Conference Paper Title*

1st Sahin Kasap
CEng
METU
Ankara, Turkey
e226456@metu.edu.tr

***Abstract*—This document is the report of Final Take Home Exam of Introduction to Image Processing Lesson. This document includes various image processing techniques.**

***Index Terms*—image processing, segmentation, loopy numbers,Hough transform, Road extraction**

I. Introduction

This report consists of three parts:
1-Algorithm to classify the characters according to whether they have loops or not.
2-Segmenting animals from the background.
3-Road extraction from aerial map images.

II. Part 1-Detecting loopy of loopless characters

A. Introduction

a) Difficulties: The numbers may be blurry.
The numbers may be in a different shape like circle or rectangle
The numbers may be not in the middle

b) Possible Solutions: Handwritten Digit Recognition Using Image Processing and Neural Networks [6]. This is for whole number detection but includes what we are doing also. But this method has machine learning in it. More than enough. Handwritten Digit Recognition Using Blob Detection and Machine Learning [7]. This document also includes what we are doing.
I am planning to enhance images, take them into bounding box, apply skeleton and get the number of loops.

c) Why I chose this one: Because this solution is easier and understandable.

d) Novelty: I don't think there is a novelty on this method.

B. Theoretical Presentation of Methods

a) Techniques And Suggestions: I used image thresholding for enhancing images, and then took the numbers into bounding box and then applied skeleton. I suggest you to seperate every number so we can look for every number seperately.

b) Pseudo Code: Here is pseudo code:
images = Get Every image in directory
enhance images
take number into bounding box
apply skeleton on every image
get Number Of Loops
Test with correct Results

Fig. 1. Thresholding 1.png

Fig. 2. Thresholding 2.png

c) Explanations of Parameter Settings: I used 177 for image thresholding, this number fit perfectly for these images, since the numbers were black or white.

C. Experimental Results

a) Outputs And Plots:

b) Parameters: I chose 115 for threshold, just because this was the best one.

Fig. 3. Bounding box 1.png

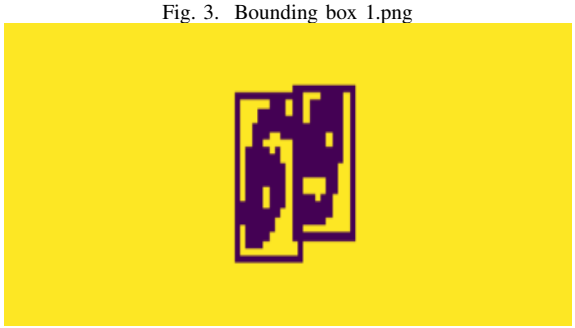
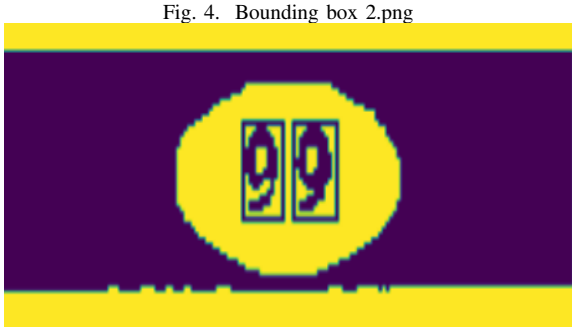


Fig. 4. Bounding box 2.png



c) Performance Measures and Complexity: The time complexity of thresholding is O(n) and for bounding box it is I think O(n*log(n)) but this depends on how many contours we have.

d) Implementation issues: I coded this on Python with VSCode, I used a 6-7 year old computer with intel i7-4th gen CPU. Even though the computer wasn't fast, the result came pretty fastly.

D. Conclusion

I think this method is quite hard since the numbers may come in different types. But, I did whatever I can do.

a) Critique: I could maybe make a more generic algorithm, that can even find the blobs on a handwritten text.

b) Comparison: The other programs generally used machine learning to completely identify the characters, so I can't compare them with mine.

c) Advantages and Disadvantages: Advantages: this method does not need machine learning etc. Disadvantages: this method may become useless in a different kind of picture set.

