CENG466 - THE3 Solutions

Ertuğrul Aypek, 2171270
Computer Engineering
Middle East Technical University
Ankara, Turkey
e217127@metu.edu.tr

Kemal Baş, 2237048
Computer Engineering
Middle East Technical University
Ankara, Turkey
e223704@metu.edu.tr

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Part 1

In this part of the homework, we are required to segment flying jets on images and write its output as mask output. In addition to this, we are also required to print the number of flying jets to the terminal.

Although there are five images to deal with, we have applied similar operations with different parameters for all images. Below are the operations and their explanations:

Working with one channel images: In order to reduce complexity, we get single channel images by converting RGB images to grayscale using rgb2gray() or working with R, G or B channels individually.

Thresholding: After obtaining the single channel image, we reduced the complexity further by thresholding the image and getting the binary image. Note that every threshold value pick in this report is based on trial and error method.

Closing, opening and dilating images: These are the most common mathematical morphological operations used in image processing and we used them a lot. In order to disjoin connected components, we used opening. On the other hand, for the purpose of joining near regions, we used closing and dilating.

Binary masking: We created binary masks and using these masks, we stored valuable information for some parts of the image.

Opening the area: After getting a series of operations applied, there might be some irrelevant and small regions remaining. In order to discard them, we used *bwareaopen(image, threshold)* function.

Filling a hole: After the operations, some holes might occur in a region. To fill them, applying closing or dilation operations might distort the image. However, there is a morphological operation for this purpose with the interface in Matlab as *imfill(image, 'holes')*.

A. Image A1

For this image, we used threshold value 100. Then we applied image closing with a 5x5 structuring element all ones.

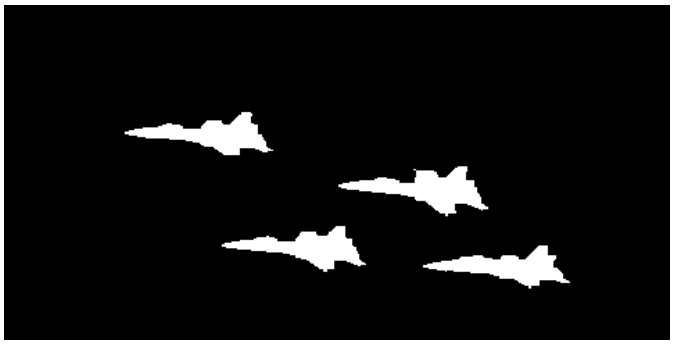

Fig. 1. A1 output image

B. Image A2

We first thresholded the image such that the pixel values between 75 and 100 will be 1 and 0 otherwise. Then, in order to store the clearly extracted upper plane, we created a binary mask. Top half of the mask was all 1s and bottom half was all 0s. We bitwise and this mask with thresholded image and saved upper plane.

After that, in order to get rid of tremendous noise around below plane, we used image opening. Then we merged the saved upper plane with opened image by using bitwise or operation. In order to connect the components, we dilated the image with 15x15 structuring element with all 1s. We filled the image to get rid of holes. As a final step, we discarded the regions whose areas are smaller than 1000 in pixels. Below image represent the final output.

