Extracting Road Vector Data from Raster Maps

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, USA
{yaoyichi,knoblock}@isi.edu

Abstract. Raster maps are an important source of road information. Because of the overlapping map features (e.g., roads and text labels) and the varying image quality, extracting road vector data from raster maps usually requires significant user input to achieve accurate results. In this paper, we present an accurate road vectorization technique that minimizes user input by combining our previous work on extracting road pixels and road-intersection templates to extract accurate road vector data from raster maps. Our approach enables GIS applications to exploit the road information in raster maps for the areas where the road vector data are otherwise not easily accessible, such as the countries of the Middle East. We show that our approach requires minimal user input and achieves an average of 93.2% completeness and 95.6% correctness in an experiment using raster maps from various sources.

Key words: GIS, raster maps, road vectorization, map processing

1 Introduction

Humans have a long history of using maps. In particular, paper maps have been widely used since the early years for documenting geospatial information. Because of the availability of low cost and high-resolution scanners and the Internet, we can now obtain a huge number of scanned maps in raster format from various sources. Since maps commonly contain road networks, raster maps are an important source of road vector data for areas where road vector data are not readily available. Moreover, we can use the road vector data as features to register maps to other geospatial data, such as imagery, and create an integrated view of heterogeneous geospatial data sets [3].

Extracting road vector data from raster maps is a challenging task. First, the extraction of road pixels is difficult since raster maps very often contain noise from image compression and scanning processes and roads often overlap with other map features. Further, for converting the road pixels to vector format, the previous work commonly uses the thinning operator [11] or line grouping and parallel-line matching techniques [1] to identify the road centerlines. The thinning operator can produce distorted lines around intersections and hence the extracted road vector data are not accurate without manual adjustment [1]. The line grouping and parallel-line matching techniques require manual settings on various parameters to identify the accurate centerlines, such as the maximum difference between the slopes of two line segments to be merged [11] .

In this paper, we present a general technique that requires minimal user input for extracting accurate road vector data from raster maps with varying

map complexity (e.g., overlapping features) and image quality. We exploit our previous work on extracting road pixels from raster maps [5] and utilize the thinning operator to determine the road centerlines. We then automatically correct the distortions near road intersections caused by the thinning operator using our previous techniques on extracting accurate road-intersection templates from raster maps [4; 6] to extract accurate road vector data. We tested our road vectorization technique on a variety of maps including scanned and digital maps from different sources and compared our results to a commercial map digitizing application.

The remainder of this paper is organized as follows. Section 2 discusses related work on road extraction from maps. Section 3 presents our approach to extract the road pixels from raster maps. Section 4 describes our approach to generate the road vector data from the extracted road pixels. Section 5 reports on our experimental results, and Section 6 presents the conclusion and future work.

2 Related Work

Much research work has been performed in the field of extracting road information from raster maps, such as separating lines from text [2; 14], detecting road intersections [8], and extracting road vector data [1; 11] from raster maps. In the previous work on text/graphics separation from raster maps, Cao and Tan [2] and Li et al. [14] utilize a preset grayscale threshold to remove the background pixels from raster maps and then detect text labels from the remaining foreground layers of the maps, and the road pixels are the by-product (i.e., only the road pixels are extracted) after the text pixels are identified. Since in their work, the main goal is to recognize the text labels, they do not process the raster maps further to extract the road vector data.

Some of the previous work assumes a simpler type of raster maps for their algorithms. Habib et al. [8] extract road intersections from raster maps that contain road lines only. Itonaga et al. [11] employ a stochastic relaxation approach to first extract the road areas from digitally generated maps (i.e., not scanned maps) and then apply the thinning operator to extract the road vector data. The distorted lines around the road intersections are corrected based on the straightness of the roads, which is determined using user specified constraints, such as the road width. In comparison, our approach can process a variety of raster maps including scanned maps, and we avoid the distortion with no parameter settings. Bin et al. [1] work on scanned maps to extract the road vector data. Instead of using the thinning operator, in [1], the medial lines of parallel road lines are first produced and then linked to generate the road vector data. In general, the vectorization results of utilizing the medial lines of parallel road lines can be very accurate for the lines around the intersections, but the extraction processes require more manually specified parameters than the thinning operator, such as the thresholds to group medial-line segments and to produce the road intersections.

