

An Interpretation System for Cadastral Maps

E. Katona and Gy. Hudra
József Attila University, H-6720 Szeged, Árpád tér 2, Hungary
katona@inf.u-szeged.hu, hudra@inf.u-szeged.hu

Abstract

To create a spatial database for some GIS application, it is a big challenge to recognize automatically all the simple and complex map objects on scanned maps. This paper presents a robust map interpretation system developed to process Hungarian land register maps (cadastral maps). Processing starts with a raster-to-vector conversion generating a raw vector image from the scanned map. All recognition steps are performed on this raw vector image: segmentation, recognition of separated and not separated symbols, recognition of more complex objects (buildings and parcels). Finally, drawing quality is enhanced utilizing the recognition results.

Interpretation is supported by a special data structure - called DG - ensuring dynamic description of hierarchical structures of drawing objects. This data structure is an essential part of our concept, therefore it is discussed in the paper.

This research has been supported by the Hungarian research foundation OTKA T020523.

1. Introduction

Geographic Information Systems are mostly based on spatial databases. Such databases often are created on the basis of paper maps, that is, map objects are converted to vector (CAD) format and are stored as spatial database entities. Such a conversion of paper maps is a time-consuming manual work, that is why lots of efforts have been made to interpret maps automatically. The automatic processing is usually based on scanning of paper maps, vectorization and recognition. Different solution strategies have been developed for different kinds of maps [1], [3], [4], [5], [9], [11].

Present paper describes a robust map interpretation system - called MAPINT - to interpret Hungarian land register maps (cadastral maps). These maps typically contain building and parcel outlines with house numbers and parcel numbers. Different kinds of dashed lines and other symbols are used to represent further features (Fig. 1).

Main processing phases of MAPINT are as follows:

- Raster-to-vector conversion (generating a raw vector image from the scanned map).
- Segmentation (raw vectors are classified into three basic categories: symbols, dashed lines, continuous lines).
- Recognition of symbols, using a neural network model.

- Recognition of so-called connection-signs, which have special importance on Hungarian cadastral maps.
- Recognition of buildings and parcels.
- Automatic correction of vectorization anomalies.

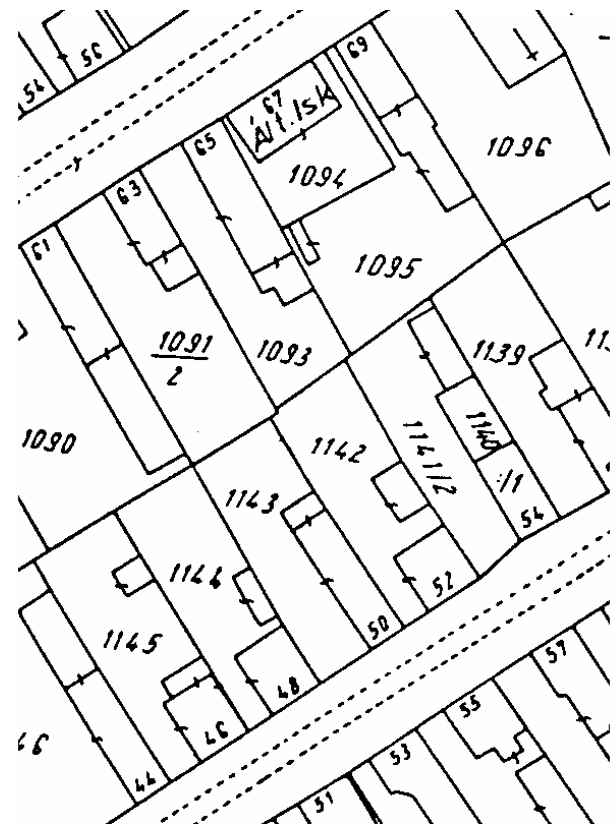


Figure 1. Fragment of a cadastral map.

During recognition a special vector data representation - called DG - is used ensuring dynamic description of arbitrarily complex drawing objects. After the raster-to-vector conversion, DG is initially created as a simple graph structure. During the recognition process DG becomes more and more complex, finally describing a hierarchical structure of map objects.

2. The raw vectorization process

Although there are different sophisticated vectorization methods (see for instance [1], [6] and [10]) to get better image quality, we take less effort in raw vectorization and concentrate on object recognition based on the raw vector image. Vectorization anomalies (see Figures 2 and 3) are corrected after recognition, because recognition results help us to decide: what to correct and how to correct.



