Classification of Line and Character Pixels on Raster Maps Using Discrete Cosine Transformation Coefficients and Support Vector Machines

Yao-Yi Chiang and Craig A. Knoblock
University of Southern California
Department of Computer Science and Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
yaoyichi@isi.edu, knoblock@isi.edu

Abstract

Raster maps are widely available on the Internet. Valuable information such as street lines and labels, however, are all hidden in the raster format. To utilize the information, it is important to recognize the line and character pixels for further processing. This paper presents a novel algorithm using 2-D Discrete Cosine Transformation (DCT) coefficients and Support Vector Machines (SVM) to classify the pixels of lines and characters on raster maps. The experiment results show that our algorithm achieves 98% precision and 85% recall in classifying the line pixels and 83% precision and 96% recall in classifying the character pixels on a variety of raster map sources.

1. Introduction

Today, more and more sources provide raster maps on the Internet (e.g., Google Map, Yahoo Map, and MSN Map). For applications to exploit the information on these raster maps, the first step is to automatically recognize the lines and the characters on them. For example, a geospatial conflation system [1] utilizes the street information (i.e., the orientations and intersections of streets) to align raster maps with satellite imagery. To manually extract lines and characters from a raster map, users have to manually select the pixels of lines or characters. This is a very tedious and time consuming process. In this paper, we present an algorithm, which uses the 2-D discrete cosine transformation (DCT) coefficients and support vector machines (SVM) to automatically classify pixels on raster maps into line or character classes. The classification results (i.e., line and character image) can be further used in vectorization components and OCR components to pull out the information such as the geometries and names of streets from the raster maps.

DCT has played an important role in many texture classification applications for its outstanding ability to generate distinct features for different texture representations [2]. It transforms an image into the frequency domain where the strength of each frequency is represented by one of the DCT coefficients. Within a local area (i.e., a DCT window), the textures of the foreground and the background are different since the colors of the background are consistent while the colors of the foreground change frequently. Among the foreground objects, lines and characters also have different texture representations as shown in Figure 1.

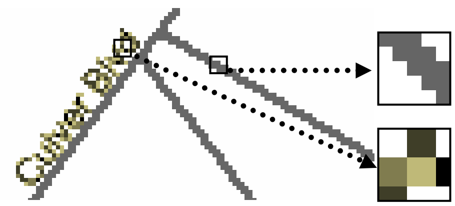

Figure 1. The areas of lines and characters on a raster map (TIGER/Line Map)

In our algorithm, the classification is pixel-based. Initially every pixel is automatically classified into background or foreground classes using a threshold. The foreground pixels alone are sent to the SVM for line or character pixel classification. The training of our SVM model will be described in Section 3. Support Vector Machines are widely used in many research fields that require classification [3], especially in the area of pattern recognition. In the training process of our algorithm, the SVM constructs hyperplanes in a multidimensional space (i.e., the feature space of the DCT coefficients) that separates pixels of different classes (i.e., line and character classes). SVM can quickly generate a model from a small set of training data and it is robust to noisy data.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes our approach to classify line and character pixels on the raster map. Section 4 reports on our experimental results and Section 5 is the discussion and future work.

2. Related Work

Much research work has been performed in the field of text and graphics separation from documents [4, 5, 6, 7, 8]. Some of the previous work assumes that the line and character pixels are not overlapping [5, 6, 7] and they extract characters by tracing and grouping connected objects. Others detect characters from more complex documents (i.e., characters overlap lines) using the differences of the length of line segments in characters and lines [4]. Li et al.[8] separate the characters from lines and rebuild both of them within a local area on the raster map, but assume that all character areas are detected beforehand. These previous algorithms perform geometrical analyses of the foreground pixels and focus on finding characters. Our algorithm detects the line and character pixels by exploiting the differences between line and character textures in the frequency domain, and we do not make any geometrical assumptions about the lines or characters.

To achieve the best results, the previous algorithms require users to setup the geometrical parameters. These parameters are not straight forward and they are source dependent. Some examples of the geometrical parameters are: the size of a character, the length of a word, the gap between characters and between words. In our algorithm, there is no parameter adjustment required. We utilize a comparatively simple training process to build the SVM classifier. The SVM classifier can be applied on different maps even if they have not been seen during the training process.

Furthermore, in order to trace the geometries of lines and characters, the foreground pixels need to be correctly separated from the background. The previous algorithms, however, usually assume that the foreground is already separated from the background or the foreground can be easily separated using histogram thresholding. This assumption does not hold on most of the computer generated raster maps. Computer generated raster maps usually contain shadows around the characters to enhance the visualization. The colors of the shadows and characters are similar and it is difficult to remove the shadows by histogram thresholding. If the shadows are not completely removed before processing, the geometries of characters are not correct for geometrical analysis. Our algorithm finds the continuous colors in the raster map to separate the foreground from the background, and the SVM classifier works well even if the shadows are not removed since it utilizes the differences of the texture representations between lines and characters.