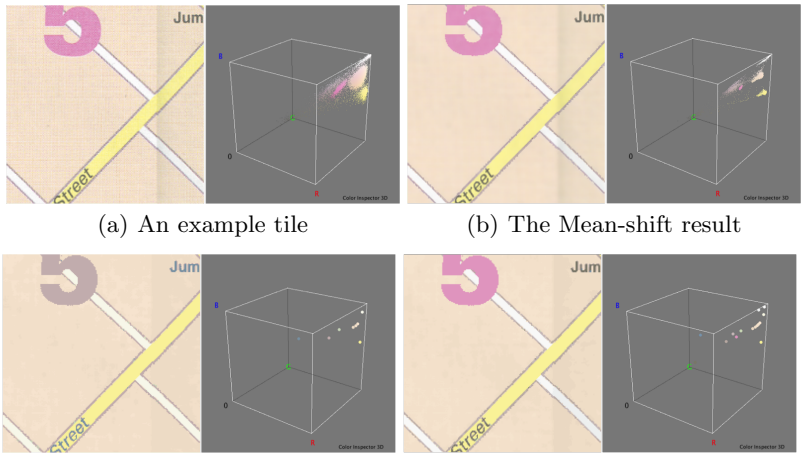
In addition to the research work, a commercial application called R2V from the Able Software is an automated raster to vector conversion software specialized in digitizing raster maps. To vectorize roads in raster maps using R2V, the

user needs to first manually specify samples of road pixels or select a set of color threshold to identify the road pixels. The manual work on specifying samples of road pixels can be laborious, especially for scanned maps with numerous colors, and the color thresholding function does not work if one set of threshold cannot separate all of the road pixels from other pixels. In comparison, our approach automatically identifies road colors from a few user labels for extracting the road pixels. After the road pixels are extracted, R2V can automatically trace the centerlines of the extracted road pixels and generate the road vector data. Our approach detects the road format and road width automatically and uses the detected road information to extract accurate road vector data. In our experiments, we tested R2V using our test maps and show that our automatic technique generates better results.

3 Extracting Road Pixels

Distinct colors commonly represent different layers (i.e., a set of pixels representing a particular geographic feature) in a raster map, such as roads, contour lines, and text labels. By identifying the colors that represent roads in a raster map, we can extract the road pixels from the map. However, raster maps usually contain numerous colors due to scanning and/or compression processes and the poor condition of the original documents (e.g., color variation from aging, shadows from folding lines, etc.). For example, Figure 1(a) shows a 200x200-pixels tile cropped from a scanned map. The tile has 20,822 distinct colors, which makes it difficult to select the road colors manually. To overcome this difficulty, many techniques have been developed to first group the colors of individual feature layers into clusters based on the assumption that the color variation within a feature layer is smaller than the variation between feature layers [5; 10; 12; 13]. Therefore, the feature layers can be extracted by selecting specific clusters. In this paper, we utilize our supervised map decomposition technique in [5] to extract the road pixels, which requires minimal user input and is capable of handling various types of raster maps, especially scanned maps.

