Graph-Based Feature Recognition of Line-Like Topographic Map Symbols

Rudolf Szendrei, István Elek, and Mátyás Márton

ELTE University, Faculty of Informatics, Budapest
swap@inf.elte.hu, {elek,matyi}@map.elte.hu
http://team.elte.hu/

Abstract. Paper-based raster maps are primarily for human consumption. Todays computer services in geoinformatics usually require vectorized topographic maps, while the usual method of the conversion has been an error-prone, manual process.

The system in development separates the recognition of point-like, line-like and surface-like objects, and the most successful approach appears to be the recognition of these objects in a reversed order with respect to their printing. During the recongition of surfaces, homogeneous and textured surfaces must be distinguished. The most diverse and complicated group constitute the line-like objects.

In this article, a possible method of the conversion is discussed for line-like topographic map objects. The results described here are partially implemented in the IRIS project, but further work remains. This emphasizes the tools of digital image processing and knowledge-based approach.

Keywords: Geoinformatics, topographic maps, raster-vector conversion.

1 Introduction

Paper-based raster maps are primarily appropriate for human usage. They always require a certain level of intelligent interpretation. In GIS applications vectorized maps are preferred. Especially, government, local authorities and service providers tend to use topographic maps in vectorized form. It is a serious challenge in every country to vectorize maps that are available in raster format. This task has been accomplished in most countries — often with the use of uncomfortable, "manual" tools, taking several years. However, it is worth dealing with the topic of raster-vector conversion. On one hand, some results of vectorization need improvement or modification. On the other hand, new maps are created that need vectorization.

The theoretical background of an intelligent raster-vector conversion system has been studied in the IRIS project [2]. Several components of a prototype system has been elaborated. It became clear very early that the computer support of conversion steps can be achieved at quite different levels. For example, a map symbol can be identified by a human interpreter, but the recognition can be

Y. Tan et al. (Eds.): ICSI 2011, Part II, LNCS 6729, pp. 291–298, 2011.

attempted with a software, using the tools of image processing [3]. A computer system can be fairly valuable and usable even if every important decision of interpretation is made by the expert user. However, the system designed and developed by the authors is aimed at to automatize the raster-vector conversion of line-like symbols as much as possible. This aim gives an emphasis to a knowledge-based approach.

This paper deals with a part of raster-vector conversion applied in cartography, with knowledge-based approach [1]. The line-like map symbols used in topographical maps will be introduced, together with the algorithms used to recognize them. The organization of expertise into knowledge base will also be presented.

The following must be considered in connection with good quality and automated vectorization. Raster maps can be adequately understood only by human expert. After the vectorization, the relationships used for interpretation are no more contained in the vectorized map — it consists only of numerical and descriptive data. Automatic interpretation of image contents requires sophisticated image processing tools, whiches are not comparable to human perception in the majority of cases. Therefore, the level of automatic recognition must also be appropriately determined.

2 Line-Like Map Symbols

The topic of this article is how to interpret printed variant of line-like map symbols and how to represent them in computer systems. This process is considered basically as the result of interpretation and processing of map symbols. To accomplish this task it is very important to understand maps, and specifically, map symbols. To gain a comprehensive survey, refer to [5]. Although human cognition can not be completely understood, it is necessary to know to a certain extent how the human expert interprets graphical information. Regarding human perception, primarily points, lines (see Fig. 1) and textured surfaces are sought and distinguished. It must be realized that human perception may reveal finer or hidden information, for example how roads crossing at different levels hide each other. Human mind is also capable of abstraction, for example when it disregards the actual texture of surface, and investigates only its shape. Human eye can make some corrections, for example in the determination of shades of color layers printed over each other.

Map interpretation process and the complexity of knowledge based object recognition can be visualized via any example of the four different object types — that is, point, line, surface and inscription.

Line-like elements can cover larger area on map than their real size. For instance in the case of a highway, a zero-width center line can represent the theoretical position of the road in the database. Beyond the graphical properties of lines the database may contain real physical parameters, such as road width, carrying capacity, coating (concrete, asphalt) etc. Hiding is a very inherent phenomenon in maps when line-like objects, landmarks (typically roads, railways

and wires) located at different elevations intersect. This results in discontinuity of objects in map visualization. However, in map interpretation continuity must be assumed.

3 Recognition of Line-Like Symbols

Line-like symbols are usually the trace of a road like object, or edge of a polygon with a given texture/attribute.