Figure 2. Scanned raster image.

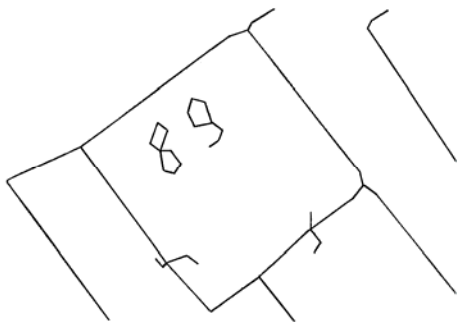


Figure 3. Raw vector image.

The raster-to-vector conversion is executed on the whole scanned binary image, independently of its contents. We use the most common vectorization process:

- (i) Thinning of the raster image and removing interior pixels.
- (ii) Detecting endpoints and junction points.
- (iii) Line tracing. Starting from endpoints and junction points, lines are traced to detect straight sections and breakpoints.

The above procedure may result in redundant breakpoints along quasi-straight lines, therefore we need some *optimization*. Denote P_i a point given with coordinates (x_i, y_i) , A chain of vectors $P_1P_2, P_2P_3, \dots, P_{n-1}P_n$ is optimized so that we search for an i where the distance of P_i from the line $P_{i-1}P_{i+1}$ is minimal and less than a given tolerance value T . If so, the point P_i is marked to be deleted. We continue the search on the chain

$P_1P_2, \dots, P_{i-1}P_{i+1}, \dots, P_{n-1}P_n$, and the process is followed until points can be deleted. We should emphasize, that tolerance T is checked not only for the current point but also for the previously deleted points.

The result of the above process is a raw vector image (see Fig. 3), which is the basis of all further recognition and correction procedures.

3. Data representation

Although internal data representations are not usually published, now we decided to discuss it because it is an essential part of our concept. Our data structure - called *DG* - can be regarded as a *set of cross-referenced objects*: each object represents a simple or complex part of the drawing, and has a unique identifier number to offer the possibility for other objects to reference it.

The DG array may contain four different types of objects:

NODE object is a tuple $(x, y, edge_1, \dots, edge_n)$ which represents a point with coordinates (x, y) , and with references $edge_1, \dots, edge_n$ to edges starting out of that point.

EDGE object is a tuple $(node_1, node_2, pat_1, \dots, pat_m)$ which represents a straight line section given with references $node_1$ and $node_2$ to the endpoints, and with optional references pat_1, \dots, pat_m to the patterns containing that edge.

TEXT object is a tuple $(x, y, height, angle, string)$ representing an inscription on the map. It is given by centerpoint coordinates (x, y) , text height, rotation angle, and the ASCII string itself.

PAT object represents a pattern in general sense. It is a tuple $(x, y, obj_1, \dots, obj_k)$, where (x, y) are centerpoint coordinates and obj_1, \dots, obj_k are references to arbitrary type DG objects which are considered as components of the pattern. As a typical case, a PAT object is a set of edges that has been recognized as a pattern on the drawing. But in general, a PAT may have other PATs as components, an EDGE may be included in two different PATs, etc.

Note that each object has a *layer* attribute defining a CAD-like layer number to the object. The system currently uses 20 layers, the most important ones are as follows:

- continuous line layer (contains NODEs and EDGEs),
- dashed line layer (contains PATs),
- separated symbols (contains PATs),
- connection sign layer (contains PATs and EDGEs),
- null-circle layer (contains PATs and NODEs),
- building layer (contains PATs and TEXTs),
- parcel layer (contains PATs and TEXTs),
- recognized text layer (contains TEXTs),
- polygon layer (contains PATs).

Initially the DG structure contains only NODE and EDGE objects in the "continuous line" layer, forming a

graph-like representation of the raw vector image. During the recognition process new objects are added to this graph in different layers, and finally a complex, hierarchical structure can be achieved to describe the drawing.

4. Segmentation

Segmentation here means classifying drawing lines into different categories (layers). First the dashed lines are recognized and marked as PAT objects in "dashed line" layer. After that, small connected patterns are detected and marked as PAT objects in "separated symbol" layer. (The symbols are not recognized in this phase.) As a result, the raw vectors are classified into three disjoint layers:

- dashed lines,
- separated symbols,
- continuous lines (the remaining vectors).

