Automatic Extraction of Road Intersection Position, Connectivity, and Orientations 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
yaoyichi, knoblock@isi.edu

ABSTRACT

The road network is one of the most important types of information on raster maps. In particular, the set of road intersection templates, which consists of the road intersection positions, the road connectivities, and the road orientations, represents an abstraction of the road network and is more accurate and easier to extract than the extraction of the entire road network. To extract the road intersection templates from raster maps, the thinning operator is commonly used to find the basic structure of the road lines (i.e., to extract the skeletons of the lines). However, the thinning operator produces distorted lines near line intersections, especially at the T-shaped intersections. Therefore, the extracted position of the road intersection and the road orientations are not accurate. In this paper, we utilize our previous work on automatically extracting road intersection positions to identify the road lines that intersect at the intersections and then trace the road orientations and refine the positions of the road intersections. We compare the proposed approach with the usage of the thinning operator and show that our proposed approach extracts more accurate road intersection positions and road orientations than the previous approach.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial Databases and GIS*

General Terms

Algorithms, Design

Keywords

Raster map, road layer, road intersection template, road orientation, thinning

1. INTRODUCTION

Raster maps are widely available and contain valuable information, such as road lines, labels, and contour lines. For instance, a digital raster graphic (DRG), which is a georeferenced scanned image of a United States Geological Survey (USGS) topographic map, can be purchased from the USGS website or accessed freely from TerraServer-USA.[1] Map repositories like the University of Texas Map Library[2] contain information-rich scanned maps for many areas and even historical scanned maps. Moreover, web mapping service providers such as Google Maps,[3] Microsoft Live Search Maps,[4] and Yahoo Maps[5] provide high quality digital maps covering many countries with rich information layers such as business locations and traffic information.

To extract information from the raster maps, one approach is to process the vector data used to generate the raster maps and then extract the desired information, such as the road vectors. However, it is generally difficult to access the original vector data for many raster maps. For example, the web mapping services provided by Google, Microsoft, and Yahoo all provide their maps only in the raster format and there is no public access to the original vector data. Furthermore, many raster maps found on the Internet are simply images without any auxiliary information of the original vector data. Hence, a more general approach is to utilize image processing and graphics recognition technologies to separate the information layers (e.g., the road layer and text layer) on the raster maps and then extract and rebuild selected layers [7].

Among the information layers on the raster maps, one of the most important layers is the road layer. Instead of extracting the entire road layer, the set of road intersection templates, which consists of the road intersection position, connectivity, and orientations, provides the basic elements of the road layout and is more accurate and easier to extract than the extraction of the entire road network. Since the road layers commonly exist across many different geospatial layers (e.g., satellite imagery, vector data, etc.), by matching the set of road intersection templates from a raster map with another set of road intersection templates from a georeferenced data set (e.g., vector data), we can identify the geospatial extent of the raster map and align the raster map with other geospatial sources [3]. An example of an inte-

[1]http://terraserver-usa.com/
[2]http://www.lib.utexas.edu/maps/
[3]http://maps.google.com/
[4]http://maps.live.com/
[5]http://map.yahoo.com

grated and aligned view of a tourist map and satellite imagery is shown in Figure 1. Accurate road intersection templates (i.e., road intersection templates with accurate positions, road orientations, and connectivities) help to reduce the search space during the matching process by selecting road intersections with the same connectivity and similar road orientations as initial matching candidates. The accurate positions of the templates also enhance the alignment results. Furthermore, the extracted road intersection templates with accurate positions and road orientations can be used as seed points to extract roads from aerial images [10] as shown in Figure 2, especially when the access to vector data is limited.

To extract the road intersection templates from a raster map, the first step is to detect the positions of the road intersections on the raster map and then convert the road lines

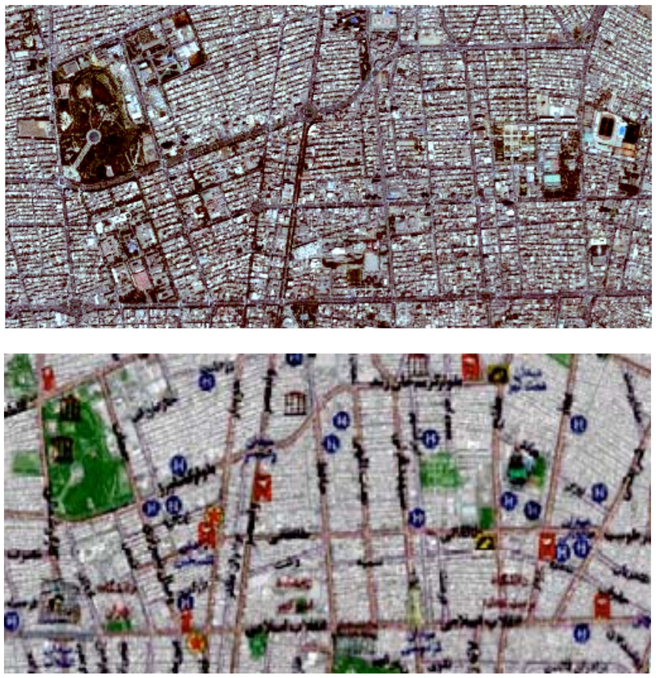


Figure 1: The integration of imagery from Google Maps and a tourist map (Tehran, Iran).

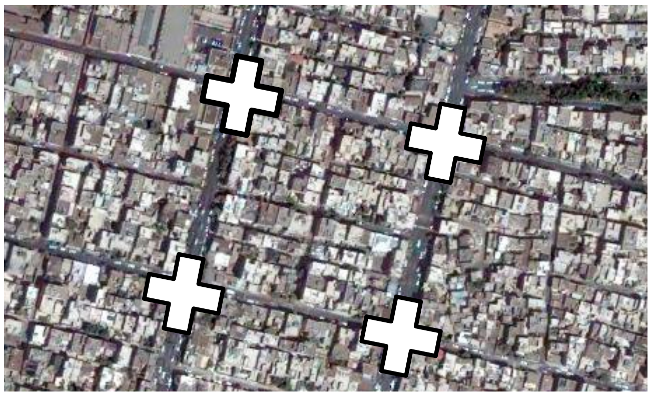
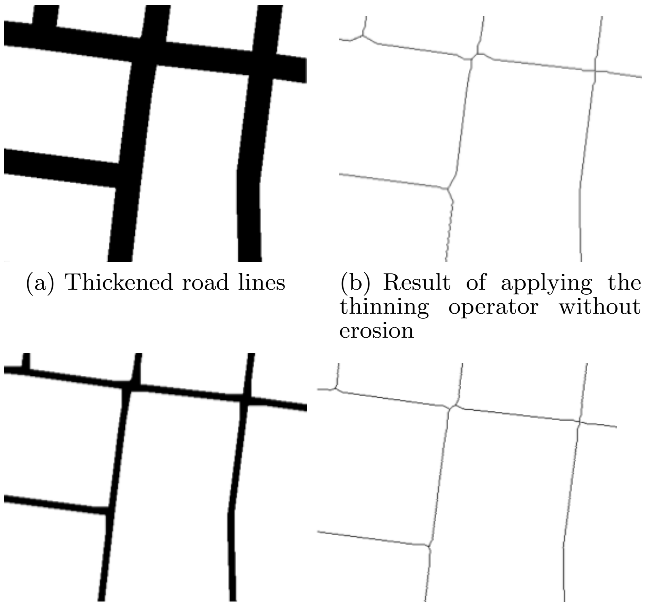


