Optoelectronics, Instrumentation
and Data Processing

Avtometriya
UDC 681.3.04/05

A PARALLEL ALGORITHM FOR THINNING OF BINARY IMAGES

A. N. Kolesnikov and E. V. Trishina

A parallel algorithm based on the distance transform for thinning of binary images is developed. The represented algorithm is intended for implementation on multiprocessor workstations with distributed memory. The checking of the algorithm on actual images demonstrated its validity and efficiency.

INTRODUCTION

Thinning (skeletonization) of images has found a wide use in image processing and pattern recognition for the structural description, parameters measurement, and redundancy reduction of images [1-6]. Due to the large size of images to be processed, for treatment of applied tasks the thinning of images is very time-consuming process. One way to reduce the time cost of computations is to use multiprocessor computer systems.

Our aim is to develop a parallel algorithm for thinning of binary images, which is intended for implementation on multitransputer workstations with distributed memory.

In Section 1 the sequential algorithm for thinning of images is briefly described. Section 2 discusses a hardware which the algorithm under development is intended for, as well as a model for parallelization. The parallel algorithm for four processors is discussed in Section 3. In Section 4 the parallel algorithm for a 2D mesh network is presented. In Section 5 a program implementation for a transputer workstation is described and results of checking the program are discussed.

1. The sequential algorithm for image skeletonization. An algorithm based on the distance transform (DT) for image skeletonization has been selected for parallelization. This method has two important advantages: the skeleton of an image can be produced in a fixed number of passes through the image, and every point of the skeleton labeled by distance to the nearest background point, thereafter the distance can be converted to the local width of the object. The algorithm for skeletonization consists of three phases:

– distance transform;
– detection of skeletal points;
– distance-to-width conversion and reduction to unit width of the skeleton (for lines of even width).

The distance transform. The type of distance transform is defined by the type of distance metrics. In this work we used the distance transform with the "chessboard metrics" [7,8]. The distance transform of this kind can be performed by two raster scans through the whole image: the first scan is performed by a 3×2 window in the lexicographic order; the other is performed in vice versa fashion. Hereafter we call these scans as forward and backward passes. We suppose that the input image has a unit width border of background color. In addition, we suppose that the image has initial labeling for the forthcoming DT as follows: the background pixels are set equal to zero, the foreground ones are set equal to a sufficiently large number.

During the forward pass the DT value is calculated for every pixel p of an object by the following formula (Fig. 1):

$$p = \min(p, n_1+1, n_2+1, n_3+1, n_4+1).$$

© 1995 by Allerton Press, Inc.
Authorization to photocopy individual items for internal or personal use, or the internal or personal use of specific clients, is granted by Allerton Press, Inc. for libraries and other users registered with the Copyright Clearance Center (CCC) Transactional Reporting Service, provided that the base fee of $50.00 per copy is paid directly to CCC, 222 Rosewood Drive, Danvers, MA 01923. An annual license may be obtained only directly from Allerton Press, Inc., 150 5th Avenue, New York, NY 10011.

$$\begin{matrix} n_2 & n_3 & n_4 \\ n_1 & p & n_5 \\ n_8 & n_7 & n_6 \end{matrix}$$

Fig. 1. The relative location of the pixel p and its eight neighbors.

During the backward pass the DT value is calculated by means of the similar formula:

$$p = \min(p, n_5+1, n_6+1, n_7+1, n_8+1).$$

The results of the distance transform for a test object after the forward and backward passes are represented in Fig. 2 to illustrate an essence of the technique.

The skeletal points detection from the local extremums of the DT is also performed in two passes through the whole image by a 3 × 3 window [8–9]. During the forward pass the skeletal points detection is performed by analyzing the distance transform data (Fig. 3, *a*). During the backward pass a connectivity of the skeleton is restored (Fig. 3, *b*). Due to the fact that the skeleton for even-width lines has a 2-pixel width, one additional pass for the skeleton thinning is needed. Concurrently, for every point of the skeleton the distance to the nearest background point can be converted to the local width of the line.

Conversion of DT values to width of the line. As it can be readily appreciated, the skeleton of even-width lines has a two-pixel width. If for forthcoming processing a skeleton of a unit width is needed, "extraneous" skeletal points are removed in one pass by a standard Arcelli algorithm. During this pass the distance transform D is converted into local width of lines W by the obvious formula: $W = 2D - d$, where d ($d = 0, 1$) is defined by the skeleton shape in a 3 × 3 window

So, several sequential passes through the whole image in two directions have to be done to perform the described algorithm for thinning. Due to this fact, the parallelization of this algorithm is by no means a trivial problem.