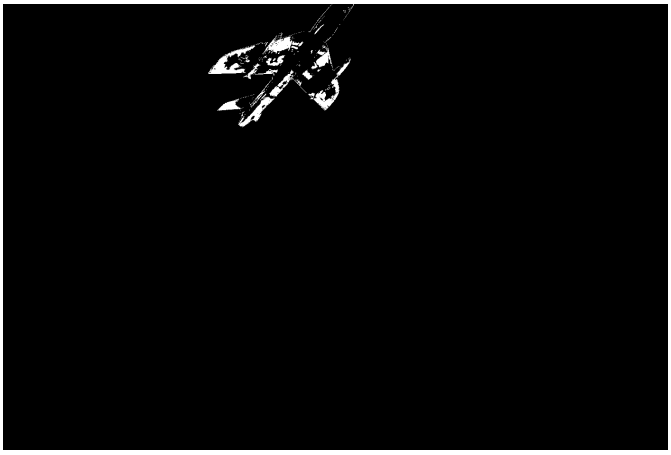

Fig. 2. Saved upper jet plane

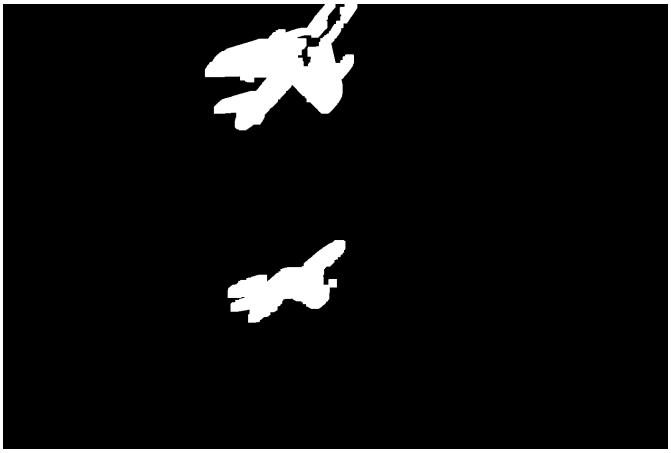

Fig. 3. Output for A2

C. Image A3

For image A3, we binarized the image with threshold pixel value 55. Then closed the image with all 1s 5x5 structuring element. Finally, we applied *bwareaopen()* to get rid of regions smaller than 10 pixels. Below is the output for this image.

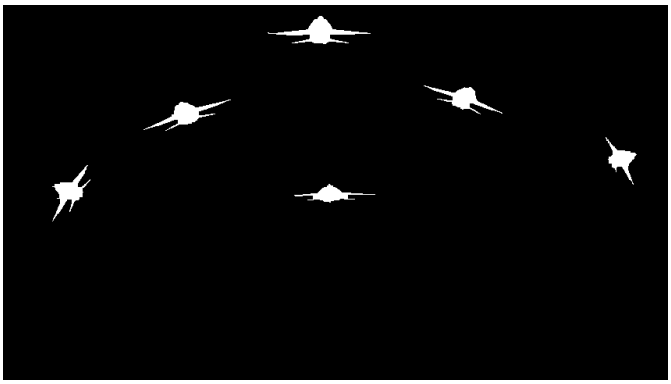

Fig. 4. Output for A3

D. Image A4

For image A4, we used 70 as threshold value. Then we eliminated the small regions with areas up to 200 pixels. After that, filling the holes was sufficient to get the jet planes. Note that, since right two jets intersect with each other, our count result for this image was 3 instead of 4. We were unable to separate these two jets with consistent segmentation output.

Fig. 5. Output for A4

E. Image A5

For image A5, we used 150 as threshold value. As different from other images, we directly worked on blue channel of this image instead of grayscale version of colored image. We thought that it would be easier to discard background by thresholding if we work on blue channel. Then we closed the image with 11x11 structuring element, discarded small regions and filling the holes. Below is the final output for A5.

Fig. 6. Output for A5

F. Image A6

For image A6, we used 55 as threshold value. However, there were a lot of small noises, so we discarded smaller than 10 pixel regions. Then we dilated the image to fill the small gaps with 5x5 structuring element all 1s. In order to fill the gaps bounded by regions, we used *imfill* method. Applying area opening again, we observed that there are just all 7 planes and a big region remained from mountain. To get rid of mountain part, we created a binary mask with 0s for a rectangle containing the mountain are and all other pixel values of 1s. Using logical and operation pixel-wise, we discarded the mountain and get the final result as shown below.

Fig. 7. Output for A6

II. Part II

In this part, we are given 25 different sample images to apply image segmentation on them. To apply image segmentation we should implement different segmentation algorithm and after that we should select the best 2 of them with their own parameters. Because of there are too many images, we will not explain all of them but instead we will give a general explanation.

