Proceedings of the Conference on Document Recognition and Retrieval XVII of SPIE-IS&T Electronic Imaging, Volume 7534, 2010.

A General Approach to Discovering, Registering, and Extracting Features from Raster Maps

Craig A. Knoblock[ab], Ching-Chien Chen[b], Yao-Yi Chiang[a], Aman Goel[a], Matthew Michelson[c], and Cyrus Shahabi[ab]

[a]University of Southern California, Information Sciences Institute and Department of Computer Science, Los Angeles, CA, USA;
[b]Geosemble Technologies, 841 Apollo Street, El Segundo, CA, USA;
[c]Fetch Technologies, 841 Apollo Street, El Segundo, CA, USA

ABSTRACT

Maps can be a great source of information for a given geographic region, but they can be difficult to find and even harder to process. A significant problem is that many interesting and useful maps are only available in raster format, and even worse many maps have been poorly scanned and they are often compressed with lossy compression algorithms. Furthermore, for many of these maps there is no meta data providing the geographic coordinates, scale, or projection. Previous research on map processing has developed techniques that typically work on maps from a single map source. In contrast, we have developed a general approach to finding and processing street maps. This includes techniques for discovering maps online, extracting geographic and textual features from maps, using the extracted features to determine the geographic coordinates of the maps, and aligning the maps with imagery. The resulting system can find, register, and extract a variety of features from raster maps, which can then be used for various applications, such as annotating satellite imagery, creating and updating maps, or constructing detailed gazetteers.

Keywords: raster maps, map discovery, map extraction, road extraction, OCR, registration, conflation

1. INTRODUCTION

Maps provide an incredibly rich source of information and are available for most of the world. Figure 1 provides a 1946 Ordnance Survey map that shows the road network, railway lines, waterways, building structures, property lines, and industries, along with text labels for many of these features. But many maps, such as this one, are only available in either paper or as scanned raster files, which makes it difficult to automatically extract the features from them. Modern day maps are typically made from a set of vector layers and rendered as a raster layer for display; but even for many recently constructed maps, the raster or printed maps are around long after the vector layers have been lost or discarded. The challenges are how to find maps of a given area, how to determine the coverage of the discovered maps, how to register the maps with other geospatial sources, such as aerial imagery, and how to extract features, such as the road and text layers, from them.

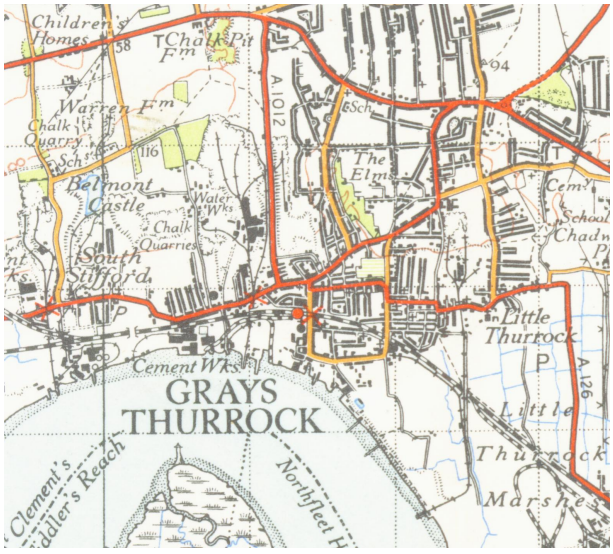


Figure 1. 1946 Ordnance Survey Map

Further author information: Craig A. Knoblock: knoblock@isi.edu, Ching-Chien Chen: jchen@geosemble.com, Yao-Yi Chiang: yaoyichi@isi.edu, Aman Goel: amangoel@isi.edu, Matthew Michelson: mmichelson@fetch.com, Cyrus Shahabi: shahabi@usc.edu

There has been a significant amount of previous work on automatic map processing. This includes work on extracting road intersections,[1] extracting road networks,[2,3] extracting topographic lines,[4–6] performing optical character recognition on the text,[7,8] registering maps with imagery,[9] extracting other types of symbols from maps (e.g., the symbol for a swamp),[10] and so on. While there has been a great deal of work, the one thing that all of this work has in common is that each technique focuses on a particular type of map (for example, the USGS topographic maps* have been a focus in many research papers). Many of the proposed techniques could be applied to other types of maps, but they would first need to be modified or tuned to work well on these maps. The problem here is that there are almost as many types of maps as there are maps since each new map often uses different colors, symbology, fonts, and layers for the different features. In contrast, we developed a general approach to finding and processing raster maps. Unlike the previous work, we do not assume that we have previously seen a given type of map. However, we do assume that the map is drawn to scale, the orientation of the map is known (most maps are oriented with North being up and if not they indicate on the map the North direction), the map has a road network on it, and the map provides a detailed road network (e.g., it does not simply show the freeways or highways). Given these assumptions, our approach can process a large and diverse set of raster maps.

In the remainder of this paper we present our end-to-end approach to discovering raster maps, registering their precise location, and extracting road and text features from the maps. Our approach starts with a name of a city and first searches for maps that cover that location. This is done by finding images using a search engine and then classifying the images into the ones that are likely to be maps (Section 2). Subsequently, for each of these maps the next step is to extract the road and text layers from these maps (Section 3). For the road layer, the system then identifies the intersections in the road layer, including the intersection location, road connectivity, and road orientation for each intersection (Section 4). The extracted intersections are then compared to the knowledge of existing road networks to register the map and determine the mapping between the intersections on the map and the ones on the road network (Section 5). Given this mapping, the system can then align the map with the corresponding imagery using a technique called conflation (Section 6). We then briefly describe the planned next steps in our research on feature recognition from the extracted road and text layers (Section 7). We compare our work to the related work on these topics (Section 8). Finally, we conclude with a summary of the contributions and promising directions for future research (Section 9).

2. RETRIEVING MAPS ON THE INTERNET

There are many interesting maps freely available on the World Wide Web, ranging from street maps to public transportation route maps. Yet, there is no single data source that collects all of them for a given region. Online maps are spread across sites and exist in many different formats. Sometimes they are embedded within documents. Even commercial image search engines are inadequate for retrieving maps. We found that the number of maps returned among the top results for queries like "Tehran maps" is small. For example, at the time of writing this paper, the first result page on Yahoo Image Search† for this query, had only two maps among 21 images on the page.

This motivates a need for automatically and accurately finding maps on the Web. This task breaks into two distinct sub-tasks. First, our method must find candidate map images. This process is based solely on context, rather than deep image analysis, since analyzing the content of each image on the Web would be too expensive both in terms of time and computational effort. The second sub-task then classifies these candidate images as maps or non-maps, based on their image content. We use a K-Nearest Neighbors classifier based on Content Based Image Retrieval (CBIR),[11] where each image is represented by a feature vector. We have developed a set of features based on the Water-filling algorithm[12] that capture the prominent characteristics of maps. In our previous work,[13] we describe our work on this classifier in detail. Figure 2(a) shows the architecture of our system. The rectangles in the diagram represent collections of data (e.g., documents, images) and the parallelograms depict processes.