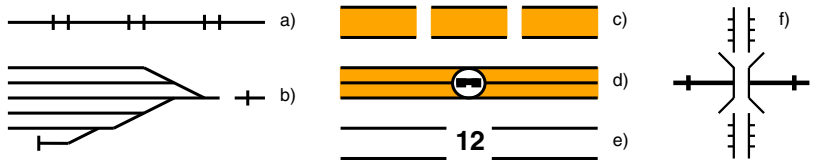

Fig. 1. Examples for line-like symbols: a) railway b) railway network at a railway station, c) highway under construction d) highway with emergency phone. e) road with a specified width f) bridge above a canal

Recognition of line-like symbols is one of the most difficult tasks of raster-vector conversion. These symbols are often complex, and it is permitted for two symbols to differ only in their size, to cross each other or to join to form a single object (see Fig. 1a, b, respectively). Difficulties are posed by parallel lines belonging to the same object (see Fig. 1d, e) versus lines running in parallel which belong to separate objects. Further difficulties are the discontinuous symbols (see Fig. 1c, e, f).

It is beyond the aim of the current article to solve all the difficulties mentioned above, so for the purpose of this paper we assume that line-like symbols

1. do not cross each other,
2. do not join to form a single object, and
3. are continuous.

A classic way of line-like symbol vectorization is introduced in [4], where cadastral maps in binary raster image format are vectorized. The additional features of color topographic maps, like road width, capacity, coating etc. can not be recognized the classical way [6]. Each of these features are represented by a corresponding graphics, color and structure. The following method is able to recognize the trace of the line-like symbols of a topographic map.

1. Do image segmentation and classify each pixel.
2. Create a binary map, where each black pixel belong to eg. road.
3. Apply thinning and morphological thinning on the binary map.
4. Vectorize the one pixel thin skeletons.

The first step is the segmentation, which works as follows. Define an object color set O and a surface color set S. The amount of colors in each color set is approx.

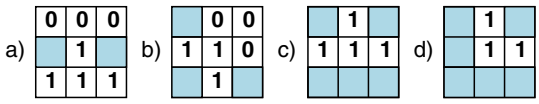

Fig. 2. a), b) structuring elements of morphological thinning based skeletonization, c), d) structuring elements of morphological fork detection on binary images. Values of the elements can be: 0 - *background*, 1 - *foreground* or *undefined*.

5-7 in the case of topographic maps. We assume that on a printed map each pixel color can be defined as a linear combination of a surface and an object color. In optimal case, this can be written as a $c = \alpha * c_o + (1 - \alpha) * c_s$ equation, where c is the value of the current pixel, and c_o, c_s are the respective object and surface colors, so the segmentation can be done by solving the minimalization task $\min_{o \in O, s \in S} |c - \alpha * c_o + (1 - \alpha) * c_s|$ for each pixel.

As the second step is a simple selection on the segmented pixels, it can be done easily. The third step consists of two different thinning methods. A general thinning method is used first to avoid creating unneeded short lines by morphological thinning. The general thinning can be described as it iteratively deletes pixels inside the shape to shrink it without shortening it or breaking it apart.

Because the result of the general thinning algorithm may contain small pixel groups, a morphological thinning should be performed. This morphological thinning can be done by using the structuring elements shown in Fig. 2. At each iteration, the image is first thinned by the left hand structuring element (see Fig. 2 a) and b)), and then by the right hand one, and then with the remaining six 90° rotations of the two elements. The process is repeated in cyclic fashion until none of the thinnings produces any further change. As usual, the origin of the structuring element is at the center.

The skeletonized binary image can be vectorized in the following way. Mark all object pixels *black* and surface pixels *white*. Mark those *black* pixels *red*, where $N(P1) > 2$, and then mark the remaining *black* fork points *blue* by using structuring elements c) and d) of figure 2 in the same way as structuring elements are used in morphological thinning. The *red* fork points are connecting lines, while *blue* fork points are connecting other fork points. Mark *green* each *black* pixel, if at most one neighbour of it is *black* (end point of line segment). It can be seen that a priority is defined over the colors as *white* < *black* < *green* < *red* < *blue*. The following steps vectorize the object pixels

1. Select a *green* point, mark *white* and create a new line segment list, which contains that point.
2. Select a *black* neighbour if it exists and if the current point is also *black*. Otherwise select a higher priority point. Mark *white* the point and add to the end of the list.
3. Go to Step 2, while a corresponding neighbour exists.
4. Go back to the place of the first element of the list and go to Step 2. Be careful that new points should be added now to the front of the list. (This step processes points in the opposite direction.)

