CENG466 - Fundamentals of Image Processing Report 3

Bengisu Ayan
Computer Engineering
Middle East Technical University
Ankara, Turkey
e223697@metu.edu.tr

Ceren Gürsoy
Computer Engineering
Middle East Technical University
Ankara, Turkey
e2237485@ceng.metu.edu.tr

I. Introduction

In this assignment, we aimed to familiarize ourselves with fundamental morphological image processing techniques and image segmentation algorithms. We have used python3.5.2 along with numpy and cv2 libraries.

II. Question 1 - Object Counting

In this part of the assignment, we are required to count the number of flying jets in the images A1.png, A2.png, A3.png, A4.png, A5.png and A6.png using only mathematical morphology techniques.

Since these images are different from each other, we applied different operations on them. However, there are some common techniques that we applied. Firstly, all these images, except for A5.png, were converted to gray scale by $openCV$. Secondly, they were converted to binary images with a threshold value, which is different for each of them as expected. With binarizing operation, we tried to obtain flying jets as white and everything except them as black. Then, after some morphology techniques using $getStructuringElement$ and $morphologyEx$ functions of $openCV$, the number of flying jets on them were counted with $connectedComponents$ function of $openCV$. Since this function counts connected components on the image, it also counts the background, therefore we subtracted one from its result to obtain the number of flying jets. Other different techniques are mentioned in section of images:

A. A1

A1 was the simplest image compared to others, because after applying the threshold, flying jets were seen clearly as can be seen in Figure 1. After multiple tries, we got the best result with threshold value 80, then we converted each pixel smaller than 80 into white color, and others became black. After that operation, some parts on the top of the flying jets also became black and looked like gaps. In order to connected these parts of the flying jets without changing their sizes, we used closing operation. Since edges of flying jets and transition between their wings and body are sharp, we used rectangular structuring element. Also, in order to prevent wrong connections, we kept size of closing elements as small, which was 3. Our result image can be seen in Figure 2

(a) A1 before (b) A1 binarized

Fig. 1: A1

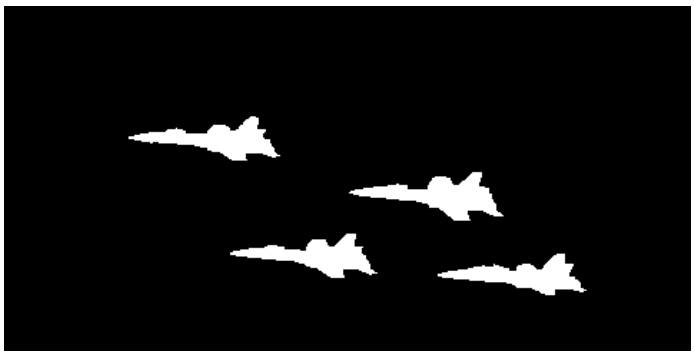


Fig. 2: A1 result

B. A2

A2.png was one of the most difficult image for us, because of the flying jet above. After many tries, we found most suitable threshold values as 75 and 110. By making values smaller than 110 as white, we got rid of sky, but flying jet below was lost into forest. Then making values also bigger than 75 as white, we separated it. However, this threshold also removed very big part of the flying jet above. These three operation can be seen in Figure 3.
After binarization, upper bound of forest region remained left. In order to remove it, we applied opening operation, because they are tiny edges like noise, and opening operation provided either removing them or keeping the size of flying jets. Since these edges were appeared sharp, we selected a rectangular structuring element. We wanted to select the size of structuring element as small as possible, in order not to lose valuable part of the flying jets, and after many tries, we selected kernel size as 5 and obtained result shown in Figure 4. As can be

seen from the figure, boundary of the forest was removed, but some parts of the flying jets were separated from each other. In order to merge them back, we applied closing operation with a rectangular kernel with size 45. When we use ellipse structuring element, the sharp transition on the wings and behind of the flying jets was disappeared. Also the size was arranged to obtain optimal appearance for both flying jets. Increasing that size caused joining more indentations, and since the kernel shape is rectangular, the images were becoming more similar to a rectangular. Therefore, we obtained a result with kernel size 45 as good as possible as shown in Figure 5.

