Road Segmentation from Satellite Images using Support Vector Machine Classification

Daniel Valério Sampaio, Yves Bieri
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—There is no abstract.**

I. Introduction

The goal of this paper is to introduce a new road segmentation algorithm for urban areas from satellite images. Road detection is important for several applications from autonomous driving to tracking of road change and urban planning.

The idea of road segmentation is not a new one and has been implemented numerous times, although with different success. The great difficulty in urban areas is the occurrence of trees and cars partially blocking the view on roads and thus making detection much harder. Additionally parking lots and railways have a lot of identical features to roads which leads to them being misclassified.

Our approach consists of extracting 19 features from out satellite images and using those to build a Support Vector Machine (SVM) classification model. Our features contain the original image, a smoothed version of said image, parts of the Singular Value Decomposition (SVD) and a region map, based on a skeletonization approach using an euclidean-like distance map, to reveal the underlying road structure.

The image preprocessing and feature extraction is implemented in *python* with the help of mostly *numpy* and *cv2*. Afterwards the features are fed to *MATLAB* which is used to produce our SVM classifier. We decided on this approach because *MATLAB* is faster and easier to run on the ETH cluster. The predicted results are then fed again to our *python* program for post-processing and to produce our final results.

Report's Structure

In the next section we describe the feature extraction used in our approach, as well as our detailed implementation. The final results achieved are shown in Section IV. We conclude the paper with a comparison to related work in Section V and give ideas for future improvement in Section VI.

II. Models and Methods - Feature Extraction

To detect roads from satellite images we decided to use Support Vector Machines (SVM) for classification. We were given 100 training images with corresponding ground truth, which we used to classify 50 test images in a second step.

For the data-driven classifier to be effective, we decided on a set of key features which we extracted from both the training images and the test images. Contrary to our initial implementation, we decided to not consider every single pixel in the image but to form patches instead. We mostly worked with 4x4, 5x5 and 10x10 patches and eventually decided on keeping the 5x5 patching, which offered a good computational speedup without loosing a lot of information of the image features.

The training and test images were taken from urban areas, which apart from roads contain mostly buildings, trees and cars. A road is a long and compared to its length thin object, often in horizontal or vertical direction and colored in a shade of gray. By concentrating on those key characteristics of a road we build a set of features which we will extract to train our model.

A. Original Image

The first feature we use is the color in the image patch. It is only natural to include this feature, as a road is usually gray. We can decide whether an image patch is gray by looking at its *RGB* values. They not only help us to find roads that look gray, but also let us recognize different objects in other colors. On one of the images for example there is a pool next to a house, which naturally is blue. Of course, blue objects cannot be roads, and are therefore excluded right from the beginning. In contrast, if the values of red, blue and green all are approximately the same, the resulting color is some kind of gray. So adding the color is a first useful feature to build our model.

B. Smoothed Image

As a second feature we decided to add the standard deviation of the color. We stated earlier, that a pixel is gray, if its three *RGB* values are close. This means that if the standard deviation is small, the red, blue and green values are all close together which is a strong indicator that the pixel is gray.

After a few trial runs on the training images we noticed that there are a lot of small imperfections which perturb our results. Cars, trees and shadow lead to a pretty big error rate. We concluded that we have to smoothen the images to get rid of this source of noise.

(a) original

(b) mean-shift filtered

Figure 1: Smoothing the original image

To do that, we used a mean-shift filter, also used in a previous work by Banerjee at al.[1]. This filter allows for edge preserving smoothing of our image, which is essential, considering that roads have distinct borders. For a neighborhood around the pixel the new spatial center and mean color value are calculated and at the end assigned to the pixel. As parameters we used a spatial radius and a color distance of 10, which seemed to produce reasonably good results in our experiment.

In addition to the previous mean and standard deviation *RGB* values of the original image, we added the ones obtained by the mean-shift filtered image as our third and fourth parameters of our feature vector.

Besides the original image we also wanted a smoothed version with less noise, as we are convinced that the combination of those two pictures are a good basis for road detection.