First, we started with looking for alternative algorithms to apply for segmentation. We first looked at official MATLAB page for segmentation [1]. We saw that there are some pre-implemented segmentation functions. These are: imsegfmm (that segments the image as binary), imseggeodesic (that segments the image with respect to geodesic distance(geodesic distance is a graphical term. It represents the number of edges in shortest path between two nodes)), imsegkmeans (that uses k-means algorithm to segment image), and lastly watershed (that segments images into many areas as river systems [2]). Also there was imsegkmeans3 segmentation function. But it was for 3D space domain images so, it is useless in our condition.

We first tried to apply k-means segmentation function because we know that it gives comparatively good results, when we tried to run our code on MATLAB we got ”Undefined function or variable ’imsegkmeans’”. We were sure that we wrote the syntax correct and installed all the image processing tools when we installed MATLAB. So we did not understand the problem so we searched it on the internet. We found that k-means image segmentation function was introduced not for so long time ago. The release version was 2018b but we were obliged to use 2018a version that is only one previous version. So we can not use built in k-means segmentation function. We decided to checkout other algorithm functions

We moved on to next function that is imsegfmm. It is a binary image segmentation function for grayscale images. So we first transformed our images into grayscale. Then we added a mask that is for seed pixels. we created two types of masks one for horizontal images one for vertical images. We were lucky because the size of the images were all same but some of them were just vertical. We took the size of one of the sample vertical image then set the middle point(240,160) true, others all false. So our seed pixel was in the middle of the image. We did the same for mask2, but with two differences, because the size was different(transpose) middle point was also different, so we changed the mask size and set the middle point as (160,240). Now we set the masks, then we will calculate the difference weight, after that we will group the image pixels with respect to their difference. We will put a threshold to decide grouping. Because every image has a different pixel difference weight, we set individual threshold values for every images. After that by the help of the imsegfmm function, we generated our segmented image. After that we saved the resulting images into the requested folder, named as ”Segmentation_results_algo1/original_name.png” with imwrite function of the MATLAB.

We decided to use png image format. Because we tried with jpg format at first. When we tried to save the images, we got an error saying incompatible data type to save as jpeg. We needed to change the data format after the calculation in order to save images as jpg. But this would create a big overhead and it was not crucially necessary. So we decided to save the images as png files. This gave us no problem so, we moved on.

Here are some example of the results of binary segmentation (Not every image was perfect because we had to use grayscale versions and in some images, the object pixel values were too close -nearly the same- so that two objects merged as one object. This is observed in especially in images that includes water):

