CENG466 - Take Home Exam III

Berk Güler
2310092

Onur Demir
2309870

I. Introduction

This document is prepared for CENG466 - Take Home Exam III. This report includes the solutions of the given problems, and also explains our methodology and includes the analysis of the results and our comments on them.

II. Question 1 - Object Counting

In this part our task was counting the number of flying balloons in the given 5 images using only mathematical morphology operations. Since there are 5 different images given, we applied different operations each on them. However, main idea and operations were same. These operations are read image as gray-scale, apply gamma correction if necessary, then apply thresholding to get binarized image. Then, apply some morphological operations by using cv.getStructiringElement and cv.morphologyEx functions. After all these operations, we used cv.connectedComponents to find the number of flying balloon objects in the image.
Using general threshold function might not be good in all cases, e.g. if an image has different lighting conditions in different areas. In that case, adaptive thresholding can help. Here, the algorithm determines the threshold for a pixel based on a small region around it. So we get different thresholds for different regions of the same image which gives better results for images with varying illumination. Therefore, in this homework, we used adaptivethreshold function for most of the given input images. Parameters of adaptivethreshold:
src = source 8-bit single-channel image.
maxValue = non-zero value assigned to the pixels for which the condition is satisfied.
adaptiveMethod = adaptive thresholding algorithm to use.
thresholdType = Type of the threshold, we used cv.THRESH_BINARY_INV in order to get balloon objects as white
blockSize = size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on.
C = constant subtracted from the weighted mean. Normally, it is positive but may be zero or negative as well.

A. A1

First we read the image as gray-scale.

Then, we applied gamma correction with value 1.1 to get better binarized image in thresholding. Without gamma correction our algorithm was not able to find the balloon near to the sun.

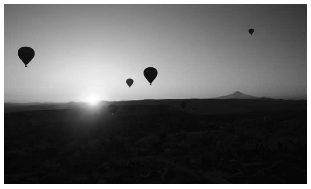
Fig. 1. A1 in grayscale

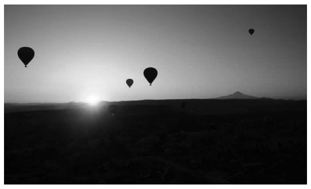
Fig. 2. A1 with gamma correction

After that, we applied adaptivethreshold function with src=gamma corrected image, maxValue=255, adaptiveMethod= ADAPTIVE_THRESH_GAUSSIAN_C, thresholdType= THRESH_BINARY_INV, blockSize = 235 and C = 68. Then, we got the following output:

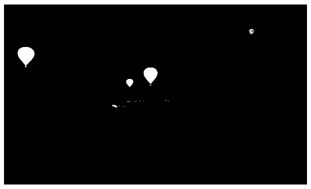
Fig. 3. A1 threshold

Then, we applied opening operation by using cv.MORPH_ELLIPSE with size 3x3, in order to remove noise in the image, and got the following output:

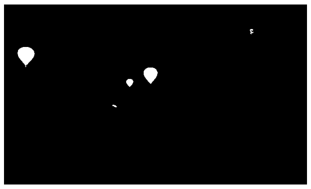
Fig. 4. A1 opening

Finally, we applied closing operation by using cv.MORPH_ELLIPSE with size 29x29. This kernel is way bigger than opening kernel since there are some gaps in the images, and we wanted to fill them by using larger kernel. Our script found 5 balloons in the image by using cv.connectedComponents. We substracted 1 from the result in order to remove background component. Result image can be seen in Figure 5.

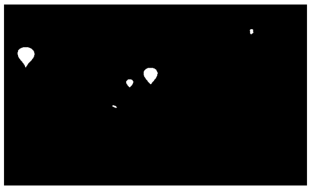
Fig. 5. A1 result

B. A2

First we read the image as gray-scale.