Figure 2: Use the extracted road intersection templates to extract roads from imagery

around the road intersections into vector format, which is difficult since the raster maps are usually complicated and contain many different layers such as characters and road lines. In our previous work [7], we presented a technique that works on the pixel level of the raster maps and utilizes the distinctive geometric properties of the road lines and road intersections to automatically extract road intersections from raster maps. During the extraction process, our previous technology reconnects the broken lines by first thickening the lines and then utilizing the thinning operator to generate one-pixel width lines to identify possible road intersections.

The idea of the thinning operator [13] is to produce one-pixel width lines that represent the central lines of the thick lines (i.e., to preserve the basic structures of the thick lines). The thinning operator first identifies every foreground pixel that connects to one or more background pixels as candidate to be converted to the background (i.e., to reduce the region of foreground objects). Then a confirmation step checks if the conversion of the candidate will cause any disappearance of original line branches to ensure the basic structures of the original objects will not be compromised. The thinning operator is robust and efficient; however, the thinning operator distorts the lines near the intersections. As shown in Figure 3, if we apply the thinning operator directly on the thick lines shown in Figure 3(a), the lines that are near intersections are all distorted as shown in Figure 3(b). Our approach in [7] to minimize the extent of the distortions is to first erode the lines using an binary erosion operator [13] as shown in Figure 3(c) and then apply the thinning operator. However, there are still distortions even after the erosion operator is applied as shown in Figure 3(d).

In the previous work, the proposed technologies focused on efficiently extracting the positions of road intersections and



(a) Thickened road lines

(b) Result of applying the thinning operator without erosion

(c) Thickened road lines after erosion

(d) Result of applying the thinning operator after erosion

Figure 3: Distorted road lines near road intersections with the thinning operator

extracting the road orientations using a simpler technique that works on the one-pixel width road lines generated by the thinning operator. Therefore, the extracted road orientations suffer from the distortions produced by the thinning operator and the positions of the road intersections are not accurate. To overcome the distortion problem and to extract accurate road intersection templates, in this paper, we present an approach that employs the results of our previous work to trace the road lines for accurate road orientations and then utilizes the road orientations to refine the positions of the road intersections.

The remainder of this paper is organized as follows. Section 2 explains our previous road intersection extraction work. Section 3 describes our improved approach to automatically extract road intersection templates. Section 4 reports on our experimental results. Section 5 discusses the related work, and Section 6 presents the conclusion and future work.

2. BACKGROUND WORK

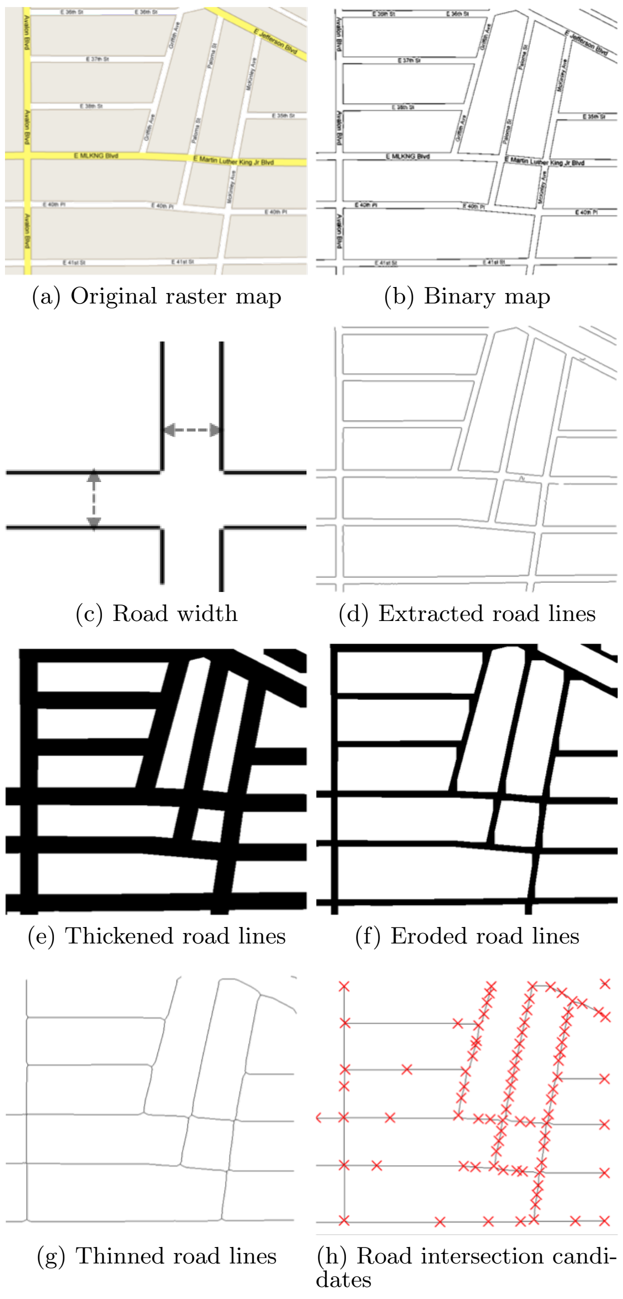
In our previous work [7], we proposed a technique to automatically extract the road intersections from various raster maps. There are three major steps in our previous approach. First, we remove the background pixels from the raster maps. Next, we separate the road lines from other objects in the foreground pixels and rebuild the road lines. Finally, we detect road intersection candidates on the road lines and utilize the connectivity (i.e., the number of lines that intersect at a given road intersection candidate) to find actual road intersections. The road orientations are also computed in the last step as a by-product. We briefly explain these three steps of the previous approach in turn in the following subsections.

