A Legend-Driven Geographic Symbol Recognition System *

Hanan Samet
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Science
University of Maryland at College Park
College Park, Maryland 20742
E-mail: hjs@umiacs.umd.edu

Aya Soffer
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Science
University of Maryland at College Park
College Park, Maryland 20742
aya@umiacs.umd.edu

July 12, 1994

Abstract

The problem of converting paper maps into digital formats is a major concern today with the emergence of geographical information systems (GIS) as replacements of the traditional paper map. One of the major problems in this conversion is that a paper map is nothing more than an abstraction. The information found in maps is mainly symbolic rather than an accurate graphical description of the region covered by the map.

A system is presented that utilizes the symbolic knowledge found in the legend of the map to drive geographic symbol recognition. The geographic symbol layer(s) of the map are first scanned. The legend of the map is located and segmented. The geographic symbols are identified, and their semantic meaning is attached to them. This information is used to build an initial training set that is then used to classify geographical symbols in input maps using statistical pattern recognition. User interaction is required at first to assist in building the training set to account for variability in the symbols. The training set is built dynamically by entering only instances that add information to to it. The training set library is stored in an appropriate spatial data structure, and a highly efficient nearest neighbor finding algorithm is used to search it. The system then proceeds to identify the geographical symbols in the input maps automatically. The system can be fine-tuned by the user to suit specific needs. An experimental study was conducted on a large amount of data. Results of the experiment are presented. The system can achieve recognition rates of over 95%.

1 Introduction

The paper map has long been the traditional representation of spatial data. Today, we are seeing the emergence of geographic information systems (GIS) as a replacement. One of the central issues in GIS is data acquisition. In particular, we must find ways of converting the data that is stored on maps to a representation that is compatible with the GIS. The most common means of performing this conversion is by use of a digitizing tablet. This process is very time consuming, expensive and inaccurate. Recently optical scanners have been put to use for this purpose. The maps are first scanned, the raster data is then converted into vector format with very heavy user intervention in order to assure the quality of this conversion [14].

There has been some research in recent years on automating this process. Most researchers have focussed on raster-to-vector conversion [10]. Unfortunately, this does not yield accurate and useful results. One of the reasons for the poor performance is that the conversion must be accomplished by some knowledge, which we term *map recognition*. One problem in map recognition is that a paper map is nothing more than an abstraction. The information found in maps is mainly symbolic rather than an accurate graphical description of the region covered by the map. For example, the width of a line representing a road has little to do with its true width. Instead, most often it is determined by the nature of the road (i.e., highway, freeway, rural road, etc.). The color and size of city names on the map convey information about the population of a city. Many graphical symbols are used to indicate the location of various sites such as hospitals, post offices, recreation areas, scenic areas etc. The key to this symbolic information may be found on the map itself in the form of the legend.

In this paper we describe a system built by us that uses the legend of the map to drive the geographic symbol recognition. The legend of the map is first located and segmented. The geographic symbols are identified, and their semantic meaning is attached to them. This information is used to build an initial training set that is then used to classify geographical symbols in input maps using statistical pattern recognition [5]. User interaction is required at first to assist in building the training set to account for variability in the symbols. The system then proceeds to identify the geographical symbols in the input maps automatically.

Another problem in map recognition is the high level of obstruction of geographical symbols due to the map making process. A map is composed of several layers. The symbols in each layer do not occlude each other. But, once these layers are composed, the objects from the different layers may intersect and occlude each other making the problem of segmentation and object recognition very complex. To alleviate this problem, the input to our system are raster images of the separate map layers. Separate map layers may not always be as readily available as integrated layer maps, yet the extra work required to get this data is well worth it. The results of the map recognition are guaranteed to be an order of magnitude better than those that would result from using the composite map. Thus much less human time will be required to verify and correct the results of the automatic process.

Most of the prior research in map recognition has concentrated on skeletonization and vectorization methods [1, 16]. Some research has been done on separating the layers of scanned maps [15, 17]. The maps included road maps [7] and cadastral maps [3, 8]. In the latter, the focus was on locating polygons representing parcels of land, buildings, and roads. As mentioned above, the emphasis of our approach is on utilizing the legend of the map to build a system to extract the symbolic information in the map layers, rather than trying to vectorize the entire map. The focus here is on tourist symbols such as campsites, hotels, recreation areas, etc., although we have used our methods to handle other types of graphical documents such as floor plans [12].

The rest of this paper is organized as follows. Section 2 outlines the main components of our system. Section 3 describes the legend-driven training of our system. Section 4 discusses the geographic object recognition process. Section 5 presents our evaluation method along with experimental results. Section 6 contains concluding remarks.

2 System Overview

The input to the system is a raster image of the symbols layer. This map image is divided into *tiles* of size 512×512 pixels (see Figure 1). These tiles are processed one-by-one.

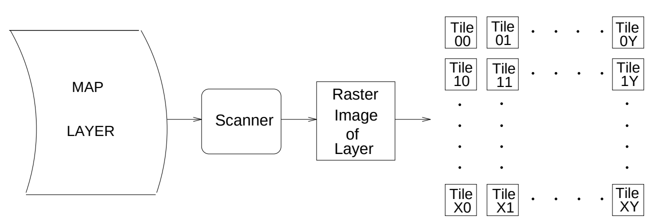


Figure 1: Map layer acquisition and splitting process.