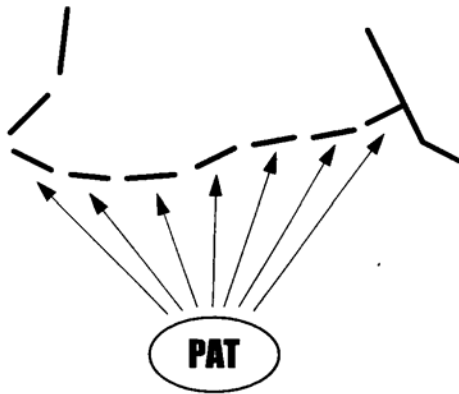


Figure 4. Dashed line recognition.

For dashed lines we use a recognition process similar to that applied in [3] but with certain modifications. We search for pairs of endpoints (that is, NODE objects with only one outgoing edge), which are close to each other. If we find such a pair A and B, then we analyse the edges AA_1 and BB_1 starting out from the endpoints. If the sequence of points A_1, A, B, B_1 can be covered by a straight line (with certain tolerance), then the vectors A_1A and BB_1 form a dashed line segment. We try to continue the dashed line in both directions in a similar way. When it cannot be continued, then a PAT object is inserted into the DG involving the vectors of the recognized dashed line (Fig. 4). Finally we mark all EDGE objects belonging to dashed lines to exclude them from further investigations.

Note that dashed curves are also recognized in this way.

Separated symbols are typically inscription characters (letters or digits), or other geographic notations on the

map. If we find a connected set of vectors, then we determine its circumscribing rectangle. If the longer size of the rectangle is between some given parameters d_1 and d_2 , then a PAT object is inserted into the DG covering this set of vectors, and it is handled as a symbol candidate in the future.

5. Recognizing separated symbols

Inscriptions on maps may have arbitrary rotation angle. This circumstance makes the recognition much more difficult than that of text document images. For instance, the decision that a digit is 6 or 9, is possible only if the global structure of the map is interpreted. It is an important help to solve this problem, that in most cases inscriptions are strings rather than single characters. If the string direction α is determined, then there are only two rotation angle possibilities: α and $180^\circ + \alpha$.

Thus, the first step of symbol recognition is to detect groups of symbols forming strings. After that the rotation angle can be determined, and symbols are rotated into normal position. The recognition itself is performed by a back-propagation neural network model, which is a useful tool in recognizing vectorized drawings [7].

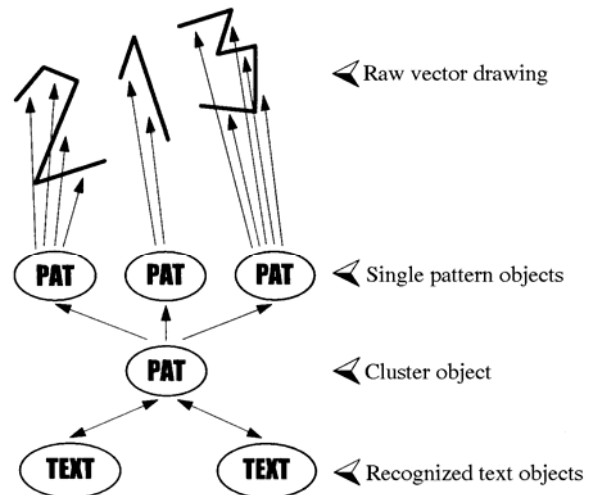


Figure 5. Recognition of strings.

We search for symbol PAT objects to be close to each other and to be along a straight line. If so, a PAT object of type "cluster" is inserted into the DG having the symbol PAT objects as components (Fig. 5). The rotation angle α is determined and inserted as extra attribute to the string PAT. After that, two empty TEXT objects are inserted into the DG: one for recognition with rotation angle α , and one for recognition with $180^\circ + \alpha$ (Fig. 5).

The neural recognition is performed symbol by symbol. The current symbol first is rotated by $-\alpha$ or $-(180^\circ + \alpha)$, to be in normal position. After that the following features are determined from the vector set of the symbol:

- number of EDGE objects,

- number of NODE objects,
- number of endpoints (NODEs with one outgoing edge),
- number of breakpoints (NODEs with two outgoing edge),
- number of junction points (NODEs with more than two outgoing edge),
- upper "max. vector" (see below),
- middle "max. vector" (see below),
- lower "max. vector" (see below).