*http://topomaps.usgs.gov/

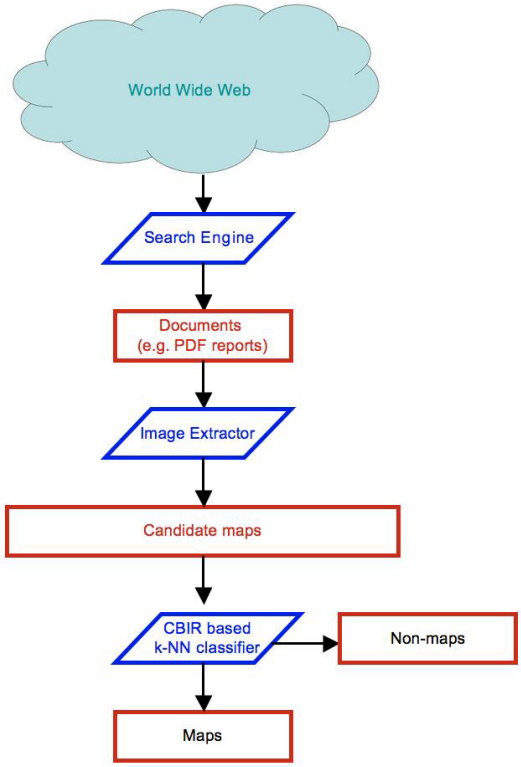
†http://images.search.yahoo.com/

We have observed that most high quality maps on the Web are found embedded within PDF reports and other documents because governments, companies, and other organizations include maps in many of their location-specific reports. Figure 2(b) shows an example of a detailed street map embedded within a larger PDF document. Further, these images are large (from 2MB to 50MB), resulting in high-quality, high-utility maps for our collection. This is in contrast to many of the images that exist on the Web independently of documents, since these images are smaller (to lessen the burden on casual browsers), and hence of lower quality. Therefore, in order to collect candidate maps, we query commercial search engines, such as Google, for PDF documents related to the area. We then download the PDFs, parse them, and extract their images. We have developed a PDF parser, based on the official PDF specification[‡] by Adobe, to extract the embedded images out of the downloaded files.

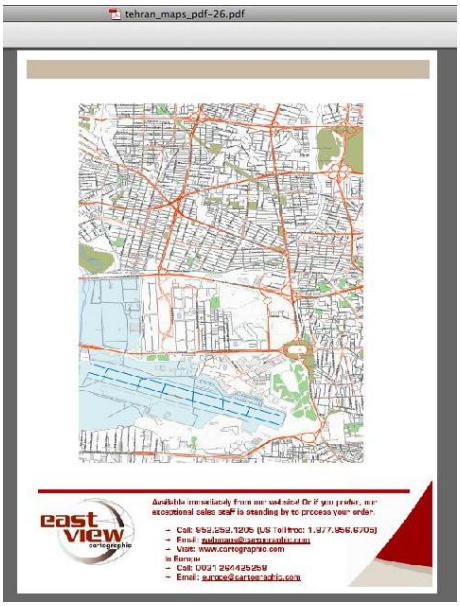
2.1 FEATURES FOR CLASSIFYING IMAGES

Once we collect the initial set of candidate images from documents, such as PDFs, we extract a set of features from these candidate images that we believe capture the defining characteristics of maps and distinguish them from other images. These features are based on the Water-filling algorithm.[12] This algorithm simulates pouring water through the lines of an image. As such, it operates on the edge-map of an image, which makes it color invariant.[§] Then, statistics are generated based on the traversal of the water through the edge map. These statistics include the time required to fill up the sections of the edge map ("Filling Time"), the amount of water required to fill it ("Water Amount"), and the number of points at which the flow of water was divided into two or more streams ("Fork Count"). We note that an edge map can result in a number of disjoint edge sections, and therefore our method calculates these Water-filling statistics for all of the sections.[¶]

Once our approach calculates the statistics for each edge section, it creates a histogram of values for each feature, putting the values into one of eight different bins representing different intervals of values. Thus we get a 24 element feature vector that represents the image. Since the number of edge sections is roughly proportional to the size of the image, we normalize the



(a) Architecture of the map classifier



(b) A sample PDF document with an embedded map

Figure 2. Extracting and classifying maps on the Web

[‡]http://www.adobe.com/devnet/pdf/pdf_reference.html

[§]We use the Canny edge detector[14] to create our edge maps.

[¶]A mid-sized image (800 pixels by 800 pixels) has about 1000 such disjoint edge sections.

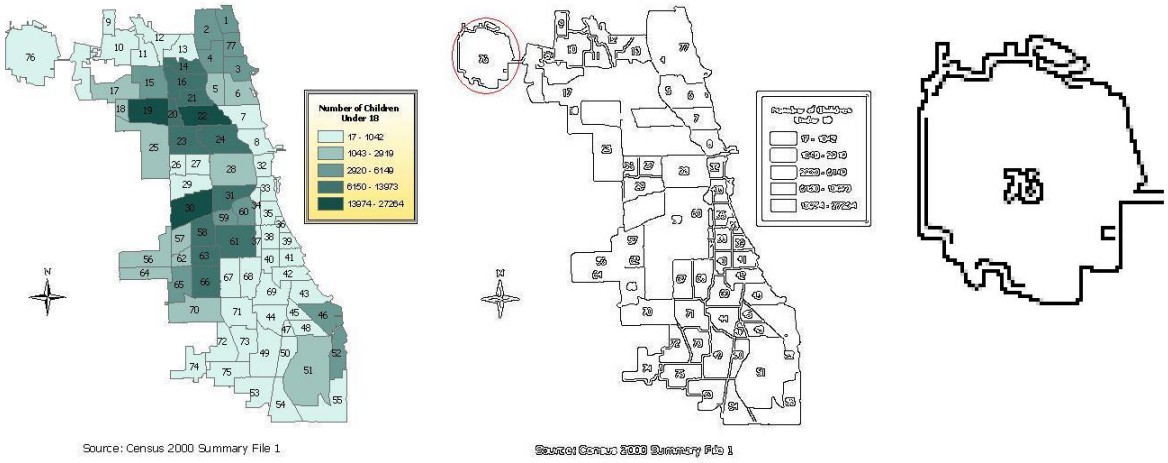


Figure 3. A sample image, its Canny edge map, and a part of it zoomed in to show the disjoint edge section

vector to make it size-invariant by scaling the histograms so that the total number of edge sections is equal to 1000. Figure 3 shows an example image of a map, its Canny edge map, and a section of it enlarged to show the disjoint edge sections.

Map images, in general, have relatively longer lines, which represent streets, freeways and borders with other regions. Accordingly, the Filling Time for most edge sections tends to be high. Also, maps have higher Fork Counts because of the repeated branching of roads or borders. On the other hand, a typical image that is not a map does not have such long lines. For example, a picture of vegetation will have many short segments for edges of leaves on trees and other growth. Therefore non-map images will have Fork Counts of mostly zero or one. The generated histograms reflect this property. A map generally skews toward larger Filling Times, Fork Counts and Water Amounts, whereas a non-map image skews toward shorter segments and lower Fork Counts.

2.2 CLASSIFYING IMAGES USING A CBIR K-NEAREST-NEIGHBOR METHOD

Maps share many common characteristics in general, such as those described above. Yet, they come in a wide variety (street maps, weather maps, physical maps, etc.), where each type has its own characteristics and distribution of features, which are slightly different for each of them. For example, street maps have sparse, straight lines, whereas contour maps have a large number of curved lines. This implies that in addition to boundaries that exist between non-maps and all types of maps, there also exist various overlapping clusters in the feature space corresponding to these subtypes. Yet, defining these sub-classes a priori for classification presents a number of difficulties. For instance, should an urban-hydrographical map be its own class or should it be a member of urban/hydrography maps?

