COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 40, 30–40 (1987)

A One-Pass Thinning Algorithm and Its Parallel Implementation

ROLAND T. CHIN AND HONG-KHOON WAN

Electrical and Computer Engineering Department, University of Wisconsin, Madison, Wisconsin 53706

AND

D. L. STOVER AND R. D. IVERSON

Applied Vision Systems, Inc., Minneapolis, Minnesota 55413

Received April 26, 1986; revised November 14, 1986

This paper describes a one-pass thinning algorithm that requires only a single cycle of parallel operations per iteration. The major difference between this algorithm and the usual multiple-pass 3×3 operators is that it uses two restoring templates (a 1×4 and a 4×1 operator) to deal with the breakage and disappearance of horizontal and vertical two pixel wide limbs. Connectedness of skeletons has been verified and limitations of the algorithm have been discussed. Results of a comparative study are also presented. In addition, a realization of the thinning algorithm is suggested with processing speed limited only to the propagation delays of logic gates. © 1987 Academic Press, Inc.

1. INTRODUCTION

In analyzing the structure of an object in an image, it is often necessary to represent and characterize the object in a compact and approximate representation. For example, edges are used to represent the object boundary and *skeletons* are used as "stick figures" to represent the general shape of objects. Skeletons which provide a simple line drawing representation of the object are useful for topological analysis and classification of the shape. They may be derived by *thinning* algorithms that transform a digital figure to a skeleton of unit width. Thinning may be defined as the successive removal of the outermost layers of an object until only a skeleton remains [2]. It is an expensive procedure when conventional sequential processors are employed. In this paper we describe a parallel, one-pass thinning algorithm for reducing binary objects into skeletons. We also present some results of a comparative study and describe a hardware realization for the algorithm.

Most thinning algorithms delete from an object (say, a connected component), at each *iteration*, boundary pixels whose removal does not disturb the general configuration and connectivity. Some thinning algorithms also guarantee that the final connected skeleton does not change or vanish, even if the iteration process continues. In other words, thinning *converges* to a connected skeleton which is a reasonable topological representation of the object.

Many thinning algorithms (see [2, 3, 5–8, 10, 11] for a few examples) have been proposed to date; and in general, they differ from each other because of the lack of exact definition of skeletons and differences in implementation. However, all thinning algorithms can be classified into two general types: sequential and parallel algorithms. The main difference between these two types is that sequential thinning

0734-189X/87 $3.00

operates on one pixel at a time and the operation depends on preceding processed results, while parallel thinning operates on all the pixels simultaneously. Sequential thinning algorithms generally generate a better skeleton, but parallel thinning is substantially faster.

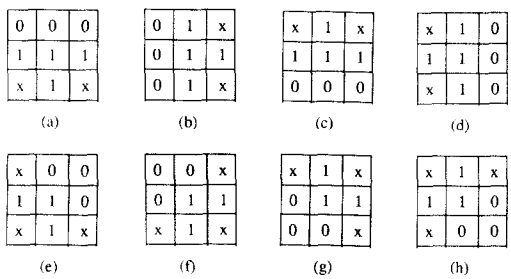
Most existing parallel thinning algorithms use several passes (subcycles) in one thinning iteration. Each pass is a parallel operation removing boundary pixels from a given direction. To complete one iteration, successive passes are applied to remove pixels from all directions. A thinning iteration can also be stated simply as follows: Remove all boundary pixels from the north side of the object; save the intermediate result, and from it remove pixels from the east; save the result, and remove pixels from the south; save the result, and remove pixels from the west. The intermediate result at each pass must be stored for subsequent passes; thus, multipass thinnings are in fact sequential in nature. On the other hand, a one-pass thinning is a true parallel algorithm.

In addition, a one-pass thinning algorithm is very well suited for implementation on a cellular array processor [14], in which a processor unit is assigned to each pixel, so that the thinning operations can be performed at all the pixels simultaneously. In the case of binary images, such an algorithm can be realized using combinational logic. Thus, it is possible to build a device such that, given a binary image of an object as an input to the device, a skeleton of the object will be obtained at the output with processing time limited to the propagation delays of the logic gates.

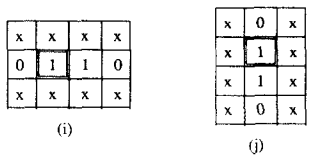
Section 2 describes our one-pass thinning algorithm. Section 3 presents its detailed properties and results of a comparative study. Section 4 describes the

TABLE 1

One-Pass Thinning Algorithm



Thinning Templates



Restoring Templates
(x's are don't cares)

hardware implementation of the algorithm. Section 5 provides a summary of the paper.