As can be seen, the distance transform is the crucial stage of the whole algorithm: if it is possible to develop a parallel algorithm for the distance transform, then the task of parallelization of the algorithm for skeletal points detection will be solved. In this connection the main attention in this paper is paid to describing the parallel

a

```
00000000000110000000000000
00000000000110000000000000
01111111111111111111111110
01222222222222222222222210
01233333333333333333333210
01234444444444444444443210
01234555555555555555543210
01234566666666666666543210
01234567777777777776543210
01234567888888888876543210
01234567899999999876543210
00000000000110000000000000
00000000000110000000000000
```

b

```
00000000000110000000000000
00000000000110000000000000
01111111111111111111111110
01222222222222222222222210
01233333333333333333333210
01234444444444444444443210
01234555555555555555543210
01234444444444444444443210
01233333333333333333333210
01222222222222222222222210
01111111111111111111111110
00000000000110000000000000
00000000000110000000000000
```

Fig. 2. The test image after the forward (*a*) and backward (*b*) passes of the distance transform.

a

```
00000000000110000000000000
00000000000110000000000000
01111111111111111111111110
01222222222222222222222210
01233333333333333333333210
01234444444444444444443210
01234555555555555555543210
01234444444444444444443210
01233333333333333333333210
01222222222222222222222210
01111111111111111111111110
00000000000110000000000000
00000000000110000000000000
```

b

```
00000000000110000000000000
00000000000110000000000000
01111111111111111111111110
01222222222222222222222210
01233333333333333333333210
01234444444444444444443210
01234555555555555555543210
0123444444××4444444443210
0123333333××3333333333210
0122222222××2222222222210
01111111111111111111111110
00000000000110000000000000
00000000000110000000000000
```

Fig. 3. The test image after the forward (*a*) and backward (*b*) passes of the skeleton points detection algorithm:
a - the skeletal points detected in the forward pass are underscored;
b - the skeletal points detected in the backward pass are marked by a cross.

algorithm for the distance transform.

2. Hardware and the model of parallelization. In developing a parallel algorithm suitable for to solving the applied tasks in image processing, the multiple-instruction multiple-data (MIMD) distributed memory multiprocessor systems were preferred because nowadays they are available on the market and give a good performance/cost ratio. Most of the MIMD computers have a regular constant-valence architecture topology and can generally be scaled proportionally without essential structure changing.

The character of the problem and the type of hardware dictate to select the geometric model of parallelism. In our case it means the following algorithm: the input image is divided into blocks that are distributed between processors combined into a 2D mesh processors network. Each processor performs processing of its own part of data and from time to time exchanges information with its neighbors. The image blocks are gathered in the host computer on completing the processing and are stored in the disk memory.

3. The parallel algorithm for skeletonization for four processors: $N_p = 2\times2$. Let us view a solution of the problem in hand for four processors. This particular solution will be used subsequently to develop a general solution for the 2D mesh processors network.

To parallelize the distance transform algorithm, first of all we take into consideration that its two passes have a commutativity property, i.e., the order of passes (namely, "left-to-right, top-to-bottom" and "right-to-left, bottom-to-top") through the image can be reversed. In addition, we can take into consideration the symmetry of the algorithm in the horizontal direction: the "left-to-right" processing procedure is commutative with the "right-to-left" one.

So, the distance transform algorithm under consideration can be parallelized for execution in four processors. According to this approach, the input image is divided into four image blocks (four quadrants). As all the distance transform procedures in fact are performed in a 3×3 window, the image blocks are to be overlapped in the 2-pixel width strip (Fig. 4, *a*)

Forward pass. The distance transform is performed in the four processors in converging directions (Fig. 4, *a*). The data exchange is performed after reaching the border as follows: every processor gives the last processed row and the last processed column to its neighbors, in its turn, the processor gets one column from one of the neighbors and a row from another one.

Backward pass. The processors perform the distance transform in the opposite direction (Fig. 4, *b*, *c*).

The data exchange is performed on the border of partition after the backward pass in the opposite direction, thereafter two passes (forward and backward) through the image are performed to detect the skeletal points. Clearly, the data exchange has to be performed between the two passes.