To identify maps we use a K-Nearest Neighbor (K-NN) classifier because it can take advantage of clusters in the feature space and implicitly model the subclasses of maps. Therefore, even as new subclasses arise, the set of neighbors can be extended (say by adding new training maps) rather than having to train a new classifier for a newly defined subclass. To quickly find the closest neighbors to a given image, we use Content-Based Image Retrieval,[11] where an image is represented by a set of features, and similar images (e.g., close neighbors) are those that are closest in the space of these features. We use L1-distance in Euclidian space as a measure of the similarity between images, where the L1-distance between two points is the sum of the absolute differences of their coordinates. Our resulting CBIR-based K-NN classifier takes as input an image represented in our Water-filling feature space, and returns the nine most similar images based on our CBIR approach. If the majority of neighbors are maps, then the input image is classified as a map. Figure 4 shows the schematic of our classifier.

To evaluate this approach, we built a repository of 4000 images (2000 maps and 2000 non-maps) by manually labeling images taken from the Web. We included maps of US and non-US cities and of large and small geographical regions, such as towns as well as continents. This repository covers a large range of subtypes of

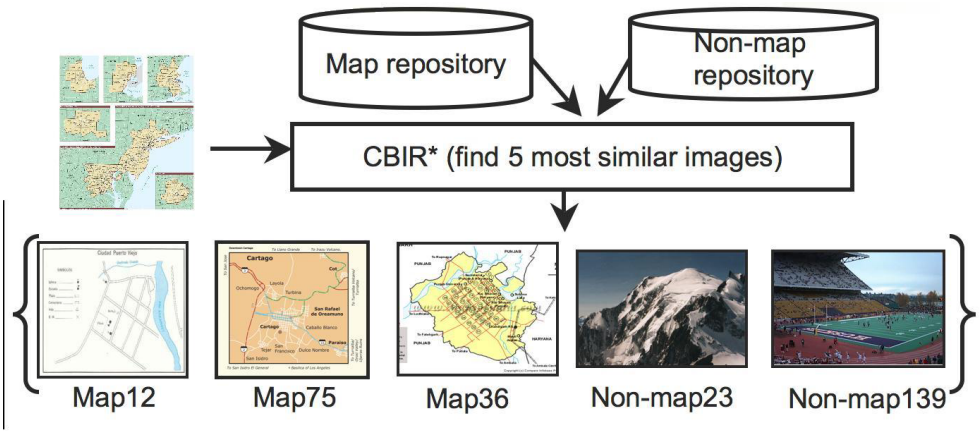


Figure 4. An example of how classification is done using a K-NN classifier based on Content Based Image Retrieval

Table 1. Performance of the two approaches

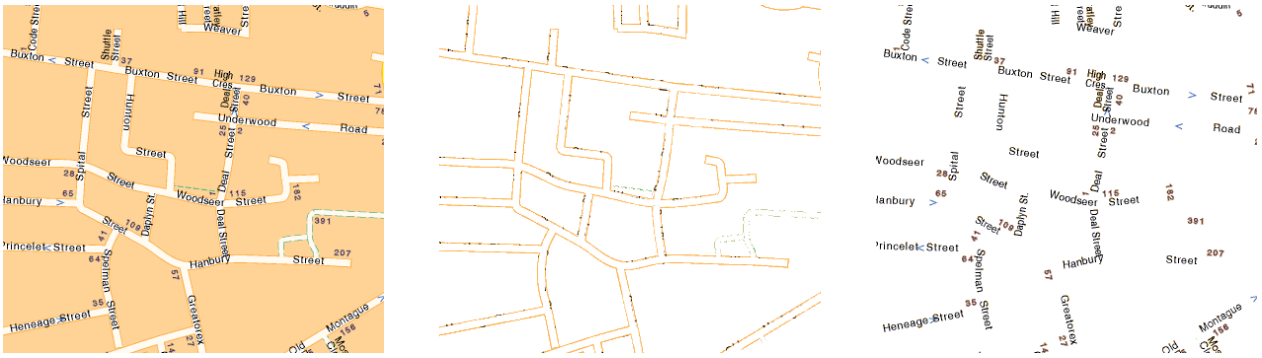
Type of set-up	Precision	Recall	F1-measure
CBIR approach	77.39	71.20	**74.17**
Desai et al.	69.23	47.62	56.43

maps. To test our classifier, we collected 4000 new images by querying Yahoo Image Search for 14 different cities (e.g., Los Angeles, Tehran, Shanghai, etc.) and 2 continents (Asia and Africa). We retained only those images that did not exist in the repository. We then classified these images using our CBIR-based approach as well as using the method proposed in Desai et al.[15] as a baseline comparison. The results of classification using both approaches are given in Table 1. As shown in the table, our approach achieves an F1-measure of 74%, which is about an 18% improvement over an earlier approach to the same problem.

3. EXTRACTING ROAD AND TEXT LAYERS FROM MAPS

Once a set of maps has been collected from the Web, the next step is to unlock the geospatial information hidden in those maps. The first step in this process is to separate the pixels of each geographic feature from a map. A set of extracted pixels representing a particular geographic feature in a raster map is called a feature layer, where the road pixels constitute the road layer and the character pixels constitute the text layer. Figure 5 shows a raster map from the Via Michelin website[‖] and the extracted road and text layers from the map. The extracted feature layers can be used to annotate other geospatial data; for example, we can align the extracted text layer

[‖] http://www.viamichelin.com



(a) A raster map from Via Michelin (b) The extracted road layer (c) The extracted text layer

Figure 5. An example of a raster map and its extracted road and text layers

with imagery to label the roads in the image, as shown in Figure 6. Further, we can convert the raster layers to machine-readable geographic features (e.g., road vectors) and create map meta data for indexing the map (as described in Section 7).

Figure 6. Labeling roads in an image using a map text layer

The extraction of feature layers is a challenging task because of the varying image quality of raster maps (e.g., poor quality scanned maps), the complexity of maps (i.e., overlapping features in maps), and the typical lack of meta data (e.g., map geocoordinates, map source, vector data source). To overcome these difficulties, we developed map decomposition techniques, each requiring a different amount of user effort to extract the road and text layers from heterogeneous raster maps.[16–20] Figure 7 shows our overall approach to handling maps of varying quality. Figure 8(a) shows an example of a map with good image quality from the Census TIGER/Line source.** Since we retrieved the raster map directly from the TIGER/Line website without compressing the image and the map is generated digitially from vector data without scanning, the map has consistent background colors and contains fewer colors overall compared to the scanned maps shown in Figure 8(b) and Figure 8(c). For the raster maps with good image quality such as the TIGER/Line map, we developed an automatic approach that exploits common map properties to extract the feature layers from the maps.[16,18] We utilize the fact that the map background generally has a dominant numbers of pixels compared with the foreground and the foreground has a high contrast against the background to first remove the map background and then extract the foreground pixels. We then exploit the distinctive geometry between the road lines and character strokes (e.g., road lines are longer compared to character strokes, character strokes are near each other) to separate the road and character pixels from the foreground pixels. The automatic approach has the benefit of using no prior knowledge of the map and no user input, but it cannot handle raster maps with poor