III. Part 2- Animal Segmenting From Background

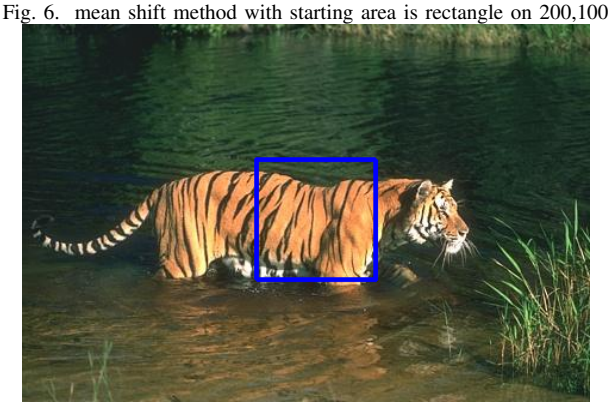
A. Introduction

This section of the report is about segmenting wild animals from forest and water background. Sample pictures have zebras, tigers, and other wild cat. In this section I tried to use Mean-Shift Clustering. Mean-Shift Clustering finds the center of mass on a data (which is image in this case), by calculating the center of mass in the shape and making the new center the point that is center of mass. So that, it iteratively finds the center of mass nearby. The problem with this method is, we can't find the exact place of a thing by just using this, since the animals or another thing may not always be in the center of mass. Actually the starting point of the iteration affects the result so much too. And we can only predetermine the shape of the cluster, so this is not the perfect method for animal segmentation.

As you see, the results are so different and we can't choose a shape that is the same as the animal shape so that we can segment. Because of these, I didn't choose to use mean-shift clustering. Instead, I used simple linear iterative clustering method with super pixels.

Fig. 5. mean shift method with starting area is rectangle on 0,0

Fig. 6. mean shift method with starting area is rectangle on 200,100



a) Difficulties: There are some difficulties for this algorithm:

- The animals can have similar colors as background.
- The animals can be too noisy, since they have camouflage skills.
- Some of the animals are colorful.

So, we can't solve this problem with an edge detection algorithm. We can't generally solve it by thresholding too. We need to find a better way.

b) Possible Solutions: We can use Superpixels and Slic algorithm, which will segment the image into smaller parts. This may be the perfect case for this problem. But the sizes of small segments will effect our results. [3]

Some other solutions include machine learning, like YOLO. [4]

Some other solutions include Threshold Segmentation, which maybe a good solution. [5] This approach chooses a threshold gap, which the animals are in and selects the areas with animals by selecting the areas that is inside this threshold.

c) Mean-Shift Segmentation: This method is a clustering method. The greatest mean value is chosen in this algorithm. This maybe useful in some images but in this report's images, I didn't use this method, since the outputs were not good enough.

d) Why I chose this one: I chose to use Superpixels and Slic algorithm. This is because the parts of the animals are segmented well by this method, and then the mean values

Fig. 7. input - 5.jpg

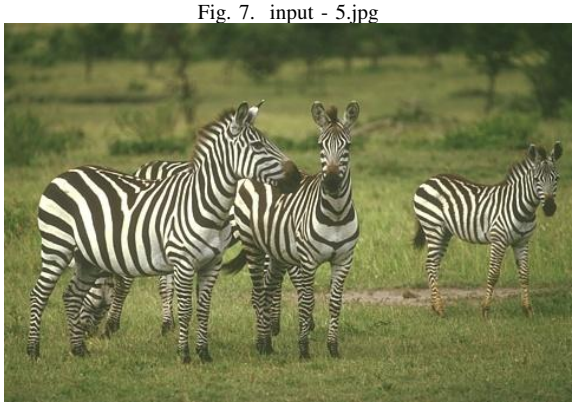


Fig. 8. input - 3.jpg

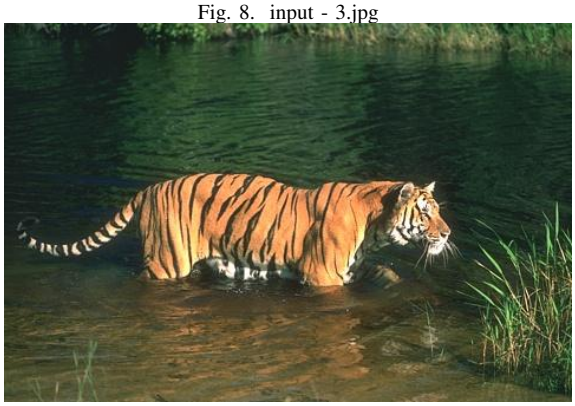


Fig. 9. 3.jpg segmented

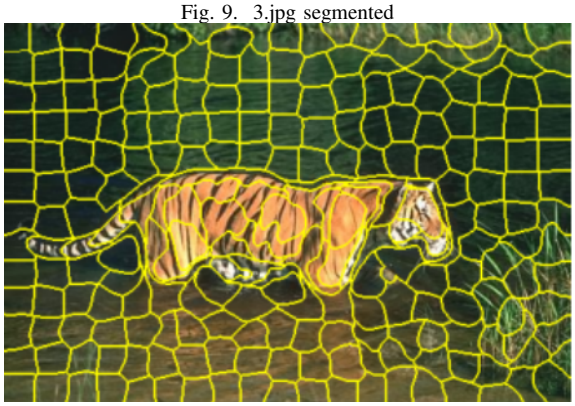
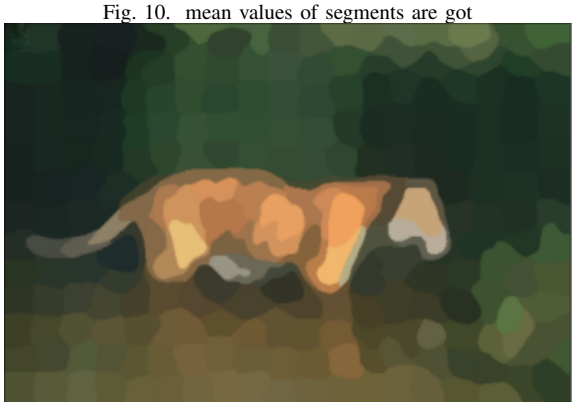