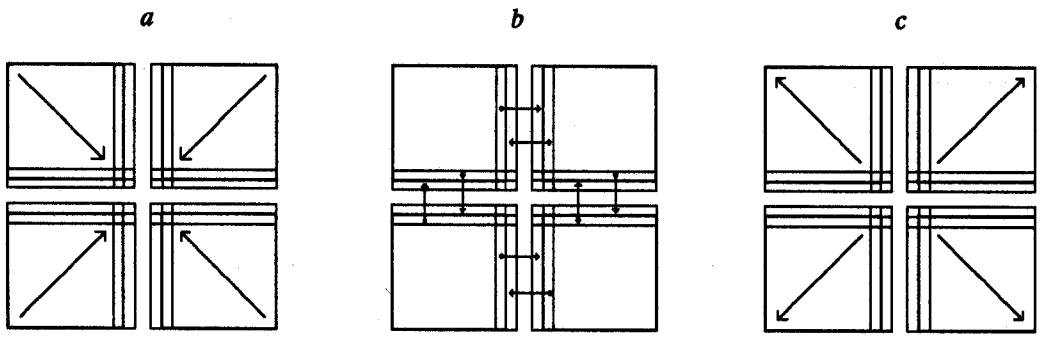

Fig. 4. A scheme of the image blocks processing for four processors: *a* - the forward pass direction; *b* - the scheme of data echange between the processors; c - the backward pass direction .

4. The parallel algorithm for a 2D mesh processor network ($N_p = N_x \times N_y$). As mentioned above, this algorithm is inherently sequential and its parallelization is by no means a trivial problem. Taking into account the symmetry of the algorithm permitted us to parallelize the processing for execution in four processors.

As for the parallel algorithm for a 2D mesh processor network, let us consider the following solution of the problem according to accepted model of parallelization: the input image is divided into blocks which have to be distributed between the processors, thereafter the forward and backward passes of the distance transform are performed with proper data exchange at the block borders. The distance transform of the whole image can be obtained by execution of sufficient number of passes. It is obvious that this algorithm is effective if the number of passes does not depend on the size of the image and, in addition, is not too large.

Let us view in detail the listed above phases of DT.

The image partition. So, the input image is divided into rectangular blocks with two-pixel width overlapping. The size of the block is:

$$H_x \times H_y = (\text{size\_}x/N_x + 2) \times (\text{size\_}y/N_y + 2),$$

where size_x × size_y is the size of the image; $H_x \times H_y$ is the size of the block; $N_x \times N_y$ is the size of the 2D mesh processors network. Every four processors are combined into one quadruplet of processors. In Fig. 5 the quadruplets are marked by dotted lines. For the sake of simplicity we assume that the number of processors is a multiple of 4; for majority of workstation models this condition is, as a rule, satisfied.

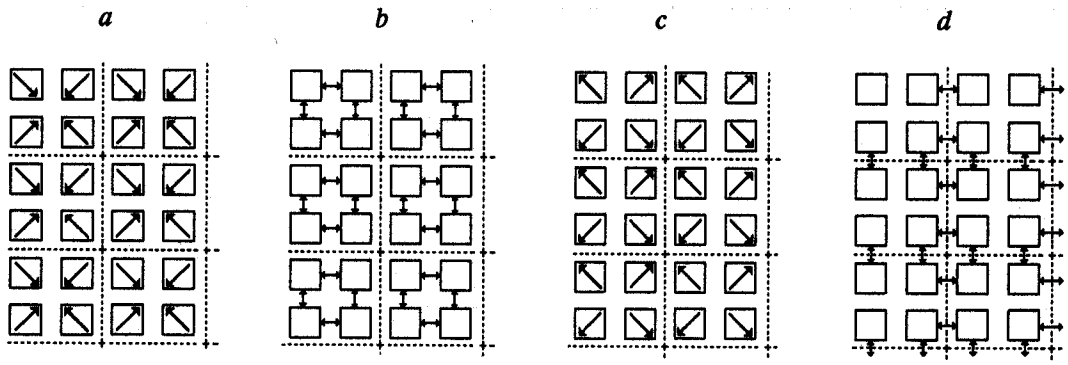

Fig. 5. A scheme of the image blocks processing for a 2D mesh processors network: *a* – the direction of processing for the forward pass; *b* – a scheme of the data exchange in quadruplets after the forward pass(data exchange inside quadruplets); *c* – the direction of processing for the backward pass; *d* – a scheme of the data exchange between quadruplets after the backward pass.

As for the problem of the optimal selection of the block size, ignoring possible limitations caused by the memory size of the processors, a solution can be obtained by minimization of the data amount of data exchange between the passes, that is, minimization of the perimeter of the rectangular block under condition of fixed area.

The forward pass. In every block the forward pass is performed in accordance to the scheme in Fig. 5, *a*; the direction of the pass is defined by number of the processor in the quadruplet.

After the first pass the data exchange is performed between blocks which belong to the same quadruplet (Fig. 5, *b*); every block gives the last processed row and column and gets two proper vectors which have to be disposed to the last row and the last column (the internal data exchange in quadruplets).