**http://www.census.gov/geo/www/tiger/

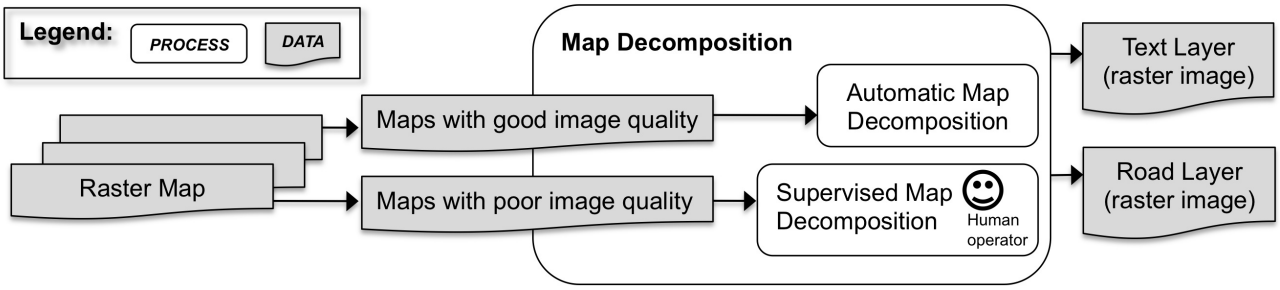
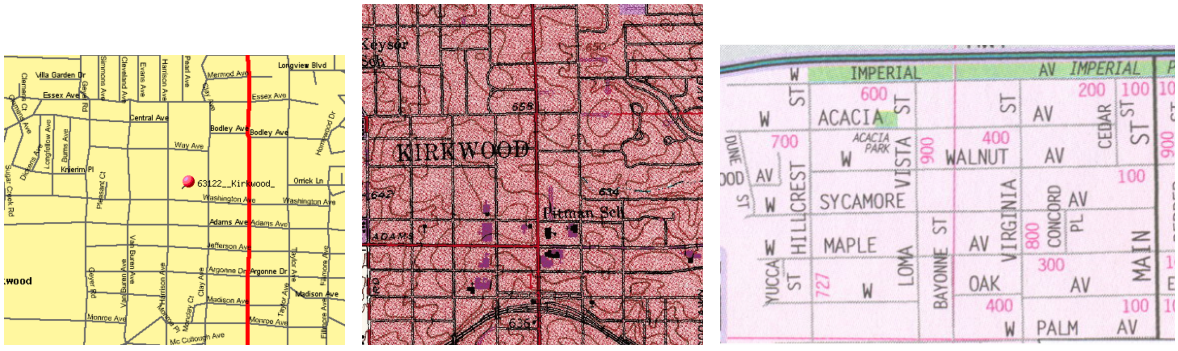


Figure 7. The overall approach to extract road and text layers from raster maps

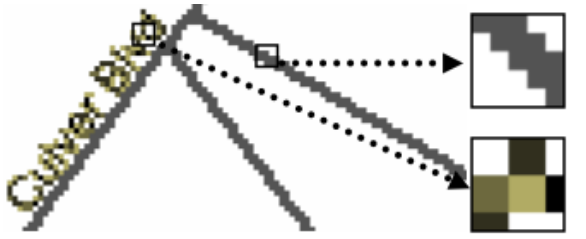


(a) A TIGER/Line map (b) A USGS topographic map (c) A scanned Thomas-Guide map

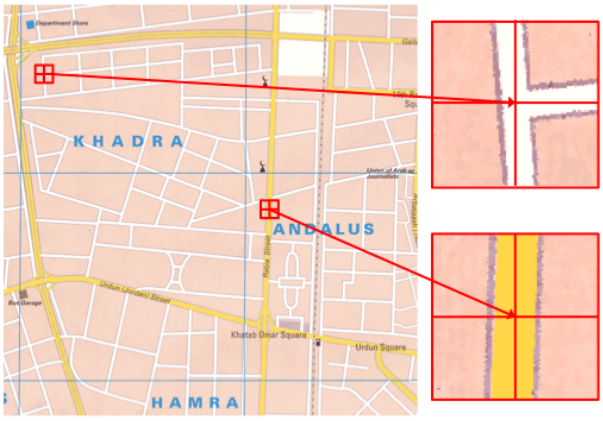
Figure 8. Example raster maps

image quality. This is because the raster maps with poor image quality, such as scanned maps, usually contain numerous colors and the background does not have a consistent color usage, so the map properties assumed in our automatic approach do not hold for these maps.

To achieve broad coverage of various map sources and varying image quality, we developed a supervised approach, which requires user labeling to extract the road and text layers from raster maps.[17,20] The supervised techniques can be used on scanned and compressed maps that are otherwise difficult to process automatically and tedious to process manually (i.e., manually classify each pixel/color in the raster map based on the geographic feature the pixel/color represents). One of the supervised techniques that we developed is a pixel-based classification technique that exploits the texture differences between areas around text pixels, line pixels, and background pixels to automatically classify each map pixel for extracting the feature layers.[17] Within a local area, the textures of the foreground and the background are different since the colors of the background are more 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 9(a). This pixel-based supervised approach enables the extraction of road and text layers from poor quality maps, but the training process requires the user to provide individual locations of roads, characters, and background pixels in the raster map, which can be a laborious task if the image quality is very poor and requires many training samples.



(a) Texture differences between text and line areas



(b) User labels centered at road lines

Figure 9. The properties that the two supervised approaches exploit for extracting the feature layers

We also developed a color-based supervised map decomposition approach that analyzes user labels of road areas (i.e., rectangular areas specified by the user) to identify the road colors used in a raster map for extracting the road layer.[20] The approach that the color-based technique uses to minimize user input is to exploit the fact that a user label is required to be centered at a road line or a road intersection, as shown by the two user labels in Figure 9(b). Since a user label is centered at a road line, if the pixels of a particular color in the user label constitute straight lines (i.e., we detect Hough lines[21] for identifying straight lines) and the lines are near the center of the user label, the pixels are road pixels and we classify the pixel color as a road color. Using this approach, the user only has to provide enough user labels to cover each road color in the raster map. For example, Figure 9(b) shows an example scanned map and the two user labels required (i.e., one for the white roads and one for the yellow roads) to extract the road layer from the map. In the experiments on the color-based approach,[20] we show that the average number of user labels to extract the road layer from 100 test maps is less than four per map, which is significantly less user input compared to providing pixel samples in the pixel-based approach. In order to avoid having a user label every map, we also developed a method of classifying maps based on the spatial relationship between colors in the maps, which allows the system to determine whether it can use the learned road colors from maps on which the system had been previously trained.[22]

4. RECOGNIZING ROAD INTERSECTIONS

From the extracted road layer, we developed an automatic approach to extract a set of road-intersection templates,[16,18,19] which represents an abstraction of the road network in a raster map. Figure 10 shows an example of a road-intersection template, which consists of the road intersection position, the road connectivity, and the road orientation. Since road networks commonly exist in various geospatial data, the set of road-intersection

templates of a raster map can serve as a reference feature in a conflation system[23,24] to compute a transformation matrix for aligning the map with other geospatial layers, such as imagery. Further, the aligned road-intersection templates can be used as seed templates for extracting roads from imagery.[3]

