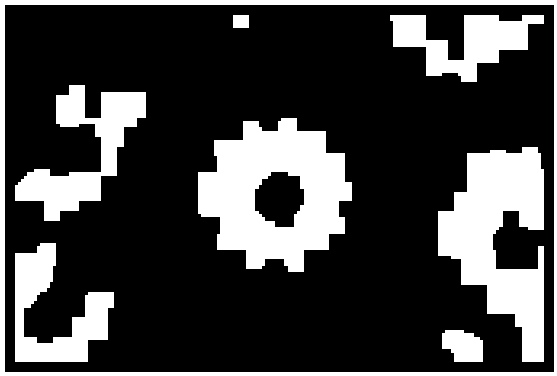

Fig. 8. Final A2.png

ones. Than we used bit plane slicing to get rid of the last 4 bits of the 8 bit integer colors, this helped us deal with almost half the number of colors than the original and color palette's size is a factor in computation time. We lastly down sampled the images for faster computation time and more smoothing. It was exceptionally handy for large input images.

A. Mean Shift Algorithm

Mean shift algorithms main idea is using rgb color values like coordinates on a 3d space and finding local maxima points. Mean shift algorithm starts by choosing a point and calculating local mean of points that are closer than a certain distance. After finding mean algorithm compares this mean value with the original point, if distance between them is more than a certain threshold we will shift the point. If the distance between new mean value and original point is less than the threshold we will say it is converged and jump to the next pixel. Why are we doing this? By following the closest mean we expect to gather all surrounding pixels to nearest local maxima and get a same colored region on the image. We have two important parameters while using this algorithm, first being the threshold and second being the radius of the area surrounding the chosen pixel. How does these effect the output;

- **Radius(Bandwidth):** Using a bigger radius means the algorithm will consider more points around the chosen

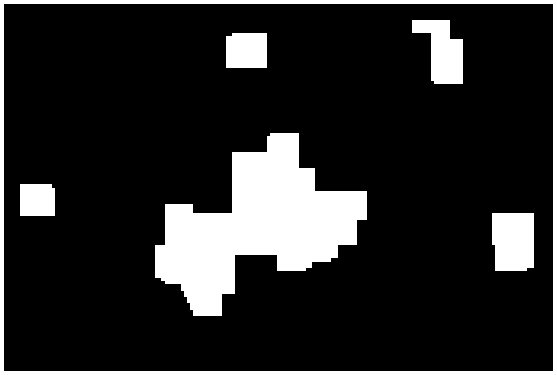

Fig. 9. Final A3.png

pixel and at the output this will create smoother boundaries but this means increasing this value too much will cause loss on percision and make it harder to differanciate neighbouring regions. Algorithm we uses find the best bandwidth according to image's color values.

- **Threshold:** Threshold effect the convergence time directly, smaller threshold means smaller changes will be counted and our algorithm will continue. Convergence time means more computation time but the results will be more accurate.
- **Cluster Size:** We can determine a minimum cluster size and eliminate clusters under this value by mapping them to the closest cluster. This will help us reduce the tiny clusters that cause noise on the output.

Choosing these values is a matter of problem at hand, design choice. We can determine the best values for this by doing a grid search or via intuition.

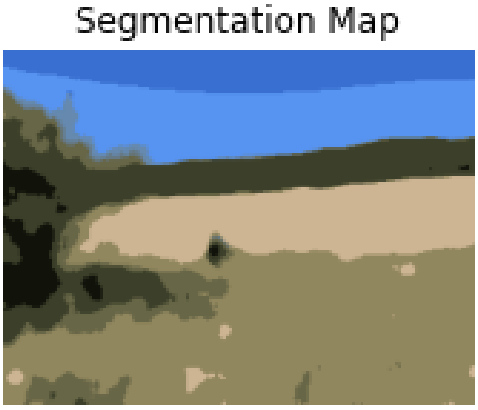

Fig. 10. Segmentation output of mean shift algorithm by eliminating clusters that contain under 50 pixel using B4

We can see the effects of this tiny clusters on our boundary overlay images too.

The image with the cluster with size under 150 has cleaner boundaries.