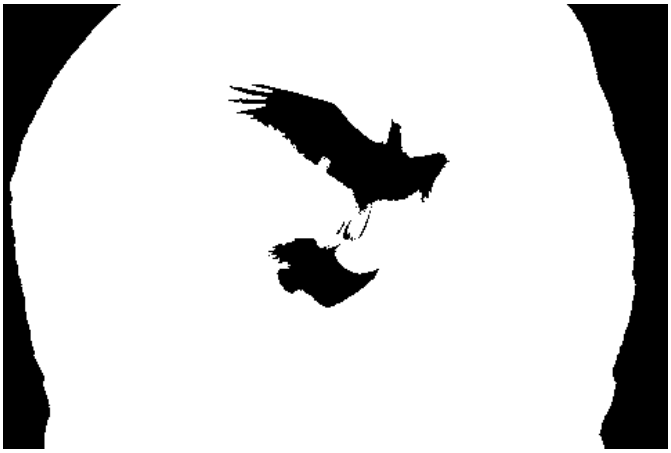

Fig. 8. Output for image 135069.jpg in binary image segmentation algorithm

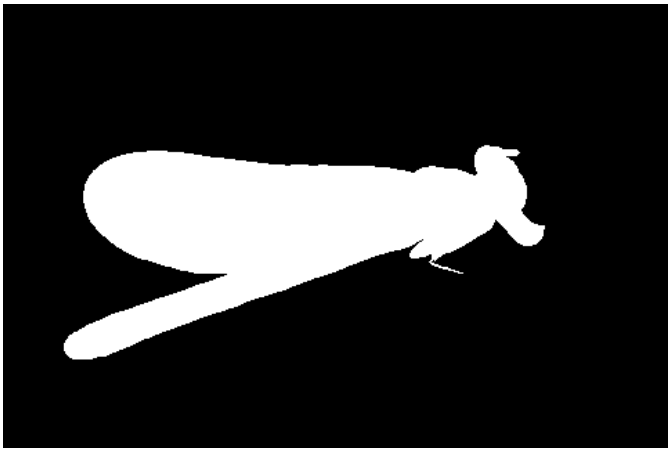

Fig. 9. Output for image 35070.jpg in binary image segmentation algorithm

After that we looked for imseggeodesic, we saw the implementation of that segmentation algorithm is too much area specific. Before using the function, we need to determine the areas to segment one by one, so, this one was actually not a smart algorithm. it was really nonsense to segment after we give the field of the objects in images to function. The algorithm should find and then determine the objects and boundaries by itself, not by us. So we passed this algorithm and decided to not choose it as one of the two best algorithm that we used.

Later on, as last algorithm, we used watershed segmentation algorithm. It divides images into many smaller regions rather than as a few big objects. ”The watershed transform finds ”catchment basins” or ”watershed ridge lines” in an image by treating it as a surface where light pixels represent high elevations and dark pixels represent low elevations. The watershed transform can be used to segment contiguous regions of interest into distinct objects” [3]. So we find the small boundaries rather than object itself. But we can see the general object structures after these small boundaries come together in an image.

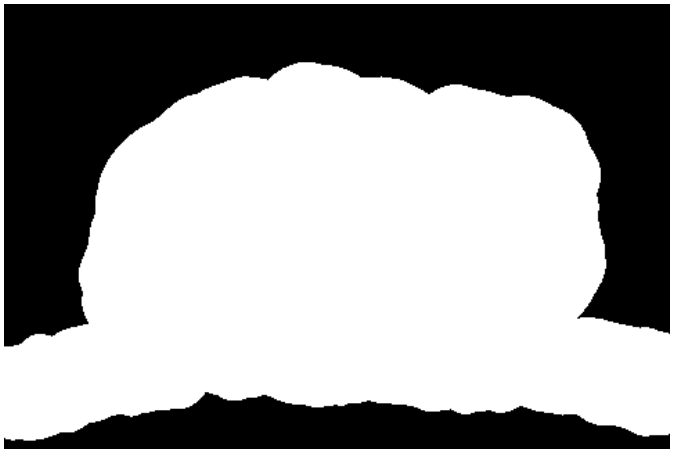

Fig. 10. Output for image 42044.jpg in binary image segmentation algorithm

It is highly abstracted function, we only give the image to segment and optionally connectivity type as parameters. We tried with all possible connectivity types and looked the results, but the best one was the not indicating connectivity type. MATLAB probably does optimisations for connectivity type to get best segmentation when the connectivity option is not given on purpose.

Here are some output samples of watershed algorithm function:

Fig. 11. Output for image 12003.jpg in watershed image segmentation algorithm

III. Part III

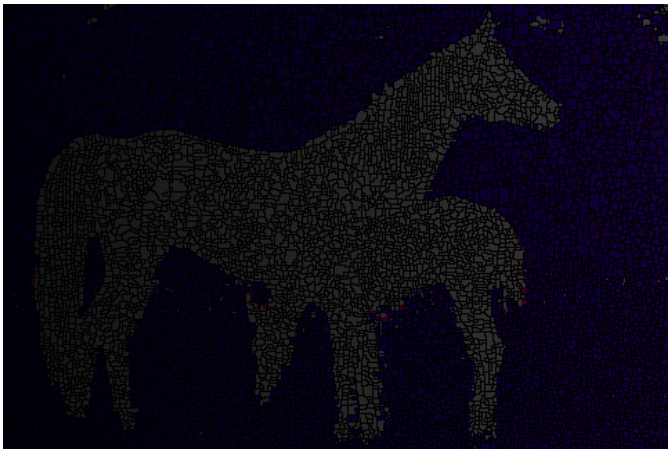

Fig. 12. Output for image 113044.jpg in watershed image segmentation algorithm

REFERENCES

[1] MathWorks, 'Image Segmentation', *https://www.mathworks.com/help/image segmentation.html*, 2019.

