*Optica Applicata, Vol. XXXII, No. 1–2, 2002*

# Letters to the Editor

## Fast thinning algorithm ready for real-time application

Jerzy Siuzdak

Institute of Telecommunications, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa, Poland.

A fast thinning algorithm is presented. It is based on the pixel index computation and application of a look-up table. The pixels whose removal does not cause line breaking are iteratively set to zero. The algorithm runs quite fast and requires moderate number of iterations of the entire image.

### 1. Introduction

Thinning is the process of reducing the width of a line-like object from many pixels wide to just a single pixel. It is used to reduce the volume of binary data obtained from the preliminary image processing stages such as the edge detection or thresholding. There are numerous thinning algorithms [1], [2] but they are either relatively slow or require massive parallel processing [3]–[5]. Here, an algorithm is presented which lacks these deficiencies.

### 2. Description of the method

We assume that the input image is binary with ones corresponding to the lines and zeros corresponding to the background. During each step of the algorithm an 8-element neighbourhood (shown in Fig. 1**a**) of the current non-zero pixel is tested. Based on the neighbourhood the decision is taken whether or not the central pixel may be removed (set to zero). Examples of the pixels which should be removed are shown in Fig. 2**a** and examples of pixels which must not be removed are depicted in Fig. 2**b**. The reasons why the pixel must not be removed are simple: either it is a line end or its removal breakes the line connectivity.

| | | |
|---|---|---|
| $p_0$ | $p_1$ | $p_2$ |
| $p_7$ | | $p_3$ |
| $p_6$ | $p_5$ | $p_4$ |

**a**

| | | |
|---|---|---|
| 1 | 2 | 4 |
| 128 | 0 | 8 |
| 64 | 32 | 16 |

**b**

Fig. 1. Notation of the pixels in the 8-element neighbourhood of a pixel (**a**). The mask convolved with (**b**).

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 0 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 0 |
| 1 | 0 | 0 |

a

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

b

Fig. 2. Examples of 8-element neighbourhoods of removable (**a**) and not removable (**b**) pixels.

To avoid comparison with several templates, as done in some other algorithms [1]–[5], we calculate an index of each non-zero pixel convolving its 8-element neighbourhood with a mask shown in Fig. 1 **b**. Following the notation of Fig. 1 we have

$$\text{index} = \sum_{i=0}^{7} p_i 2^i. \tag{1}$$

The index ranges from 0 to 255 and it defines in a unique way the type of the current neighbourhood of the pixel (out of the $2^8$ possible ones). Considering each possibility defined by the index value one can decide on the possible removal of the central pixel. These decisions are taken before processing and are grouped in a look-up table. Thus in each step the value of the central pixel is determined from the look-up table (TABLE) shown in Fig. 3.

$$\text{Central pixel} = \text{TABLE(index)}. \tag{2}$$

This step is iteratively applied to all non-zero pixels in the image until no further deletion is possible. The required number $N$ of entire image iterations is rather small (typically $N = 3, \ldots, 5$) and it depends on the line thickness of the image being processed.

An example of the initial edge image obtained by gradient modulus calculation followed by the comparison with a threshold is given in Fig. 4**a** and processing results are shown in Fig. 4**b**.

We compared the execution time of ours and two other thinning algorithms [4], [5] and it turned out that our algorithm was a few times faster. All the programs were written in high-level language (Delphi) and run on IBM PC. The following typical execution times have been obtained for various $640 \times 480$ pixel edge images:

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 10 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 20 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 30 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 40 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 50 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 60 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 70 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 80 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 90 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 100 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 110 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 120 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 130 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 140 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 150 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 160 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 170 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 180 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 190 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 200 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 210 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 220 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 230 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 240 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 250 | 1 | 1 | 1 | 0 | 1 | 1 | X | X | X | X |

**Fig. 3.** Look-up table TABLE. To get the index value add columns' and rows' indices.

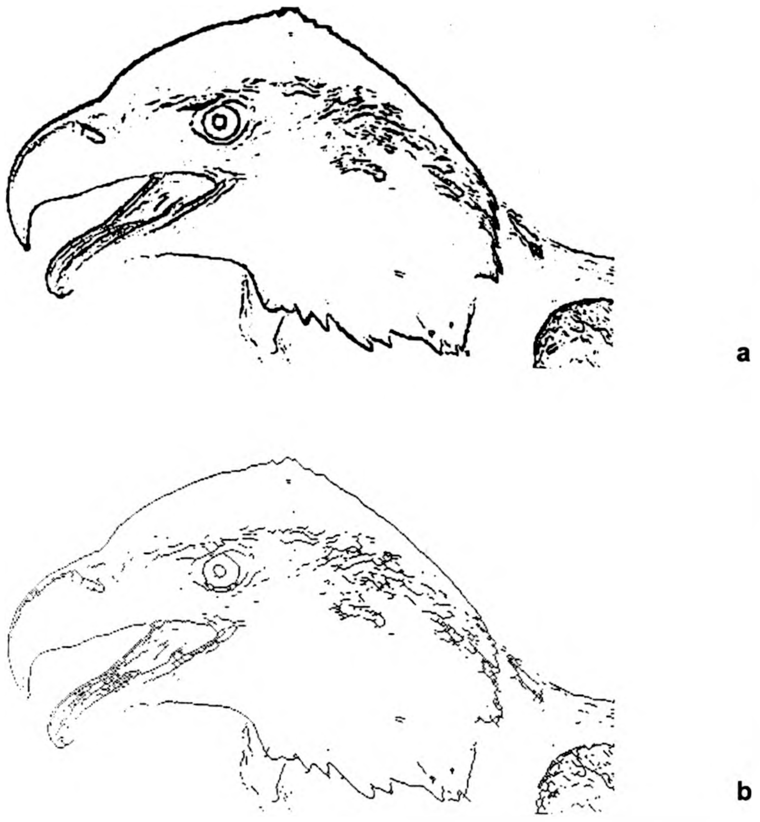

Fig. 4. Initial image (a), processing results ($N = 4$) (b).

method [4] 90 s,
method [5] 60 s,
our method 15 s.
These results show the method potential.

## 3. Conclusions

A fast thinning algorithm is presented. It is based on the pixel index computation and application of the look-up table. It performs faster than other thinning algorithms [4], [5] and requires moderate number of iterations of the entire image.

## References

[1] Naccache N. J., Shinghal R., IEEE Trans. Syst. Man. Cybern. **14** (1984), 409.
[2] Smith R. W., Pattern Recognition **20** (1987), 7.
[3] Celenk M., Choon K. L., Opt. Eng. **30** (1991), 275.
[4] Bowen C., Bagherzadeh N., Computers Electr. Eng. **19** (1993), 351.
[5] Datta A., Parui S. K., Pattern Recognition **27** (1994), 1181.

*Received November 5, 2001*