2.1 Legend Acquisition

The tile(s) containing the legend is identified. This tile is passed to the legend acquisition module (see Figure 2). The legend is segmented. A feature vector

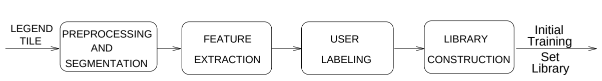


Figure 2: Legend acquisition module.

is computed for each connected component. The geographic symbols are isolated and their semantic meaning is attached to the feature vectors corresponding to them. An initial training set is constructed containing one instance of each symbol in the map. This is done manually. Automating this process is a subject of future research.

2.2 Symbol Recognition

Each non-legend tile is input into the symbol recognition system. The system may work in two modes. In the *user verification mode*, the user verifies the classifications before they are input to the GIS. The training set is modified to correct the erroneous classifications. In the *automatic mode*, the classifications are generated by the system and input directly to the GIS. The user determines the mode in which the system operates. In general, the first tiles will be interpreted in user verification mode. Once the user is satisfied with the recognition rate achieved, the system is placed in automatic mode to process the remaining tiles. There are five basic modules in the system (see Figure 3 for a block diagram).

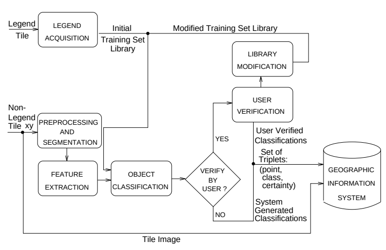


Figure 3: Block diagram of the legend-driven geographical symbol recognition system.

2.2.1 Preprocessing and Segmentation Module

In the preprocessing and segmentation module, various image processing techniques are applied to enhance the image. The image is then segmented into its constituent elements using a connected component labeling algorithm (e.g., [11]). The output of this module is a labeled image in which each pixel has a region number as its value. Regions that are smaller than a certain threshold are labeled 0 (undefined).

2.2.2 Feature Extraction Module

The input to this module is the labeled image output by the preprocessing and segmentation module. For each region in the labeled image, a set of *features* is computed. The system uses some global and some local shape descriptors [9] that have been identified as features that best discriminate between geographic symbols. These features are invariant to scale, orientation, and translation. The results of the feature computation are composed into a *feature vector*. The center of gravity (i.e., centroid) of each region is also computed, and the x and y coordinates of this location are termed a *location vector*. The output of this stage is the feature vector which is a point in the n-dimensional *feature space*, and the location vector which is a point in the 2-dimensional *location space*.

2.3 Object Classification Module

The input to this module are the feature and location vectors output by the feature extraction module, the current training set library, and some parameters set by the user. The training set library, constructed by the legend acquisition module, initially consists of one feature vector for each geographical symbol along with its semantic meaning (i.e., its *classification*). This is the class that the system should assign to each instance of the same symbol. Depending on the quality of the raster image, this may suffice. However, in most cases, the instances of each symbol vary and several instances of each symbol are required in order to build a representative training set library that will produce reasonable recognition rates. The current training set library is used to assign candidate classifications to each input feature vector. A value approximating the certainty of the correctness of these classifications is attached to each classified object (see Section 4 for more details). The output of this module consists of the classifications that were made along with the corresponding location of the symbols on the map and the certainty of the classifications. In terms of a GIS, the output is point data where the classifications and certainty values are alphanumerical attributes of the point.

2.3.1 User Verification Module

This module is only active when the system in working in user verification mode. The input to this module are the feature vectors and the classifications that were output by the object classification module. An image containing the symbols that were identified is composed. This image is displayed next to the original image. The user indicates which symbols have been classified incorrectly, and informs the system of the correct classification. This module outputs the location, classification (as approved or corrected by the user), and a certainty of 1 for all the symbols found in the map tile. In addition, it outputs the feature vectors corresponding to the erroneous classifications along with their correct classifications.

2.3.2 Library Modification Module

This module is only active when the system is working in user verification mode. The input to this module is a list of feature vectors along with their respective classifications. These vectors will be used to classify subsequent input symbols, and thus they comprise the training set for the classification process. The library is stored as an adaptive k-d tree [6]. See section 3 for more details about the storage and retrieval methods employed for managing the library. The output of this module is the current training set library, which is used by the object classification module.