2.1 Automatic Segmentation

Since the foreground pixels of the raster maps contain the road layers, the first step for extracting road intersections is to extract the foreground pixels. We utilize a technique called segmentation with automatically generated thresholds to separate the foreground pixels from the background pixels of the raster maps. Because the background colors of raster maps have a dominant number of pixels and the foreground colors have high contrast against the background colors, we generate the segmentation thresholds by analyzing the shape of the grayscale histogram of the raster map [15]. We first identify the largest luminosity cluster in the histogram as the background cluster and then classify other clusters as either background clusters or foreground clusters by comparing the number of pixels in the clusters. After we remove the background clusters from the original map shown in Figure 4(a), we produce the binary map shown in Figure 4(b)

2.2 Pre-Processing – Extract and Rebuild Road Layer

With the extracted foreground pixels, we first utilize our parallel-pattern tracing algorithm [7] to detect the road width of the majority of roads on the raster map. For a given foreground line, the parallel-pattern tracing algorithm employs two convolution masks that work on the horizontal and vertical directions to search for corresponding parallel lines to determine the road width. The detected road width is shown with gray dashed lines in Figure 4(c). The road width is important in the latter steps to rebuild the road layer.


(a) Original raster map (b) Binary map

(c) Road width (d) Extracted road lines

(e) Thickened road lines (f) Eroded road lines

(g) Thinned road lines (h) Road intersection candidates

Figure 4: Automatic extraction of road intersections

After we obtain the road width, we use the text/graphic separation technique [1] to remove the foreground pixels that do not hold the properties that constitute roads (e.g., pixels for labels, contour lines, etc). The extracted road layer is shown in Figure 4(d). Finally, to extract the structure of the road layer, we utilize the binary dilation operator with the number of iterations determined by the detected road width to first thicken the road line as shown in Figure 4(e). Then we apply the binary erosion operator and the thinning operator to generate one-pixel width road lines as shown in Figure 4(f) and Figure 4(g).

2.3 Determine Road Intersections and Extract Connectivity with Road Orientation

With the one-pixel-width road lines, we utilize the corner detector [14] to detect road intersection candidates as shown with the cross marks on the road lines in Figure 4(h). For each road intersection candidate, we draw a box around it and then use the number of foreground pixels that intersects with this rectangle as the connectivity of the intersection candidate as shown in Figure 5. If the connectivity is less than three, we discard the point; otherwise it is identified as a road intersection point. Subsequently, without tracing the line pixels, we link the road intersection candidate to the intersected foreground pixels on the rectangle boundaries to compute the orientations of the road lines.

The final road intersection extraction results are shown in Figure 6. Although we successfully extract most of the road intersections, the positions of some extracted road intersections are not at the center points of the intersected lines and the road orientations are not accurate, especially the ones on the intersection of a T-shape roads. This is because the extracted road lines near the intersections are distorted by the morphological operators (i.e., the binary dilation operator, the binary erosion operator, and the thinning operator) as shown in Figure 4(g); and the method we use to compute the road orientations does not take into account the distortions. In the next section, we present our improved approach to overcome the distortion problem and build an accurate road intersection template by utilizing the extracted road intersection position, the extracted one-pixel-width lines, and the detected road width.

3. AUTOMATIC EXTRACTION OF ROAD INTERSECTION TEMPLATES

The overall approach to extract road intersection templates is shown in Figure 7, where the gray boxes indicate the results we use from our previous approach in [7]. Based on our previous work, we recognize that there is distortion near each road intersection, and the extent of the distortion is determined by the thickness of the line, which is determined by the number of iterations of the morphological operators. And the number of iterations of the morphological operators is based on the road width that is detected using the parallel-pattern tracing algorithm.

Therefore, with the results of the road intersection positions and the road width, we first generate a binary large object (i.e., blob, a connected foreground object on an image) for each intersection to create a blob image. We then intersect the blob image with the thinned line image to label the location of the potential distorted lines. Finally, we trace the unlabeled lines to compute the road orientations and use the road orientations to update the positions of the road intersections to generate accurate road intersection templates.

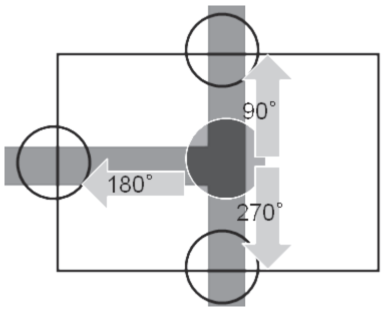


Figure 5: Construct lines to compute the road orientations

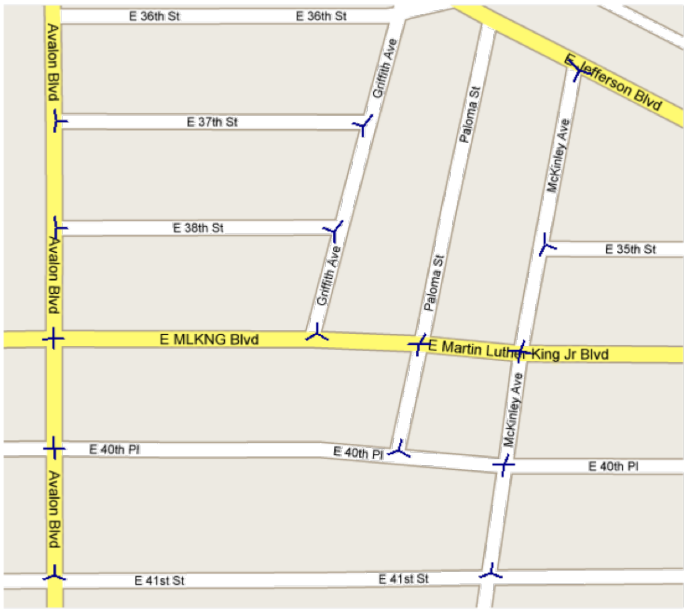


Figure 6: Extracted road intersections

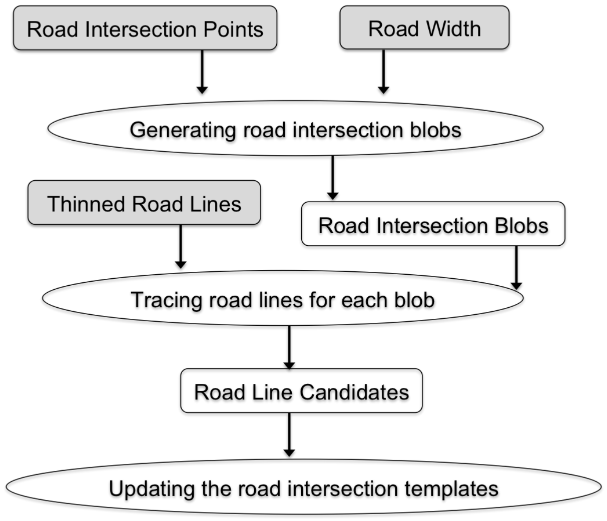

Road Intersection Points
Road Width
Generating road intersection blobs
Thinned Road Lines
Road Intersection Blobs
Tracing road lines for each blob
Road Line Candidates
Updating the road intersection templates