C. Continuity Score

One of our key problems in road detection was the lack of completeness with respect to the classification of roads in the image. As a result of that we introduced a score, which should indicate if a patch in the image is probable to be part of a road, by considering how many cross-neighboring patches are gray. First we check if the patch to be classified is gray, by thresholding the standard deviation of the before calculated standard deviation of the three color channels. If so, we do the same for its four cross-neighbors and, in the final step, calculate a score based on how many of these patches are considered gray: 0 if ≤ 2, 0.5 if 2 and 1 if ≥ 3. Given the simplicity of this score, it was not clear how much we could gain out of it. Nevertheless, we expected some positive effect on the classification. Even though it was not a major improvement, there were some holes filled and therefore our initial thoughts proved to be right.

D. Region Map

Our second to last, and most complex feature is based on the paper by Gaetano et al. [2]. The morphological analysis the paper is based on assumes that roads are objects that spread over a rather large part of the image and that have a comparatively small width compared to their total length. Building a morphological skeleton is then the base of the algorithm and consequently for this feature. Ideally a road can afterwards be approximately represented by a branch of the skeleton.

Lets take a look at a brief overview of the algorithm:

1) Edge Detection
2) Distance Map
3) Skeleton
4) Skeleton Pruning
5) Road Reconstruction
6) Watershed Transformation

First edge detection is conducted on the image. The result is then used to compute an euclidean-like distance map (EDM) which then is used to extract the morphological skeleton. Skeletonization although is not very robust, making it therefore desirable to improve it in a post-processing step. To do that, firstly the lines are thinned to a one-pixel width. As we still have too much noise, we also need to remove small branches that are very unlikely to be part of a road. This step is called skeleton pruning. Finally double thresholding is applied to classify valid branches in the skeleton. We consider the average distance in the distance map of every pixel in a skeleton branch and also the overall average distance of every pixel. In the end a watershed transform is applied to reconstruct an image containing various regions, indicating if there is a possible road or not.

1) Implementation Details:

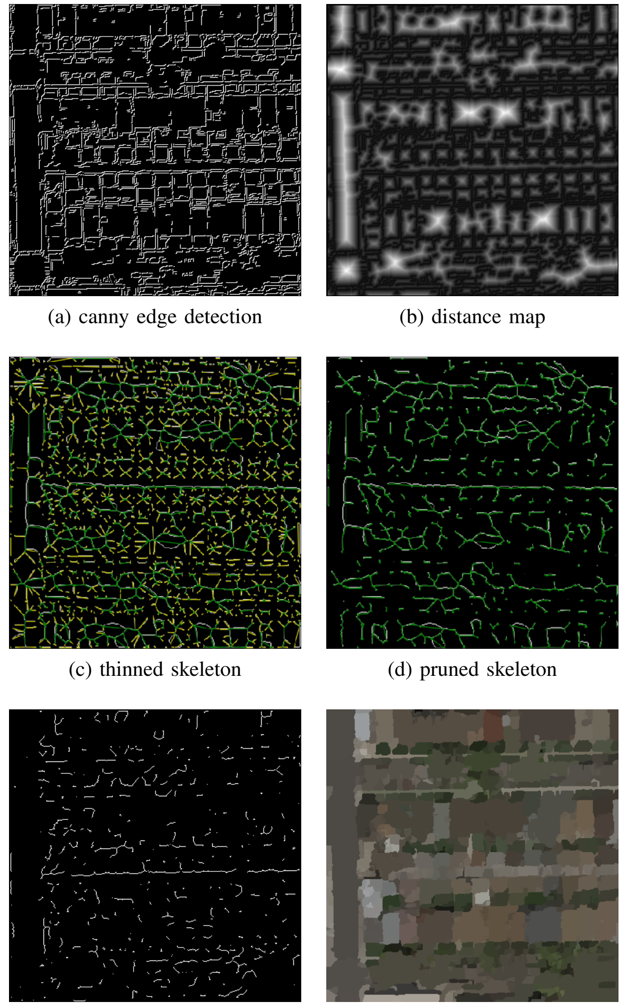
The first step is edge detection. Here, we used a *python* implementation of the canny edge detector, where we first transformed the original into a gray-scale image. Although canny edge detection has a higher computational burden than e.g. a sobel edge detector, we decided to use canny, as the results were more accurate.