(a) A2 before (b) A2 binarized with threshold value 110 (c) A2 binarized also with threshold value 75

Fig. 3: A2

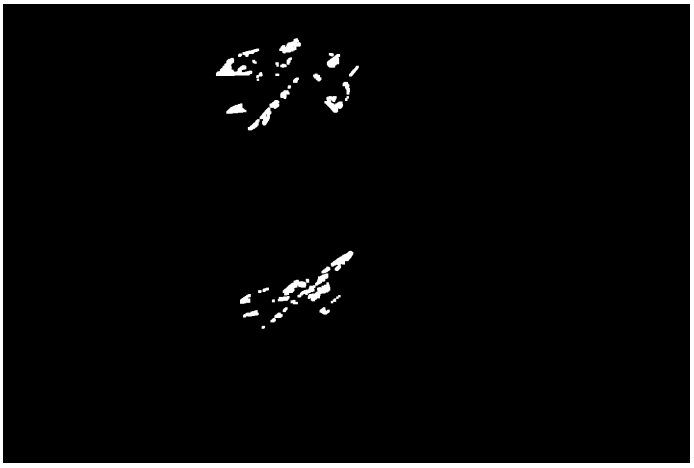


Fig. 4: A2 after opening

C. A3

A3.png was also simple one compared to others, because after applying the threshold, flying jets were seen clearly again as can be seen in Figure 6. Threshold value, which is 60, was the most suitable value for this image, because increasing it caused more parts to pass through, for example gasses under the plane on the left became visible. Decreasing it, on the other hand, took away the necessary parts of the flying jets. After this operation, there were small parts near the wings of flying jets, especially on their behind wings. Therefore, we removed them by keeping other parts of the wings (because they would also be counted as protrusion and would be deleted.) as much as possible. Then, we selected kernel size as 3. It caused to remove separate parts of the wings and gave a good result except for flying jet above and at right, because their behind

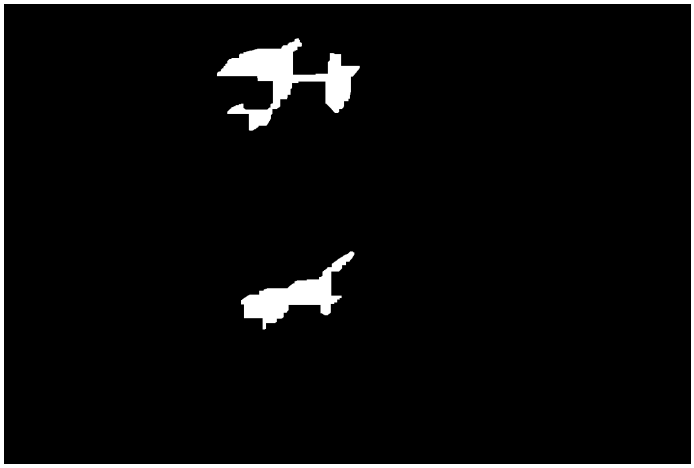


Fig. 5: A2 result

wings were very tiny. Then, there were still separate parts of the wings that we wanted to preserve as can be seen in Figure 7. Therefore, we applied closing operation to connect them with kernel size 5. Since closing operation connected parts between wings and body of the flying jets, increasing the size of kernel caused behind wings became almost invisible and made them as a part of the body. Decreasing that size, on the other hand, could not connect the separate parts, and they were counted as different objects. Therefore, we obtained the best result with kernel size 5 as shown in Figure 8. Lastly, both the shape of kernel used in opening and closing operation was ellipse instead of rectangular. The reason of it is that after making the image binary, we realized the transition on the wings and body is smooth, like circular. Also, body part of the flying jets are ellipse and using rectangular structure element would cause to lose their elliptic shape.