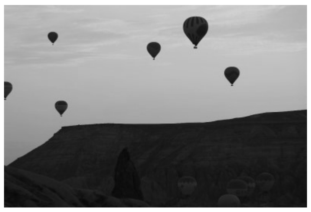
Fig. 6. A1 in grayscale

Then, we applied adaptivethreshold function with blockSize = 255 and C = 90. Even though we tried different threshold values we could not manage the get the balloons front of the mountains. After thresholding we lost them and got the following output:

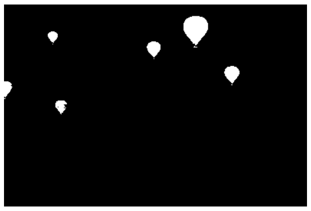
Fig. 7. A2 threshold

Finally, we applied closing operation by using cv.MORPH_ELLIPSE with size 25x25. We did not use opening in this image, since there were no noise in the image after thresholding. There were only gaps in the images, and in order to fill them we applied closing operation. Our script found 6 balloons in the image by using cv.connectedComponents. We substracted 1 from the result in order to remove background component. Result image can be seen in Figure 8.

C. A3

First we read the image as gray-scale can be seen in Figure 9.

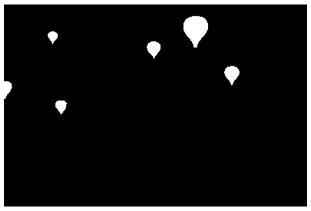
Fig. 8. A2 result

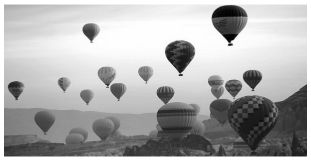
Fig. 9. A3 in grayscale

Then, we applied gamma correction with value 3 to get better binarized image in thresholding. Without gamma correction our algorithm was not able to remove mountain and get balloons correctly.

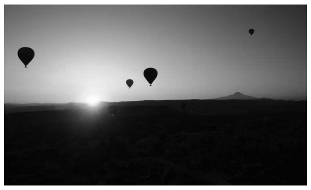
Fig. 10. A3 with gamma correction

After that, we applied adaptivethreshold function with blockSize = 141 and C = 15. We tried with many different values to get better threshold output but we got the best output with these values. Threshold output is as follows:

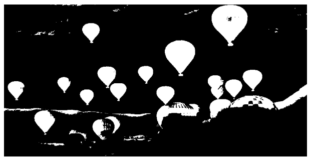
Fig. 11. A3 threshold

Then, we applied opening operation by using cv.MORPH_ELLIPSE with size 29x29, in order to remove noise in the image. Since there are so many noise arise from the mountain and the cramped balloon objects, we had to choose kernel size very large. When we choose opening kernel size very large, we saw some gap in one of the balloon, and got the output as in Figure 12.

Finally, we applied closing operation by using cv.MORPH_ELLIPSE with size 7x7. This kernel is smaller comparing to opening kernel. We tried to fill the gap in the big balloon but we could not manage it perfectly. When we chose the closing kernel larger components started to unite. Our script found 17 balloons in the image by using

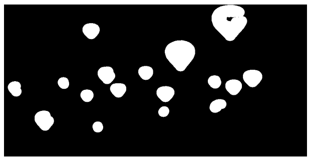

Fig. 12. A3 opening

cv.connectedComponents. We substracted 1 from the result in order to remove background component. Result image can be seen in Figure 13.

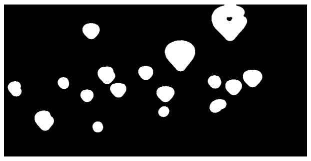

Fig. 13. A3 result

D. A4

First we read the image as gray-scale.

