CENG466-Final Take Home Exam

Seda Civelek
Computer Engineering
METU
2237147
seda.civelek@metu.edu.tr

***Abstract*—This document is a report for Ceng466-Fundamentals of Image Processing course's Final Take Home Exam. The answers of each question can be found in subsections respectively. In order to implement algorithms, matlab is used for question 1 and 3 and python is used for question 2.**

***Index Terms*—number detection, animal segmentation, line detection**

I. Question 1

I found an article published by F. Budiman and E. Sugiarto that offering solution for extracting features of number from a given image [1]. My implementation developed based on this solution.

To be able to detect numbers correctly, we should have a clear image that the numbers are accentuated successfully. However, background differs from image to image. Also, given images were noisy. So, preprocessing step and getting a clear image was difficult. Binarizing image with some treshold suppresses the background and accentuates the numbers.

Just making binarized image is not enough since there was an image like 2.png in Dataset1. Since color around the numbers are so similar to the numbers color, I couldn't successfully accentuate the numbers. So, I decided to oversegment image using superpixels() function. I set the number of superpixels to be generated value to 500. Setting higher superpixel values increase the quality of the segmented image. After I over segment the image, I binarized image using 0.5 threshold. Higher threshold values doesn't correctly accentuates numbers. You can see the images after preprocessing step.

Fig. 1. Output of Step1 with given input image 1

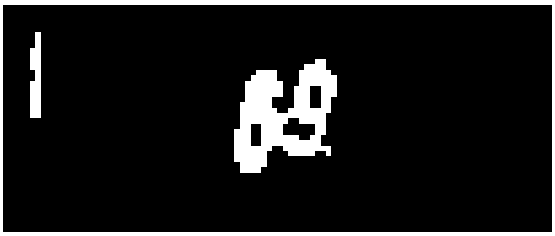

Fig. 2. Output of Step1 with given input image 2

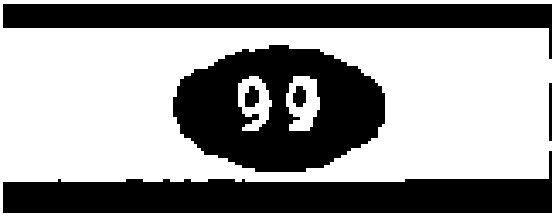

Fig. 3. Output of Step1 with given input image 3

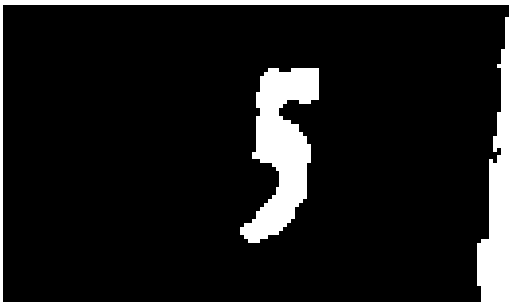

Fig. 4. Output of Step1 with given input image 4

Fig. 5. Output of Step1 with given input image 5

As you can see, other objects around number couldn't suppressed successfully, but the best outputs I could generate are these ones.

To implement minimal bounding box, we should detect connected components and label every distinct connected components. We can store the positions of connected pixels, so that we can determine minimum and maximum values to create minimal bounding box. In matlab, regionprops function with "BoundingBox" property returns the measurements of "BoundingBox" property. Returned object contains minimum x and y values of a bounding box and width and height information of the bounding box. I used that function to generate bounding box for every distinct connected component. Since numbers are counted as a distinct connected compenent,it draws the rectangles that encloses the numbers.

To skeletonized numbers, we can delete pixels on the boundaries of connected components without breaking hole component [2]. Hence, we can generate a skeletonized image. In matlab, we can do that operation just using bwmoprh(image,operation) function. Note that, operation argument of the function must be 'skel'. So, I decided to use that function to skeletonize accentuated numbers. You can see the output images generated so far.

Fig. 6. Output after Step3 with given input image 1

Fig. 7. Output after Step3 with given input image 2

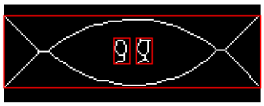

Fig. 8. Output after Step3 with given input image 3

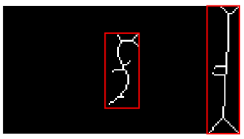

Fig. 9. Output after Step3 with given input image 4

From that part, I couldn't implement the algorithm since the outputs generated in step1-2-3 are not very successful. But I developed an algorithm to label numbers.