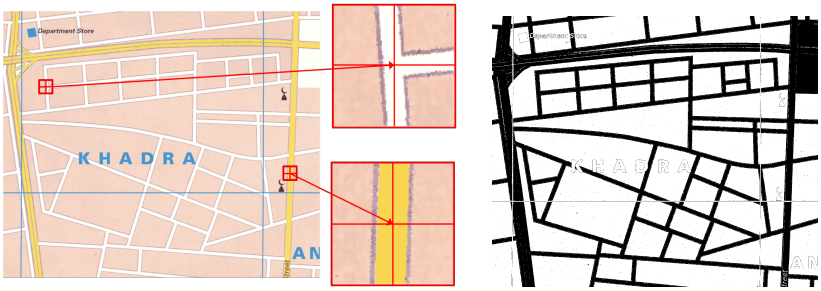
The supervised map decomposition technique first employs two color quantization techniques to reduce the number of colors in the raster map. To preserve object edges while clustering the colors in a raster map, we first employ the Mean-shift algorithm [7], which merges two colors into one by considering their distance in the color space (we use a color distance of 25 in the red, blue, and green color space) as well as in the image space (we use a spatial distance of 3 pixels). The Mean-shift algorithm reduces the number of colors in Figure 1(a) by 72% as shown in Figure 1(b). To further merge similar colors in the raster maps for reducing the user input to select the road colors, we apply the K-means algorithm with a user specified K to generate a quantized map image with at most K colors. The K-means algorithm can significantly reduce the number of colors in a raster map by maximizing the inter-cluster color variance; however, since the K-means algorithm considers only the color space, it is very likely that the resulting map has merged features with a small K. For example, Figure 1(c) shows the quantized map with K as 8 and the text labels have the same color as



(a) An example tile (b) The Mean-shift result

(c) The K-means result, K=8 (d) The K-means result, K=16

Fig. 1. An example map tile and the color quantization results with color cubes



(a) User labels centered at road lines (b) Extracted road pixels

Fig. 2. Extracting road pixels using road color identified by analyzing user labels

the road edges. Therefore, the user would need to select a larger K to separate different features, such as in the quantized map in Figure 1(d) with K as 16.

With the quantized map, the user provides labels of road areas such as the two user labels shown in Figure 2(a), and the map decomposition technique then exploits the fact that a user label is required to be centered at a road line or a road intersection to identify the road colors. Using this approach, the user only has to provide enough user labels to cover each road color in the raster map, such as one for the white roads and one for the yellow roads in Figure 2(a). Figure 2(b) shows the extracted road pixels by using the road colors identified using these two user labels.

4 Vectorizing Road Pixels

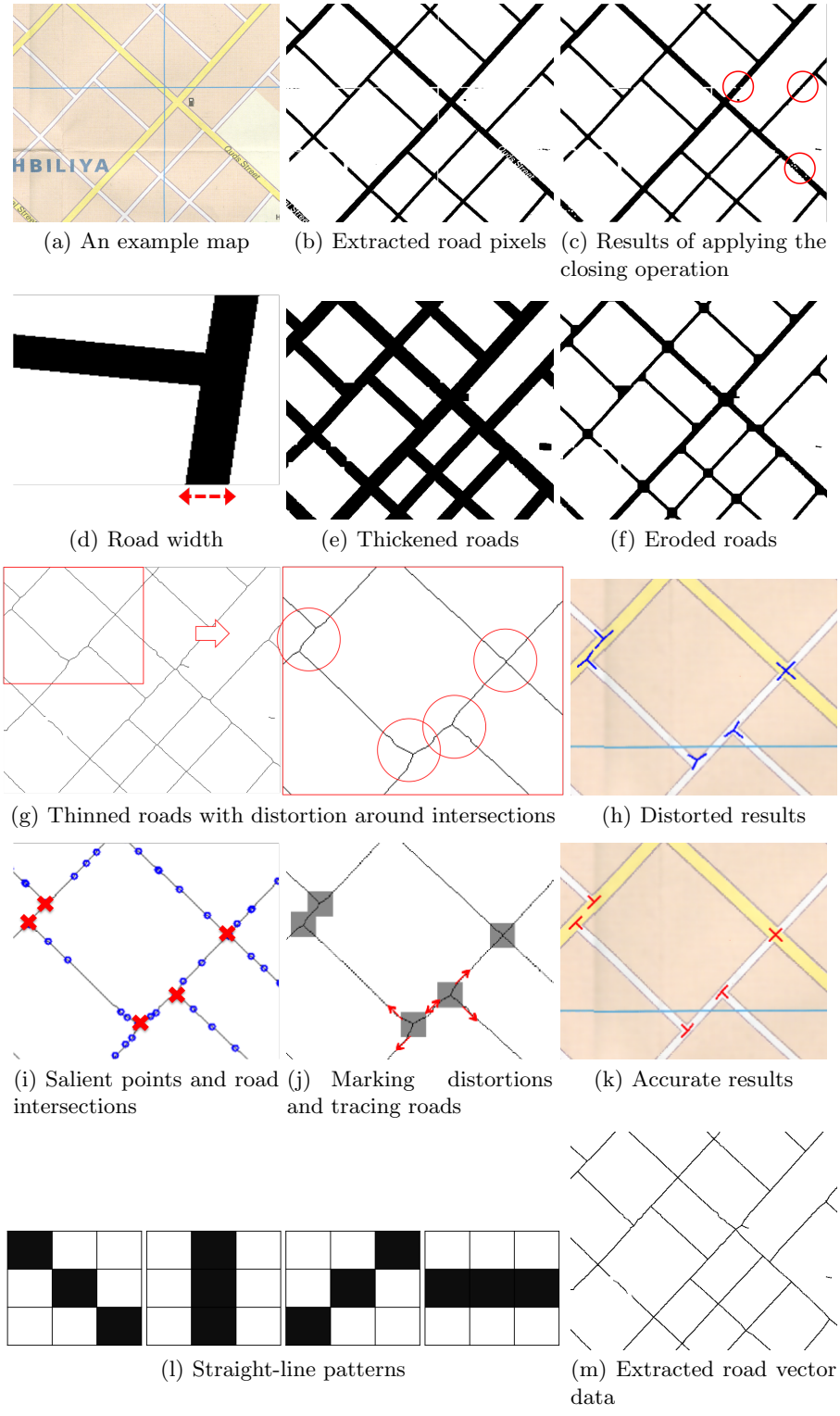
Once we have the road pixels, we generate the road vector data by first determine the road centerlines and then vectorize the centerlines. Figure 3(a) and Figure 3(b) show an example map tile from a scanned map and the road pixels extracted from the map using the approach described in the previous section. The extracted road pixels contain objects other than roads since they are drawn using the same color as roads. In addition, some of the road lines in the extracted