2. THE ONE-PASS THINNING ALGORITHM

Ten basic templates are defined in our one-pass thinning algorithm (Table 1). Eight of them are used for deleting boundary pixels, while the other two are used for disabling the deletions of pixels if certain conditions occur. Let the former be known as the "thinning" templates, and the latter as the "restoring" templates.

Templates (a)–(d) are used for removing boundary elements of an object from the north, west, south, and east directions, respectively, while templates (e)–(h) are used for removing corners and diagonal boundary elements from the northeast, northwest, southwest, and southeast direction, respectively. They are designed to be applied to the binary image simultaneously. Any pattern that matches one of the templates (a)–(h) will have its center "1" pixel removed. Since the entire set of thinning templates are applied in parallel, it can easily be seen that objects that are two pixels wide will disappear completely. Much of the difficulty related to this problem is in maintaining connectedness of the skeleton; this difficulty cannot be resolved using the usual 3×3 window operator. To counter this problem, the 1×4 and 4×1 restoring templates are used. Template (i) will restore two pixels wide vertical lines, while template (j) will restore two pixels thick lines. That is, for any pattern that matches templates (i) or (j), its center "1" pixel will be saved regardless of the outcomes of thinning templates (a)–(h).

The thinning algorithm is formally stated below.

(1) *Step* 1. Remove all points in the image that match one of the templates (a)–(h), but leave the points alone that match template (i) or (j).

(2) *Step* 2. Do Step 1 successively until no further changes occur.

This algorithm was applied to several binary images, and their resulting skeletons are shown in Fig. 5. Observations and properties of this algorithm will be presented in the next section.

3. PROPERTIES AND OBSERVATIONS

There exist a number of comparative studies examining various thinning algorithms [1, 2, 4], but no precise standard has yet been established. However, it has been generally accepted that a good skeleton must possesses some necessary properties, such as those concerning topology, shape, connectedness, and sensitivity to boundary noise.

A skeleton must be topologically equivalent to the object; if the object is connected, the skeleton must be connected, it must run along the medial axis of the object (i.e., the thinning algorithm is isotropic), and it must be one pixel thick (see Serra [12], and Davies and Plummer [2] for details). Maintaining connectedness is a basic requirement of all thinning algorithms. If the usual 3×3 thinning templates are used, the algorithm must be a multipass operation in order to avoid the breaking of skeletons in certain situations, thereby increasing the cost or reducing the speed. Fundamentally, the thinning operation of a one-pass algorithm requires larger than 3×3 templates to ensure connectedness.

In some applications, the skeleton is required to run along the medial axis of the object. To satisfy this condition, the thinning operation must delete pixels symmetri-

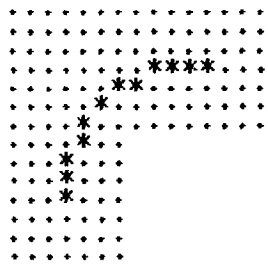


FIG. 1. The bias skeleton. The algorithm produces skeletons which are not medial near sharp corners.

cally to avoid bias in certain directions. Multipass algorithms are suitable for this requirement, but again are more expensive than one-pass algorithms.

The ability to handle boundary noise is another desired property of thinning algorithms. The lack of this capability will lead to the generation of skeletons with unpredictable accuracy.

The one-pass thinning algorithm presented here possesses most of the above mentioned properties. The connectedness of skeletons has been confirmed by an experiment, in which all possible connected components within a 5×5 window were exhaustively generated and used as input patterns to the thinning algorithm. The resulting skeletons were examined topologically, and no breaking of skeletons was found.