To determine the last three features, the circumscribing rectangle of the symbol is divided into upper, middle and lower segments (Fig. 6). For each segment the longest vector (x_1, y_1, x_2, y_2) , starting from an endpoint, is determined, and a "max. vector"

$(\max(0, x_1 - x_2), \max(0, y_1 - y_2), \max(0, x_2 - x_1), \max(0, y_2 - y_1))$ is calculated for each segment (see the features above).

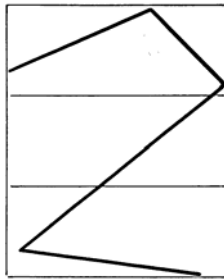


Figure 6. Dividing symbols for segments.

A back-propagation neural network is used for recognition. The network gets the 17-element feature vector described above and puts a n -element output vector concerning to the symbol types to be recognized. The highest output value means the recognition result. If the highest value is under a certain threshold, we consider the symbol unrecognized.

All symbols are recognized both with α and $180^\circ + \alpha$ rotation and recognition results are stored in the TEXT objects (Fig. 5). After that a global recognition rate is calculated for both TEXT objects, which is weighted with a "preferred angle" parameter given for the whole map. (For instance, if the map has been scanned with a rotation of 90° , then "preferred angle" should be set to 90° .) We choose the TEXT object, that has greater recognition rate.

However, we do not want only a correct drawing as recognition result, but we need to build a spatial database, that is why we should distinguish between *house numbers* and *parcel numbers*. Distinction is made by size (parcel numbers are a bit larger) and by length (parcel numbers usually have more than 3 digits while house numbers have at least 3 digits), but this distinction is unsure and needs manual checking (see the last chapter).

6. Recognizing not separated symbols

Symbols that are not separated from parcel and building lines, remain in the "continuous line" layer after segmentation. There are two such symbols on Hungarian cadastral maps:

- *connection signs*, expressing that a given building belongs to a given parcel or to another building (see Figures 2 and 7),
- *null circles*, denoting distinguished measured points on the cadastral map, typically occurring at the corners of parcels (Fig. 8).

There are different techniques to recognize such symbols, mostly used for recognizing different types of diagrams. For instance, [8] searches for loops in the raw vector graph, while [12] uses generic rules to describe symbols. Although these techniques cannot be applied directly in our case, we follow a similar approach.

To recognize connection signs, we search for the three typical situations of Fig. 7 in the "continuous line" layer. After recognition, edges of the connection sign are deleted, segments of the holding line are unified into one straight line, and this EDGE object is marked with an attribute meaning "this edge holds a connection sign".

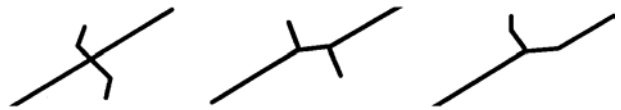


Figure 7. Three typical occurrences of connection signs on the raw vector image.

Null-circles denote distinguished identifier points on the cadastral map, and typically occur at the corners of parcels. The recognition is based on finding small closed polygons on the "continuous line" layer. After recognition, null-circles are compressed into their center and are stored as a NODE object with attribute "null-circle" (Fig. 8).

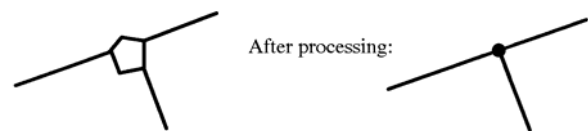


Figure 8. Appearance of null-circles at the raw vector map and at the processed map.

7. Recognizing buildings and parcels

The problem is, how to distinguish parcel lines from building lines. Hori and Shimotsuji [5] proposed probabilistic relaxation for a similar problem. However, in our case we have an easier method: we can utilize the information offered by the recognized connection signs, as follows.

After recognizing and processing connection signs and null-circles, the "continuous line" layer contains a clear polygon structure, that is, a set of disjoint polygons covering the whole image in the map frame. First we create PAT objects for all these polygons. The algorithm is as follows.

All polygons are walked around. We start out from an edge, at junctions we always choose the rightmost edge, until we reach back to the starting edge. Each affected EDGE object is labelled according to the walking direction (label 1 means walking from $node_1$ to $node_2$, label 2 means walking from $node_2$ to $node_1$). The procedure is continued until all edges are labelled both with 1 and 2. In this way all polygons are walked around clockwise and "islands" also counter-clockwise (Fig. 9). During the procedure a PAT object is generated to each polygon.