(a) A3 before (b) A3 binarized

Fig. 6: A3

D. A4

Again firstly we applied a threshold to the image and made pixel values smaller than 50 as white. When we increased that value a bit, to 55, there were no significant different inside the flying jets, but some parts of gases behind the flying jets remained. And decreasing it caused that more parts of the flying jets were deleted. Therefore, we obtained the best result with that value as shown in Figure 9. Then, we first wanted to apply closing in order to connect separate parts inside the flying jets, but as can be seen in the original image and its binarized version, tail of the flying jet at down

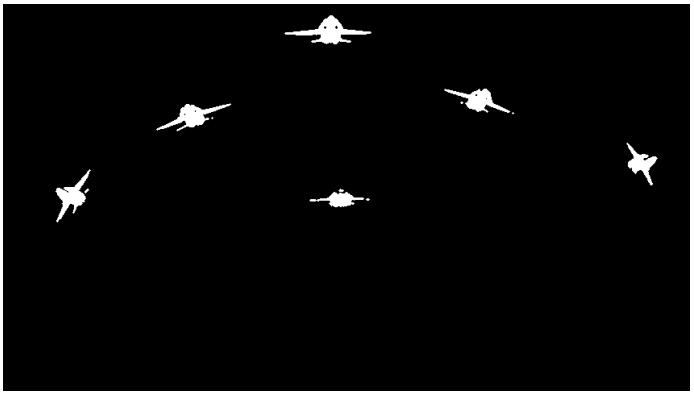


Fig. 7: A3 after opening

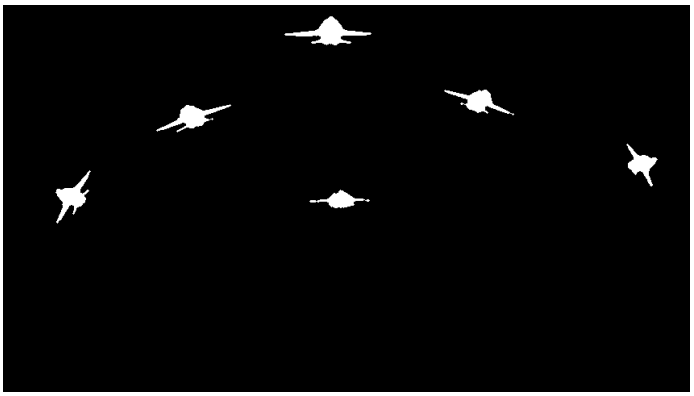


Fig. 8: A3 result

right corner looks like connected to the body of the flying jet at upper right corner. Therefore, applying any operation that connects or thickens parts like closing or dilation would make more difficult to separate them. Therefore, we applied erosion, decreased the size of flying jets and our problem was solved. Although the tail of the flying jet at down right corner was separated and joined to body of the flying jet at upper right corner, it seemed like the wing of that flying jet, therefore it wasn't a problem for us. Since we wanted to keep parts of the flying jets as much as possible, we selected a structuring element with small size, 3, and obtained the result shown in Figure 10. Also, since actual part we wanted to be eroded was in rectangular shape, we selected the shape of structuring element as rectangular. Then, since there were gaps inside the flying jets, we applied closing operation to fill them and also connect other separate parts. Since we didn't want to reconnect the tail we separated and regions that we want to fill placed in the body of the flying jets which are ellipse, we used an ellipse structuring element, and arranged its size as effective as possible, which was 7. As a result, we obtained the image shown in Figure 10. Since we arranged the size of kernel small, tails of upper flying jets couldn't be connected to flying jets. Therefore, we decided to remove them by keeping the parts inside the body of the flying jets. Then, we applied the opening operation with a small size kernel, which is 3 and obtained the result shown in Figure 10. Since some parts of connected with tiny edges were also removed, we applied the dilation to make them thicker. Since dilation is the complement of erosion operation, our size of flying jets didn't increase very much as well as we connect the all separate parts. We used a rectangular structure element as in erosion operation and its size was arranged as 6. As a result, we obtained the result image shown in Figure 11.