Figure 7: Overall approach to extract the road intersection templates from raster maps

3.1 Generating Road Intersection Blobs

Since the thinning operator produces distorted lines near the road intersections and the extent of the distortion is determined by the road width, we can utilize the extracted road intersections and the road width from our previous approach [7] to label the locations of the potential distorted lines.

For each extracted road intersection shown in Figure 6, we generate a rectangular blob using the binary dilation operator with twice the road width as the number of iterations. The result blob image is shown in Figure 8(a). We then intersect the blob image with the thinned line image shown in Figure 4(g) to label the locations of the potential distorted lines. As show in Figure 8(b), the line segments within the gray boxes are labeled by the blob image as the potential distorted lines.

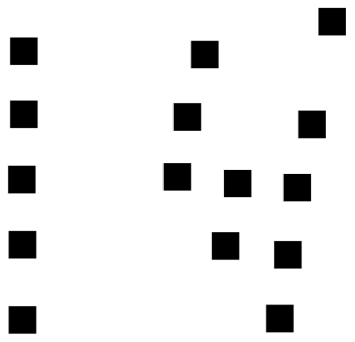
Subsequently, we use the labeled image shown in Figure 8(b) to detect road line candidates for each intersection. We identify the lines linked to each blob and their contact points by detecting the line pixels that have any neighboring pixel labeled by the gray boxes. These contact points indicate the starting points of a road line candidates for the blob. In the example shown in Figure 8(c), the road intersection associated with the left blob has three contact points that are on the top, right, and bottom of the blob, and these points will be used as starting points to trace the road line candidates for the road intersections of the blob.

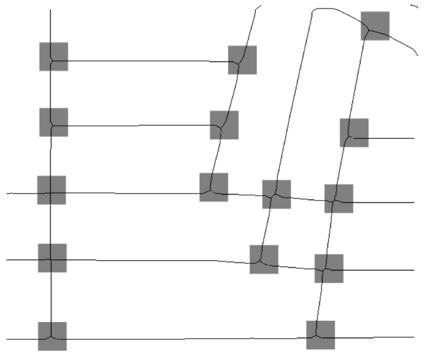
Instead of drawing a rectangle around each intersection to find the contact points as in our previous approach [7], we generate the blob image to take advantage of using image processing operations, which are more robust and easier to implement. For example, when we generate the blob image, if two intersections are very close to each other, their blobs automatically merge into one big blob by the binary dilation operator without additional implementations to compute the intersections of every rectangle box. This is important because when two intersections are very close to each other, the thinned line between them is totally distorted as shown in Figure 9. With the merged blob shown in Figure 9(d), we can associate the four lines linked to this blob with the two intersections as the road line candidates of the intersections. In the last step of updating road intersection templates, a filtering step will decide which candidates need to be discarded so that the two intersections will still have three road lines instead of four.

3.2 Tracing Road Lines

For each road intersection blob, we start to trace the road lines from its contact points using the flood-fill algorithm shown in Table 1. The flood-fill algorithm first labels the contact point as visited and then checks the eight neighboring pixels of the contact point to find unvisited points. If any of the eight neighboring pixels are not labeled as visited and are not within a blob, the neighboring pixel will be set as the next visit point for the flood-fill algorithm to consider.

When the flood-fill algorithm considers a new pixel, we also record the position of the pixel to latter compute the road orientation. The number of pixels that the flood-fill algorithm can trace from each contact point is controlled by the variable *MaxLinePixel* shown in Table 1. The flood-fill algorithm counts the number of pixels that it has visited and makes sure the count is less than the *MaxLinePixel* variable every time before it visits a new pixel or it stops. As shown in Figure 10, instead of tracing the whole curve starting from the two contact points (i.e., the one on the right and the one on the bottom), we assume that roads near the contact points are straight within a small distance (i.e., several meters within a street block on the raster map) and utilize the *MaxLinePixel* to ensure that the flood-fill algorithm traces only a small portion of the road lines near the contact points.


(a) Road intersection blob image


(b) Blob image intersected with the thinned line image

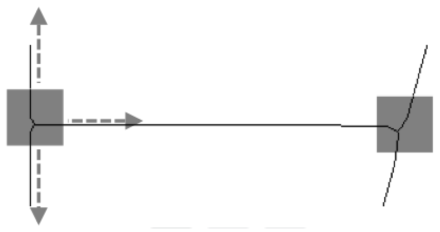

(c) Contact points

Figure 8: Generate the blob image to label the road lines

After the flood-fill algorithm walks through the lines from each contact point and records the position of the line segment pixels, we utilize the *Least-Squares Fitting* algorithm to find the linear functions of the lines. Assuming a linear function L for a set of line pixels traced by the flood-fill algorithm, by minimizing the sum of the squares of the vertical offsets between the line pixels and the line L, the *Least-Squares Fitting* algorithm finds the straight line L that best represents the traced line pixels. The algorithm works as follows:[6]

Given a set of pixel locations $\{(x_i, y_i)\}, i = 1, \ldots, n$, in the map, find the target line function L, where

$$L : Y = m \times X + b \tag{1}$$

[6]The proof of the Least-Squares Fitting algorithm can be found in most statistics textbooks or on the web at http://mathworld.wolfram.com/LeastSquaresFitting.html