3. Approach

The overall approach for this paper is shown in Figure 2. The first stage is to separate foreground pixels from the background. The second stage is to classify line and character pixels among the foreground pixels.

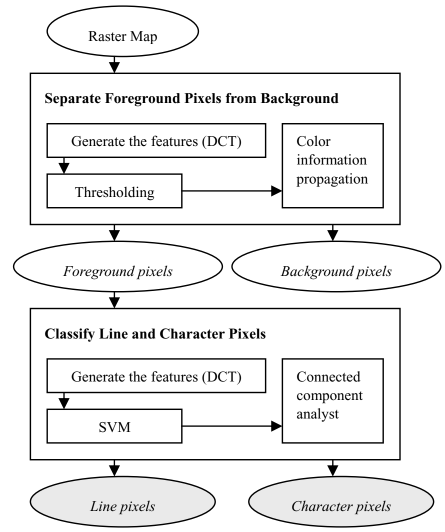

Figure 2. Overall approach

3.1. Separating Foreground Pixels From the Background

To separate foreground pixels from the background, we first generate the DCT coefficients for pixels on the input raster map using a 3-by-3 window. For efficiency, we subsample the input raster map by scanning the pixels on alternative rows/columns. The window size is intended to be small to ensure any object with continuous tones within a 3-by-3 block will be classified as background. For example, if a park is represented using a color green and it is larger than our DCT window, most of the green pixels will be classified as background except the pixels near the park boundaries.

The distribution of energies among the DCT coefficients of the background are significantly different than the foreground. The property of consistent color in background results in low (near 0) energies for high-frequency DCT coefficients while the frequent color changes of the foreground result in higher energies for these coefficients. We discard the DC term (i.e., the frequency $\{0, 0\}$) since it represents the average pixel value of the DCT window and it is color dependent. We check the summation of the absolute value of the other 8 DCT coefficients using a preset threshold of 0.0001 which allows for only minimal variation in the color. The pixel is classified as a background pixel if the summation is smaller than the threshold. After we scan through the entire map and perform the thresholding, we have two images: the background image and the foreground image. There are still some mis-classified pixels, since some of the areas used to classify the pixels contain both background and foreground areas as described in the park example. We eliminate these false-positives by exploiting the color information on the original map.

Different map sources use different colors to represent objects, however, a general property of the color usage is observed – a color is used only once to represent either the foreground or the background. For example, a TIGER/Line map uses a total of 22 colors on a particular map we tested. Among them, 2 are used for lines, 18 are used for characters, and others are used for background. Thus, we assume that a color can only be either foreground color (i.e., lines and characters) or background color and propagate the color information to update the final results. The probability of a color to be a background color is the number of pixels of the color in the background image divided by the total number of pixels of the color in the original map. Since we classify the pixels into 2 classes, according to the Maximum A Posteriori probability (MAP) rule, a color belongs to the background class if:

$$P\left(Background|Color\right) \geq 0.5 \qquad (1)$$

3.2 Classifying Line and Character Pixels

After we have the foreground pixels, we want to further classify the foreground pixels into line pixels and character pixels. Characters are generally more complex than lines, so the energies of high-frequency DCT coefficients of character textures are higher than the energies of these coefficients of line textures.

We generate the DCT coefficients for each foreground pixel and send them to the SVM for classification. We scan the foreground pixels using a larger DCT window (i.e., 5-by-5) to capture the texture differences. Instead of using all DCT coefficients as features for classification, we use only 22 out of 25 high-frequency DCT coefficients which have the highest differentiating power to classify line textures and character textures. This number was determined experimentally. Although the SVM should be able to figure out how to weight each feature, our experiments show that discarding the low-frequency DCT coefficients can help the SVM to reduce the computation time and enhance the accuracy when generating the training model.

To train the SVM model, we need training samples for line and character pixels. For the character training samples, we use a Mapquest Map that has characters in all different orientations with shadow pixels and manually remove line pixels from it. For the line training samples, we use two street maps from Google Map and one from ViaMichelin Map and manually remove the characters from them. We use more than one map for line samples because we want the training samples to cover all different orientations of lines – straight lines (i.e. horizontal lines, vertical lines, and diagonal lines) and curved lines.

Based on the 22 DCT coefficients, the SVM classifies foreground pixels into either line class or character class. Now we have one image of lines and one image of characters from the input raster map. Since the training samples are character-only-images or line-only-images, the pixels which have the DCT windows covering line areas and character areas are sometimes mis-classified. To clean up the results, we perform a 2-phase simple connected component analysis. A small connected object in either the result image of lines or the result image of characters could be a "missing part" of a larger connected component in the other image. For example, a line segment might be misclassified as character pixels because there are characters in its DCT window. If we move that line segment from the character image to the line image, it will connect to the original bigger line and the result is improved. If it connects to nothing in the line image, it is not a "missing part" of anything and we move it back to the character image. So we first find connected components which are smaller than the DCT window in the character image and move them to the line image. Then we perform the same analysis on the line image and again move small connected components to the character image. Thus, the result images are updated and only small connected components which are part of a larger connected component will be moved to the other image.