road layer are broken since the missing pixels also belong to the text labels and grid lines (i.e., overlapping features) and these pixels are not drawn using the road colors. To separate the non-road features from the road pixels, we exploit the distinctive geometric properties of road lines such as road lines are linear objects and are connected, to remove solid areas and small connected-components. Next, we apply the closing operator to reconnect one-pixel wide gaps and fill small holes. The closing operator first expands the foreground areas by one pixel (i.e., one iteration of the dilation operation) and then expands the background areas by one pixel (i.e., one iteration of the erosion operation). Figure 3(c) shows the results after we apply the closing operator, where the red circles show that some of the missing road pixels are filled if the missing parts are small, especially the places where the text labels overlap with roads.

In order to reconnect broken lines with larger gaps automatically, we expand the areas of road pixels by utilizing the binary dilation operator as shown in Figure 3(e). We determine the number of iterations of the dilation operator (i.e., how far the foreground region should expend) using the road width and road format (i.e., double-line and single-line roads) identified automatically by the Parallel-Pattern Tracing algorithm [6]. In a road layer where road lines are drawn as single lines (i.e., single-line format) as the example shown in Figure 3, the detected road width is the thickness of the majority of the road lines in the road layer as the dashed lines shown in Figure 3(d). If a road line is drawn using two parallel lines (i.e., double-line format), the road width is the pixel distance between corresponding road pixels on the parallel lines. During the thickening process, we also merge parallel lines into thick single lines if the road layer is double-line format.

To generate the centerline representation of the thickened road lines, we apply the binary erosion operator and the thinning operator as shown in Figure 3(f) and Figure 3(g). We use the erosion operator to shrink the road areas before we apply the thinning operator because the thinning operator distorts lines near the intersections and the extent of the distortion depends on the thickness of the lines before the thinning operator is applied. Although the binary erosion operator helps to minimize the extent of the distortion caused by the thinning operator, the road geometry near the intersections is still not accurate, especially near T-shape intersections. Figure 3(g) shows the distorted examples of the road geometry around a T-shape intersection and Figure 3(h) shows an inaccurate results if the distorted lines are traced to generate the road vector data.