The backward pass. The second pass is performed in the opposite direction (Fig. 5, *c*), after that the data exchange between the neighboring blocks belonging to the adjacent quadruplets is performed (the backward pass and the external data exchange between quadruplets; Fig. 5, *d*).

Analysis of the DT data. After the backward pass and the external data exchange the DT data in the proper rows and columns are analyzed. On the basis of this analysis the number of rows n_y and columns n_x to be processed during the next forward pass are calculated and the four conditions to complete the DT after this pass are checked.

The number of rows n_y is defined by the expression:

$$n_y = \max_x\left\{0, \left\{\left(D\left(x,1\right) - D\left(x,0\right)\right)/2\right\}\right\},$$

where $D(x, y)$ is the values of the distance transform in the block; $0 \le x < \text{size\_}x$.

The processing can be successfully completed after the next forward pass if the two following conditions are satisfied:

1. $n_y < H_y$,
2. $\max_x\{D(x,1)\} < 2H_y$, where $0 \le x < H_x$.

In a similar way we have the expression for the number of columns n_x to be processed in the next forward pass, and the two last conditions of successful completion of the DT after this pass:

$$n_x = \max_y\left\{0, \left\{\left(D\left(1,y\right) - D\left(0,y\right)\right)/2\right\}\right\}, \qquad \text{where } 0 \le y < \text{size\_}y,$$

3. $n_x < H_x$,
4. $\max_y\{D(y,1)\} < 2H_x$, where $0 \le y < H_y$.

Analysis of the DT on the other two sides of the block after the next forward and backward passes is performed in a similar way.

Taking a decision on completion of DT. The number of rows and columns to be processed is calculated after each pass as well as the conditions to complete processing after the current passes are determined. The information about situation at the border of the blocks are transferred to the control processor through the network. The control processor makes a decision on completion or continuation of the distance transform and informs the processors about it. Taking into consideration that, in fact, the number of passes must be no less than three, we can start to collect the information about the DT values after the third pass only.

After making a decision on completion of the DT, in every processor detection of the skeletal points is performed, in doing so the number of the forward and backward passes of procedures for skeletal points detection is exactly the same as the number of the forward and backward ones for the distance transform.

So, execution process of the distance transform can be controlled in a profitable way by analyzing the values this transform at the border of the blocks. It is the principle of distance transform that gives us a tool for the control.

In order to estimate the number of passes needed to process the actual images, the worst case should be studied, namely, a $P_x \times P_y$ rectangle, as the object to be thinned (for the sake of simplicity $P_x < P_y$). As the analysis has shown, if the width of the object is less than $4H_x$ and the object is fitted in two quadruplets (in the width), it can be processed in proper way in three passes of the DT through the image. If the object width is less than $3H_x$, it can be processed also in three passes, even if the object belongs to three quadruplets (in the width). We underline once more that we are dealing with the width of an object not with its length.

Since the algorithm for thinning is worth to apply for images of maps, schemes, engineering drawings, for

which the presence of broad objects is little likelihood, so in the majority of actual cases the number of passes does not exceed three. Moreover, the third pass will most probably be shorter than full, if the number of rows and columns to be processed is less than the block size. The weak dependence of the number of passes on the number of processors for actual images assures efficiency of the developed parallel algorithm: performance of a multiprocessor system is linearly proportional to the number of processors.

Moreover, the developed technique for analysis of the distance transform allows one to define the minimal number of computations for every block and thereby to avoid redundant computing.

For a 4×4 mesh processor network, three passes are enough regardless the size of the objects. For a 2×2 mesh processor network, as it was mentioned above, two passes are enough. The developed algorithm can be easy modified for implementation on a linear array of processors.

Properties of the parallel algorithm. Let us point out the main properties of the developed parallel algorithm for skeletonization for a 2D mesh processor network:

– the load balance is optimal, because every processor gets equal amount of data; the number of operations per processor is approximately the same;

– the communication /computation ratio is good, because the data exchanges are relatively rare; the amount of data for exchange during processing is proportional to the linear size of the image, the amount of computations is proportional to its area;

– the algorithm efficiency is proportional to the number of processors (we neglect the time cost for input/output) due to the good load balance, as well as the weak dependence of the number of passes on the number of processors.

5. Program implementation of the parallel algorithm. To testify the developed parallel algorithm and to create the correspondent software which would be suitable for solving applied image processing problems, this algorithm was implemented for a parallel workstation PARAstation (by Transtech) with four INMOS T800 transputers. The SPARCstation 10 was used as the host computer. The input images were binary ones as TIFF-files.