(a) A4 before (b) A4 binarized

Fig. 9: A4

(a) A4 after erosion (b) A4 after closing (c) A4 after opening

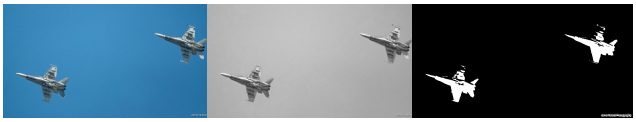
Fig. 10: A4

Fig. 11: A4 result

E. A5

In A5.png, there were nothing at the background except blue sky as shown in Figure 12. Therefore, we decided to read image as rgb instead of gray scale and discard the blue channel at first. So, we obtained the result shown in Figure 12. Then, we applied a threshold to binarize the image and made every pixel smaller than 150 as white and obtained the result shown in Figure 12. There were only jets and text left on the image. We firstly removed the text placed at the right corner using erosion with an ellipse structuring element with

size 5 and received the result shown in Figure 13. Since very small pieces of the text remained and also there were some separate parts behind the flying jet, we applied opening to remove them. We didn't increase the size of structuring element of erosion operation, because it would take off more parts inside the flying jets. For opening operation, we used an ellipse structuring element with size 7. It provides us smoother deletion on the tiny pieces like around the parts of the flying jets as shown in Figure 13. Then, we wanted to connect upper part of the flying jets to their bodies. Since these parts are completely separate, we couldn't apply closing operation directly with a small sized kernel. Therefore, we applied dilation in order to decrease the gap between upper parts and the bodies with a rectangular structuring element with size 7. By making thicker all of the parts, we received the result shown in Figure 13. Lastly, we applied closing operation in order to connect the parts with an ellipse structuring element with size 11. And, we obtained the result shown in Figure 14.



(a) A5 before (b) A5 without blue channel (c) A5 binarized

Fig. 12: A5

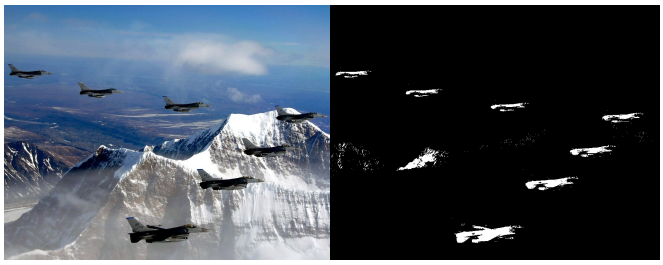
(a) A5 after erosion (b) A5 after opening (c) A5 after dilation

Fig. 13: A5

Fig. 14: A5 result

F. A6

A6.png was the most difficult one for us. The reason of it is that the color of the flying jets and the part of the mountain at front left are very similar as can be seen in Figure 15. Therefore, after binarization, a big part of that mountain was left as shown in Figure 15. We used threshold value as small as possible, 15, but decreasing it more caused removing the part of other flying jets very much without removing this big part of the mountain. Therefore, we decided to apply morphological operation without considering that part. We first applied opening to remove noises left from background and also tiny pieces around the flying jets. We selected an ellipse structuring element with size 4, because shape of the flying jets became similar to ellipse after binarization, and we didn't want to lose their shapes with a rectangular structuring element. Also, in order to save the valuable parts inside flying jets, we kept its size as small as possible, which is 4, and we obtained the result shown in Figure 16. Since some gaps on some parts of the flying jets like on their tails were occurred, we firstly made the flying jets thick with dilation operation, then connected separate parts with closing operation. Since again we wanted to preserve their shape and the smooth transition between their parts, we selected an ellipse structuring element for both of them. The results we obtained shown in Figure 16 respectively. After making flying jets as one part, we applied mask in order to remove the eighth part remaining from background. The result image is shown in Figure 17.



(a) A6 before (b) A6 binarized

Fig. 15: A6

(a) A6 after opening (b) A6 after dilation (c) A6 after closing

Fig. 16: A6

III. QUESTION 2 - SEGMENTATION

In this part of the assignment, we are required to develop an image segmentation algorithm that can separate dogs from their background. Image segmentation is the process of partitioning an image into multiple different regions. We tried several different approaches before coming up with our own algorithm.