Algorithm 1: Number classifying

```
reshape each bounding box that encloses a number to
  a fixed size box;
value ← ∑ number of pixels a number contains;
 data ←predetermine number of pixels each number
 should contain in given fixed size box;
while data.hasNext do
    if value = data then
        label value as loopy or loopless;
        break;
    else
        data ← data.next
    end
end
```

There are really useful and powerful tools for number recognition like Tesseract [3]. Tesseract is an OCR engine that uses deep learning inside. In my solution the predetermined data is so weak comparing to deep learning model of Tesseract. Also, my binarization and region of interest algorithms don't work perfectly.

II. Question 2

To be able to over segment image to generate super pixels using mean shift algorithm, we should first discuss what mean shift clustering algorithm does. Simply, mean shift algorithm tries to convert data in a feature space. Intensity values of pixels are sampled and this samples are centralized into different spaces. Then, means of this samples are computed. This process is repeat until convergence is provided. Hence, samples that have similar features are assigned to same clusters. After clustering the samples, super pixels could be represented as a weighted graph. After constructing the weighted graph we can implement n-cut segmentation. Hence, we can separate animals from background.

To over segment image using mean shift algorithm, I firstly retrieve estimated bandwidth to use in mean shift algorithm. When I give sample number with high values, I get more clear result, but estimate bandwidth uses quadratic complexity. So, I decided to give value as 8000. This value if enough for clustering and also doesn't take much time. Also, I set the quantile value to 0.1. After getting estimated bandwidth I applied mean shift and get clusters of similar featured samples. Hence, I could over segment the image. You can see the related outputs below.

Fig. 10. Output image after over segmentation with given input image 1

Fig. 11. Output image after over segmentation with given input image 2

Fig. 12. Output image after over segmentation with given input image 3

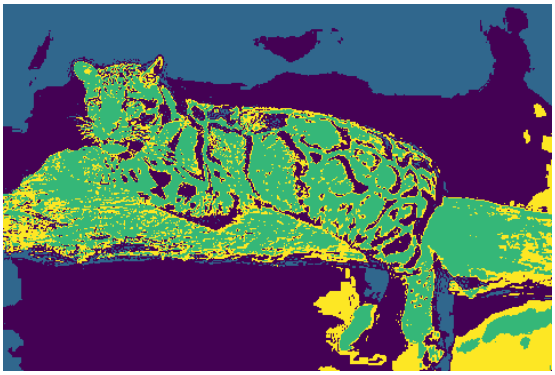

Fig. 13. Output image after over segmentation with given input image 4

Fig. 14. Output image after over segmentation with given input image 5

III. QUESTION 3

To detect roads from the given satellite images, we should firstly accentuate straight lines. After that, we can use Hough transform to detect parallel lines. To accentuate roads, I simply applied Canny edge detection to gray scaled image. I used bwareaopen to eliminate components that their size is less than 200. Then using a structuring element with the shape of straight line with 3 length, I applied closing operation. However, I couldn't get perfectly accentuated roads and suppressed areas. You can see the outputs below.