For correcting the distortion around the intersection points and generating accurate road vector data from the thinned-line image (Figure 3(g)), we first detect intersections of the thinned lines to mark potential distorted lines. We utilize the corner detector [15] to detect intersection candidates and then use the connectivity of the candidates to determine actual road intersections [6]. Figure 3(i) shows the intersection candidates in blue circles and the actual intersections with cross marks. Since the extent of the distortion around each intersection is determined by the thickness of the thickened lines (which is decided by the road width and the dilation operator), we can mark potential distorted thinned-lines



(a) An example map (b) Extracted road pixels (c) Results of applying the closing operation

(d) Road width (e) Thickened roads (f) Eroded roads

(g) Thinned roads with distortion around intersections (h) Distorted results

(i) Salient points and road intersections (j) Marking distortions and tracing roads (k) Accurate results

(l) Straight-line patterns (m) Extracted road vector data

Fig. 3. Extracting road vector data from an example map

```
CNList; // The connecting-node list
// CN.x and CN.y are the pixel location
road_vectors; // The line-segment list
/* A line segment contains two
CN indexes */
start_id; end_id; /* The IDs of the starting
and ending CNs of the line segment we
are currently tracing */

void main()  // Program starts here
  Foreach CN in the CNList {
    start_id = CN.id;
    SetVisited(CN.x, CN.y);
    floodFill8(CN.x, CN.y);
  }
  UpdateCNLocation();
```

```
Function void floodFill8(int x, int y)
  if (InsideImage(x,y) && NotVisited(x,y)
  && NotBackground(x,y)) {
    if (IsCN(x,y)){  // We found a line
      end_id = GetCNID(x,y);
      road_vectors.AddLine(start_id, end_id);
    } else {
      SetVisited(x,y);
      floodFill8(x + 1, y); floodFill8(x - 1, y - 1);
      floodFill8(x, y + 1); floodFill8(x + 1, y - 1);
      floodFill8(x + 1, y + 1); floodFill8(x – 1, y);
      floodFill8(x - 1, y + 1); floodFill8(x, y - 1);
    }
  }
```

Fig. 4. Pseudo codes for tracing line pixels

near an intersection point using a gray box with size as the thickness of the thickened lines as shown in Figure 3(j). We then trace the lines outside the gray boxes to generate accurate road orientations and update the positions of the road intersections based on the intersecting roads and their orientations. Figure 3(k) shows a portion of example extraction results. The road lines around the intersections are accurate despite the distortion of the thinned lines shown in Figure 3(g).

With the accurate positions of the road intersections and the knowledge of potential distorted areas, we start to trace the road pixels on the thinned-line image to generate the road vector data. The thinned-line image contains three types of pixels: the non-distorted road pixels, distorted road pixels, and background pixels, (as shown in Figure 3(j), they are the black pixels not covered by the gray boxes, black pixels in the gray boxes, and white pixels, respectively). We create a list of *connecting nodes* (CNs) of the road vector data. A CN is a point where two lines meet at different angles. We first add the detected road intersections into the CN list. Then, we identify the CNs among the non-distorted road pixels using a 3x3-pixels window to check if the pixel has any of the straight-line patterns shown in Figure 3(l). We add the pixel to the CN list if we *do not* detect a straight-line pattern since the road pixel is not on a straight line.

To determine the connectivity between the CNs, we trace the road pixels using an eight-connectivity flood-fill algorithm shown in Figure 4. The flood-fill algorithm starts from a CN, travels through the road pixels (both non-distorted and distorted ones), and stops at another CN. Finally, for the CNs that are road intersections, we use the previously updated road intersection positions as the CNs' positions. The CN list and the determined connectivity are the results of our extracted road vector data. Figure 3(m) shows the extracted road vector data. The road vector data around the road intersections are accurate since the distorted lines are not traced by the flood-fill algorithm and the intersection positions are updated using accurate road orientations.