A. Clustering Based Segmentation

Our first intuition was to apply image segmentation using k-means clustering. We first smoothed the images with Gaussian

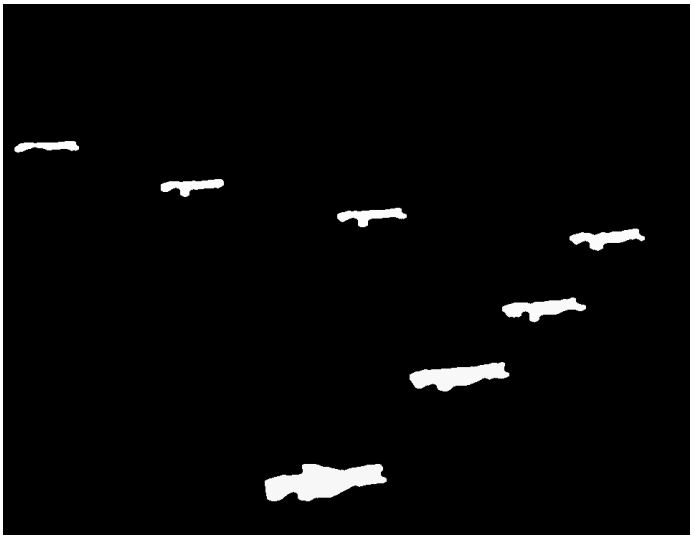
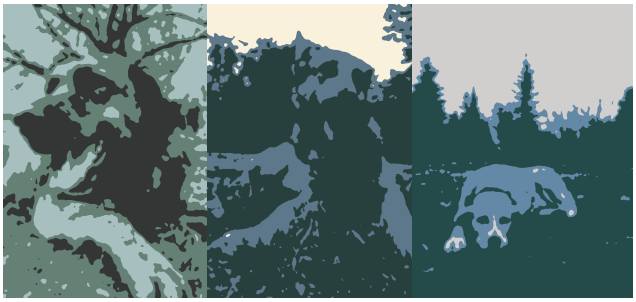


Fig. 17: A6 result

filters. We then used $cv2.kmeans$ function in order to apply this algorithm on the images. We tried different k values and settled on k=3 while analyzing the images. However, we could not obtain satisfactory results using k-means clustering algorithms. You can see some of the resulting images in Figure 18. Different parts of the dogs were clustered in different segments and we could not develop a further approach to merge these segments after k-means clustering.



(a) B1 (b) B3 (c) B5

Fig. 18: K-means clustering applied on B1, B3 and B5 with k=3

B. Thresholding Based Segmentation Using Otsu's Method

We then decided to try applying global thresholding based on Otsu's method. In a sense, image thresholding is the simplest kind of image segmentation because it partitions the image into two groups of pixels — white for foreground, and black for background [1]. Otsu's method is a global thresholding algorithm. We thought of applying this method and if the results are satisfactory, apply morphological operations to obtain better dog segments. We first smoothed the gray scale images using the Gaussian filter. We then used $cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)$ function in order to apply this algorithm on the images. As you can see in Figure 19, while the method was partially successful on some images such as B5, it was unsuccessful on images like B1 and B3, where there is a contrast between different parts of the dog's furs. In B1 and B3, dogs' legs and main body are in different segments which would be inefficient if we tried to apply morphological operations on them. Thus, we decided to try other methods.

(a) B1 (b) B3 (c) B5

Fig. 19: Otsu thresholding applied on B1, B3 and B5

C. Edge Based Segmentation

Thirdly, we tried a method that detects edges with canny edge detection algorithm, finds the largest component and separates it from the image. We used $cv2.Canny()$ function for canny edge detection with different minimum and maximum values. However, we couldn't find a way to get the boundaries of only dogs. The grass on the images created a big problem for us, also some large things at the background like trees or buildings were also detected as a big component. Therefore, we couldn't continue to apply this method. Our three results were shown in Figure 20