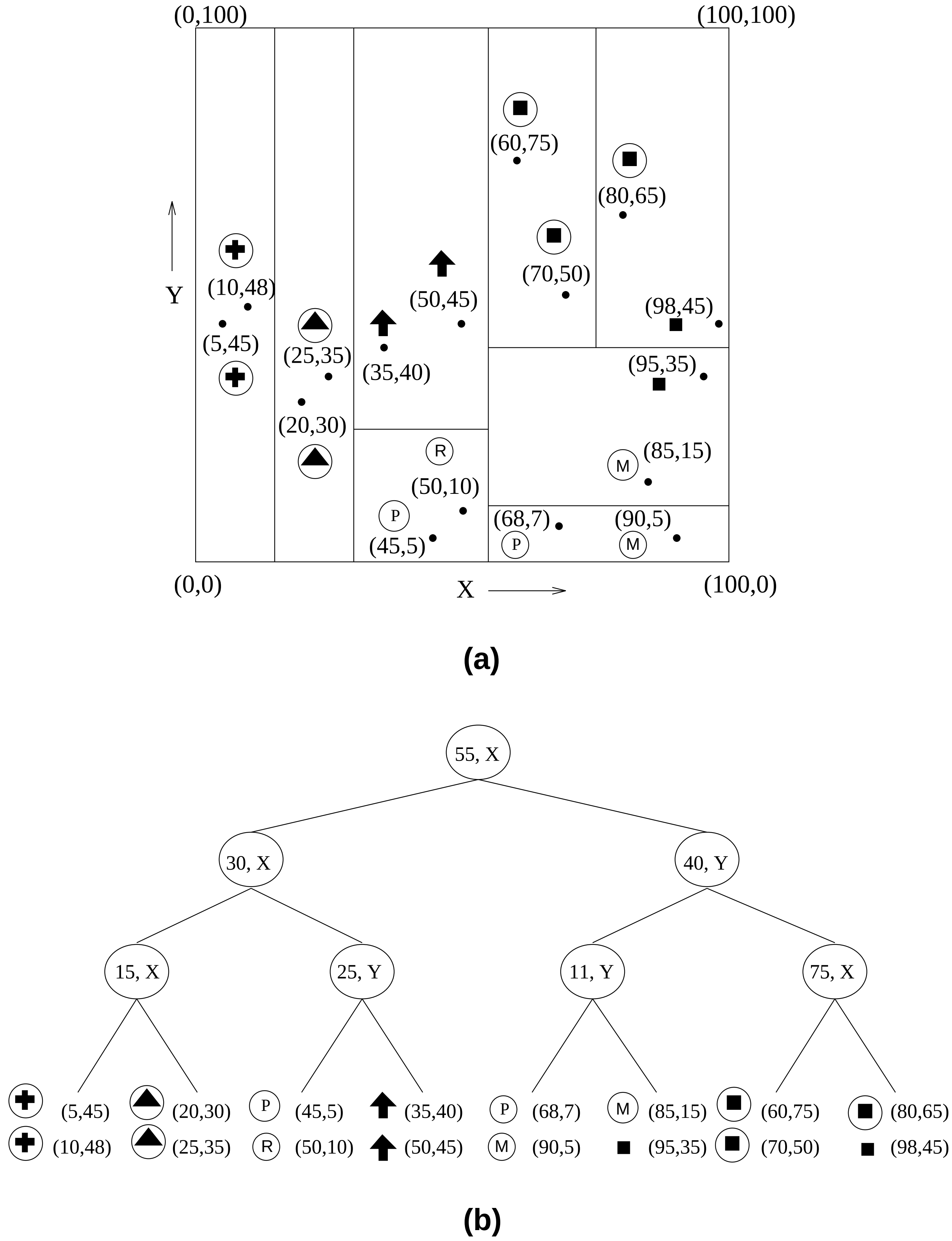


Figure 4: Adaptive k-d tree: (a) set of symbol instances in 2-space. (b) corresponding 2-d tree.The 2-space corresponds to 2 features of the symbol; not the location of the symbol.

2.4 Geographic Information System

The input to the GIS is a set of points, corresponding to the location of the symbols found in the maps. For each point, its several possible classifications and a certainty value approximating the correctness of each classification, are given. This input comes either directly from the object classification module or from the user verification module depending on the mode in which the system is working.

3 Managing the Training Set Library

Realizing that a feature vector is a point in n-dimensional space, we use methods borrowed from computational geometry and spatial data structures to spatially sort the training set. The data structure that we use is the adaptive k-d tree of Friedman, Bentley, and Finkel [6]. This is a variant of a binary search tree with two pieces of information stored at each node: a dimension number d indicating the discriminating dimension, and a discriminating value v which is usually the median value of dimension d of the set of points stored below this node. The discriminating dimension is chosen to be the dimension for which the spread of values of that dimension is a maximum. The spread is measured as the distance from the minimum to the maximum value normalized with respect to the median. All points with values less than or equal to the discriminating value are inserted in the left subtree, and all points with values greater than the discriminating value are inserted in the right subtree. This process is continued recursively until only a few points are left in the set, at which time the decomposition ceases and the result is termed a *bucket*. The components of the bucket are represented by a linked list. Figure 4 shows an example of some instances of geographic symbols in a 2 dimensional feature space (i.e., a feature vector with two components) with the corresponding adaptive 2-d tree.

In order to build an adaptive k-d tree, all points must be known a priori.The training set in our system is built in stages while the system is working in user verification mode, as described in Section 2. Initially, the training set only contains one instance of each symbol, as output by the legend acquisition module. Later, for each tile that is interpreted in user verification mode, only those instances of symbols that were not classified correctly by the current training set are added to it. At this stage, the adaptive k-d tree is reconstructed in order to ensure that it remains balanced. Since the system is working in user verification mode it is not very fast at this point. Thus it is reasonable to spend a little more time to rebuild the adaptive k-d tree, and ensure that future classifications will be made as efficiently as possible.

