MVA '92 IAPR Workshop on Machine Vision Applications Dec. 7-9,1992, Tokyo

FAST RASTER–TO–VECTOR CONVERSION OF LARGE–SIZE 2D LINE–DRAWINGS IN A RESTRICTED COMPUTER MEMORY

S.Ablameyko, V.Bereishik, O.Frantskevich,
M.Homenko, E.Melnik, N.Paramonova

Institute of Engineering Cybernetics
of the Belarusian Academy of Sciences

6, Surganov str., 220012 Minsk
Republic of Belarus
e–mail: mahaniok@adonis.ias.msk.su

ABSTRACT

An effective pipeline oriented scheme and new techniques to process large–size 2D line–drawing images in a restricted computer memory are suggested. The main differences of our approach are a modified run–length image representation on all processing steps, storing a limited amount of image lines in memory, and new raster–to–vector transform algorithms based on a set of local operations and the same principles. Experiments performed on different line–drawing types show that our approach allows:

- to process large–size binary images without dividing them into parts in a restricted computer memory;
- to develop fast processing algorithms with low computational complexity;
- to obtain acceptable time characteristics of raster–to–vector conversion on a personal computer.

This approach can be easily realised in pipeline architecture using simple transputer chain.

INTRODUCTION

The process of automatic input of 2D line–drawings into CAD systems or Geographical Information Systems (GIS) is very important for different applications. Many systems have been developed to organize this process efficiently [1–3]. The most time consuming task in this process is preprocessing of scanned binary images and their conversion to vector form. There are two main problems for the raster–to–vestor (r–t–v) transform. Line–drawings have usually large size (A2/A1) that require a large amount of memory for their storing. The second problem is a large amount of operations performed during the r–t–v conversion which takes an unacceptable processing time.

To solve the first problem, the initial image is often divided into parts that after vectorisation are joint back together. There are some difficulties in processing part frames in this way. Another approach is storing and processing in a special buffer a limited number of adjacent image lines. In this case the image can be processed line–by–line.

It is possible to reduce a volume of raster data by using different coding techniques. The powerful and popular coding technique is run–length (r–l) coding, which can be easily obtained from a scanner. In papers [4,5] the description of line–by–line processing and realization of some preprocessing operations on this representation have been given.

However, to obtain acceptable time characteristics most of the systems use either powerful workstations or specialized hardware. But for wide practical use it is necessary to have systems based on a general purpose low cost hardware.

In this paper we suggest an effective pipeline oriented scheme and new techniques for processing of large–size 2D line–drawing images represented by modified run–length coding.

The main characteristics of our approach are:

- the possibility of processing large–size binary images in a restricted computer memory without division them into parts;
- the automatic calculation of object thickness;
- the satisfactory time characteristics for the r–t–v transform.

As an input we consider: 1) engineering drawings and 2) different black–and–white layers of geographical maps.

GENERAL PRINCIPLES OF RASTER–TO–VECTOR CONVERSION

The main tasks of this conversion are well known: noise reduction, area extraction, thinning or contouring, vectorization, approximation, and information recording in data base. We assume that each of these tasks can be solved using the same principles. It is supposed that the result of any operation implementation to any line is fully determined by only few adjacent lines. These lines (image stripe) are stored in special buffers of program. Each program has access to all of these lines and can check, process and change their pixels using some basic techniques. The lines in the stripe are processed from bottom to top. The stripe moves consequently on the whole image from top to bottom and all operations are performed during this move. Therefore, we obtain that after every stripe inspection the corresponding operation is fully executed for the top line of the stripe. This line is removed from the stripe and moves to next processing function. A new bottom line is introduced in the stripe.

Every line in the stripe is represented by an ordered list of x–coordinates of black (white) pixels, which have white (black) pixels to the left. Usage the coordinates of the beginning black and white runs instead of the run length (as in a standard run–length representation) allows:

- to reduce a volume of raster data;
- to make more simple analysis of pixel neighbourhoods and runs of adjacent lines;
- to reduce a computational complexity of processing algorithms.