(a) B1 (b) B2 (c) B5

Fig. 20: Canny edge detection on B1, B2 and B5

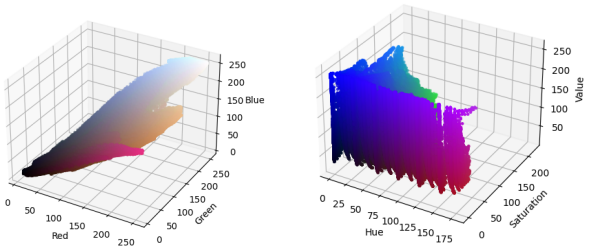
D. Our Algorithm

We decided to exploit the color information of the images while working on our own segmentation algorithm as edge based segmentation algorithms were not useful in our case. Here are the steps to our algorithm:

1) **Smooth the image**

We smooth the image using a 11x11 Gaussian kernel and the $cv2.GaussianBlur$ of cv2. We added this step to our algorithm as it was a consistent step in most of the image segmentation algorithm we have learned.

2) **Convert the image to HSV color space**
HSV stands for Hue, Saturation, and Value (or brightness), and is a cylindrical color space. The colors, or hues, are modeled as an angular dimension rotating around a central, vertical axis, which represents the value channel. Values go from dark (0 at the bottom) to light at the top. The third axis, saturation, defines the shades of hue from least saturated, at the vertical axis, to most saturated furthest away from the center. [2]
As can be seen in Figure 21, in B4 (as an example), colors are much more localized and visually separable in HSV color space. We use this property a lot in this algorithm to exploit color and contrast information of the image. We converted from RGB color space to HSV color space using $cv2.cvtColor(image, cv2.COLOR\_RGB2HSV)$



(a) RGB (b) HSV

Fig. 21: B4 color distribution in RGB and HSV spaces

3) **Extract masks for the desired colors (green, blue and white masks are provided in our implementation)**
The main point we exploited in all these dog images is the fact that dogs are lying on the grass. As the grass is always green in these images, we mask the range of green values using $cv2.inRange$ of cv2. We filter the hue values between 30 and 86 to eliminate green values in our green mask. For saturation and value, we decided on an interval of 0 and 255. This method worked exceptionally well in B1 as the image only consists of the dog and grass.

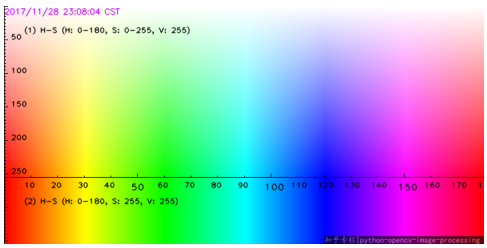
After realizing the mask works with green values, we decided to add the colors we want to eliminate as an argument to the $segmentation\_function$. For the image dataset in the homework, we only added blue, white and yellow masks as optional masks to the default green masks.

For the sky appearing in B2, B3, B4 and B5 we created a mask by filtering blue colors from the image. For this process we eliminated hue values between 80 and 125.

For white buildings in B2, we created a mask by filtering white colors from the image. We were not very hopeful about this filter as the dogs in the image also have a fair color. However, surprisingly it worked better than we expected after a few experiments with the color range. We used the hue range 0 and 145, saturation range 0 and 60 and value range 200 and 255.

We created a yellow mask for yellow leaves and trees in B2 and B5. This filter had a smaller scope in the images compared to the other ones and we were afraid it would also affect the dog in B5 but we managed to eliminate the problem. We masked hue values between 10 and 33, saturation between 0 and 255, values between 0 and 100.

The values in this step were mostly estimated by checking the HSV diagram in Figure 22 and experimentation.