On loading the input image, first of all a linear size of the image is defined. This information as well as the total number of processors is distributed through the network to prepare the processors to get image blocks. As the image rows are loading from the external memory, they are distributed through the network. In every processor, some rows destined for it are picked out from the flow of rows, the others are transferred further through the network. The loading of the obtained block to the local memory of the processor is performed in a certain order that depends on the number of processor in the quadruplet. This procedure allows one to use the same program for every processor, regardless of the direction of the first pass. In order to avoid deadlock during the data exchange between passes, the sequence of data sending and receiving in each processor is determined by its position in the processors network. After completion of the processing the image blocks are gathered in the control processor and stored in the external memory as a TIFF-file. An input binary image and its skeleton, a result of the processing, are represented in Fig. 6.

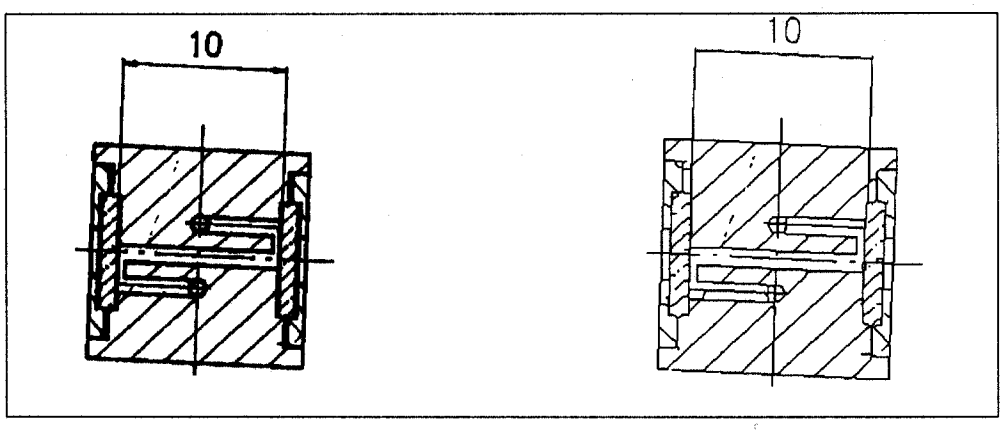

Fig. 6. The binary image (a piece of the drawing) and its skeleton.

Because of the small number of processors at our disposal, the time cost was measured for the ring configuration of the processor network only (it is a linear array with one, two, and four processors). The results of these measurements have shown that as the number of processors increased by a factor of two, the performance increased 1.7–1.8 times. The difference of this coefficient from two is explained, we suppose, by the costs for input/output, data exchange, and other overheads.

CONCLUSION

So, we have developed: a parallel algorithm for distance transform and skeletonization of binary images intended for execution on multiprocessor computers with distributed memory; a processing supervision technique which allows one to process correctly a binary image in minimal time; a method for conversion of the distance transform data to the local width of lines. Checking the algorithm on actual images has demonstrated its validity and efficiency.

The suggested technique can be used for developing parallel algorithms for skeletonization based on the distance transform with different types of distance metrics.

This work has been done at the Department of Computer Science, the University of Joensuu, Finland, thanks to kind help and assistance of Prof. Martti Penttonen and Mr. Juha Oinonen.

REFERENCES

1. M.D.Levie, *Vision in Man and Machine.* McGraw-Hill, New York, ch. 10, 1985.
2. A.D.Darwish and A.K.Jain, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-10, no. 1, pp. 56–68, 1988.
3. T.Kaneko, *Pattern Recognition*, vol.25, no. 9, pp. 963–973, 1992.
4. T.Nihomiya, K.Yoshimura, M.Nomoto, and Y.Nakagawa, *Proc. of the 11th International Conference on Pattern Recognition,* vol. 1, p. 55–56, 1992,
5. R.Kasturi, W.El-Masri, J.Shan, J.R.Gattiker, and U.B.Mokate, *IEEE Trans. on Pattern Recognition and Machine Intelligence,* vol. PAMI-12, no. 10, pp. 978–992, 1990.
6. S.Suzuki and T.Yamada, *Pattern Recognition,* vol. 23, no. 8, pp. 919–933, 1989.
7. J.I.Toriwaki and S.Yokoi, *Progress in Pattern Recognition,* vol. 1, ed. L.N.Kanal and A.Rosenfeld, Amsterdam, North-Holland, 1981, pp. 187–264.
8. C.Arcelli and G.S. de Baja, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-7, no. 4, pp. 463–474, 1985.
9. C.Arcelli and G.S. de Baja, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-13, no. 4, pp. 411–414, 1992.

4 October 1995

Novosibirsk (Russia) – Adelaida (Australia)