RASTER-TO-VECTOR TRANSFORM ALGORITHMS

Inside of the suggested approach, fast processing algorithms have been developed. These algorithms are based on a set of basic operations, which different combination is used for every algorithm. Consider these algorithms in more details.

NOISE REDUCTION. Three main algorithms for noise rediction have been developed:

- logical mask filtering based on sequential 3x3 mask analysis and parallel 8-dilation and erosion performed simultaneously;
- small spots and holes detection and removal based on a fast simplified contour following and modification of image lines containing these elements;
- threshold smoothing of horisontal, vertical and diagonal contour protrusions and dents based on an original concept of "fuzzy raster straight lines" in the mentioned directions.

The last two algorithms use maximal parameters of defects which age given a priori. All three algorithms can be used either separately or together, consequently one by another. The number of lines in the stripe depends from chosen parameters.

Usually on engineering drawings and maps there are small areas, like arrows and dots, which are better represented by their contours. For their extraction we introduce the next algorithm.

AREA EXTRACTION. Consider that we know a priori a maximal thickness of elongated objects. Then image objects are eroded according to a chosen parameter (half of maximal object thickness). As structuring element we use a "raster circle" representing discretized image of circle with a given diameter. This operation deletes from the image all elongated objects excepting internal parts of areas. Then we subtract one image from another and obtain that on the initial image all areas are represented by objects which thickness is equal approximately to the thickness of all elongated objects.

THINNING. Thinning algorithm is based on the algorithm given in [6] and is modified for our approach and representation. In the stripe, $2w + 2$ lines are stored, where w is a maximal thickness of objects. All lines in the stripe are enumerated from bottom to top. Thinning is performed from the first line (bottom) to the last (top) line. Object in every line in the stripe is thinned different amount of times (has different thickness). The lowest line is completly unthinned and the highest one is fully thinned, all middle lines are partially thinned. For thinning one line, two adjacent lines (lower and upper) are used. Under pixel analysis a mask is formed which represents a 8-bit code. Using this mask an input in a look-up-table is performed, where a new pixel value is defined. A labelling of each full thinned pixel is performed, and the label is equal to the number of iterations performed.

VECTORIZATION. The transformation of thinned image into a vector form is performed by one scan and storing three lines in the stripe. In a medial line of the stripe, black pixels are analysed using the Crossing number. Depending on the Crossing number value, the following situations are extracted: object beginning, end of object, continuation, merging, splitting, node, and isolated point. An algorithm for the situations processing has been developed [8]. In the result of vectorization, object segments bounded by end points and nodes are extracted. The simple approximation is simultaneously performed.

CONTOURING. Algorithm stores two lines in the stripe and solves the following main tasks: extracting a situation in the stripe and its solution. The possible situations on the image are known: object beginning, continuation, splitting, merging, end of the object. Algorithm for the situation processing has been developed. Under the contour extraction, special buffers are reserved. They are intended for the assembly and storing information about every contour. Then the buffers can be merged or splitted depending on the processed situation. Simultaneously with the object contour extracting, geometrical parameters of objects are calculated. The relations between processed objects (e.g. one object is located inside another one) are computed.

VECTOR DATA BASE OBTAINING. To form a vector data base, polygonal approximation of segments is performed simultaneously with the computing of geometrical parameters. For the engineering drawing, under segment approximation an arcextraction is performed. From this step we either obtain the IGES or DXB AutoCAD file represented by simple primitives or continue interpretation process. Some of these techniques are described in more detail in [7].

BASIC OPERATIONS

These basic operations are used in the described above algorithms.

3x3 NEIGHBOURHOOD ANALYSIS USING LOOK-UP TABLE. To analyse a neighbourhood of processed pixel x, we perform alignment of intervals of two adjacent lines (lower and upper) so that right end of black or white interval is not less than x. Thus pixel in a central column have the same value and it is necessary to define last 6 pixels. In the result, 9-bit code of neighbourhood is formed.To analyse a neighbourhood of next pixel it is necessary only to add a right column of neighbourhood.