(a)

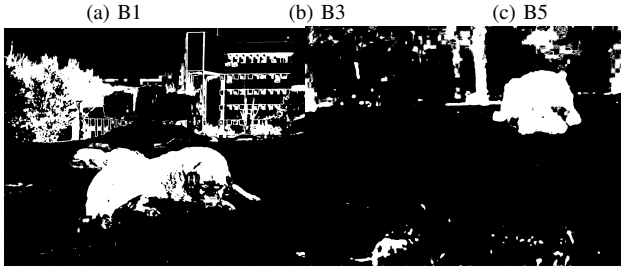
Fig. 22: HSV color ranges

4) **Apply the final mask to the image**
After creating these masks, we added them together to create a final mask and applied it to the original image to obtain a binary image. However, we obtained an image with white background and black foreground objects, therefore we inverse the image. The results of this step can be seen in Figure 23

5) **Apply Opening to the resulting binary image** As can be seen from Figure 23, there are still many white components we failed to completely get rid of except for the dogs. There are small grasses and parts of trees leftover from color filtering. We decided to apply morphological operations to get fix these problems. Therefore, we applied opening to get rid of these small components. We chose a 27X27 kernel as some of them were not that small. Furthermore, we chose a elliptical structural element as dogs have smoother corners, instead of rectangles. You can see the results in Figure 24. While applying opening, we lost the information about the dog's head in B5.

6) **Apply Closing to the image obtained from the previous step**
To fix the small wholes in the five images, we decided to apply closing and create more connected components. Results can be seen in Figure 25. As you can see, in

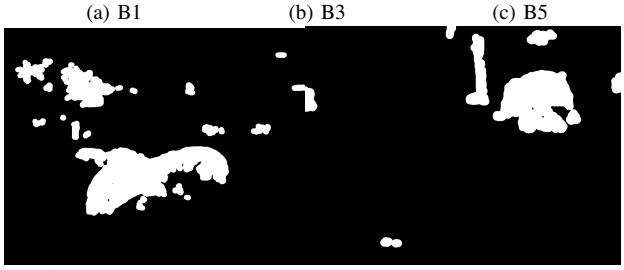
(a) B1 (b) B3 (c) B5



(d) B2 (e) B4

Fig. 23: B1, B2, B3, B4, B5 after HSV color space segmentation

(a) B1 (b) B3 (c) B5



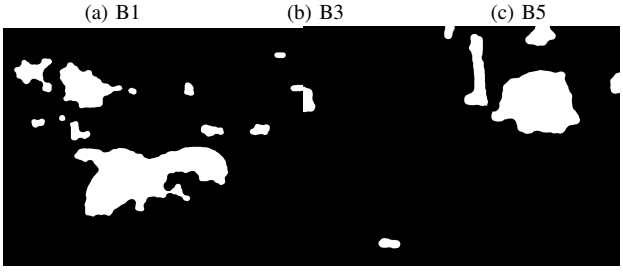
(d) B2 (e) B4

Fig. 24: B1, B2, B3, B4, B5 after applying Opening

B1, we could not fill the hole in the dog. It was a darker part of the image, which we involuntarily filtered. Currently we apply closing with kernel size of 41X41, if we increase the kernel size to close the hole, the other images are heavily distorted. We now have more concrete segments instead of small random components.

7) **Extract the biggest connected component** We then decided to get the biggest connected component to get rid of the smaller random segments and obtain only the dogs and the background. However, this step has also a drawback: Since we take the biggest segment, it is impossible for us to find separate dogs if they are not connected to each other. Thus, we can only apply image segmentation to obtain **one** dog or seemingly connected dogs like in B2. End results of our algorithm can be seen in Figure 26.

(a) B1 (b) B3 (c) B5



(d) B2 (e) B4

Fig. 25: B1, B2, B3, B4, B5 after applying Closing

(a) B1 (b) B3 (c) B5

(d) B2 (e) B4

Fig. 26: B1, B2, B3, B4, B5 after our algorithm is applied

IV. QUESTION 3 - OLD EXAM QUESTION

A. (a)

The y-axis of these histograms shows the number of pixels having a value corresponding to its histogram, and each of these histograms are one of the channel of that image. If we sum the y values of pixels on one histogram, we can get the

number of pixels of one channel of the image, because total value on all of them is the same. One channel of the image consists of 63 pixels. As we keep H,S,I channels of the image and each pixel is represented by 3 bits, image size is 63*3*3 = 567 bits.