4. Experimental Setup and Results

We tested 9 online map sources as shown in Table 1. The test maps are all disjoint from the training set and are randomly selected covering different countries. The SVM classifier is built prior to the experiment using three maps from three of the sources as described in Section 3. To evaluate the results, we manually extract an image of all lines and an image of all characters from every test map. If there are pixel overlaps between lines and characters, the pixels will appear in both images. We compared our algorithm against the work of Cao et al.[4]. We tried our best to optimize the parameters in Cao's algorithm although the parameters may not give the absolute best results. Moreover, their algorithm uses histogram thresholding to separate the foreground pixels from the background, which is not valid on our test maps. Thus, we sent the foreground pixels only to their algorithm instead of sending the original raster maps. The outputs of our algorithm are two images of either all line or all characters for each input raster map. The output of the work of Cao et al. is an image of all characters since their work focuses on character extraction. We obtain the line image by computing the pixel difference between the character image and the foreground image.

The precision is defined as the number of correctly classified pixels in the result images of lines/characters divided by the total number of pixels in the manually extracted images of all lines/characters. The recall is defined as the number of correctly classified pixels in the result images of lines/characters divided by the total number of pixels in the result images of lines/characters.

Our results show that we can detect almost every character and large portions of lines. The first thing to notice is that, because we counted the overlapped pixels twice in

Table 1. Experimental results comparing our algorithm against the work of Cao et al.

Map Source	Precision/Recall of Classification			
	Line Pixels		Character Pixels	
	Ours	Cao's	Ours	Cao's
A9	**99/91%**	95/91%	**79/98%**	77/85%
MSN	**99/79%**	91/87%	**75/99%**	81/86%
Google	**99/99%**	95/99%	**98/99%**	95/72%
Yahoo	**95/91%**	70/96%	**91/92%**	88/30%
Mapquest	**99/78%**	88/73%	**84/98%**	76/85%
Map24	**95/74%**	97/70%	**73/96%**	70/98%
ViaMichelin	**83/34%**	44/57%	**87/96%**	90/68%
Multimap	**89/82%**	98/64%	**63/90%**	46/97%
TIGER/Line	**99/94%**	97/89%	**83/99%**	67/90%
Average	**98/85%**	85/82%	**83/96%**	71/71%

both line and character class, as long as there are overlapped pixels, the recall of the two classes will not be complimentary. The precision is lower in the character classification than in the line classification but the recall is higher. This is because if the line pixel is near/overlapped with the character pixel, it will be classified as a character pixel since it's neighborhoods are complex textures rather than linear textures. In the comparison, although Cao's algorithm tried to rebuild the characters and lines, the results show that our algorithm still outperformed theirs on the precision and recall. This is because when their algorithm extracts the characters touching lines, many line pixels that do not overlap with characters are still mis-classified as characters. The recall of the line classification on ViaMichelin Map of both algorithms are lower than average since there is significant overlap between the characters and lines.

5. Discussion and Future Work

The main contribution of this paper is to provide a robust and efficient algorithm to automatically and accurately classify line and character pixels on raster maps. Our algorithm can be applied on various maps that have not been seen in the training process.

We plan to further develop our work as follows: The classified character pixels will be used in the OCR component to recognize the characters. The line pixels will be used in the vectorization component to rebuild the lines. We also want to test our algorithm on more types of raster maps. We focus on computer-generated raster maps which have more consistent texture properties on lines and characters than hand-draw maps, but we believe our algorithm can be applied to other types of maps as well.

6. Acknowledgement

We would like to thank Dr. Chew Lim Tan for his generous sharing of their code in [4]. We also want to thank Mr. Mark J. Carman for his contribution of the digital signal processing knowledge in this paper.

This research is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955, in part by the Air Force Office of Scientific Research under grant number FA9550-04-1-0105, and in part by a gift from Google. The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

[1] C.-C. Chen, C. A. Knoblock, C. Shahabi, Y.-Y. Chiang, and S. Thakkar. Automatically and accurately conflating orthoimagery and street maps. In *Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems*, 2004.

[2] T. Randen and J.H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

[3] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

[4] R. Cao and C. L. Tan. Text/graphics separation in maps. In *Proceedings of the 4th International Workshop on Graphics Recognition Algorithms and Applications*, 2001.

[5] J. P. Bixler. Tracking text in mixed-mode documents. In *Proceedings of the ACM Conference on Document Processing Systems*, 2000.

[6] A. Velázquez and S. Levachkine. Text/graphics separation and recognition in raster-scanned color cartographic maps. In *Proceedings of the 5th International Workshop on Graphics Recognition Algorithms and Applications*, 2003.

[7] L. A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):910–918, 1988.

[8] L. Li, G. Nagy, A. Samal, S. Seth, and Y. Xu. Cooperative text and line-art extraction from a topographic map. In *Proceedings of the 5th International Conference on Document Analysis and Recognition*, 1999.

[9] T. Joachims. Making large-scale SVM learning practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.