To process pixel, a special look-up-table (LUT) is used. It represents one-dimensional 256-byte array. Input to the LUT is performed by using 8-bit code that we obtain from 9-bit code of neighbourhood. Every element in the LUT denotes the pixel type or action that must be performed to process the pixel. Each processing algorithm uses its own LUT. For example, the LUT for filtering defines a changing black (white) pixels to white (black), thinning LUT contains pixel characteristics (deletable pixel, completly thinned, etc).

PIXEL LABELING. The used r-l representation is not convinient for pixel labeling. But this operation is used in many algorithms. It is performed by storing a pixel line representation together with r-l line representation. This double line representation is permissible in our approach because only a limited number of lines is stored in computer memory. It allows to have a fast access to every pixel and store label of the pixel. Every processing algorithm gives own labels to pixels.

MODIFICATION OF IMAGE LINES. All processing algorithms change an initial line representation. It is made in one of the following ways:

- delete a black run by corresponding run labeling,
- run borders are changed by changing the border coordinates;
- a new r-l line representation is formed in a new buffer.

FOLLOWING THE CONNECTED PIXEL CHAINS. This operation is used to follow branches of borders or thin objects, to merge or split them depending on a processed situation. To perform chain following, two adjacent lines are used. During the following, a calculation of branch parameters and their description in terms of Freeman code or simple polygonal line can be obtained.

SELECTION OF SIGNIFICANT PIXELS. The used r-l representation allows to process only significant for implemented operation pixels and reduce a computational complexity of algorithms and their processing time. For example, only black pixels are analysed under thinnning. A white intervals are ignored and are not considered. This principle is used in all algorithms too.

RESULTS AND DISCUSSION

The software for the realization of r-t-v conversion was developed on an IBM PC/AT computer in C language. As an input device we used a large-format scanner, constructed in our Institute and a standard scanner A4 format. The input binary images were obtained from black and white layers of maps as well as from engineering drawings with size A4-A2, digitazed with a resolution of 20 pixels per 1 mm and 300 DPI. The raster data for the processing are represented in PCX, TIFF or MSP format. Output format now is IGES or AutoCAD DXB for engineering drawings in terms of simple primitives and a special format for maps.

Two variants of r-t-v conversion have been developed: "skeletonized" variant which obtains a skeleton object representation of initial image and "contoured" one which obtains a countour object representation. The results of processing map layers and engineering drawings by "skeletonized" variant are shown in Fig.1a,b,c and by "contoured" variant in Fig.1d.

The execution time for every function usually does not exceed few tens of seconds. The total time depends on the number of objects, their length, and set of used functions. The average time for the "contoured" conversion is equal to 1-2 minutes for A4 format of line-drawing scanned by 300 DPI, and 3-5 minutes for the "skeletonized" variant on IBM PC/AT 386. The map layers with size 500x600 mm are processed by the "skeletonized" variant 7-10 minutes.

The main differences of our approach from those already proposed are:

- a modified run-length coding for the representation of image on all steps, which allows to compress image and at the same time to have an access to all pixels of adjacent lines;
- a specialized scheme for the processing, wnich is based on the storing in memory a limited amount of lines;
- the algorithms for raster-to-vector transform which are based on a set of local operations and the same principles;
- it can be easily realized in pipeline architecture using simple transputer chain.

The performed experiments allow us to use and recommend this approach for the realization of raster-to-vector transform of large-size 2D line-drawing images on personal computers with the restricted memory resources.

CONCLUSION

The approach for the processing 2D line-drawings in the restrictions of memory has been suggested. The main differences of this approach are: a modified run-length image representation, line-by-line processing, a specialized pipeline oriented scheme and processing algorithms based on a set of basic operations. This approach allows to process large-size images with satisfactory time characteristics and a satisfied quality. This approach can be easily realized in pipeline architecture using simple transputer chain.