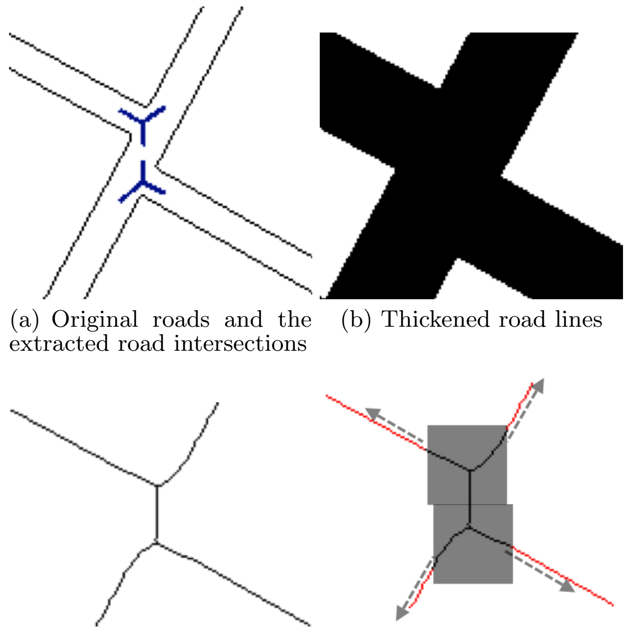
$$m = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2} \quad (2)$$

$$b = \frac{\sum y - m\sum x}{n} \quad (3)$$

The computed line functions are then used in the next step of updating road intersection templates to identify actual road lines and refine the positions of the road intersections.

3.3 Updating Road Intersection Templates

In the previous steps, we associated a set of road line candidates with a road intersection and traced the road line functions. In this step, we first compute the intersections of these road line candidates and then filter out outliers of the road line candidates.


(a) Original roads and the extracted road intersections (b) Thickened road lines

(c) Distorted thinned lines (d) Merged blobs

Figure 9: Merge nearby blobs to trace roads

Table 1: Flood-fill algorithm

```
FLOOD-FILL($Image, x, y$)
if $Image.isALinePixel(x, y) = TRUE$
AND $Image.isVisited(x, y) \neq TRUE$
AND $Image.isWithInABlob(x, y) \neq TRUE$
AND PixelCount < MaxLinePixel
    then
        $RecordPixelPosition(x, y)$
        $PixelCount{+}{+}$
        $Image.SetVisited(Image, x, y)$
        FLOOD-FILL($Image, x + 1, y$)
        FLOOD-FILL($Image, x - 1, y$)
        FLOOD-FILL($Image, x, y + 1$)
        FLOOD-FILL($Image, x, y - 1$)
        FLOOD-FILL($Image, x + 1, y + 1$)
        FLOOD-FILL($Image, x - 1, y - 1$)
        FLOOD-FILL($Image, x + 1, y - 1$)
        FLOOD-FILL($Image, x - 1, y + 1$)
```

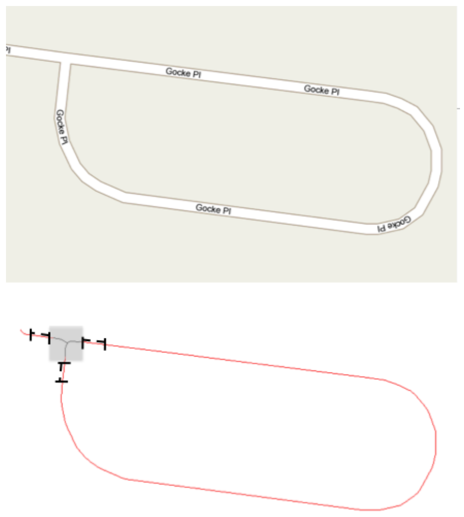


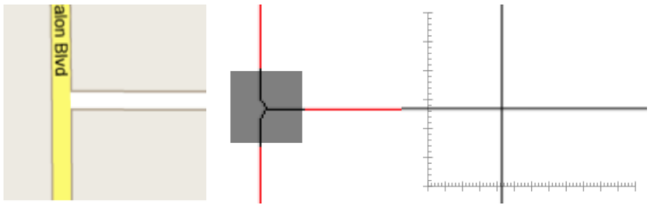
Figure 10: Trace only a small portion of the road lines near the contact points

There are three possible intersecting cases for the road line candidates of one intersection. The original maps of the three possible intersection cases are shown on the left in Figure 11. The middle images shown in Figure 11 are the thinned lines with the locations of the potential distorted lines labeled by the blob images, and the right images are the traced line functions (i.e., the line functions that are computed using the *Least-Squares Fitting* algorithm) drawn on a two-dimension plane.

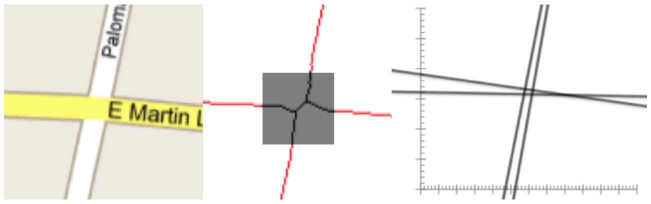
The first case is where all of the road line candidates intersect at one point as shown in Figure 11(a). The second case is where the road line candidates intersect at multiple points and the intersection points are all near the road intersection position, as shown in Figure 11(b). The third case is where the road line candidates intersect at multiple points and some of the points are not near the road intersection (i.e., the outliers), as shown in Figure 11(c).

For the first case shown in Figure 11(a), the position of the updated road intersection template is the intersection point of the road line candidates, and the road orientations are the orientations of the intersecting roads, which are 0 degrees, 90 degrees, and 270 degrees, respectively. For the second case shown in Figure 11(b), the position of the road intersection template is the centroid of the intersection points of all road line candidates and the road orientations are the orientations of the intersecting roads, which are 80 degrees, 172 degrees, 265 degrees, and 355 degrees, respectively.

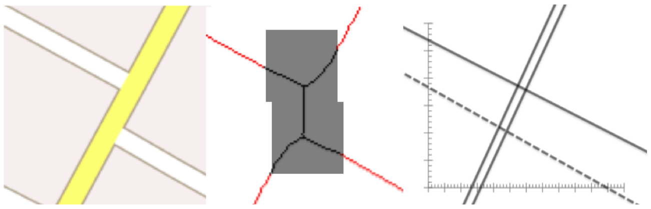
Since the extent of the distortion depends on the road width, the positional offset between any intersection of the road line candidates and the road intersection should not be larger than the road width. We first consider the upper road intersection in the third case shown in Figure 11(c). The intersections where the dashed line intersects with the other two lines are more than a road width away from the upper road intersection, so we discard the dashed line and use the



(a) Case one: the three lines intersect at one point



(b) Case two: the three lines intersect at multiple points



(c) Case three (the upper road intersection): the four lines intersect at multiple points with one outlier (the dashed line)

Figure 11: Filter out the outliers

centroid of the intersection points of the other three road line candidates to update the position of the upper road intersection. The road orientations of this road templates are 60 degrees, 150 degrees, and 240 degrees, respectively. Similarly, for the lower road intersection shown in Figure 11(c), the road line candidate that is almost parallel to the dashed line will be discarded, and the dashed line is kept as a road line for the road intersection. The road orientations of the lower road templates are 60 degrees, 240 degrees, and 330 degrees, respectively.