However, one major problem observed with this algorithm is that it generates biased skeletons. In some situations, the bias is quite apparent. This is partly due to the fact that the restoring templates are even size templates, and hence the pixel removal process is not symmetrical. Another factor contributing to the bias in the skeleton is a side effect of the thinning templates: they remove pixels from convex corners faster than from concave corners. As shown in the set of eight thinning templates (a)–(h) in Section 2, concave corners are removed by the first four templates (a)–(d), and convex corners by templates (e)–(h); it can easily be demonstrated that the two sets remove pixels near sharp corners at different rates. As shown in Fig. 1, the linear object with a sharp turn produces a skeleton that does not run along the object medial axis near the turn. Instead, it runs along the inside

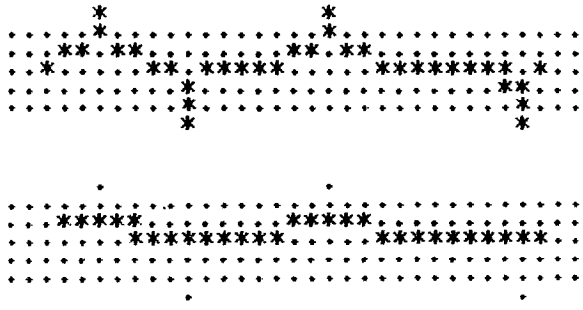
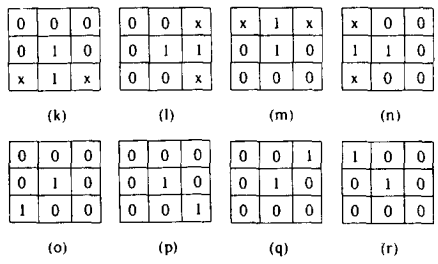


FIG. 2. (a) A noisy skeleton as a result of one pixel boundary noise pixels. (b) A clean skeleton generated by the thinning and the trimming templates.

TABLE 2
Trimming Templates



Trimming Templates
(x's are don't cares)

corner of the turn by "cutting the corner." Clearly, a specially designed set of thinning templates for concave corners can be added to deal with this situation, but the overall operational cost will be increased.

Another problem associated with thinning algorithms is that they will produce a noisy skeleton if the original object contains a noisy boundary. Figure 2a is an example generated by the proposed algorithm. Such effects can be eliminated by a prethinning smoothing procedure to smooth out the noisy boundary of the original object before the application of the thinning. This operation involves an added preprocessing step which is undesirable in some applications with real time constraints. An alternative is to apply a set of noise cleaning (trimming) templates to the binary image at each iteration in parallel with the thinning operation. The objective is to suppress the growth of spurious segments induced by boundary noise by removing noise spurs of unit length. Although this operation increases the total implementation cost, it does not affect the speed of the operation because it can be applied to the image simultaneously with the thinning and restoring templates. A set of trimming templates has been defined for this purpose.

Figure 2b is the resulting skeleton of the same image used in Fig. 2a. In this case the trimming templates were applied together with the thinning and restoring templates to produce an improved skeleton.

However, in using the trimming templates (Table 2), end points of objects are no longer preserved. For example, if the one-pass algorithm (with trimming) is applied to a one pixel wide limb, the object will converge to a single dot because the trimming templates will treat the two end points as noise spurs of unit length. One way to handle this situation is to discriminate between true ends and noise spurs, which is fundamentally impossible for 3×3 templates in the framework of the one-pass algorithm. One compromise for dealing with this problem is to assume knowledge of the approximate size of the object; that is, the number of iterations can be predetermined to terminate the thinning iterations.

Another problem associated with the restoring templates is that diagonal limbs will be reduced to step skeletons rather than true diagonal skeletons. The reason is that the removal of pixels from step skeletons by the thinning templates will be saved by the restoring templates, resulting in convergence. It is to be noted that this

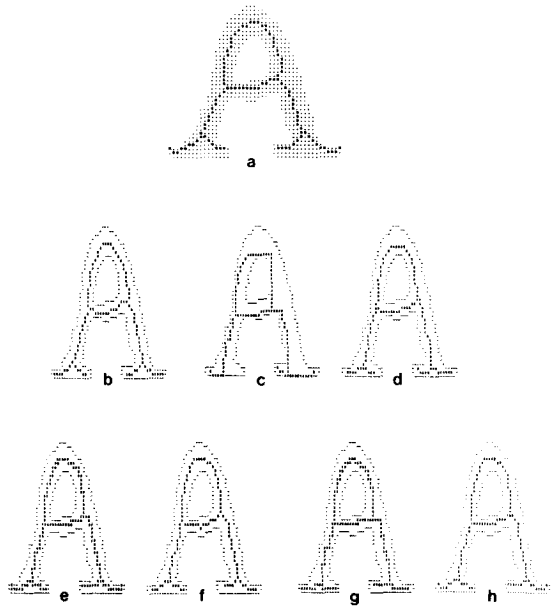


FIG. 3. (a) The proposed one-pass algorithm. (b) Hilditch's algorithm. (c) Deutsch's algorithm 1. (d) Deutsch's algorithm 2. (e) Tamura's 4-curve algorithm. (f) Tamura's 8-curve algorithm. (g) Yokoi's 4-curve algorithm. (h) Yokoi's 8-curve algorithm.

is not a major problem for most applications, since step skeletons are valid one pixel wide connected components.