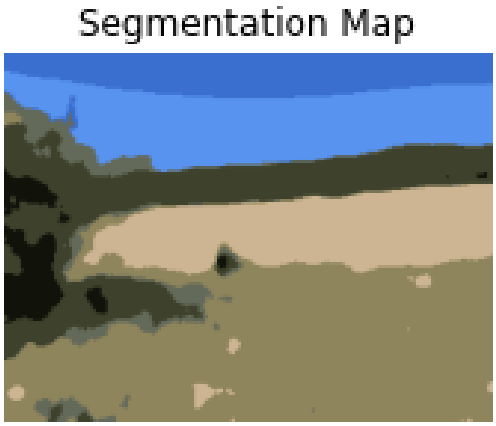

Fig. 11. Segmentation output of mean shift algorithm by eliminating clusters that contain under 100 pixel using B4

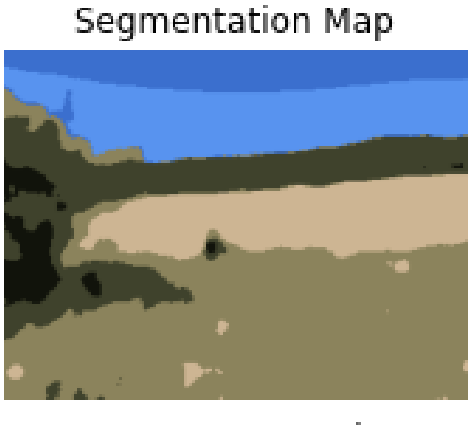

Fig. 12. Boundary overlay output of mean shift algorithm by eliminating clusters that contain under 150 pixel using B4

B. N-cut Algorithm

BEWARE: N-cut algorithm we use does not work with the same version that is needed for the $networkx$ library. N-cut only works with the version 2.6.3 of networkx and networkx works with only the latest release. We still got the both outputs by using different environments but there was no way we could think of for getting the graph and tree representations of the n-cut segmentation outputs. Coding was really similar, getting the labels for the segmentations of image through algortihms and representing part was the same. We still put the n-cut segmentation code to the end so there should not be any problems with the other parts of the code. **BEWARE END**

N-cut algorithm is basically a graph partioning algorithm with a goal that is minimizing the weights between partionings and maximizing the weights inside the partionings. N-cut algorithms outputs were sharper than mean shift algorithm's, mean shift detected the color differences on big objects as boundaries. But mean shift algorithm's outputs were more accurate when it could detect the edges. This is due to

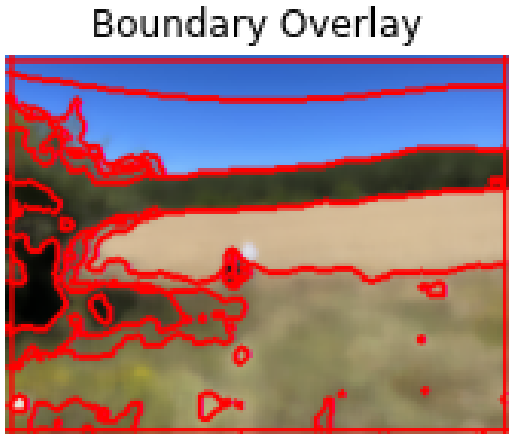

Fig. 13. Boundary overlay output of mean shift algorithm by eliminating clusters that contain under 50 pixel using B4

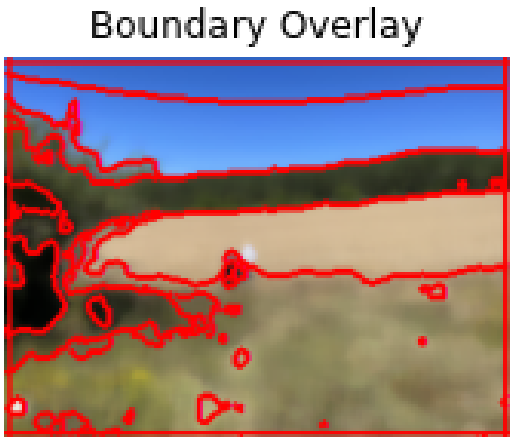

Fig. 14. Boundary overlay output of mean shift algorithm by eliminating clusters that contain under 100 pixel using B4

mean shift algorithm using only color information but n-cut segmentation using both color and spacial distance values.

When using n-cut algorithm we can change the weights of color values or spacial distance values effect on the edge values we try to minimize or maximize. Algorithm we used gave us the option to change how many clusters we wanted. Here are some examples with different values of this parameter;

Values we tried turned out to be too big to have a detectible change especially between 400 and 600. But we can still see that there is a line in the middle of the picture that separates the shady part of the ground from the sunny part. Increasing this parameter means increasing the cluster number and this means more detailed segmentations.