The last example shows how the merged blob helps to extract correct road orientations even when one of the lines is totally distorted during the thinning operation. This case holds when the distorted line is very short and hence it is very likely to have the same orientation as the other lines. For example, in Figure 11(c), the distorted line is part of a straight line that goes through the intersection, so it has the same orientation as the 240 degree line.

4. EXPERIMENTS

In this section, we evaluate our approach by conducting experiments on raster maps from various sources. We first explain the test data sets and our evaluation methodology and then analyze the experimental results and provide a comparison to our previous work [7].

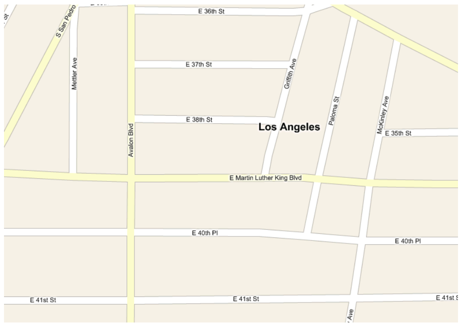
4.1 Experimental Setup

We evaluate 10 raster maps from five different sources, MapQuest Maps,[7], OpenStreet Maps.[8], Google Maps, Microsoft Live Search Maps, and Yahoo Maps, The 10 maps cover two areas in the United State, one in Los Angeles, California and the other one in St. Louis, Missouri. Figure 12 shows two example maps for these areas.

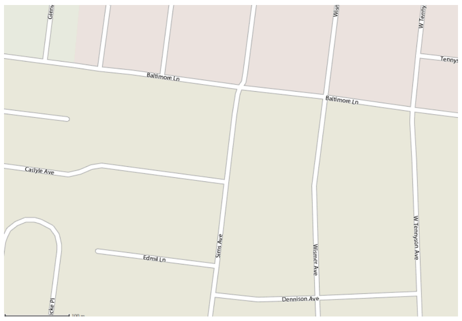
We first apply our road intersection extraction application from our previous approach to detect the road width and extract a set of road intersection positions and the thinned road lines for each map. Then we utilize the technique described in this paper to extract the accurate road intersection templates from the raster maps. We report the accuracy of the extraction results using the positional offset, orientation offset, and connectivity offset. The positional offset is the average number of pixels between the extracted road intersection templates and the actual road intersections on the raster maps. The actual road intersections on the raster maps are defined as the centroids of the intersection areas of two or more intersecting road lines. The orientation offset is the average number in degrees between the extract road orientations and the actual road orientations. The connectivity offset is the total number of missed road lines. We manually examine each road intersection on the raster maps to obtain the ground truth of the positions of the road intersections, the connectivities and the road orientations. Figure 13 shows the ground truth of two road intersection templates. For the road intersection template in Figure 13(a), the road orientations are 0 degrees, 90 degrees, and, 270 degrees, respectively.

4.2 Experimental Results

From the 10 raster maps, we extracted a total of 139 road intersection templates with 438 lines and the results are



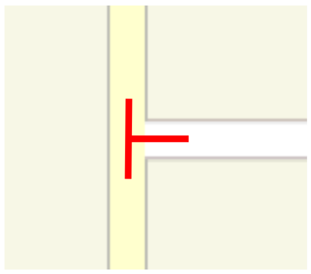
(a) Live Search Maps, Los Angeles, California



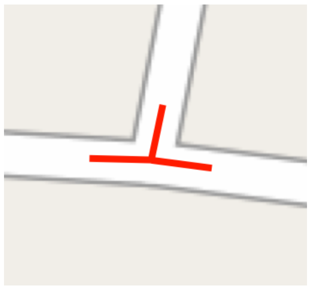
(b) Yahoo Maps, St. Louis, Missouri

Figure 12: Examples of the test maps

[7]http://www.mapquest.com/
[8]http://www.openstreetmap.org/



(a) The ground truth on a map from Google Maps



(b) The ground truth on a map from OpenStreet Maps

Figure 13: The ground truth

shown in Table 2. The average positional offset is 0.4 pixels and the average orientation offset is 0.24 degrees, which shows our extracted road intersection templates are very close to the ground truth on these test maps. In order to achieve higher accuracy in the positional and orientation offsets, we need to discard the lines that do not have accurate orientations in the filtering step (i.e., the outliers). So the connectivity is lower than our previous work; we missed 13 lines from a total of 451 lines in order to trace the lines to extract the road intersection templates. These 13 lines all belong to the road intersections that are near the boundaries of the map and hence we cannot find road lines long enough to compute the correct orientations.

We also compare the extracted template with the extracted template using the methods of our previous approach in [7], which uses only the thinning operator. The comparison results of the positional offset and orientation offset are shown in Figure 14. Our proposed approach in this paper has a large improvement on the results of every map source for both positional offset and orientation offset. Figure 16 shows example extraction results using the approach in this paper and our previous approach in [7]. This figure shows the distorted road lines from our previous work are corrected using the approach in this paper.[9]

4.3 Computation Time

The system is built using Microsoft Visual Studio 2005 on a Windows 2003 Server with Intel Xeon 1.8 GHZ Dual Processors and 1 GB of memory. The largest test map is 809 pixels by 580 pixels and it took 11 seconds to first extract the road intersection positions using the approach in [7] and another 5 seconds to extract the road intersection templates using the techniques proposed in this paper. The dominant factors of the computation time are the map size, the number of foreground pixels of the raster map, and the number of road intersections on the raster map. The average time is 10.5 seconds to extract the position of road intersections and 4.7 seconds to extract the road intersection templates, which is sufficient for many applications that need real-time intersection extraction results.