Fig. 10. mean values of segments are got



of the superpixels are taken and the image becomes easily segmentable.

e) Novelty: I don't think there is a novelty in this method.

B. Theoretical Presentation of Methods

a) Techniques And Suggestions: The Slic algorithm is used in this process. The Slic method makes the image segmented to parts of the number that we have given. And then I get the mean values of this segments and I get the animals with a cluster that is inside the image. I suggest you to make number of segments around 250-300, and always use the image multichannel, because a grayscale image makes the animal colors similar to the background.

b) Pseudo Code: here is the pseudo code:

img = Read the image
segments = Get Slic Segments Of img
segmented image = Draw Average Colors Of segments
detect edges on segmented image
clear background and put the animal on image
return the cleared img

c) Explanations of Parameter Settings:

C. Experimental Results

Some of the inputs:

a) Outputs And Plots: After applying Slic segmenting, we get a segmented image. And after that we can get the mean values of every superpixel-segment. After these processes, we can easily get the result.

b) Parameters: I have chosen the number of segments to be 250, since the tiger was in the size of this

c) Performance Measures and Complexity:

d) Implementation issues: I coded this on Python with VSCode, I used a 6-7 year old computer with intel i7-4th gen CPU. The computer was slow, so debugging and testing took longer, and since python is not compiled, detecting errors was too hard.

D. Conclusion

It is hard to detect animals on a noisy background but we can somehow overcome this issue by segmenting image with Slic technique.

a) Critique: This program may not work with different kind of images, since we need more data and testing for this.

b) Comparison: The other methods uses more data, so I couldn't fully compare. In the other methods, thresholding and edge detection is used. There are better and worse methods.

c) Advantages and Disadvantages: The advantage of this program is, it is quite useful to get a blob and an easy to implement method. The disadvantage is, it is not generic enough for a quite good rate of pictures I think.

-

IV. Part 3 -Road Extraction From Aerial Map Images

A. Introduction

Road extraction from aerial images is an important process when it comes to ruling big cities or making a map application for human use. When processing an image for road detection, there are some key points that I think very important:

- Roads are generally continuous
- Road colors are different from the other colors and roads are generally grayscale
- Roads have parallel lines

By considering these, we need to get the roads from an image.

a) Difficulties: -The images are so big, so a bad algorithm will work for a long time, I needed to build a good one. -The roads are not obvious all the time, since the shadows, cars, other noises may be on the road. -Roads may not be flat, we need to consider the curves as well.

b) Possible Solutions: I found some solutions for this problem by making a small literature survey:
-Road detection from high-resolution satellite images using artificial neural networks [1]
This solution is not good for us, since we don't have enough data for constructing a ANN.
-ROAD DETECTION FROM HIGH AND LOW RESOLUTION SATELLITE IMAGES [2]
Actually this is a good solution for exactly this problem, it includes low-high pass filtering, Sobel and Prewitt edge detection algorithms are applied, and then some machine learning classification algorithms are applied. This was quite useful, I tried sobel, but couldn't manage to have a good result.
-My way: removing the non gray, blackish and whitish colors and then applying Hough line detection algorithm. Since the roads are generally gray, I removed the other colors and then applying Hough, I got the lines. The result is further edited for continuous roads.

c) Why I chose this one: Because the other algorithms were generally machine learning algorithms, which need to have lots of data of roads. But in this experiment, we don't have any data that will help us get the roads. So I made my own algorithm.

d) Novelty: This algorithm doesn't need any large data, and I think very easy to understand.