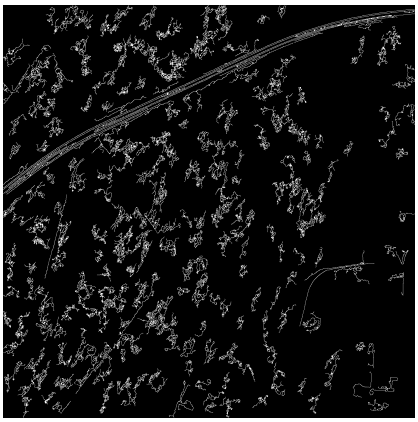

Fig. 15. Accentuated roads with given input image 1

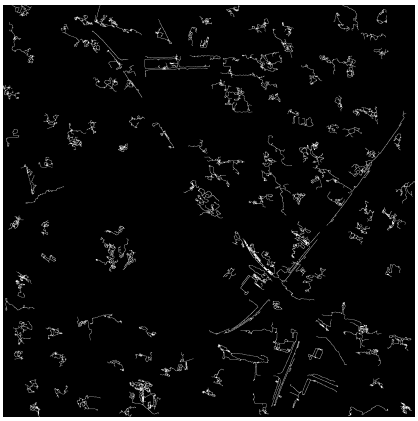

Fig. 16. Accentuated roads with given input image 2

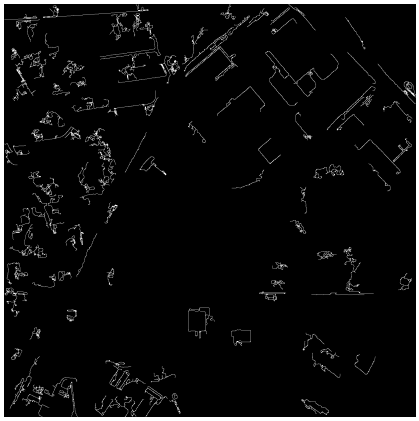

Fig. 17. Accentuated roads with given input image 3

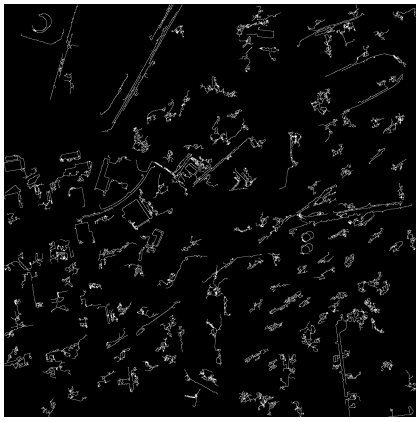

Fig. 18. Accentuated roads with given input image 4

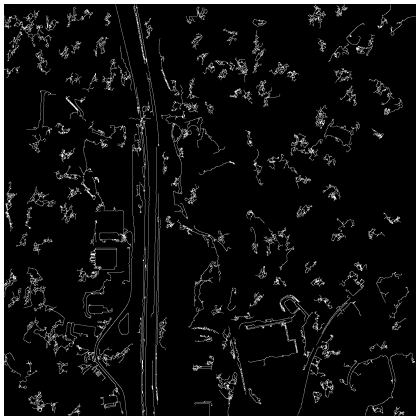

Fig. 19. Accentuated roads with given input image 5

After that, I used hough() function to get rho,theta and H matrix. This function applies Standard Hough Transform to a binary image. The function computes line parametric representation equation inside, that is:

$$rho = x * \cos(theta) + y * sin(theta) \quad (1)$$

Rho in the equation means the distance to the line along the vector that is perpendicular from its origin, and theta represents

the angle between that vector and x-axis [4]. H matrix gives values of rho and theta. So, using houghpeaks() function with H matrix gives us the peak values as a matrix coordinates of the peaks.I set number of peaks as 100 to get more peaks. Hence, the algorithm can detects the roads with small areas too. Finally, I used houghlines() function to extract line in the image using the matrix that is returned by houghpeaks(). I keep the longest line that is detected and draw them with green color on the original imaege. You can see the outputs below. Note that, there are meaningless line drawings too. I believe, if I could make the accentuation as clear as possible, I get more accurate results.

Fig. 20. Accentuated roads with given input image 1

Fig. 21. Accentuated roads with given input image 2

Fig. 22. Accentuated roads with given input image 3

Fig. 23. Accentuated roads with given input image 4

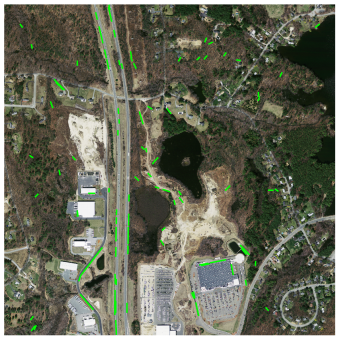

Fig. 24. Accentuated roads with given input image 5

REFERENCES

[1] Budiman, F., Sugiarto, E. (2020). Image Feature Extraction of Numbers and Letters Using Matrix Segmentation. Scientific Visualization, 12(1), http://sv-journal.org/2020-1/11/en.pdf. https://doi.org/10.26583/sv.12.1.11

[2] Morphology - Skeletonization/Medial Axis Transform. (n.d.). Skeletonization/Medial Axis Transform.

[3] Tesseract documentation. (n.d.). Tesseract OCR. https://tesseract-ocr.github.io/ http://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm

[4] Althoff, K., Abu-Gharbieh, R., Hamarneh, G. (1999, September). Automatic Line Detection. https://www.cs.sfu.ca/ hamarneh/ecopy/compvis1999_hough.pdf