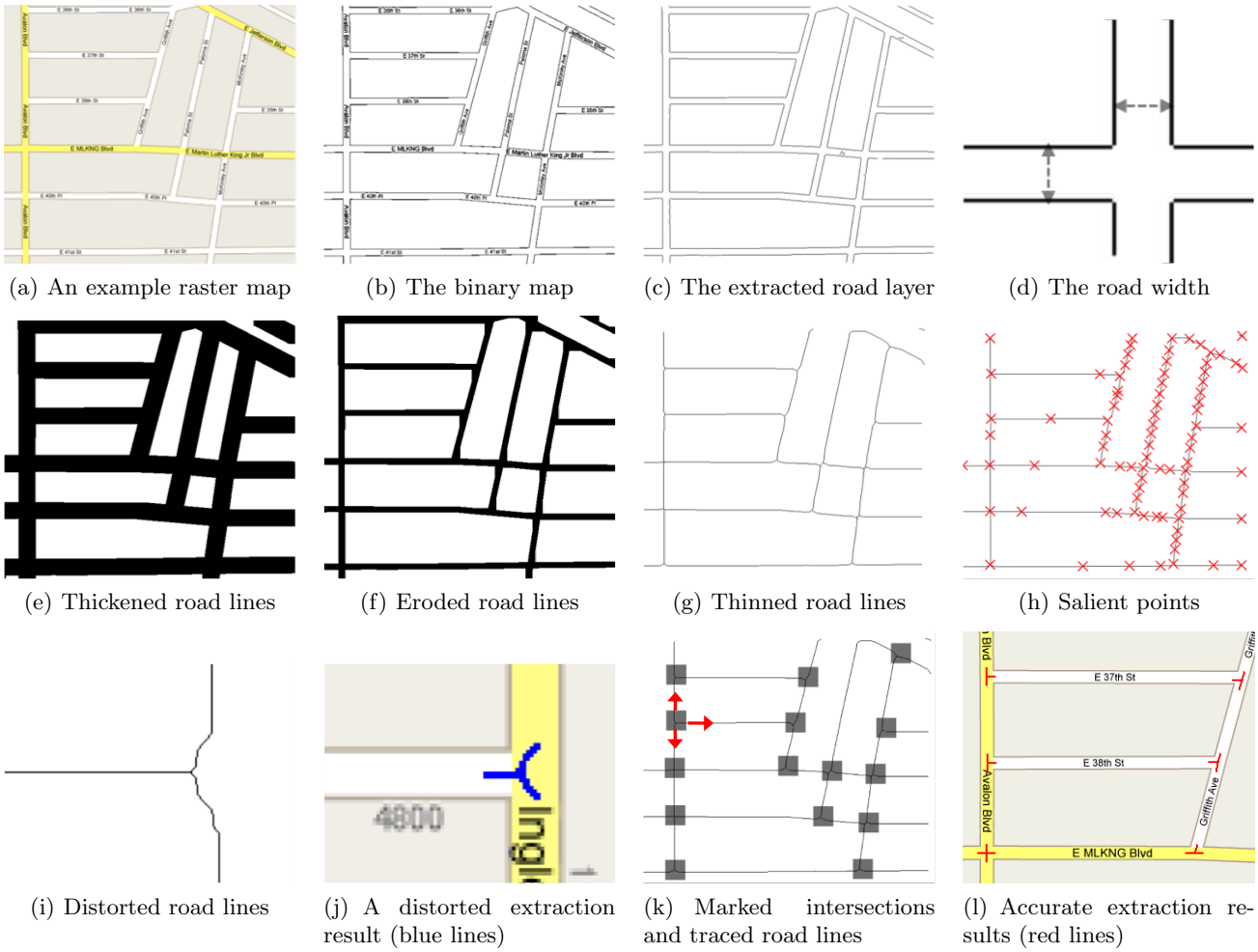
Figure 10. An example road-intersection template

Figure 11(a) and Figure 11(b) show an example map and the map's foreground layers (i.e., the binary map). Figure 11(c) shows the extracted road layer using our map decomposition approach described in the previous section. Some of the road lines in the extracted road layer are broken since a portion of the road pixels also belong to the text layer (i.e., overlapping features), such as where characters touch road lines. These shared pixels are removed during the layer extraction processes when we separate the road and character pixels from the foreground pixels. To reconnect the broken road lines and produce the road topology (i.e., the central-line representation of the road network) for extracting the road intersections, we first identify the road width and road format (i.e., double-line and single-line roads) of the road layer. We developed the Parallel-Pattern Tracing algorithm (PPT),[16,18] which employs two convolution masks working on the horizontal and vertical directions to search for corresponding pixels of parallel lines to determine the road width and road format automatically. In a road layer where road lines are drawn in two parallel lines (i.e., double-line format) as shown in the example in Figure 11(c), the road width is the pixel distance between corresponding road pixels on the parallel lines, as shown by the gray dashed lines in Figure 11(d). If a road line is drawn using one line (i.e., single-line format), the detected road width is the thickness of the majority of the road lines. The road width and road format help to determine the parameter settings in the next step, which uses morphological operations[25] to produce the road topology.

To produce the road topology, we first thicken the road lines to reconnect broken lines by utilizing the binary dilation operator as shown in Figure 11(e). During the thickening process using the dilation operator, we also merge parallel lines into thick single lines if the road layer is double-line format, and we determine the number of iterations of the dilation operator (i.e., how far the foreground region should be expanded) using the road width. Then, we apply the binary erosion operator and the thinning operator to shrink the thickened road lines and generate one-pixel width lines as shown in Figure 11(f) and Figure 11(g). We use the erosion operator to shrink the lines before we apply the thinning operator so that the thinning operator does not have to be applied directly on the thick lines. This is 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.

With the one-pixel-width road lines, we utilize the corner detector[26] to detect salient points as candidates for road intersections as shown with the cross marks in Figure 11(h). A salient point is a point where two lines meet at different angles and a road intersection is a point where *more* than two lines meet at different angles, so we can compute the connectivity of each salient point to identify actual road intersections. We draw a box around each salient point and then use the number of foreground pixels that intersect with this rectangle as the connectivity of the salient point. If the connectivity is less than three, we discard the point; otherwise, it is identified as a road intersection.

From an identified road intersection, we trace the road lines connected at the intersection to generate a road-intersection template.[19] Although the binary erosion operator helps to minimize the extent of the distortion caused by the thinning operator, the road topology near the intersections is still not accurate, especially near T-shape intersections. Figure 11(i) shows a distorted example of the road topology around a T-shape intersection and Figure 11(j) shows an inaccurate road-intersection template if the distorted lines are traced to generate the result. We use the road width detected using PPT to mark potential distortion areas as shown by the gray boxes in Figure 11(k). This is because the extent of the distortion is determined by the thickness of the thickened lines, which is determined by the number of iterations of the binary dilation operator based on the road width. 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. In the experiments on this work,[19] we show that the geometry and positions of our extracted road-intersection templates are very close to the ground truth (i.e., manually drawn road-intersection templates). Figure 11(l) shows a portion of the extraction results for Figure 11(a). With the accurate results, conflation applications that use the road-intersection templates as a



(a) An example raster map (b) The binary map (c) The extracted road layer (d) The road width

(e) Thickened road lines (f) Eroded road lines (g) Thinned road lines (h) Salient points

(i) Distorted road lines (j) A distorted extraction result (blue lines) (k) Marked intersections and traced road lines (l) Accurate extraction results (red lines)

Figure 11. Automatic extraction of road-intersection templates

reference feature can reduce their search space during the matching processes by using the road orientation and connectivity as features to select possible matches. Moreover, applications that work on extracting roads from imagery also benefit from the accurate road-intersection templates since the application can start with accurate seed templates for identifying road areas.