5 Experiments

We evaluated our road vectorization approach using three raster maps produced from different sources. Two maps are scanned maps (350dpi) covering the city of

Table 1. The number of colors in the image for user labeling of each tested map and the number of user labels for extracting the road pixels

Tile Number	ITM Map										Gecko Map										UN Map
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Colors	8	16	8	16	16	16	16	16	16	8	8	8	16	16	16	8	16	16	16	8	90
User Labels	4	3	3	4	3	2	4	3	3	2	2	2	2	2	2	2	3	3	3	2	1

Bagdad, Iraq published by Gizi Maps and International Travel Maps (ITM). We cropped and tested 10 map tiles (800x600 pixels each) from each of the scanned map. The paper maps have been folded, and the folding lines cause inevitable shadows and color differences between areas in the scanned maps, which enriches our test data since the cropped tiles from the same map have various color usages and image quality. In addition to the scanned maps, we tested a digitally generated map covering Afghanistan published by the United Nations (UN).[1] The digital map (3300x2550 pixels) shows the main and secondary roads, cities, political boundaries, airports, and railroads of the nation. We tested the digital map as a single tile in our experiments. For comparison, we also tested the automatic road vectorization function in R2V from the Able Software.

We first applied our supervised map decomposition technique described in Section 2 to extract the road pixels from the test maps. We pre-processed the scanned map tiles using the Mean-shift and K-means algorithms with K as 8, 16, 24, and 32 to generate four quantized images for each map tile. The user started the user-labeling task from the quantized image containing eight colors. If the user cannot distinguish the road pixels from other map features (e.g., background) in the quantized image, the user will then select an image containing more colors (a higher K) for user labeling. We did not apply the color segmentation algorithms on the digital map before user labeling. This is because the digital map contains a smaller number of colors (i.e., 90 unique colors) and there is only one color representing both the major and secondary roads in the map. Table 1 shows the numbers of colors in the images used for user labeling and the numbers of user labels used for extracting the road pixels. The user-labeling task is the only process that requires user input in our experiments, and for all of the scanned map tiles, only two to four labels were needed.

We tested R2V on extracting the road pixels from the test maps. Since the scanned maps contain numerous colors, we need more than one set of color thresholds to extract the road pixels (which R2V only allows one) or significant user effort to manually specify sample pixels for each of the road colors. Therefore, we did not successfully extract the road pixels from the scanned maps using R2V. For the digital map, we used one set of color threshold to extract the road pixels using R2V.

For the extracted road vector data, we report the accuracy of the extraction results using the road extraction metrics proposed in [9], which include the completeness, correctness, quality, redundancy, and the root-mean-square (RMS) difference. We manually drew the centerlines of every road lines in the

[1] http://unama.unmissions.org/

Table 2. Numeric results of the extracted road vector data from the scanned Gecko and ITM maps (four-pixel-wide buffer) using our approach

	ITM	Gecko	ITM	Gecko	ITM	Gecko	ITM	Gecko	ITM	Gecko
Tile	**Completeness**		**Correctness**		**Quality**		**Redundancy**		**RMS Diff.**	
1	98.7%	97.8%	96.7%	85.8%	95.5%	84.1%	0.07%	0%	2.34	1.69
2	99.3%	97.4%	93.6%	97.5%	92.9%	95%	0%	0%	1.23	3.51
3	98.1%	93.3%	75.8%	97.8%	74.7%	91.4%	0%	6.7%	1.52	2.46
4	91.7%	97.2%	96.0%	98.7%	88.3%	96%	0%	1.73%	2.57	1.61
5	92.0%	98.9%	94.7%	97.9%	87.5%	96.8%	0%	0%	2.79	1.32
6	92.7%	88.6%	99.0%	90.2%	91.9%	80.1%	0%	0.41%	2.50	3.2
7	97.5%	97.3%	99.2%	98%	96.7%	95.4%	3.34%	6.52%	1.81	1.65
8	95.1%	93.4%	97.1%	94%	92.5%	88.2%	0%	0%	2.02	2.56
9	93.7%	99.0%	94.6%	83.3%	88.9%	82.6%	0%	0%	2.21	1.54
10	97.1%	98.7%	85.9%	94%	83.7%	92.9%	0.7%	1.7%	2.20	1.47
Avg.	**95.6%**	**96.2%**	**93.3%**	**93.7%**	**89.3%**	**90.3%**	**0.6%**	**1.7%**	**2.12**	**2.1**