[2] S. Eddins, 'The Watershed Transform: Strategies for Image Segmentation', *https://www.mathworks.com/help/images/image-segmentation.html*, 2002.

[3] MathWorks, 'Watershed', *https://www.mathworks.com/help/images/ref/wate* 2019.

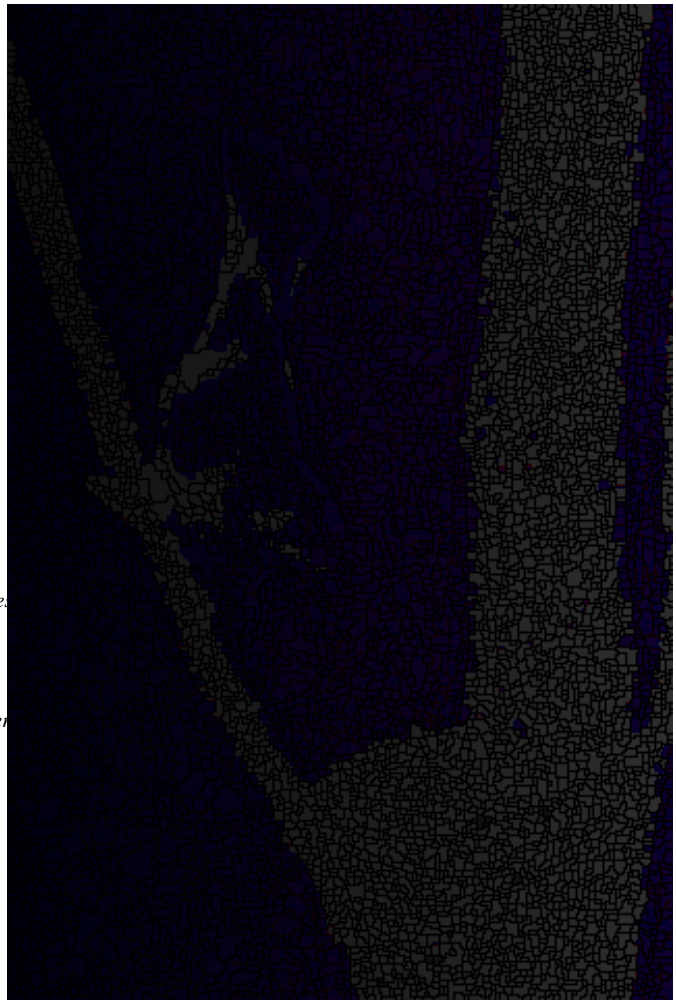

Fig. 13. Output for image 163014.jpg in watershed image segmentation algorithm

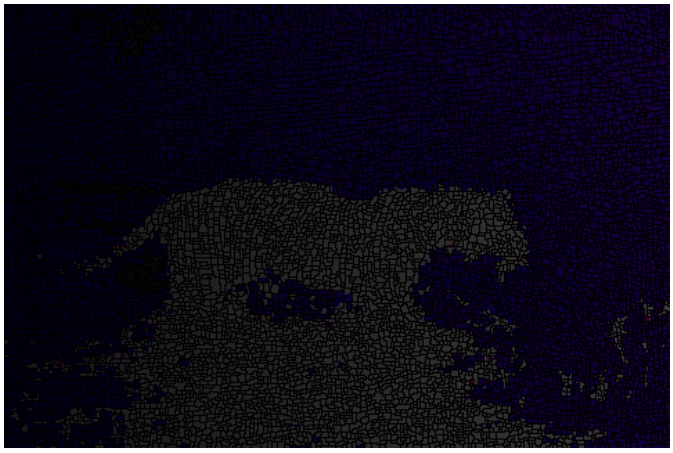

Fig. 14. Output for image 108073.jpg in watershed image segmentation algorithm