This method of dynamically constructing the training set guarantees that the training set will remain small without compromising the results of classification using this training set. The reason for this is that there is no redundant information in the training set. Thus it is very efficient and yields results similar to those obtained using a condensing technique to minimize the size of the training set [5].

4 Classifying Geographic Symbols

Geographic symbols identified in the map tiles are classified using a slight modification on the *k-nearest neighbor* rule that is widely used in non-parametric statistical pattern recognition. A classifier using the k-nearest neighbor rule first finds the k vectors in the training set that are nearest to the input vector. Each of these neighbors gives one vote for its class. The class with the most votes is chosen as the class of the input vector. Our modification is to use a *weighted k-nearest neighbor* rule. The k (chosen empirically) vectors in the training set nearest to the input vector are found, as before. To determine the class of the input vector, each one of the neighbors is given a vote whose strength is inversely proportional to its distance from the input vector, rather than giving each neighbor one vote. The vote is given by

$$Vote_L = \frac{1}{dist(F^L, F^I)^2}$$

where F^L and F^I are the feature vectors of the library instance and the input symbol, respectively. The votes for each class C are summed.

$$Votes_C = \sum_{L \ni class(L)=C} Vote_L$$

and the input vector is assigned the class C for which $Votes_C$ is maximum. The distance between two vectors is defined to be the weighted Euclidean distance given by

$$dist(F^L, F^I) = \sqrt{\sum_{i=1}^{N} w_i (F_i^L - F_i^I)^2}$$

where F_i^L is the i^{th} feature of the library vector, F_i^I is the i^{th} feature of the input vector, and w_i is the i^{th} weighting factor. The weighing factor is computed so that features with a smaller variance have a larger

certainty value for this classification. The system also computes certainty values for the classes represented by the remaining k-nearest neighbors. This value approximates the certainty that the input vector belongs to the particular class to which it is assigned. For each of the candidate classes, C_i, the certainty value is a function (not given here) of the average distance of C_i from the input vector, and the average distance of C_i from the library feature vector that is closest to the input vector. The classification module outputs all of the candidate classifications along with their certainty. The class assigned using the weighted k-nearest neighbor rule will have the highest certainty value.

5 Experimental Study

The system was tested on the red sign layer of the GT3 map of Finland. The scale of the map is 1:200000. The layer was scanned at 240dpi. See Figures 5 and 6 for a sample tile of the red sign layer of the map compared to the same tile when all layers are composed.

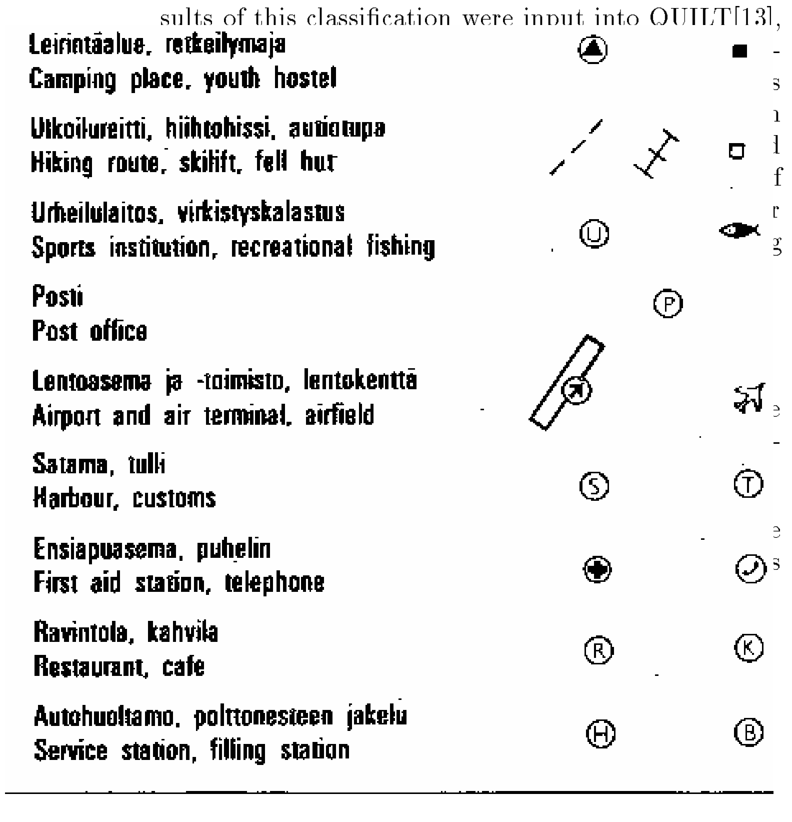
Figure 5: Example map tile - all layers

A portion of the map containing part of the legend relevant to this layer can be found in Figure 7. Notice that there are many symbols in the map tile that are not found in the legend. These are mainly numbers, names, and markers that are related to other layers. These symbols, termed *invalid symbols*, should all be classified as invalid by the system. The layer was split into 425 tiles of size 512×512. These tiles were examined in a random order to give the system a chance