5. DETERMINING THE GEOCOORDINATES OF A MAP

Once we have identified a set of road intersections from a map, we can in turn compare those intersections with existing georeferenced road vector data to determine the coordinates of the map. We can obtain road vector data with known coordinates for various regions of the world. For example, the US Census TIGER/Line files[††] contain road networks for the United States and NAVTEQ[‡‡] NAVSTREETS covers the entire US as well as many other regions worldwide. A prerequisite step is processing those vector datasets to produce a road intersection database in order to compare the intersections from the map. We have built such a database covering the entire US, where we found 11.4 million road intersections.

After detecting a set of road intersection points from each dataset (i.e., the map and the existing road vector data) separately, the remaining problem is how to match these intersection points effectively and efficiently to locate a common distribution (or pattern) in these intersections. Intuitively, our system uses the layout of the

[††]http://www.census.gov/geo/www/tiger/
[‡‡]http://www.navteq.com/

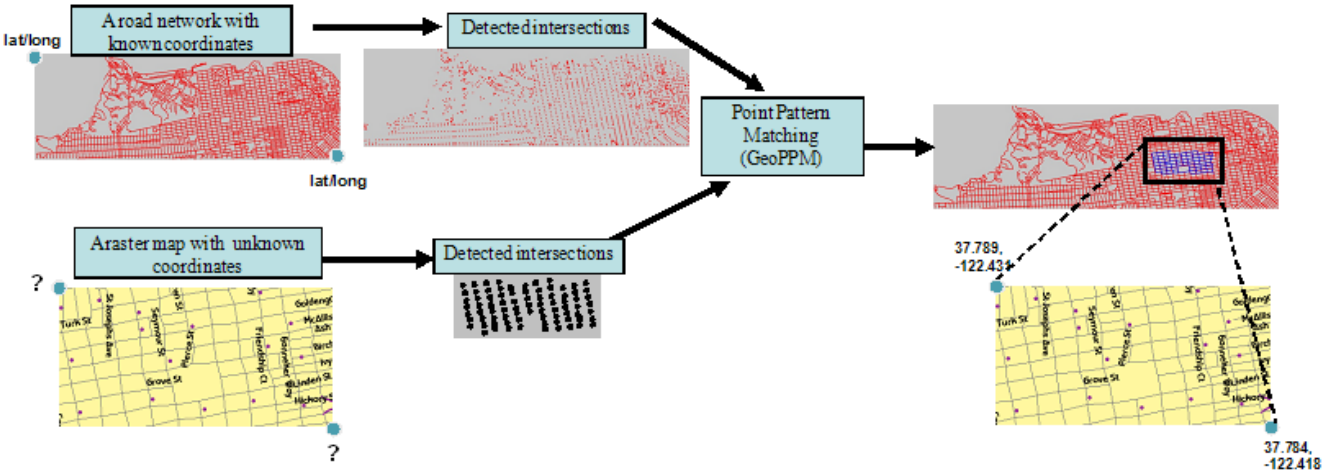


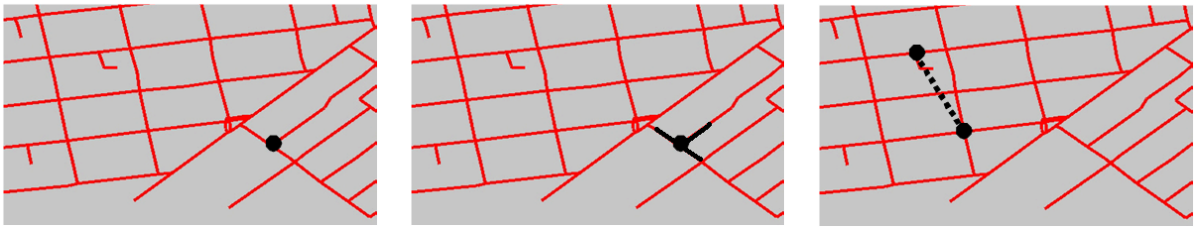
Figure 12. The overall approach to determine the coordinates of a map

detected intersections from each dataset to compute their geospatial relationship. Figure 12 shows our overall approach. Using detected road intersections as input, our system locates the common point pattern across these two point sets by computing a proper transformation T between them. The system can then utilize this transformation to determine the coordinates of the map.

The transformation T is a 2D rigid motion (rotation and translation) with scaling. Because the majority of maps are oriented such that north is up, we only compute the translation transformation with scaling. A brute-force method to resolve the point pattern matching problem would be to generate all possible matching point pairs from both point sets to obtain the translation and scaling factors. Each transformation thus generated could then be applied to the entire set of points in a map to determine whether the majority of the points can be aligned to another point set. Since this algorithm would be very time consuming, we developed a number of techniques to improve its performance by exploiting auxiliary information, such as the map scale, the connectivity of the intersections (i.e., the number of intersected road segments), and the density of these intersections. The basic idea is to exclude all unlikely matching point pairs. For example, given a point pair (x1, y1) and (x2, y2) from the map, we need to only consider pairs (lon1, lat1) and (lon2, lat2) from the road vector, such that the real world distance between (x1, y1) and (x2, y2) is close to the real world distance between (lon1, lat1) and (lon2, lat2). In addition, (x1, y1) and (lon1, lat1) would be considered as a possible matching point if and only if they have similar road connectivity and orientation. We named this approach GeoPPM and Figure 13 illustrates how GeoPPM works using an example. This enhanced algorithm improves the execution time by at least two orders of magnitude. In our previous papers[24, 27] we provide additional details on these techniques.

6. ALIGNING A MAP WITH IMAGERY

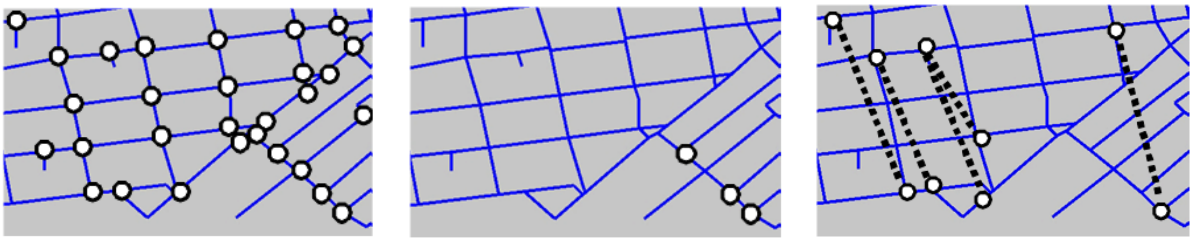
After GeoPPM generates a set of matched point pairs (called "control point pairs") for the map and road vector data, we can deform the map to align it to the road vector data utilizing these identified control point pairs. Furthermore, in our previous work,[23] we developed a technique to efficiently and automatically align road vector data with orthorectified imagery. This implies that using the road vector as glue, we can align a map with the corresponding imagery. We first align a road vector data with an image, and then we match map intersections with the road vector data. Finally, we apply a technique, called rubber-sheeting, to align the map and imagery based on the matched control-point pairs. Intuitively, imagine stretching a map as if it was made of rubber. The rubber-sheeting algorithm deforms a map algorithmically, forcing registration of control points over the map with their corresponding points on the imagery to accomplish the overall alignment. Rubber-sheeting is a commonly-used method for aligning various geospatial datasets.[9] Figure 14 illustrates how the rubber-sheeting technique partitions both datasets into triangular regions to define the localized transformations based on the



(a) A georeferenced road vector dataset with a selected intersection

(b) An selected intersection from the vector data with connectivity and orientation information

(c) A selected point pair from the vector data including the angle between the selected points



(d) The candidate points on the map for the point selected in (a) if there is no filtering to prune the search space

(e) The candidate points on the map for the point selected in (b) using the intersection connectivity and orientation

(f) The candidate point pairs on the map for the selected point pair in (c) based on the angles between the points

Figure 13. Comparing a road vector dataset (a-c) with a map (d-f) using GeoPPM to prune the search space

Figure 14. Partition each dataset into corresponding triangular regions based on the control point pairs (black points on the image and corresponding white points on the map)

control point pairs. Figure 15 illustrates the map-imagery alignment result after replacing the image pixels semi-transparently with the corresponding pixels on the map by using the computed transformation coefficients.