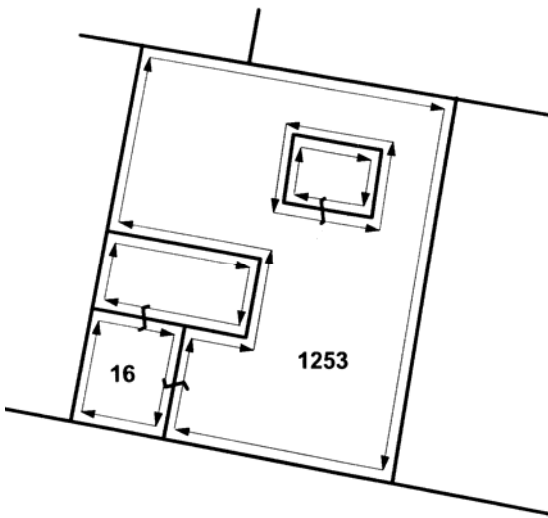


Figure 9. Recognizing polygon structure.

If we compute the area of each polygon, we get positive values for clockwise polygons and negative values for counter-clockwise ones. Because latter ones represent "islands" in a containing polygon, their edges are joined to the PAT of the containing polygon.

The result is a set of PATs, some of them represent buildings, others represent parcel-fragments (that is parcel territories excluding the buildings). Latter ones can be identified by the enclosed parcel number and will be deleted. In this way the ambiguous situations like in Fig. 10 can be managed too.

To detect parcel polygons, the "continuous line" layer is processed again. First, all edges holding connection signs are temporarily deleted. In the remaining structure all edges, starting from endpoint, are also temporarily deleted. The deletion process is repeated until all edges starting from endpoints are eliminated. We can realize that the remaining polygon structure is just the set of parcel polygons. Finally, the parcel PATs can be supplemented with a reference to the TEXT object of the enclosed parcel number.

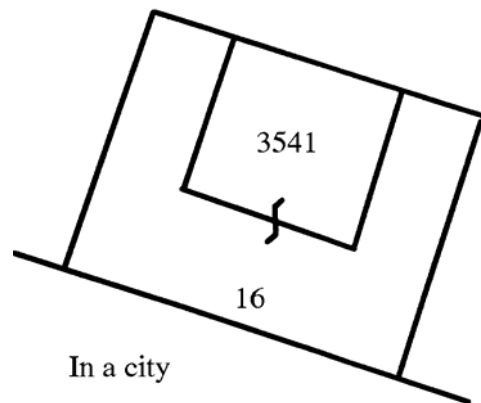
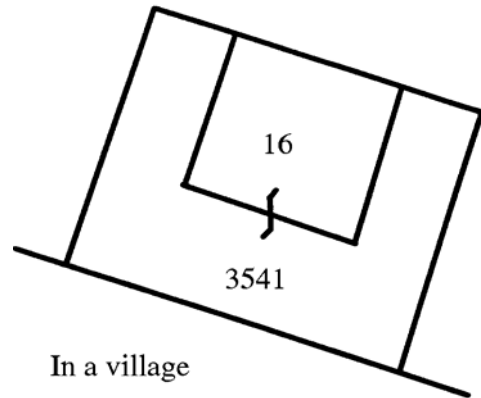


Figure 10. Example of an ambiguous situation: small building on a big parcel, and big building with a small courtyard (with house number 16 and parcel number 3541 in both cases).

8. Drawing correction

The raw vector map contains many small inaccuracies (see Figures 2, 3), which do not influence the recognition process, but should be eliminated to get a correct digital map. Fig. 11 shows typical anomalies occurring at corners and T-junctions.

We consider the drawing correction problem also as a *recognition problem*: we search for situations like in Fig. 11, and create a "correction PAT object" to each situation. These PAT objects can be accepted or rejected during a manual checking procedure.

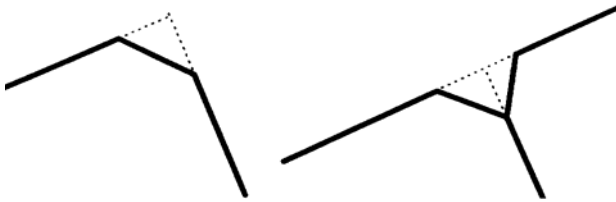


Figure 11. Typical anomalies on the raw vector image and their correction.

The recognition itself is based on searching for short edges and analysing their neighborhood. A similar concept is applied in [3].