maps as the ground truth. The completeness and correctness represent how complete/correct the extracted road vector data are (the optimum is 100%). The quality is a combination metric of completeness and correctness (the optimum is 100%). The redundancy shows the difference in percentage between the correctly extracted line and the matched ground truth (the optimum is 0). The RMS difference is the average distance between the extracted lines and the ground truth, which represents the geometrical accuracy of the extracted road vector data. To generate these metrics, the authors in [9] suggest using a buffer width as half of the road width in the test data. In our test maps, the roads are five and eight pixels wide in the digital map and are seven to ten pixels wide in the scanned maps. We used a buffer width of four pixels.

Table 2 and Table 3 show the numeric results. The average completeness, correctness, and quality are around 90% to 96% and the average redundancy numbers are around 1% for the scanned and digital map. Figure 5 shows some example results, where the geometry of the extracted road vector data are very close to the road centerlines in the maps. Some broken lines are not connected (causing lower completeness numbers, such as the digital map) since the gaps are larger than the iterations of the dilation operations (we automatically detected the road format as single-line roads and used three iterations of the dilation operator to fix the gaps smaller than six pixels). The broken lines could be reconnected with post-processing on the road vector data since the gaps are now smaller than they were in the extracted road layers resulting from the dilation operations. The tiles 3 and 10 of the ITM map and tiles 1 and 9 of the Gecko map have lower correctness since parts of non-road features are also extracted using the identified road colors and those parts contribute to false-positive road

Table 3. Numeric results of the extracted road vector data from the UN digital map (four-pixel-wide buffer) using our approach and R2V

Tested Technique	Completeness	Correctness	Quality	Redundancy	RMS Diff.
This Paper	87.9%	99.9%	87.8%	0%	3.75
Able R2V	76.1%	96.7%	74.2%	18.92%	3.91

vector data. Figure 5(a) to Figure 5(c) show the ITM tile 3 where the runway is represented using the same color as the white roads and hence are extracted as road pixels. This type of false-positives could be further removed by including user validation steps after the road pixels were extracted. Some tiles have higher redundancy numbers such as the Gecko tiles 3 and 7, which is because some of the straight road lines in these tiles were extracted as shorter line segments with a small orientation variation and their buffers overlap with each other. The average RMS differences are under three pixels for scanned maps and under four for the digital map, which shows that the thinning operator and our approach to correct the distortion result in good quality road geometry. Table 3 shows our approach achieved better results than R2V.[2] The lower completeness of R2V is because R2V did not automatically connect broken road pixels. The lower correctness and high redundancy of R2V is because R2V generated small line segments instead of long and smooth lines and did not generate accurate road lines near the intersections.

For the computation time, we built our test system using Microsoft Visual Studio 2008 running on a Windows XP Professional Virtual Machine installed on 2.4 GHz Intel Core 2 machine with one GB of memory. The average processing time for vectorizing the road pixels for a map tile (800x600 pixels) is 13 seconds. The dominant factors of the computation time are the image size, the number of road pixels in the raster map, and the number of road intersections in the road layer.