Figure 6: Example map tile - red sign layer

Figure 7: Legend portion containing tourist symbols

to see a large variety of symbols in the user verification phase as some symbols tend to appear clustered in the map. The legend was identified manually, and the classifications were attached to the feature vectors representing the geographic symbols that the system should identify. There were 25 such symbols. The system processed the first 50 tiles in user verification mode. At that stage the training set contained 80 instances of symbols and the current recognition rate was determined sufficient. The remaining tiles were processed automatically. See Section 5.3 for the results of this fully automatic recognition. The results of this classification were input into QUILT[13],



- *deletion errors* — an object that should have been classified as a valid symbol was classified as invalid.

- *insertion errors* — an invalid symbol was classified as a valid symbol, or a valid symbol was classified as another valid symbol in addition to being classified correctly. The latter situation arises when all of the candidate classifications among the k-nearest neighbors are considered.

Substitution errors and deletion errors may cause the GIS to overlook tiles that should be retrieved for a given query. Insertion errors and substitution errors may cause the GIS to retrieve superfluous map tiles for a given query. In the context of a GIS or an image database, the insertion errors are the least severe. The purpose of the map recognition is to enable the system to retrieve just those map portions that are relevant to a given query. Retrieving too many tiles is not as severe as missing tiles. The user can always weed out those tiles that do not actually conform to the query.

5.2 Experiment Description

50 sample tiles were chosen from the tiles that were not used for training the system. These tiles were used as input to the system. For each symbol in each tile, the system output the classes of the k nearest neighbors to the symbol's feature vector along with a certainty value as described in Section 4. The class

assigned to the symbol using the *weighted k-nearest neighbor* rule (i.e., the one with the highest certainty value) was compared manually to the correct classification, as found in the raw image. The number of errors of each type for each tile was recorded. In addition, the classes of the remaining neighbors which were output by the system with a lower certainty value were compared to the correct classification. The number of errors of each type for this case were recorded. The certainty value assigned by the system for each correct classification was noted. This was done in order to determine whether inserting into the GIS more than one candidate classification per symbol with an appropriate certainty value yields better results than just inserting the class selected using the *weighted k-nearest neighbor* rule (i.e., the one with the highest certainty value).

This experiment was repeated five times varying the value of the maximum distance in the normalized feature space allowed between an input symbol and the nearest symbol in the training set (termed the *window size*). Any symbol whose nearest neighbor is not within the window defined by the window size is classified as invalid. The values selected for the window size were $0.01, 0.02, 0.04, 0.08, 0.16$. As the window size increases, more neighbors representing more classes will be found in the window. Thus we expect to have fewer substitution and deletion errors, at the cost of more insertion errors. See Section 5.3 for the results of our experiments.

5.3 Results

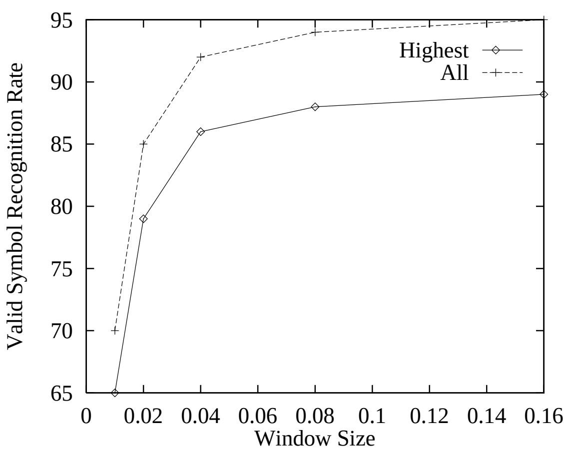


Figure 8: Valid symbol recognition rate for various window sizes.

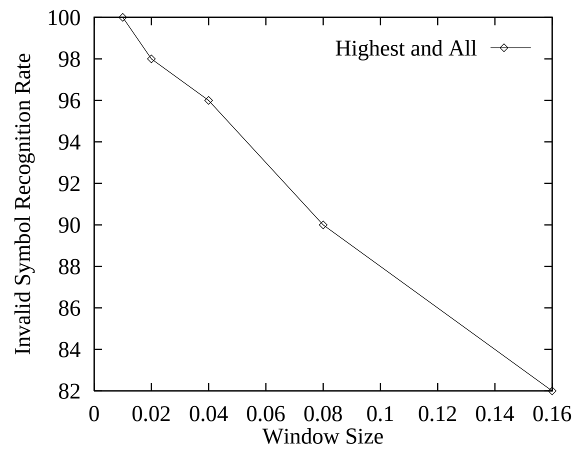


Figure 9: Invalid symbol recognition rate for various window sizes.