9. Implementation and experiments

The MAPINT program has been developed in C++ language under MS-Windows environment. Recognition of each map object type is split into three steps:

- *Automatic recognition*: new PAT objects are created in the DG structure.

- *Manual checking and correction*: PAT objects can be accepted or rejected one by one. This is an optional step, we can say “accept all” if the recognition rate is good enough.

- *Processing of accepted objects*: recognition results are made final by modifying the DG structure. For instance, in the case of separated symbols, original vectors and intermediate PATs are deleted and only the accepted TEXT object is kept (Fig. 5).

When checking separated symbols, a *training file* is generated automatically storing the generated feature vectors and the corresponding answers entered by the user. This file can be used to train the neural network.

The system has been tested with cadastral maps from different territories, each scanned with 300 dpi resolution. Recognition rates strongly depend on image quality, summarizing we give the following experimental results:

<i>object type</i>	<i>min.</i>	<i>max.</i>	<i>average</i>
dashed lines:	84 %	99 %	94 %
numeric characters:	70 %	97 %	85 %
connection signs:	87 %	97 %	92 %
buildings	79 %	98 %	93 %
parcels	83 %	97 %	91 %
drawing correction	91 %	96 %	93 %

Currently the system is used in the Phare HU905.0203 Land Consolidation Project, the aim of which is to develop a full technology for Hungarian land registry offices to create a uniform digital map database for the whole country. For this purpose, MAPINT has been supplemented with a control point based affin-transform to convert scanned maps into the required map projection.

As future development, we plan to extend the role of neural recognition for different types of drawing objects. The aim is to get a *universal learning system*, which is not specialized to a given type of drawing, but - during manual digitization - it generates training files (see above) and can learn how to handle the current drawing type.

References

- [1] L. Boatto et al.: An Interpretation System for Land Register Maps, *Computer (IEEE)*, July 1992, pp. 25-33.
- [2] Y. Chen, N. A. Langrana and A. K. Das: Perfecting Vectorized Mechanical Drawings. *Computer Vision and Image Understanding*, Vol. 63, No. 2, pp. 273-286 (1996).
- [3] N. B. Ebi: Image Interpretation of Topographic Maps on a Medium Scale via Frame-Based Modelling. *Proc. of Int. Conf. on Image Processing*, Vol. I, pp. 250-253 (1995).
- [4] L. Heute, J.M. Ogier, Y. Lecourier, C. Olivier: Two Aspects of Automatic Map Treatment: Road and Texture Extractions. *11th IAPR Internat. Conf. on Pattern Recognition*, IEEE Computer Science Press, pp. 109-112 (1992).
- [5] O. Hori and Sh. Shimotsuji: Probabilistic relaxation method for line-drawing interpretation. *11th IAPR Internat. Conf. on Pattern Recognition*, IEEE Computer Science Press, Vol. 2, pp. 158-161 (1992).
- [6] R. D. Janssen and A. M. Vossepel: Adaptive Vectorization of Line Drawing Images. *Computer Vision and Image Understanding*, Vol. 65, No. 1, pp. 38-56 (1997).
- [7] E. Katona, K. Palágyi, N. Tóth: Signature verification using neural nets. *Proceedings of 9th Scandinavian Conference on Image Analysis*, pp. 1115-1122 (1995).
- [8] S. H. Kim, J. W. Suh and J. H. Kim: Recognition of Logic Diagrams by Identifying Loops and Rectilinear Polylines. *Proc. 2nd Int. Conf. on Document Analysis and Recognition*, pp. 349-352 (1993).
- [9] T. Nagao, T. Agui and M. Nakajima: An Automatic Road Vector Extraction Method from Maps. *Proc. 9th Internat. Conf. on Pattern Recognition*, Vol. I, pp. 585-587 (1988).
- [10] P. Vaxivière, K. Tombre: Celestin - CAD Conversion of Mechanical Drawings, *Computer (IEEE)*, July 1992, pp. 46-54.
- [11] H. Yamada, K. Yamamoto, T. Saito and K. Hosokawa: Recognition of Elevation Value in Topographic Maps by Multi-Angled Parallelism. In: *Advances in Structural and Syntactic Pattern Recognition* (ed.: H. Bunke), World Scientific (1992).
- [12] Y. Yu, A. Samal and S. C. Seth: A System for Recognizing a Large Class of Engineering Drawings. *IEEE Transactions on Pat. Anal. and Machine Int.*, Vol. 19, No 8, pp. 858-889 (1997).