In the second step we prepare for the morphological skeletonization. With help of the previously generated edge map we create an euclidean-like distance map. We start out by giving all edge pixels from the canny edge map a distance of 1. For the future development of the distance map we consider the 8-neighborhood of each edge pixel (8-neighborhood meaning the 8 pixel that form a one-pixel wide border around the center pixel). The four horizontal and vertical neighbors are given a distance of +1, meaning that if the center pixel had distance 1, they get assigned distance 2. For the four diagonal pixels we add $\sqrt{2}$ to the center pixel's value. By assigning a higher value to the diagonal pixels we generate smoother curves and preciser distance values for further computations. We continue those steps iteratively until all pixels are assigned a value. From now on we refer to the result of this step as the euclidean-like distance map (EDM).

Now, in the third step, we create the morphological skeleton out of our EDM. For this we use a crest line following approach, as described in paper [2]. With the crest line approach we want to find a relatively simple skeleton. One single straight line is enough to represent a road pretty accurately. As in urban areas most roads are either horizontal or vertical without curves this approach gives us a pretty good result.

Now we look at all points in the EDM which have a maximum of two neighbors with higher value. These points are considered crest line points. With this approach we recursively add the neighboring points with the steepest ascent to our crest line. Finally we apply morphological thinning to reduce our skeleton to a one-pixel width.

The fourth step consists of pruning the skeleton, as the road skeleton obtained in step 3 has a lot of small branches that most likely are not part of a road. They therefore generate more noise than benefit to the scene, creating a more splintered final image. By pruning the skeleton we can enhance the representation of our roads by getting rid of some of this splintering. As you can see in our skeleton map in Figure 2 we mark all leave nodes and their connections yellow and all intersection nodes as well as their connections to other intersection nodes in green. Finally we consider all nodes with connection degree of one or lower, i.e. all nodes and connections in yellow, and remove those nodes together with the respective edge/branch.



(a) canny edge detection

(b) distance map

(c) thinned skeleton

(d) pruned skeleton

(e) final skeleton

(f) region map

Figure 2: Building the Region Map

In the final and concluding step we do road detection and map reconstruction. For this we use a double thresholding method. Recall that we can describe a road as a branch of the skeleton with a large length compared to its width.

Therefore we can introduce two main cost/score functions, that we will later use for thresholding. The first score is an average over the distance map on the skeleton branch's points S_i

$$\{D_i\}_{avg} = \sum_{p \in S_i} \frac{\text{EDM}(p)}{\text{card}(S_i)},$$

where i indicates the i-th skeleton branch. The second one is the so-called road skeleton score (RSS):

$$RSS_i = \frac{\{D_i\}_{avg}}{\text{card}(S_i)}$$

The paper on which the method is based on [2] proposes to set the threshold to $threshold_{RSS} = 0.04$ and $threshold_D = 12$. The points in the skeleton satisfying both conditions, i.e. having a score below these thresholds, are then kept for the final skeleton.

Lastly we apply a marker based watershed transform on the distance map with the proposed skeleton to reconstruct a region map. The watershed transformation summarizes large common patches that we color with the help of our constructed mean-shift filtered image in every single region. Finally we obtain a segmented region map where each connected component contains exactly one connected skeleton segment.

E. Singular Value Decomposition (SVD)

In a last and final step we extracted the first, second and third rank of the Singular Value Decomposition (SVD). As SVD in general extracts useful information, in our case particularly the first two ranks, we decided to use them and therefore complete our feature extraction.

III. Models and Methods - Post-Processing

Based on an earlier lecture on *Visual Computing* at *ETH Zurich* by Prof. Markus Gross we knew that binary morphological closing could come of help, if we had problems with getting straight lines in the classification. Therefore we decided on doing a post-processing step after the image was classified.

We could see significant improvements with respect to completion, but had difficulties in problematic areas, as the binary morphing amplified erratic regions as well.