Figures 8 and 9 show the recognition rate for valid and invalid symbols respectively. Valid symbols are those input symbols that were identified as valid geographic symbols (i.e., the ones acquired in the legend acquisition phase). For the purpose of this analysis, an input symbol is categorized as valid or invalid according to the ground truth data, i.e. by inspection of the raw images that were input to the system. The number of valid and invalid input symbols in the 50 test tiles was recorded. All percentages reported here are of these numbers. The valid symbol recognition rate indicates what percent of the valid input symbols were classified by the system as the correct geographic symbol. Invalid symbols are input symbols occurring in the map that did not appear in the legend and will not be recognized (i.e, classified as invalid). They include numbers, letters, etc. The invalid symbol recognition rate indicates what percent of the invalid input symbols were in fact classified by the system as invalid. The "Voted" plot shows the recognition rate when considering only the class assigned to the symbol by the voting process among the k-nearest neighbors (referred to as the *voted classification* and corresponds to the one with the highest certainty value). The "All" plot shows the rate when considering the classes of all of the k-nearest neighbors regardless of their certainty (referred to as the *probabilistic classification*).

As expected, as the window size increases, the valid symbol recognition rate increases and the invalid symbol recognition rate decreases. The reason for this is that there are more candidate classes when the window is larger. Thus the chance that a symbol from the correct class lies in the window increases as does the

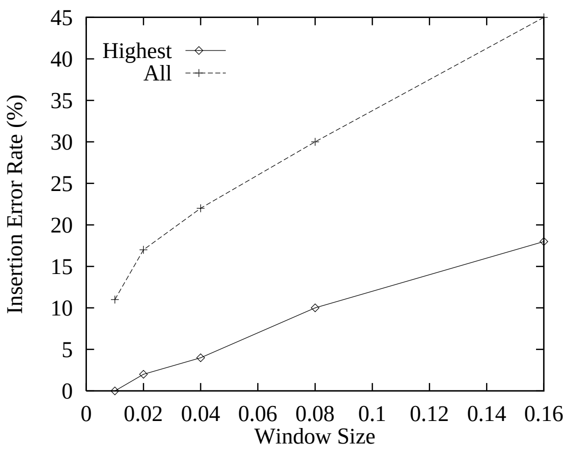


Figure 10: Percent of invalid input symbols classified as valid symbols or valid symbols classified as other valid symbols in addition to being classified correctly (i.e., an insertion error occurred) for various window sizes.

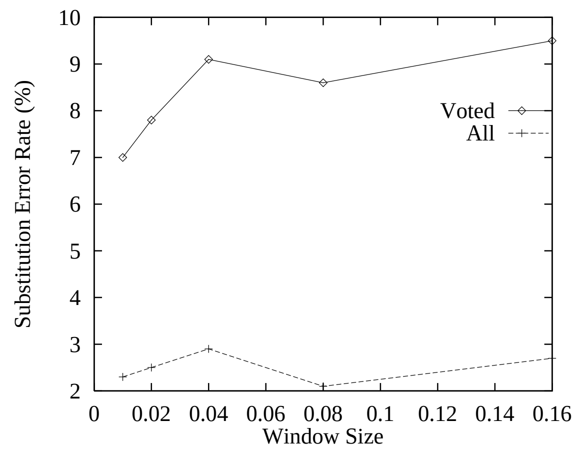


Figure 11: Percent of valid input symbols classified as the wrong valid symbol, (i.e., substitution error occurred) for various window sizes.

chance that some valid symbol from the library will lie in the window of an invalid symbol. Recall that a symbol is classified as invalid if it is has no neighbors in the window. From Figure 8 we also see that considering the classes of all the neighbors increases the valid symbol recognition rate. The requirement in this case is only that the correct classification be in the window with any certainty value. Thus more valid symbols will be classified correctly. However, this will increase the insertion error rate since more than one classification may be recorded in the GIS per symbol. This can be seen in Figure 10 which shows the percent of invalid input symbols classified as valid symbols or valid symbols classified as other valid symbols besides being classified correctly (i.e., an insertion error occurred). In particular, notice the higher values of the "All" plot.

Figures 11 and Figure 12 analyze the cause of the erroneous valid symbol classifications. Figure 11 shows what percentage of those input symbols that were determined to be valid symbols by the ground truth data were classified by the system as valid symbols but not as the correct symbol (i.e., a substitution error occurred). Figure 12 shows what percentage those input symbols that were determined to be valid symbols by the ground truth data were classified by the system as invalid symbols (i.e., a deletion error occurred). Recall that a deletion error occurs if no neighbors are found within the specified window. From these two figures we see that the substitution error rate is only slightly effected by the window size, whereas the deletion error rate is highly effected by the window size and shows a sharp drop at a window size of 0.04. From this we may conclude that the increase in the valid symbol recognition rate with an increase in window size is mainly attributed to a sharp reduction in the number of deletion errors rather than to a significant change in the number of substitution errors. The only way to decrease the substitution error rate is to consider all classes found among the k-nearest neighbors in the given window as indicated in Figure 11 by the lower values of the "All" plot compared to the "Voted" plot. These above conclusions are also apparent in Figures 13 and 14 which show the rate of the various error types as a function of the window size. The plot representing the substitution errors is rather flat, whereas the plots representing insertion and deletion errors change significantly with the window size.