Now this work is continued on the recognition of basic engineering drawing primitives and cartographical objects.

ACKNOWLEDGMENT

We wish to acknowledge the assistance of A.Grenov, O.Patsko, N.Bereishik, O.Okun and I.Ljatkevich in implementing and testing the system described in this paper.

REFERENCES

1.R.Kasturi, S.T.Bow, W.El-Mastri, J.Shah, J.R.Gattiker, U.B.Mokate, A system for interpretation of line drawings, IEEE Trans. on PAMI, 12, pp.978-992, 1990.

2.A.Antoine, S.Collin, K.Tombre, Analysis of technical documents: The REDRAW system, Proc. IAPR Workshop on Syntactic and Structural Pattern Recognition, Murray Hill, pp.1-20, 1990.

3. Computer. Special issue on systems of document image analysis, July 1992, 113p.

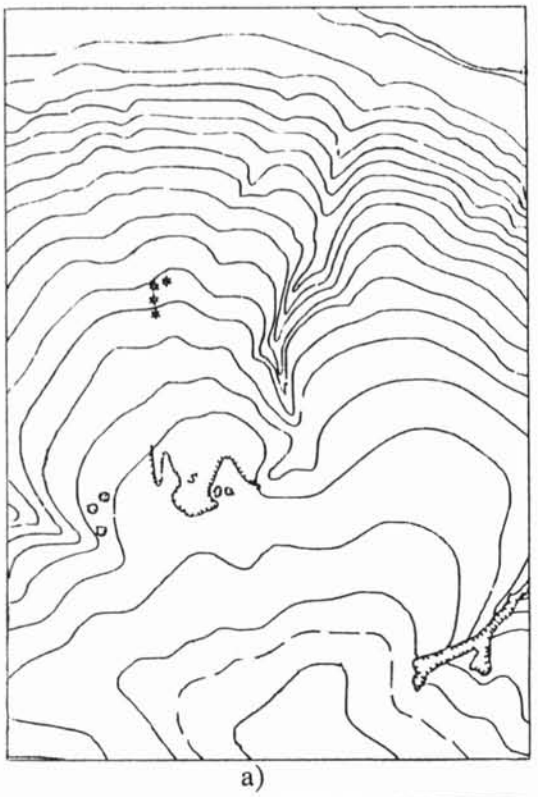
4.D.Rutovitz, Efficient processing of 2-D images, in Progress in Image Analysis and Processing, V.Cantoni, L.P.Cordella, S.Levialdi and Sanniti di Baja, Eds., World Scientific, Singapore, pp.229-253,1990.

5.J.Piper, Efficient implementation of skeletonisation using interval coding, Pattern Recognition Letters, No.3, pp.389-397, 1985.

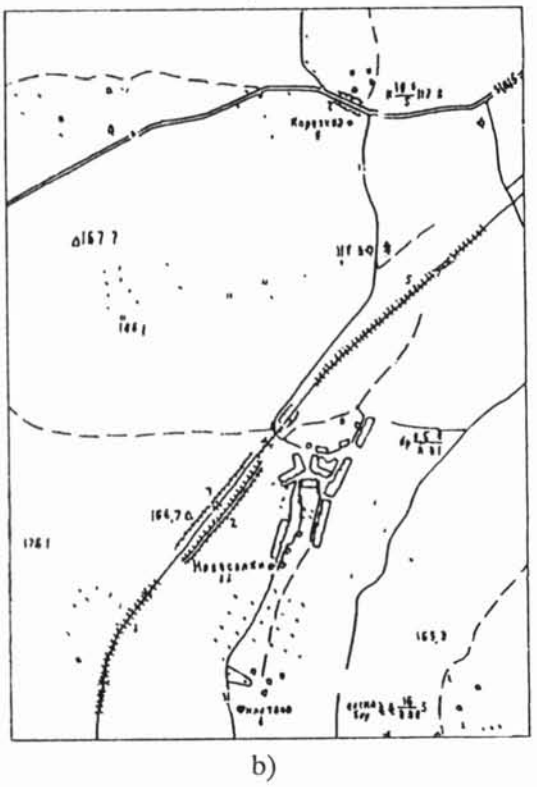
6.J.Hildich, Linear skeletons from square cupboards, in Machine Intelligence 4, B.Meltzer, Ed., Edinburgh University Press, pp.404-420, 1969.