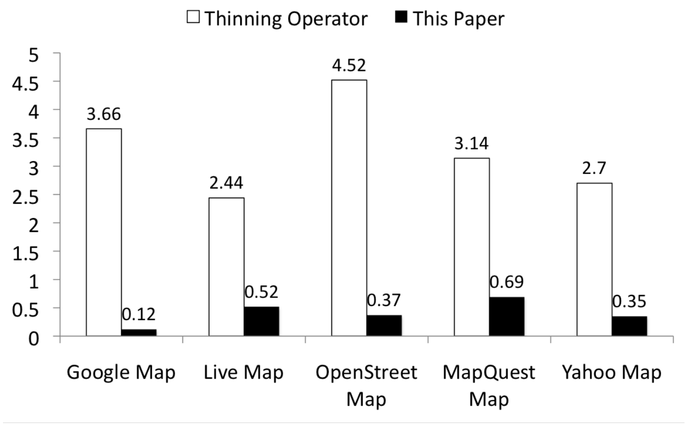
5. RELATED WORK

Much research work has been performed in the field of extracting and recognizing graphic objects from raster maps, such as extracting road intersections [6, 7, 8], separating lines from text [1, 5, 11, 12], and recognizing contour lines [4, 9] from raster maps.

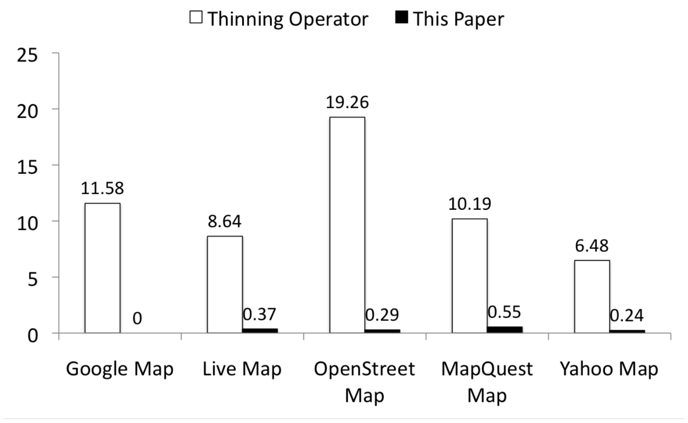
In our previous work [6], we combine a variety of image processing and graphics recognition methods such as the morphological operators to automatically separate the road layers from the raster maps and then extract the road intersection points. Since the thinning operator is used in the work, the extracted road intersections have high positional and orientation offsets when the road width is wide. To overcome this problem, in [7] we utilize the Localized Template Matching (LTM) [2] to improve the positional offset. However, since the road orientations are not accurate, it is difficult for LTM to find the correct matches to improve the positions and further correct the road orientations. In this paper, we utilize the detected road width, the positions of the road intersections, and the thinned lines to trace the road lines and then update the road intersection positions to achieve the best results on the extraction of road intersection templates compared to our previous work.

For the other related work specifically working on extracting road intersections, Habib et al. [8] utilize several image processing algorithms to automatically extract road intersections from raster maps. In [8], they detect the corner points on the extracted road edges and then classify the

[9]The comparison results for this map using only the thinning operator is shown in Figure 6



(a) Positional offset comparison (in pixels)

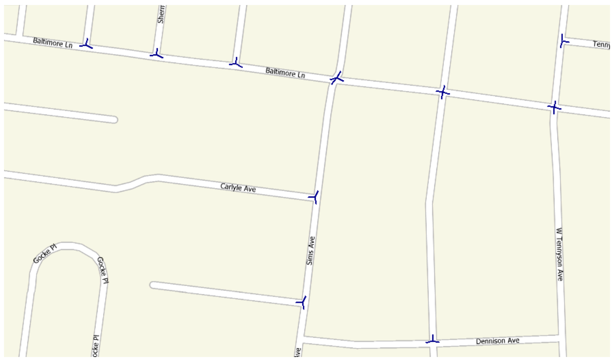


(b) Orientation offset comparison (in degrees)

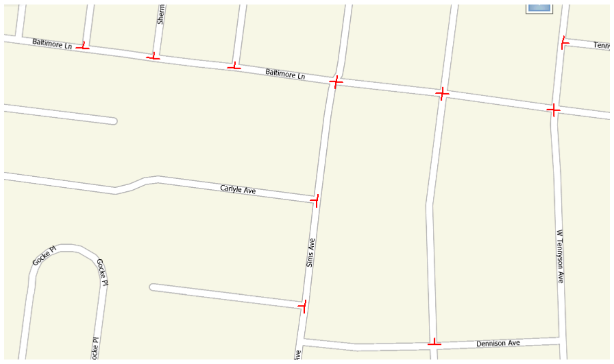
Figure 14: Experimental results using the approach from this paper and our previous work using the thinning operator

Table 2: The positional offset, orientation offset, and the connectivity offset

Map Source	Number of Extracted Road Intersections	Number of Extracted Road Lines	Positional Offset (in pixels)	Orientation Offset (in degrees)	Connectivity Offset (number of lines)
Google Maps	28	87	0.12	0	3
Live Search Maps	28	91	0.52	0.37	2
Yahoo Maps	27	82	0.35	0.24	5
MapQuest Maps	25	83	0.69	0.55	0
OpenStreet Maps	31	95	0.37	0.29	3



(a) MapQuest Maps (thinning operator)



(b) MapQuest Maps (this paper)

Figure 15: Experimental results using the approach from this paper and our previous work using the thinning operator

corner points into groups. The centroids of the classified groups are the road intersections. Without tracing the road lines, false-positive corner points or intersections of T-shape roads significantly shift the centroids away from the correct locations. In comparison, our approach explicitly traces the road lines for accurate positions of the road intersection templates and accurate orientations of the intersecting roads.

To extract text and lines from documents or maps, the research on text/graphics separation [1, 5, 11, 12] performs geometrical analyses of the foreground pixels or exploits the differences between line and character textures to achieve their goals. Cao et al. [1] detect characters from maps where characters overlap lines using the differences in the length of line segments of characters and lines. Nagy et al. [11, 12] first separate the characters from the lines using connected component analysis and then focus on local areas to rebuild the lines and characters using various methods. Our other work in [5] detects the line and character pixels by exploiting the differences between line and character textures in the frequency domain.

Furthermore, Khotanzad et al. [9] utilize a color segmentation method and exploit the geometric properties of contour lines such as that contour lines never intersect each other to extract the contour lines from scanned USGS topographic maps. Chen et al. [4] latter extend the color segmentation method from Khotanzad et al. [9] to handle common topographic maps (i.e., not limited to USGS topographic maps) using local segmentation techniques. However, these text/graphics separation and line extraction approaches do not perform further analysis of the lines to determine the locations of the line intersections and orientations; hence they provide only partial solutions to the problem of extracting the road intersection templates.