The window sizes of 0.04 and 0.08 appear to be best for this data set. A valid symbol recognition rate of over 92% was achieved, while about 20% of the invalid symbols were classified as some valid symbol (i.e., an insertion error) with this window size. The ideal window size should however be selected according to the requirements of the application. If it is critical for the GIS not to miss any tiles when responding to a query, then a larger window size should be selected. This will result in incorporating more insertion errors into the GIS, while ensuring that a minimum of deletion errors are incorporated into the GIS. This means that the GIS overlooks fewer tiles at the cost of retrieving superfluous tiles that will need to be weeded out

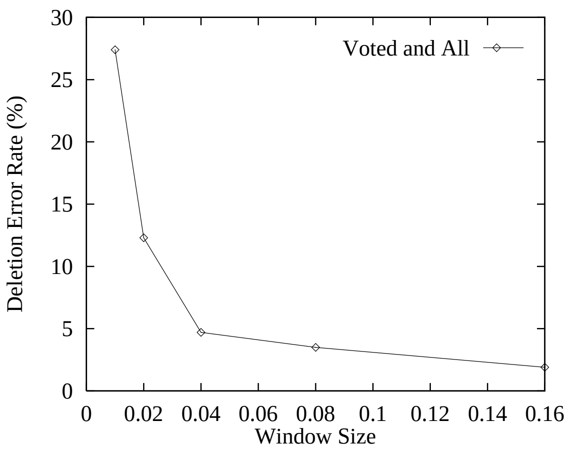


Figure 12: Percent of valid input symbols that were classified as invalid symbols (i.e., deletion error occurred) for various window sizes.

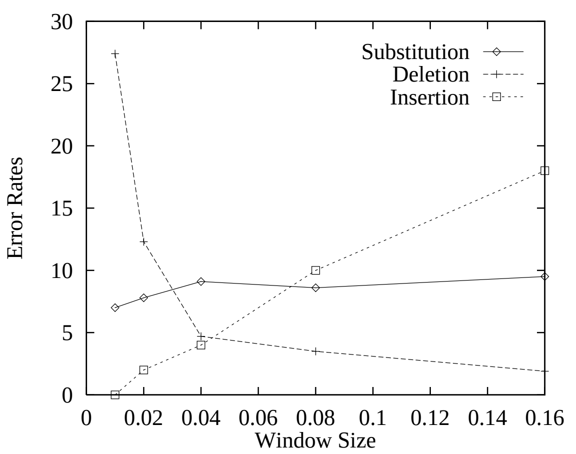


Figure 13: Error rates using the voted classification for various window sizes.

manually. If accuracy is not as important as the time required to weed out the tiles manually, then a smaller window size should be selected. As our results indicate, it is possible to achieve valid recognition rates of over 95% with a large enough window, and invalid symbol recognition rates of 100% with a small enough window.

Figure 15 shows the certainty values assigned to those input symbols that were determined to be valid symbols by the ground truth data and were classified by the system as the correct valid symbol (i.e., no error occurred). Similarly, Figure 16 shows the certainty values assigned to those input symbols that were determined to be valid symbols by the ground truth data and were classified by the system as the

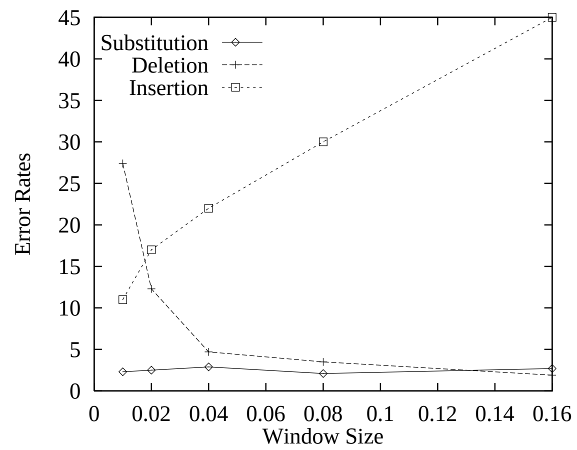


Figure 14: Error rates using the probabilistic classification for various window sizes.

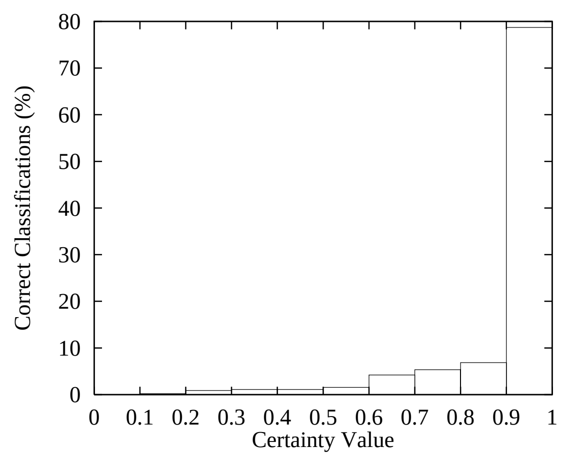


Figure 15: Correct valid symbol classifications per certainty value range.

wrong valid symbols (i.e., an erroneous classification due to a substitution or insertion error). The vast majority of correct classifications were assigned a certainty value above 0.9, whereas the certainty values of the erroneous classifications are more spread out. The user may set the minimum certainty value required for considering a classification. By selecting a minimum certainty value of about 0.7 the user may eliminate many of the substitution and insertion errors. Thus weeding out automatically most of the superfluous tiles while overlooking only a small number of the required tiles.