5. Go to Step 1, while a *green* point exists.
6. Select a *black* point, mark *white*, and create a new line segment list, which contains that point.
7. Select a *black* neighbour of the current point, mark *white*, and put it at the end of the list.
8. Go to Step 7, while a *black* neighbour exists.
9. Select a *red* point p, mark *white* and create a new line segment list, which contains that point. Let $NeighbourSelect = RedSelect = Counter = 0$, $BlueFirst =$ false, $where =$ back, $q = p$.
10. Let $PrevPoint = q$.
11. If the $NeighbourSelect$th neighbour r of q exists, let $q = r$, let $BlueFirst = (Steps = 0$ and $where$=back), let $n = q$, and increment $NeighbourSelect$ by 1. Put q into the list at $where$ and go to Step 13.
12. If the $RedSelect$th neighbour r of q exists,
    (a) If q and n are neighbours and $where$ =front, then let $q = PrevPoint$ and increment $RedSelect$ by 1. Go to Step 10.
    (b) Put q into the list at $where$, mark q *white*, let $NeighbourSelect = 0$ and increment $Counter$ by 1. Go to Step 10.
13. If $where$=back, then let $where$=front, $q = p$ and go to Step 10.
14. Go to Step 9, while a *red* point exists.

Although, the algorithm above vectorizes all the objects, it merges the several object types and colors. Hence, pixels of a given object color are copied onto a separate binary image before they are vectorized.

We introduce an approach, which is able to recognize the features of line-like objects, so the corresponding attributes can be assigned to them. This assumes that the path of each object exists in the corresponding vector layer. In order to recognize a specific feature, its properties should be defined for identification.

Two properties of vectorised symbols are recognized: forks ($F \sim Fork$), and end-points ($E \sim End$). Both are well known in fingerprint recognition where they are called *minutiae*. In the case of fingerprints, a fork means an end-point in the complement-pattern, so only one of them is used for identification. In our case, we can not define a complement-pattern, so both forks and end-points are used.

Representation of line-like symbols is based on weighted, undirected graphs. An EF-graph is an undirected graph with the following properties:

- Nodes are either of type E or F. The color of a node is determined by the corresponding vector layer.
- Two nodes are connected if the line-segment sequence connecting the nodes in the corresponding vector layer does not contain additional nodes. Edges running between nodes of different colors can be defined by the user (in case of multicolor objects). The weight of the edge is equal to the length of the road connecting the two nodes, and it has the color of the corresponding symbol part.
- There are special nodes, denoted by an index P, which occur on the trace of a line object. These will be used to produce the final vector model.

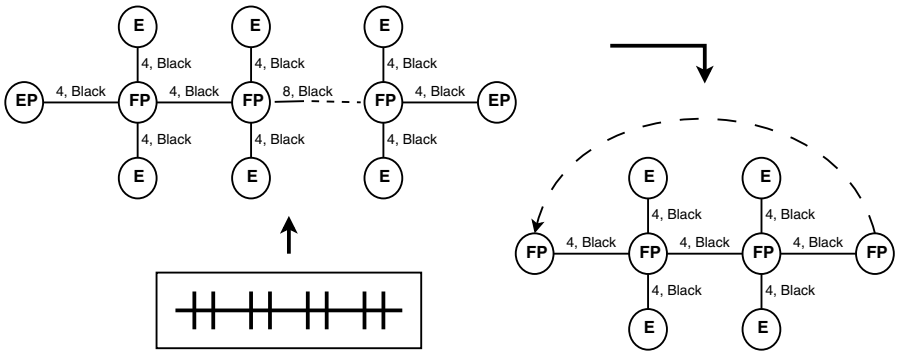

Fig. 3. The EF graph and the elementary EF graph of a double railway line. Distances are relative values and refer to the scale of the map. The dashed line in the elementary EF graph represents its cyclical property.

An EF-graph can also be assigned to the vectorised map, not only to the vectorised symbols, where line-like symbols are not separated to their kernels. For recognition we use the smallest units of the symbol, called the kernel. The smallest unit is defined as the one which can be used to produce the entire symbol by iteration. In the EF-graph there is usually only two nodes participating in the iteration; these are type F with only a single edge, so become the entry and exit points to the graph. In the very few cases, where the entry and exit points of the smallest unit can not be identified, the kernel of the line-like object is itself. Smallest unit can not be defined for the whole vectorised map.

Figure 3 shows how a symbol is built up from its smallest units by iteration. Weights represent proportions and depend on the scale of the map. Beside weights, we can assign another attribute to edges, their color. In the figure almost all edges are coloured black.