B. (c)

1) i: As in adaptive thresholding we split the image into regions and compute their thresholds separately, we normally apply Algorithm 1 to each separate region. In our calculations for this question, we took region number = 1, as we have no spatial information about the image to divide it into different regions. For thresholding of each region, we chose to apply Otsu's thresholding algorithm. Algorithm 1 is our pseudocode for it. After applying Otsu's algorithm, we found threshold h_H, h_S, h_I to be 3 for each histogram.

2) ii: Otsu's thresholding algorithm separates the image into 2 regions: foreground and background. However, in our version of the algorithm, we have calculated a threshold value for each histogram. Thus, we need to chose which value to use to segment our image. In each case, we find 2 homogenous regions by separating values to be bigger and lower than 3. Values lower than 3 represent the background and bigger than 3 represent the foreground.

REFERENCES

[1] https://learnopencv.com/otsu-thresholding-with-opencv/
[2] https://realpython.com/python-opencv-color-spaces/

Algorithm 1 Thresholding algorithm

procedure OTSU_THRESHOLDING(h_H, h_S, h_I, N){Adaptive Histogram thresholding using Otsu}
  $p_H[L]$
  $p_S[L]$
  $p_I[L]$
  for `i in L-1` **do**{Compute normalized histograms}
    $p_H[i] \leftarrow h_H[i]/N$
    $p_S[i] \leftarrow h_S[i]/N$
    $p_I[i] \leftarrow h_I[i]/N$
  end for
  $P_H[L] \leftarrow p_H$
  $P_S[L] \leftarrow p_S$
  $P_I[L] \leftarrow p_I$
  for `i=1 in L-1` **do**{Compute cumulative sums}
    $P_H[i] \leftarrow P_H[i-1] + p_H[i]$
    $P_S[i] \leftarrow P_S[i-1] + p_S[i]$
    $P_I[i] \leftarrow P_I[i-1] + p_I[i]$
  end for
  $m_H[L] \leftarrow 0$
  $m_S[L] \leftarrow 0$
  $m_I[L] \leftarrow 0$
  for `i=1 in L-1` **do**{Compute cumulative means}
    $m_H[i] \leftarrow m_H[i-1] + i \times p_H$
    $m_S[i] \leftarrow m_S[i-1] + i \times p_S$
    $m_I[i] \leftarrow m_I[i-1] + i \times p_I$
  end for
  {Compute global means}
  $m_{HG} \leftarrow m_H[L-1]$
  $m_{SG} \leftarrow m_S[L-1]$
  $m_{IG} \leftarrow m_I[L-1]$
  {Compute between class variance}
  $var_H[L] \leftarrow 0$
  $var_S[L] \leftarrow 0$
  $var_I[L] \leftarrow 0$
  for `i=0 in L-1` **do**
    $var_H[i] \leftarrow \frac{m_{HG} \times P_H[i] - m_H[i]^2}{P_H[i] \times [1 - p_H[i]]}$
    $var_S[i] \leftarrow \frac{m_{SG} \times P_S[i] - m_S[i]^2}{P_S[i] \times [1 - p_S[i]]}$
    $var_I[i] \leftarrow \frac{m_{IG} \times P_I[i] - m_S[i]^2}{P_I[i] \times [1 - p_I[i]]}$
  end for
  {Find i* as the value of i for which var[i] is max}
  $i_H \leftarrow 0$
  $i_S \leftarrow 0$
  $i_I \leftarrow 0$
  for `i=0 in L-1` **do**
    if $var_H[i] > i_H$ **then**
      $i_H \leftarrow var_H[i]$
      $threshold_H \leftarrow i$
    end if
    if $var_S[i] > i_S$ **then**
      $i_S \leftarrow var_S[i]$
      $threshold_S \leftarrow i$
    end if
    if $var_I[i] > i_I$ **then**
      $i_I \leftarrow var_I[i]$
      $threshold_I \leftarrow i$
    end if
  end for
  return $threshold_H, threshold_S, threshold_I$
end procedure=0