B. Theoretical Presentation of Methods

a) Techniques And Suggestions: I used a kind of color thresholding and applied Hough and processed the result for better solutions. I suggest, using sobel and canny edge detection algorithms if possible, since they can make a good edge map.

b) Pseudo Code: The algorithm is like this:
img = read image
roads = Accentuate The Roads(img)
roads = Apply Hough (img , roads)

Fig. 11. 5.png, an input for our method

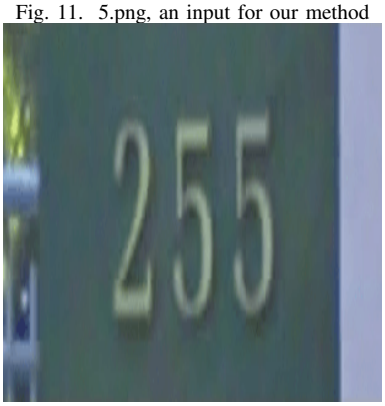


Fig. 12. output for 5.png

roads = Post Process (roads)
return overlaid roads on image

c) Explanations of Parameter Settings: While accentuating the roads we need to prepare the image for road detection. The images are generally too noisy, but the roads are smooth, having only cars and a few shadows. Without preparing, we can't apply edge detection easily.
So in this process, I cleared all the colors except grayish ones. For this purpose, I cleared the pixels that have a dominant color(The colors in pixel are not similar, for example closer to green) (max of pixel - min of pixel ¿20) and I cleared the pixels that are near black (min of pixel ¡80) and that are near white (min of pixel ¿180).
And then in Apply Hough method, I applied Hough Transform and got the straight lines. Then drew the straight lines on the roads image. I chose the min line lengths as 100 and chose max gap between lines 8. I actually tried and saw these were the best solutions.
And then in post processing, I deleted the lines that doesn't have any neighbors, so they are alone, since the roads are generally long and continuous.
And in the overlaying part, I just make the roads on the image green, so that they are obvious.

C. Experimental Results

a) Outputs And Plots: As you can see from the images above, the roads are detected and dyed to green. The results

Fig. 13. 4.png, another input for our method

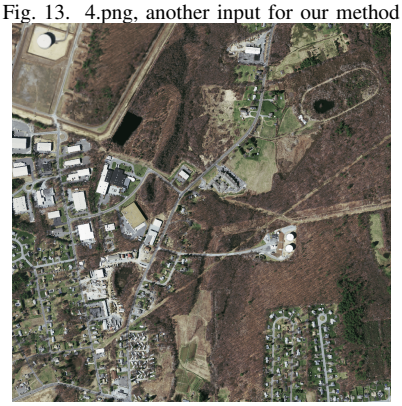
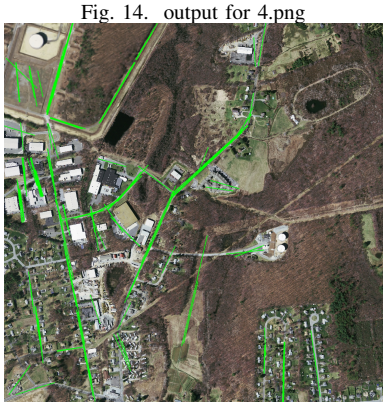


Fig. 14. output for 4.png



are not perfect.

b) Parameters: The parameters for the algorithm were chosen the same for all the images because I tried to make a generic algorithm. These were told before, the same parameters were chosen.

c) Performance Measures and Complexity: The time complexity for color segmentation is O(N), and the time complexity for Hough transform is O(N$\hat{3}$), but my method took longer, since my method is written in Python, which is way slower than OpenCV which is written in C and C++.

d) Implementation issues: The implementation took a lot longer, because of the device that this project is tested was not a good one. Only Python was used. I coded the program in VS Code, with a computer which is 6-7 year old and runs with Intel i7-4th generation.

D. Conclusion

a) Critique: I think my algorithm was a nice one, but it had a lot of missing things of course. I could make an edge detection, I tried but I mustn't have decided not to use one. By some thresholding and adjustments, I may have found a good edge detection. Other than that, the results will not be consistent if the image is taken on a night time, and if the road color is different.

b) Comparison: The other algorithms that I have found were generally using machine learning, since roads may differ in a lot of ways. But this method uses only a few data to get the roads from images. I actually couldn't find a similar project that uses a few data to get the roads, so I can't compare according to the speeds, results etc.

c) Advantages and Disadvantages: The advantages of this project are the long and continuous roads appear quite good, when the images are taken in the daytime it works very well, and this algorithm works in rural areas better. The disadvantages are, this algorithm can't be used on dissimilar images, it is not good for a crowded city since there are so many buildings with smooth walls that may be understood as road.

REFERENCES

[1] M. Mokhtarzade, M.J. Valadan Zoej
Road detection from high-resolution satellite images using artificial neural networks, 2007, Pages 32-40,http://www.sciencedirect.com/science/article/pii/S0303243406000171
[2] You can look here
[3] https://infoscience.epfl.ch/record/149300
[4] https://ieeexplore.ieee.org/abstract/document/8575770
[5] Click Here
[6] http://iaeng.org/publication/WCE2007/WCE2007_pp648-651.pdf
[7] I. Penev, M. Karova and D. Todorov, "Handwritten Digit Recognition Using Blob Detection and Machine Learning," ANNA '18; Advances in Neural Networks and Applications 2018, St. Konstantin and Elena Resort, Bulgaria, 2018, pp. 1-6.