6. CONCLUSION AND FUTURE WORK

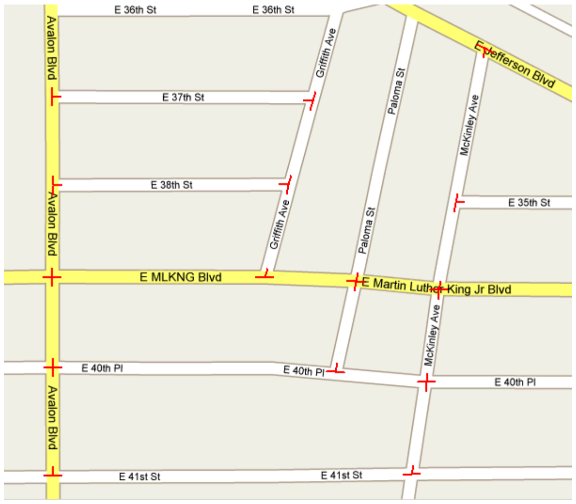
In this paper, we proposed a technique to automatically extract road intersection templates from raster maps. Our experiment shows efficient and accurate results with the positional offset as 0.4 pixels and orientation offset as 0.24 degrees on average. The result is a set of road intersection templates, which consists of the positions and connectivities of the road intersections and the orientations of the roads that intersect at the intersection for each raster map. With the accurate road intersection templates, conflation applications can significantly reduce the possible matching candidates during the point matching process by using the road orientations as one feature to select possible matches. Moreover, applications that work on imagery to extract roads can also benefit from the accurate road intersection templates to enhance the quality of their extraction results.

We plan to extend our work to fully vectorize the road lines from the raster maps. For example, we plan to discover the relationship between the road intersections as well as to extract feature points from the road lines other than road intersections, such as corner points on curve roads. The fully vectorized road network then can be used to match with other vector data to fuse the raster map with other geospatial data even more efficiently.

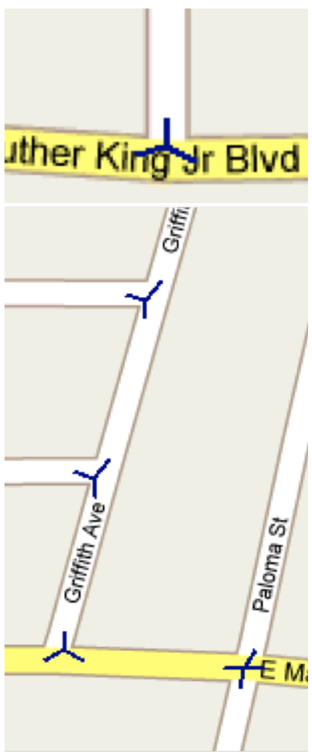
7. ACKNOWLEDGMENTS

This research is based upon work supported in part by the University of Southern California under the Viterbi School Doctoral Fellowship, in part by the National Science Foundation under award number IIS-0324955, in part by the United States Air Force under contract number FA9550-08-C-0010, and in part by a gift from Microsoft.

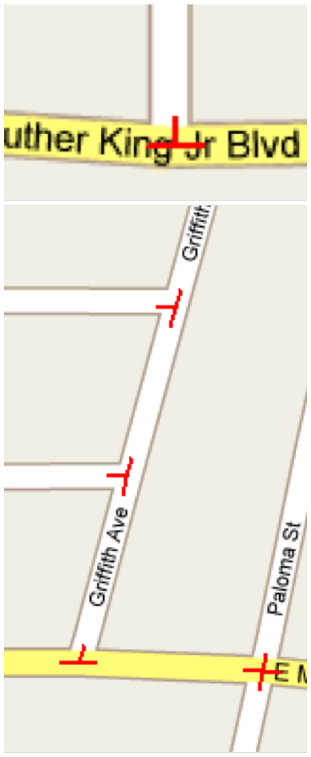
The U.S. Government is authorized to reproduce and dis-



(a) Google Maps (this paper)



(b) Thinning operator



(c) This paper

Figure 16: Detail view of various road intersection templates

tribute 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.

8. REFERENCES

[1] R. Cao and C. L. Tan. Text/graphics separation in maps. In *Proceedings of the Fourth International Workshop on Graphics Recognition Algorithms and Applications*, pages 167–177, Springer-Verlag, London, UK, 2002.

[2] C.-C. Chen, C. A. Knoblock, and C. Shahabi. Automatically conflating road vector data with orthoimagery. *GeoInformatica*, 10(4):495–530, 2006.

[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. Chen, R. Wang, and J. Qian. Extracting contour lines from common-conditioned topographic maps. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):1048–1057, 2006.

[5] Y.-Y. Chiang and C. A. Knoblock. Classification of line and character pixels on raster maps using discrete cosine transformation coefficients and support vector machine. In *Proceedings of the 18th International Conference on Pattern Recognition*, pages 1034–1037, 2006.

[6] Y.-Y. Chiang, C. A. Knoblock, and C.-C. Chen. Automatic extraction of road intersections from raster maps. In *Proceedings of the 13th Annual ACM International Symposium on Geographic Information Systems*, pages 267–276, 2005.

[7] Y.-Y. Chiang, C. A. Knoblock, C. Shahabi, and C.-C. Chen. Automatic and accurate extraction of road intersections from raster maps. *Geoinformatica*, 2008.

[8] A. F. Habib and R. E. Uebbing. Automatic extraction of primitives for conflation of raster maps. Technical report, The Center for Mapping, 1999.

[9] A. Khotanzad and E. Zink. Contour line and geographic feature extraction from USGS color topographical paper maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):18–31, 2003.

[10] G. Koutaki and K. Uchimura. Automatic road extraction based on cross detection in suburb. In *Proceedings of the SPIE, Image Processing: Algorithms and Systems III*, volume 5299, pages 337–344, 2004.

[11] 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 Fifth International Conference on Document Analysis and Recognition*, pages 467–470, 1999.

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

[13] W. K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley-Interscience, 3rd edition, 2001.

[14] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[15] G. W. Zack, W. E. Rogers, and S. A. Latt. Automatic measurement of sister chromatid exchange frequency. *Journal of Histochemistry and Cytochemistry*, 25(7):741–753, 1977.