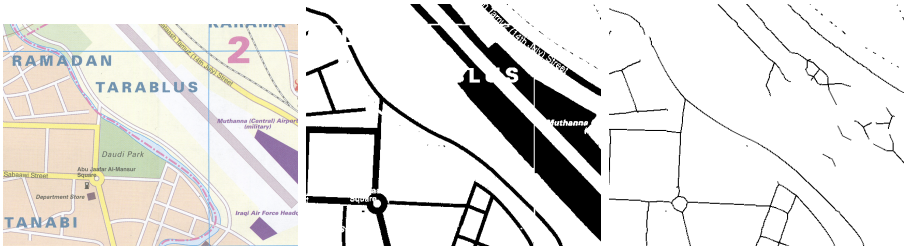
6 Conclusion and Future Work

We present a general technique that extracts accurate road vector data from heterogeneous raster maps with minimal user input. We utilize our previous work [5] to handle raster maps with varying image quality and exploit the accurate road-intersection templates [4; 6] to prevent distorted extraction results. We show that our technique extracts accurate road vector data from three raster maps with varying color usage and image quality. In the future, we plan to test our approach on more maps from various sources and test to include post-processing on the road vector data to improve the results.

7 Acknowledgments

This research is based upon work supported in part by the University of Southern California under the Viterbi School Doctoral Fellowship, and in part by the United States Air Force under contract number FA9550-08-C-0010. 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.

[2] We used the "Auto Vectorize" function in R2V without manual post-processing



(a) ITM tile 3 (b) Road pixels of (a) (c) Road vector data of (a)

(d) ITM tile 6 (e) Road pixels of (d) (f) Road vector data of (d)

(g) ITM tile 9 (h) Road pixels of (g) (i) Road vector data of (g)

(j) Gecko tile 7 (k) Road pixels of (j) (l) Road vector data of (j)

(m) UN map (portion) (n) Road pixels of (m) (o) Road vector data of (m)

Fig. 5. Examples of the road vectorization results

Bibliography

[1] D. Bin and W. K. Cheong. A system for automatic extraction of road network from maps. In *Proceedings of the IEEE International Joint Symposia on Intel ligence and Systems*, pages 359–366, 1998.

[2] R. Cao and C. L. Tan. Text/graphics separation in maps. In *Proceedings of the Fourth GREC Workshop*, pages 167–177, 2002.

[3] C.-C. Chen, C. A. Knoblock, and C. Shahabi. Automatically and accurately conflating raster maps with orthoimagery. *GeoInformatica*, 12(3):377–410, 2008.

[4] Y.-Y. Chiang and C. A. Knoblock. Automatic extraction of road intersection position, connectivity, and orientations from raster maps. In *Proceedings of the 16th ACM GIS*, pages 1–10, 2008.

[5] Y.-Y. Chiang and C. A. Knoblock. A method for automatically extracting road layers from raster maps. In *Proceedings of the Tenth ICDAR*, 2009.

[6] Y.-Y. Chiang, C. A. Knoblock, C. Shahabi, and C.-C. Chen. Automatic and accurate extraction of road intersections from raster maps. *GeoInformatica*, 13(2):121–157, 2008.

[7] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on PAMI*, 24(5):603–619, 2002.

[8] A. Habib, R. Uebbing, and A. Asmamaw. Automatic extraction of road intersections from raster maps. Project Report submitted to the Center for Mapping, The Ohio State University, 1999.

[9] C. Heipke, H. Mayer, C. Wiedemann, and O. Jamet. Evaluation of automatic road extraction. In *International Archives of Photogrammetry and Remote Sensing*, pages 47–56, 1997.

[10] T. C. Henderson, T. Linton, S. Potupchik, and A. Ostanin. Automatic segmentation of semantic classes in raster map images. In *the Eighth GREC Workshop*, 2009.

[11] W. Itonaga, I. Matsuda, N. Yoneyama, and S. Ito. Automatic extraction of road networks from map images. *Electronics and Communications in Japan (Part II: Electronics)*, 86(4):62–72, 2003.

[12] V. Lacroix. Automatic palette identification of colored graphics. In *the Eighth GREC Workshop*, 2009.

[13] S. Leyk and R. Boesch. Colors of the past: color image segmentation in historical topographic maps based on homogeneity. *GeoInformatica*, 14(1):1–21, 2010.

[14] L. Li, G. Nagy, A. Samal, S. C. Seth, and Y. Xu. Integrated text and line-art extraction from a topographic map. *IJDAR*, 2(4):177–185, 2000.

[15] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.