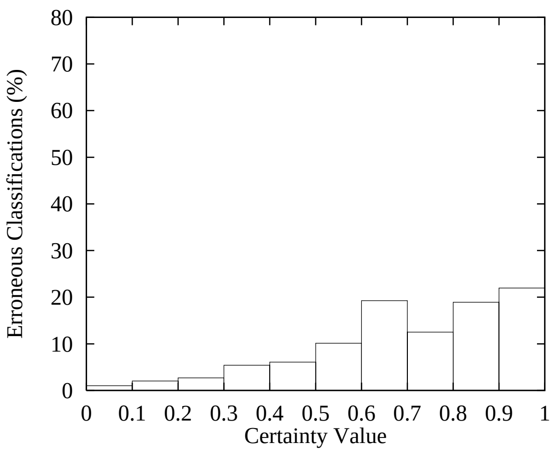


Figure 16: Erroneous classifications per certainty value range.

6 Concluding Remarks

A legend-driven geographic symbol recognition system has been described. The system utilizes the fact that most of the data found in maps is symbolic, and that the key to understanding the symbols can be found in the legend of the map. The system is efficient and flexible. The training set is built dynamically by entering only instances that add information to to it thereby creating a small but effective training set. The training set library is stored in an appropriate spatial data structure, and a highly efficient nearest neighbor finding algorithm is used to search it, thus enabling quick classification. Users may fine-tune the performance of the system to their requirements by setting the window size and minimum certainty values to fit their particular application.

An experimental study was conducted on a large amount of data. The experimental results showed that the system can achieve recognition rates of 95% with little user intervention. However, with more user intervention the system may reach 100% recognition rates. At present, the legend acquisition process is mostly manual. Automating it is subject of future research. The system can be easily adapted to interpret other graphical documents and we have used similar methods for the interpretation of floor plans [12]. This system is designed for map layers containing geographic symbols. In order to provide full map recognition, other methods need to be developed to interpret layers containing additional types of symbolic information such as roads, bodies of water, etc. Once this is done, the results can be integrated into a GIS to provide a comprehensive tool to utilize the vast amount of data that is found in paper maps.

7 Acknowledgements

We are grateful to Karttakeskus, Map Center, Helsinki, Finland for providing us the map data.

References

[1] S. V. Ablameyko, B. S. Beregov, and A. N. Kryuchkov. Computer-aided cartographical system for map digitizing. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 115–118, Tsukuba Science City, Japan, October 1993.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, Arlington, VA., January 1994.

[3] L. Boatto, V. Consorti, M. Del Buono, S. DiZenzo, V. Eramo, A. Esposito, F. Melcarne, M. Meucci, A. Morelli, M. Mosciatti, S. Searci, and M. Tucci. An interpretation system for land register maps. *Computer*, 25(7):25–33, July 1992.

[4] G.L Cash and M. Hatamian. Optical character recognition by the method of moments. *Computer Vision, Graphics, and Image Processing*, 39(3):291–310, September 1987.

[5] P. Devijver and J. Kittler. *Statistical Pattern Recognition*. Prentice Hall, New-Jersey, 1982.

[6] J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[7] M. Ilg. Knowledge-based understanding of road maps and other line images. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 282–289, Atlantic City, June 1990.

[8] R. D. T. Janssen, R. P. W. Din, and A. M. Vossepoel. Evaluation method for an automatic map interpretation system for cadastral maps. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 125–128, Tsukuba Science City, Japan, October 1993.

[9] M.D. Levine. *Vision in Man and Machine.* McGraw-Hill, New York, 1982.

[10] D. J. Peuquet. An examination of techniques for reformatting cartographic data part 1: The raster-to-vector process. *Cartographica*, 18(1):34–48, January 1981.

[11] A. Rosenfeld and A.C. Kak. *Digital Picture Processing.* Academic Press, New York, second edition, 1982.

[12] H. Samet and A. Soffer. Automatic interpretation of floor plans using spatial indexing. In *Proceedings of the 7th Internatianl Conference on Image Analysis and Processing*, page ?, Bari, Rome, September 1993. World Scientific.

[13] C.A. Shaffer, H. Samet, and R.C. Nelson. QUILT: a geographic information system based on quadtrees. *International Journal of Geographical Information Systems*, 4(2):103–131, April 1990.

[14] J. Star and J. Estes. *Geographic Information Systems*, chapter 6, pages 85–91. Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[15] S. Suzuki and T. Yamada. MARIS: Map recognition input system. *Pattern Recognition*, 23(8):919–933, August 1990.

[16] N. Tanaka, T. Kamimura, and J. Tsukumo. Development of a map vectorization method involving a shape reforming process. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 680–683, Tsukuba Science City, Japan, October 1993.

[17] T. Xu and X. Lin. A new algorithm separating string from map images. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 910–913, Tsukuba Science City, Japan, October 1993.