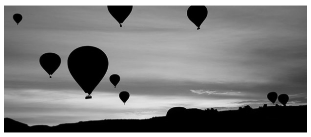

Fig. 14. A4 in grayscale

After that, we applied simple thresholding function with low threshold value=0 and high threshold value as 10. Since both balloons' and mountain's pixel values are very close to 0 we could not manage to exclude mountains from the threshold. When we tried to remove mountains we lost the ballons close to mountain. Therefore, we decided to keep mountain and the balloons together. Threshold output is as follows:

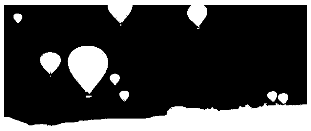

Fig. 15. A4 threshold

Then, we applied opening operation by using cv.MORPH_ELLIPSE with size 5x5, in order to remove the baskets under the balloons and to seperate 2 balloons that are in the right corner from the mountain. Then, we got the output as in Figure 16.

Finally, we applied closing operation by using cv.MORPH_ELLIPSE with size 2x2. This kernel is smaller comparing to opening kernel. We tried to smooth the edges of balloons by applying closing operation. When we chose the closing kernel larger, components started to unite with each other and the mountain. Our script found 9 balloons in

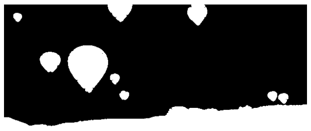

Fig. 16. A4 opening

the image by using cv.connectedComponents. We substracted 2 from the result this time, in order to remove mountain component together with the background component. Result image can be seen in Figure 17.

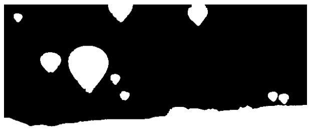

Fig. 17. A4 result

E. A5

First we read the image as gray-scale.

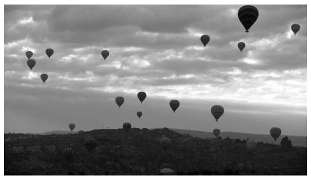

Fig. 18. A5 in grayscale

After that, we applied adaptivethreshold function with blockSize = 85 and C = 36. We tried with many different values to get better threshold output but we got the best output with these values. Threshold output is as follows:

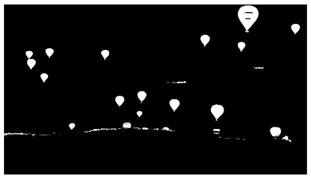

Fig. 19. A5 threshold

This time we applied closing operation first. We applied closing operation by using cv.MORPH_ELLIPSE with size 7x7, fill the gaps in the balloons. We chose the kernel size small since we do not want forest and cloud noises in our image. Closing output as in Figure 20.

Finally, we applied opening operation by using cv.MORPH_ELLIPSE with size 9x9. This kernel is slightly larger comparing to closing kernel. We tried to remove forest, cloud and rock noise from the image. 9x9 kernel size was the best size for our image. Our script found 18 balloons in the image by using cv.connectedComponents. We substracted 1 from the result in order to remove background component. Result image can be seen in Figure 21.

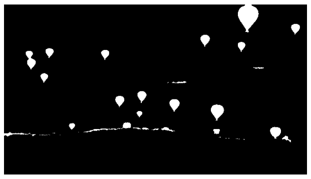
Fig. 20. A5 closing

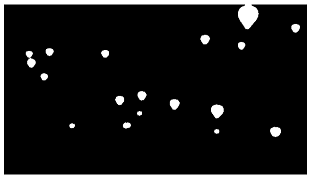
Fig. 21. A5 result

III. Question 2 - Segmentation

In this part, we used mean shift and n-cut segmentations to separate water, land and sky in the given images. Image segmentation is the process of partitioning an image into multiple different regions. We tried several different parameters to optimize parameters etc. to achieve the optimal solution.
We first implemented the mean shift and n-cut segmentations by using several libraries.
For mean shift;
We used sklearn library to apply algorithm. But before applying algorithm we smoothed the image by using Gaussian filter from opencv library.Then, we resized the image in order to reduce execution time of bandwith estimation. After applying meanshift algorithm, we replaced each pixel region with its average color ('centroid').
For n-cut;
We used skimage library to apply algorithm.We first obtained superpixeled-image by using "segmentation.slic" function. Then we used superpixeled-image to build rag (graph) out of the superpixeled image labels. After that, we applied n-cut segmentation method. Then, we replaced each pixel region with its average color ('centroid').