C. Representations of the segmented images

Main reason of segmentation is getting the possible objects, backgrounds or parts of the objects separatey. After this we may want to look at the realationships between these

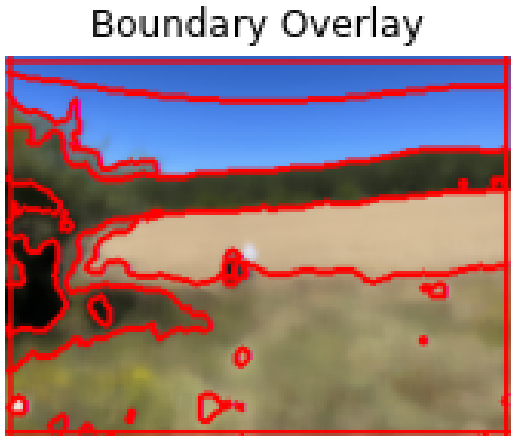

Fig. 15. Boundary overlay output of mean shift algorithm by eliminating clusters that contain under 150 pixel using B4

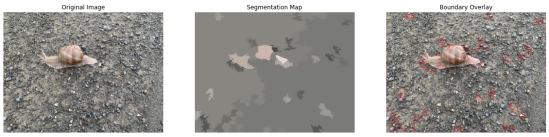

Fig. 16. Boundary overlay and segmentation outputs of n-cut algorithm by using image B1 and $n_segments$ parameter as 200

segments to get a sensei some information about the context. To see more clearly and use these relations we may want to represent/visualize them in different ways. For this assignment we had two options. A region adjacency graph or a relation tree structure. Region adjacency graphs give us information about which regions are neighbours/touching each other and relation tree give information about what are the main components of the image and if there is what do they contain.

Especially on the second figure we can see that graph is almost like made out of two smaller and denser graphs. We might say that this image can be separated into two more connected, almost independent regions. And yes we can see that separating the original image as sky and ground would not be that wrong.

From the tree relation we can say that this image' components/segments are more connected to one segment and there are lone segments too and yes again there is a lonely sky segment and more crowded ground segment in the image.

III. Dependencies

We used following libraries for the described reasons.

- **os:** Handling non-existent input or output paths.
- **numpy:** Executing array and matrix operations.
- **PIL:** Reading images and converting them to arrays.
- **mathplotlib:** Writing arrays as image files.
- **sklearn:** Using MeanShift method and estimate_bandwidth methods
- **networkx:** Creating graphs and trees
- **skimage:** Implementing n-cut segmentation

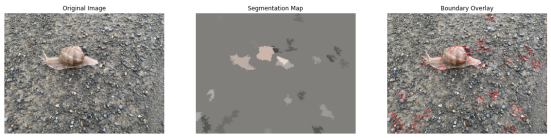

Fig. 17. Boundary overlay and segmentation outputs of n-cut algorithm by using image B1 and $n_segments$ parameter as 400

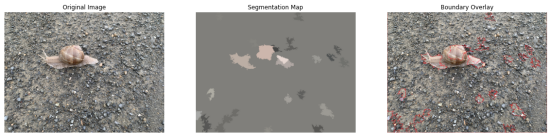

Fig. 18. Boundary overlay and segmentation outputs of n-cut algorithm by using image B1 and $n_segments$ parameter as 600

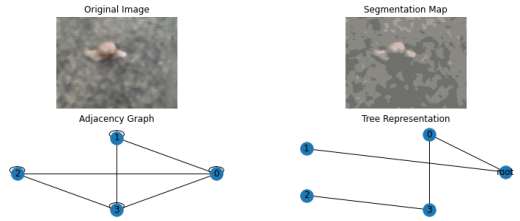

Fig. 19. Original image, segmentation map, graph representation and tree representation of the image B1

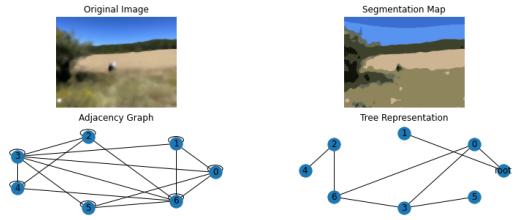

Fig. 20. Original image, segmentation map, graph representation and tree representation of the image B4