7. NEXT STEPS: RECOGNIZING THE ROAD AND TEXT LAYERS

Now that we can extract the map layers and determine the precise geocoordinates of a map, our next step is to recognize the features of the extracted layers. In particular, we want to build accurate road layers by turning the extracted road layer into vector data. We also plan to work on the problem of performing optical character recognition on the text layers. This problem can be especially

Figure 15. The map to imagery alignment result

challenging on maps with a lot of text close to the road lines and text that follows the directions of the roads. Subsequently, given the resulting features, we plan to work on the problem of how to associate the text labels with the road vectors. In this section, we describe our initial work on road and text recognition. In Section 3, we already described how we can extract the road and text layers from a map. However, the output of that step is the corresponding raster layers for each feature. The next step is to turn the road layer into vector data and the text layer into text. This builds on the previous steps for extracting the raster layer for the roads and text, extracting accurate intersection points, and aligning the road and text layers with the imagery.

There are two general challenges in turing a road layer from a map into an accurate vector layer. First, the location of the vectors may not be accurate because of inaccuracies or lack of meta data about the original map. This problem is addressed by our approach to automatically registering a map with an image as described in Section 5. The output of the registration process is a set of control-point pairs that were used in Section 6 to align a map with an image. The same set of control point pairs can be applied to the extracted road-vector data to conflate the road vector data with the orthorectified aerial imagery, which will ensure that the vector data will be properly aligned for a given region. The second challenge is that the road layer extraction process may introduce additional inaccuracies through the use of the morphological operations needed to fill in missing pixels and construct the single-line road network. To address this problem, we use the method for accurately recognizing the road intersections as the starting point for the extraction of the road vector data. As described in Section 4, we are able to accurately extract both the location and angles of the roads that comprise an intersection. We use these extracted intersections as the seed points to trace the road lines and extract the vector data from the road layer.[28]

Recognizing the text layers is even more challenging. Some of the issues are that there may be a variety of fonts and font sizes, the characters may be broken where they intersect with the roads, the road names may have a variety of orientations, and on some maps the names may even follow the curvature of the roads. We plan to start with commercial optical character recognition (OCR) software and apply it to the text of the maps since these systems have a lot of knowledge and training to handle various fonts and character sets. However, in order to use these systems we have to address the problem of orienting the text so that it is displayed horizontally from left to right. One way to do that is to use the fact that the road names typically follow the orientations of the roads to determine the correct orientation for the OCR system. We can also use the fact that characters are grouped into words and we can group the characters to find the most likely orientation. Another problem is that given a noisy text layer, the OCR system may make recognition errors on the individual characters. To address this problem we plan to use background knowledge for a given region that provides the names of roads and other features for the area. Thus, instead of just matching individual characters, OCR can be applied in the context of entire words such that feature names in the knowledge base are preferred over unknown feature names.

8. RELATED WORK

In our previous work on harvesting maps from the internet,[15] we used Laws' Texture[29] as the representative feature set to differentiate between maps and non-maps, with Support Vector Machines[30] as the classifier. In this work we use Water-filling features with a CBIR-based K-NN classifier. As demonstrated by our results, the CBIR approach with Water-filling features surpasses the previous performance by about 18% in F1-measure and yields a substantial processing improvement in both time and memory requirements. A lot of research has been done in the area of finding the right features for comparing images. While we use Water-filling features as they accurately capture the key features of a map, other studies have proposed "salient point" features based on wavelets[31] and shape similarity.[32]

The CBIR-based, K-Nearest Neighbor approach has been previously used to classify medical images.[33] That work also includes Water-filling features for the CBIR component. However, this combination performs the worst among the five different classification systems they tested. This is because Water-filling features work best on images with sharp boundaries and no color gradient and medical images like X-rays have amorphous boundaries and gradual color gradients. On the other hand, maps satisfy the requirement of the Water-filling algorithm precisely and we have exploited it to our benefit. In addition, the context in which they apply these algorithms is very different from ours. Our system is geared towards automatically harvesting maps from the Web, while their system is used to classify images so that they can be queried categorically.

In the previous work on text/graphics separation from raster maps, Cao and Tan[7] and Li et al.[8] 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. The road pixels are the by-product after the text pixels are identified. Since in their work[7,8] the main goal is to recognize the text labels, they do not perform further processing to extract the road topology of the raster map. For the previous work that focuses on recognizing road features (e.g., road lines and intersections) from the raster maps, that work assumes a simpler type of raster map for their automatic processing. Habib et al.[1] extract road intersections from raster maps that contain road lines only. Itonaga et al.[2] employ a stochastic relaxation approach to first extract the road areas and then apply the thinning operator to extract a one pixel-width road network from digitally generated maps (i.e., not scanned maps). These approaches each handle a specific type of map. In comparison, the approach presented in this paper is not limited to a specific map type and can be used on heterogeneous raster maps, including scanned maps and maps that contain overlapping feature layers.

Other map processing techniques require user training to process raster maps with low image quality. Salvatore and Guitton[4] use a color thresholding method as their first step to extract contour lines from topographic maps. Khotanzad and Zink[5] utilize a color segmentation method with user annotations to extract the contour lines from the USGS topographic maps. Chen et al.[6] extended the color segmentation method in Khotanzad and Zink's work[5] to handle common topographic maps (i.e., not limited to the USGS topographic maps) using local segmentation techniques. The techniques with user training are generally able to handle maps that are more complex. However, the user training processes are complicated and labor intensive, such as manually generating color thresholds for every input map[4] and labeling all combinations of line and background colors.[5] In comparison, our supervised color-based map decomposition approach requires the user to label only a few road areas, which is simpler and more straightforward.

There have been a number of efforts to automatically or semi-automatically detect matched features across different GIS datasets.[9,34–36] Given a feature point from one dataset, these approaches utilize different matching strategies to discover the corresponding point on another dataset within a predetermined distance. Typically, these existing algorithms only handle the matching of geospatial datasets in a known geometry systems (e.g., the same coordinate system). Moreover, various commercial GIS systems, such as ESEA MapMerger, have been implemented to achieve the matching of vector datasets with different accuracies. However, most of the existing commercial systems require manual work to transform two GIS datasets into the same geocoordinates beforehand. There are no other automatic methods to resolve the matching and alignment of a map of unknown coordinates with other datasets (i.e., road vector data or imagery). Furthermore, our technique infers the coordinates and scale of the map.