The recognition of line-like objects is reduced to an extended subgraph isomorphism problem: we try to identify all the occurrences of the EF graph of the symbol (subgraph) in the EF graph of the entire map. The weights of the EF graphs are normalized with respect to the scale of the map, and the collection is sorted in decreasing order of node degrees. Call this collection of sorted EF graphs S. Since the EF graphs created to maps do not contain the edges those connecting nodes with different colors, this case should be handled. In this article, the potential edges are identified by searching the corresponding neighbour on its own layer in the given distance of the node. The validity of a found potential edge is verified by comparing the color of the edge and the color of the segmented image pixels lying *under* the edge.

Subject to the conditions above it is possible to design an algorithm for the recognition of subgraphs. While processing the map, recognized objects are removed, by recoloring the corresponding subgraph. Two colors, say blue and red, can be used to keep track of the progress of the algorithm and to ensure termination.

The following algorithm stops when there are no more red nodes left in the graph.

1. Choose an arbitrary red node U with the highest degree from the EF graph of the map.
2. Attempt to match the subgraph at node U against S, that is the sorted collection of EF graphs, until the first successful match in the following way:
   (a) Perform a "parallel" breadth-first search on the EF graph of the map and the EF graph of kernel of the current symbol with origin U. This is called successful if both the degree of all nodes match, and weights are the same approximtely.
   (b) In the case of success, all matching nodes become blue, otherwise they remain red.

Upon successful matching the EF-graph of the symbol is deleted from the EF-graph of the map. Entry and exit points must not be deleted unless they are marked as E, and the degree of remaining F nodes must be decreased accordingly. The equality of edge weights can only be approximate, due to the curvature of symbols. The algorithm above can be implemented, as it keeps track the given object by using the line segments as paths in vector data. The other difficulty is that edges with differently colored nodes are not directly defined by the vector layers. In practice, we have created a spatial database, which has contained the vectorized line segments and their color attribute. The potential edges was determined by a query, looking for existence of a neighbour with the right color in a given distance from the corresponding node.

4 Results

In this article a feature extraction method is introduced for line-like symbol vectorization in the IRIS projects. The project aims to automate and support the recognition of raster images of topographic maps, with the combination of digital image processing and a knowledge-based approach.

The interpretation of line-like symbols is the most difficult issue in topographic map vectorization. An EF graph representation is developed, which is used for the recognition of curved, line-like objects with regular patterns.

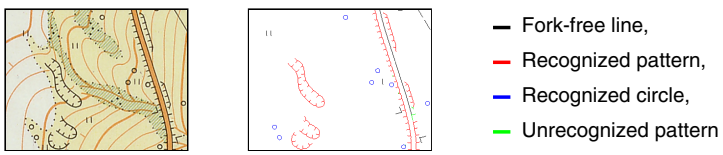

Fig. 4. Recognition results of a Hungarian topographic map line networks at map scale 1:10000 and scanning resolution of 300dpi

The method was tested on a 6km × 4km section (6990 × 4680 pixels) of a large scale 1:10 000 topographic map (see Fig. 4). In our experience some spatial filters, like Kuwahara and conservative smoothing improved the quality of segmentation. During a large number of tests, symbols completely appearing on maps were identified at a high rate (> 90%), while symbols disappearing partially, (like junctions and discontinuities) remained mostly unidentified. In the latter case, the level of identification can be enhanced with some heuristics using the neighbour segments.

Acknowledgement

The work was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TAMOP 4.2.1./B-09/1/KMR-2010-0003).

References

[1] Corner, R.J.: Knowledge Representation in Geographic Information Systems. Ph.D. thesis, Curtin University of Technology (December 1999)
[2] Dezső, B., Elek, I., Máriás, Z.: IRIS, Development of Automatized Raster-Vector Conversion System. Tech. rep., Eötvös Loránd University and IKKK (November 2007) (in Hungarian)
[3] Dezső, B., Elek, I., Máriás, Z.: Image processing methods in raster-vector conversion of topographic maps. In: Karras, A.D., et al. (eds.) Proceedings of the 2009 International Conference on Artificial Intelligence and Pattern Recognition, pp. 83–86 (July 2009)
[4] Janssen, R.D.T., Vossepoel, A.M.: Adaptive vectorization of line drawing images. Computer Vision and Image Understanding 65(1), 38–56 (1997)
[5] Klinghammer, I., Papp-Váry, Á.: Földünk tükre a térkép (Map, mirror of the Earth). Gondolat (1983)
[6] Liang, S., Chen, W.: Extraction of line feature in binary images. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E91A(8), 1890–1897 (2008)