IV. Results

Our final result, an SVM model trained with 95 out of the 100 training images given (five contained nearly no roads and would have compromised the general case), achieves a correctness of up to 81.48%, according to the lecture's online evaluation on *kaggle*.

V. Conclusion - Comparison to Related Work

To know where we lie in comparison to related work, we looked at a few papers. One of them is a comparative study by Anastasiia Volkova and Peter W. Gibbens [3], where different road extraction techniques were analyzed — one of them being an SVM classification model too, as in our approach. The proposed SVM classifier in [3] gave high completeness results (75-91%) — a value we can relate to with our classifier — but achieved very low correctness values (35%) — in comparison: our classifier achieves more than 81%. This proves that our feature extraction is richer in information, i.e. there is more relevant detail in our data.

There are other related works that are helped by human interaction, as e.g. a work done by Yuan and Cheriyada [4]. There, we already have a vector map — we can also say a one-pixel road map — from which roads are then built upon. Our approach relies on no user interaction and is therefore fully automatized.

VI. Future Work

As we can see in the final result, there is still potential for improvement, both in the feature extraction and in the post-processing. The score function explained in Section II-C for instance could be extended to work with a greater neighborhood to deal with streets that are covered by small trees, cars or shadows from buildings.

In the post-processing step, a second skeletonization attempt could be done, to extinguish erratic classification as we can see in the sub-figure (d) below.

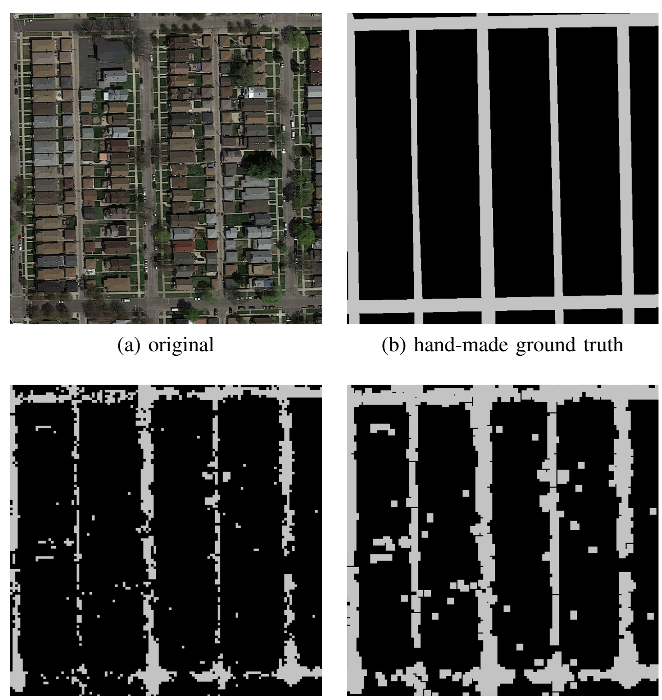
Acknowledgments

The authors thank the producers of Game of Thrones for being such an inspiration in the early Monday mornings.

References

[1] B. Banerjee, S. Buddhiraju, and B. K. Mohan, "Automatic extraction of road segments from high resolution satellite images: A region and boundary based approach," in *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, ser. ICVGIP '12. New York, NY, USA: ACM, 2012, pp. 59:1–59:8. [Online]. Available: http://doi.acm.org/10.1145/2425333.2425392

[2] R. Gaetano, J. Zerubia, G. Scarpa, and G. Poggi, "Morphological road segmentation in urban areas from high resolution satellite images," in *2011 17th International Conference on Digital Signal Processing (DSP)*, July 2011, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6005015&isnumber=6004864



(a) original (b) hand-made ground truth

(c) classified image (d) post-processed image

Figure 3: Our final Result

[3] A. Volkova and P. W. Gibbens. (2015) A comparative study of road extraction techniques from aerial imagery. [Online]. Available: https://www.academia.edu/15375100/

[4] J. Yuan and A. M. Cheriyadat, "Road segmentation in aerial images by exploiting road vector data," in *Computing for Geospatial Research and Application (COM.Geo), 2013 Fourth International Conference on*, July 2013, pp. 16–23. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6602035&tag=1