7.O.Semenkov, S.Ablameyko, V.Bereishik, V.Starovoitov, Information processing and display in raster graphic systems, Nauka i Technika, Minsk, 183p., 1989, (in Russian).

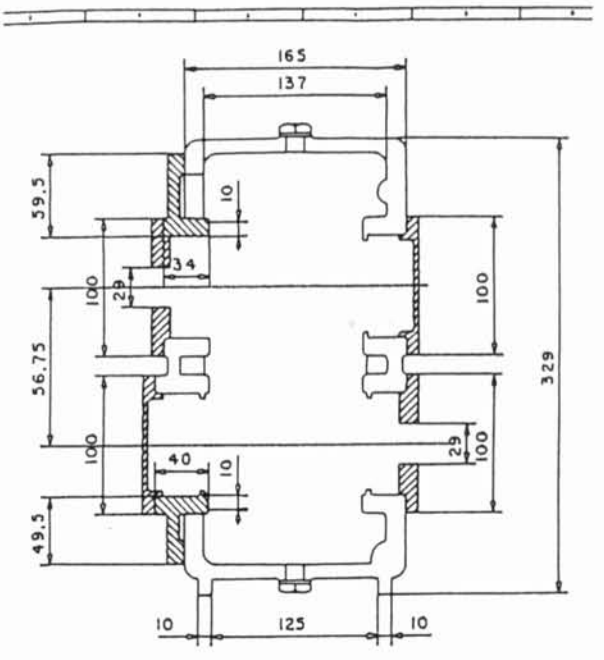
8.S.Ablameyko, V.Bereishik, N.Paramonova, A.Marcelli, S.Ishikawa, K.Kato, Line-drawing description: from skeleton to hierarchical vector representation, Proc. IEICE Workshop on Pattern Recognition and Understanding, PRU 91-75, 24-25 October 1991, Toyama, Japan, pp.23-30, 1991.



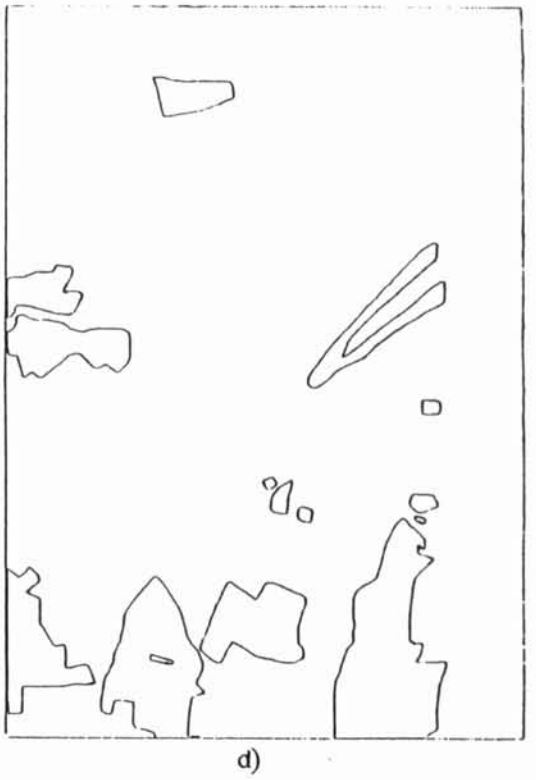
a)



b)



c)



d)

Fig.1. Examples of 2D line–drawing processing by a–c) "skeletonized" variant, d)"contoured" variant of raster–to–vector conversion.