A. B1

Our input image was given as Fig 22. After mean shift with

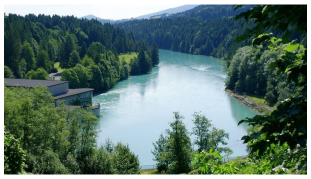
Fig. 22. B1 input

quantile -¿ 0.7 and number of samples -¿ 2000; After n-cut with compactness = 20 and number of segments = 1000; As a result by comparing two result with different parameters we decided on these final parameters. For these image ncut was better than mean shift since it detected sky, land and water.

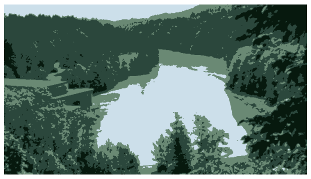
Fig. 23. the3 B1output meanshift

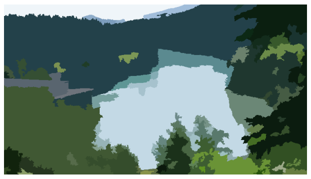
Fig. 24. the3 B1output ncut

B. B2

Our input image was: After mean shift with quantile = 0.7

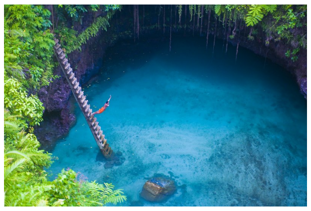
Fig. 25. B2 input

and number of samples = 2000; After n-cut with compactness=

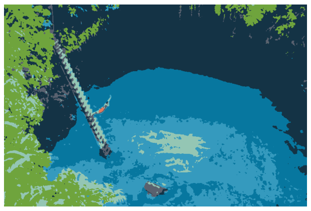
Fig. 26. the3 B2output meanshift

20 and number of segments = 1000; As a result by comparing two result with different parameters we decided on these final parameters. For these image meanshift was better than n-cut since it detected sky, land and water more precise.

C. B3

Our input image was: After mean shift with quantile = 0.7 and number of samples = 2000; After n-cut with compactness= 20 and number of segments = 1000; As a result by comparing two result with different parameters we decided on these final parameters. For these image ncut was better than mean shift since it detected sky, land and water.N cut is dedected less number of label than meanshift.
Finally overall execution time was less than 10 minutes.

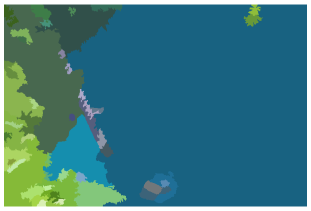

Fig. 27. the3 B2output ncut

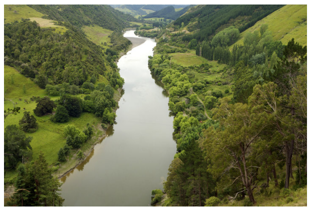

Fig. 28. B3 input

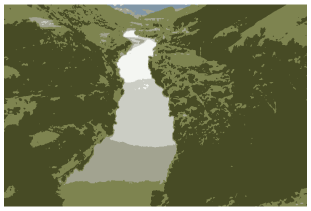

Fig. 29. the3 B3output meanshift

References

[1] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., USA.

[2] Jpeg Compression. (2015). Arjunsreedharan.Org. https://arjunsreedharan.org/post/146070390717/jpeg-101-how-does-jpeg-work

[3] Noise reduction. (2021). Geeksforgeeks. https://www.geeksforgeeks.org/how-to-remove-noise-from-digital-image-in-frequency-domain-using-matlab/

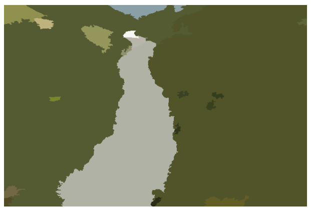

Fig. 30. the3 B3output ncut