9. CONCLUSION

In this paper we described a general approach to discovering, registering, and extracting features from raster maps. The contributions of this work include the ability to identify maps from other types of images, the ability to extract road and text layers from a raster map, the automatic recognition of road intersection, including the accurate extraction of the cardinality and angles of the roads, the algorithms to automatically determine the geocoordinates of a map using background data on the road network, and the techniques for aligning a map with aerial or satellite imagery. A key part of this contribution are techniques that are not specific to a given map type and can be applied to a wide range of raster maps. The result of the combination of all of these techniques means that we can build systems that can automatically discover maps for a region, determine their precise coverage and scale, accurately overlay the maps on top of imagery, and build a database of the road and text layers that can be used for linking other types of information or even building new maps. Overall, these techniques provide the ability to exploit the tremendous amount of information contained in raster maps that was previously unavailable.

Beyond the current work on recognizing the road and text features in a map and associating the two, there are a number of interesting directions for future research. We would like to remove or reduce some of the assumptions identified in the beginning of the paper in terms of what types of maps can be processed. First, we would like to be able to handle abstract road maps, which are maps that do not contain fully detailed road networks and maps where the orientation is not known. Second, we would like to eliminate the need to know the general area

of a map in order to register a map. Once we can perform the OCR on the text layer, we can then use the recognized text to focus on the most likely area or areas of a map, which will make it possible to process a map without any knowledge of its general location. Third, we would like to remove the requirement that we have a vector dataset of an area in order to align a map with that location. Once we are able to extract a road vector layer from a map, we can then use our previous work on aligning road vector data with imagery[23] to create a vector layer for areas where no such vector layer is available. Then other maps of that region can be registered and aligned using this new vector dataset.

ACKNOWLEDGMENTS

This research is based upon work supported in part by the United States Air Force, Air Force Research Laboratory, Office of Scientific Research under contract number FA9550-08-C-0010 and grant number FA9550-07-1-0416, and in part by a gift from Microsoft. 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. A. Habib, R. Uebbing, and A. Asmamaw, "Automatic extraction of road intersections from raster maps," tech. rep., Center for Mapping, 1999.
2. 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), pp. 62–72, 2003.
3. G. Koutaki and K. Uchimura, "Automatic road extraction based on cross detection in suburb," in *Proceedings of the SPIE, Computational Imaging II* **5299**, pp. 337–344, 2004.
4. S. Salvatore and P. Guitton, "Contour line recognition from scanned topographic maps," in *Proceedings of the Winter School of Computer Graphics*, 2004.
5. 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), pp. 18–31, 2003.
6. Y. Chen, R. Wang, and J. Qian, "Extracting contour lines from common-conditioned topographic maps," *IEEE Transactions on Geoscience and Remote Sensing* **44**(4), pp. 1048–1057, 2006.
7. R. Cao and C. L. Tan, "Text/graphics separation in maps," in *Proceedings of the Fourth International Workshop on Graphics Recognition Algorithms and Applications*, pp. 167–177, 2002.
8. 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), pp. 177–185, 2000.
9. A. J. Saalfeld, *Conflation: automated map compilation.* PhD thesis, University of Maryland, College Park, MD, USA, 1993.
10. G. K. Myers, P. G. Mulgaonkar, C.-H. Chen, J. L. DeCurtins, and E. Chen, "Verification-based approach for automated text and feature extraction from raster-scanned maps," in *Lecture Notes in Computer Science*, **1072**, pp. 190–203, Springer, 1996.
11. A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(12), pp. 1349–1380, 2000.
12. X. Zhou, Y. Rui, and T. Huang, "Water-filling: A novel way for image structural feature extraction," in *Proceedings of the International Conference on Image Processing*, pp. 570–574, 1999.
13. M. Michelson, A. Goel, and C. A. Knoblock, "Identifying maps on the World Wide Web," in *Proceedings of the 5th International Conference on Geographic Information Science, LNCS 5266*, pp. 249–260, Springer, (New York), 2008.
14. J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence* **8**, pp. 679–714, 1986.
15. S. Desai, C. A. Knoblock, Y.-Y. Chiang, K. Desai, and C.-C. Chen, "Automatically identifying and georeferencing street maps on the Web," in *Proceedings of the 2nd International Workshop on Geographic Information Retrieval*, pp. 35–38, 2005.

16. Y.-Y. Chiang, C. A. Knoblock, and C.-C. Chen, "Automatic extraction of road intersections from raster maps," in *Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems*, pp. 267–276, 2005.
17. Y.-Y. Chiang and C. A. Knoblock, "Classification of line and character pixels on raster maps using discrete cosine transformation coefficients and support vector machines," in *Proceedings of the 18th International Conference on Pattern Recognition*, pp. 1034–1037, 2006.
18. 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), pp. 121–157, 2008.
19. 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 SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 1–10, 2008.
20. Y.-Y. Chiang and C. A. Knoblock, "A method for automatically extracting road layers from raster maps," in *Proceedings of the Tenth International Conference on Document Analysis and Recognition*, 2009.
21. D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall Professional Technical Reference, 2002.
22. Y.-Y. Chiang and C. A. Knoblock, "Classification of raster maps for automatic feature extraction," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009.
23. C.-C. Chen, C. A. Knoblock, and C. Shahabi, "Automatically conflating road vector data with orthoimagery," *GeoInformatica* **10**(4), pp. 495–530, 2006.
24. C.-C. Chen, C. A. Knoblock, and C. Shahabi, "Automatically and accurately conflating raster maps with orthoimagery," *GeoInformatica* **12**(3), pp. 377–410, 2008.
25. W. K. Pratt, *Digital Image Processing: PIKS Scientific Inside*, Wiley-Interscience, 3rd ed., 2001.
26. J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.
27. C.-C. Chen, C. A. Knoblock, C. Shahabi, S. Thakkar, and Y.-Y. Chiang, "Automatically and accurately conflating orthoimagery and street maps," in *Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS'04)*, 2004.
28. Y.-Y. Chiang and C. A. Knoblock, "An approach to automatic road vectorization of raster maps," in *Proceedings of the 8th IAPR International Workshop on Graphics RECognition (GREC'09)*, 2009.
29. K. Laws, *Textured Image Segmentation*. PhD thesis, University of Southern California, 1980.
30. V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.
31. Q. Tian, N. Sebe, M. Lew, E. Loupias, and T. Huang, "Image retrieval using wavelet-based salient points," *Journal of Electronic Imaging* **10**(4), pp. 835–849, 2001.
32. L. Latecki and R. Lakamper, "Shape similarity measure based on correspondence of visual parts," *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(10), pp. 1185–1190, 2000.
33. T. Lehmann, M. Guld, T. Deselaers, D. Keysers, H. Schubert, K. Spitzer, H. Ney, and B. Wein, "Automatic categorization of medical images for content-based retrieval and data mining," *Computerized Medical Imaging and Graphics* **29**, pp. 143–155, 2005.
34. P. Dare and I. Dowman, "A new approach to automatic feature based registration of SAR and SPOT images," *International Archives of Photogrammetry and Remote Sensing* **33**(B2), pp. 125–130, 2000.
35. G. Seedahmed and L. Martucci, "Automated image registration using geometrically invariant parameter space clustering (GIPSC)," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* **34**(3A), pp. 318–323, 2002.
36. V. Walter and D. Fritsch, "Matching spatial data sets: A statistical approach," *International Journal of Geographic Information Sciences* **13**(5), pp. 445–473, 1999.