3.1. Comparison with Other Thinning Algorithms

In this section, results obtained by the proposed algorithm are compared with those from various other thinning algorithms. In one experiment, the binary test pattern (the letter "A") as used in Tamura's study [1] was employed; the result, shown in Fig. 3, compares favorably among the rest.

In another experiment, the "standards" for skeleton precision as proposed by Davies and Plummer [2] were used to compare with our results. Figure 4 presents a number of standard skeletons generated by the Davies and Plummer method. Figure 5 presents the results of our proposed thinning algorithm using the same set of images as in Fig. 4; our results conform closely with those standards.

4. HARDWARE IMPLEMENTATION

The thinning, restoring, and trimming operations used in the proposed algorithm require a vast amount of neighborhood operations per image; therefore, implementation on a general purpose computer is not feasible for most real-time applications. It is desirable for most applications to realize the algorithm as a parallel processing device to achieve the necessary speed. In this study, we assume that inputs are binary images; therefore, the implementation can be realized by combinational logic circuits with processing speed limited only to the propagation delays of the logic gates.

With the assumption that image data are raster scanned and continuously acquired, row by row, into the processing device, a serpentine memory structure is

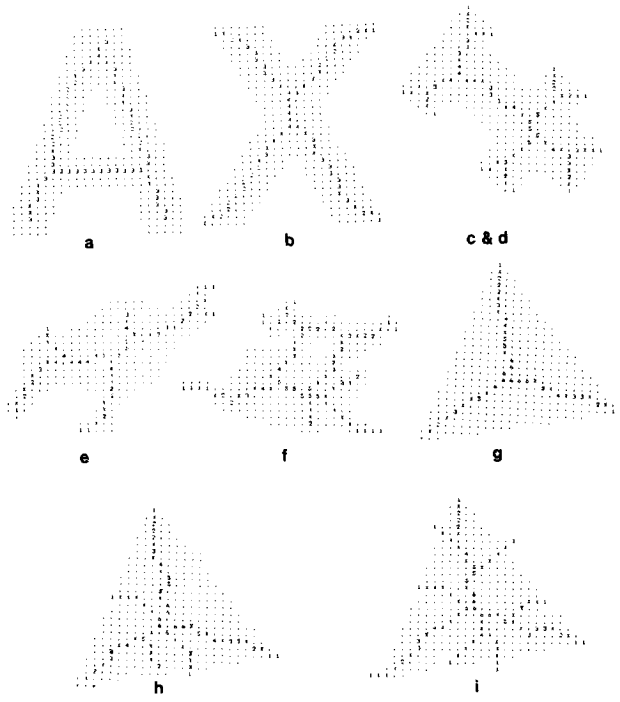


FIG. 4. (a)–(i) Thinning results obtained from Davies and Plummer [2].

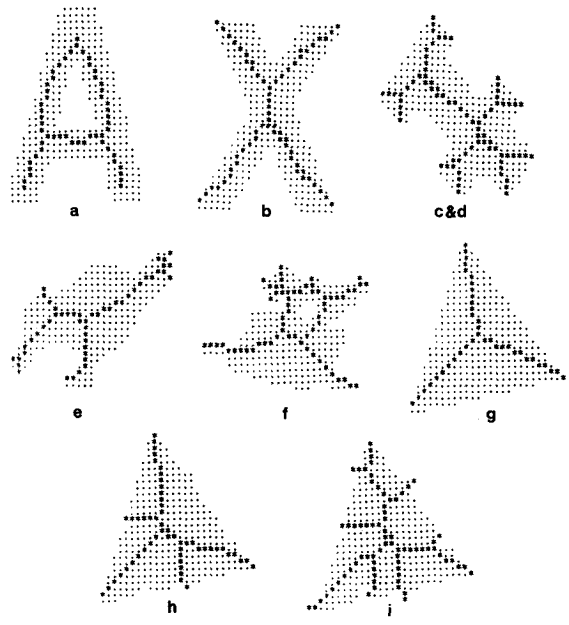


FIG. 5. (a)–(i) Thinning results of the same images used in Fig. 4a–i, obtained by the proposed one-pass algorithm.

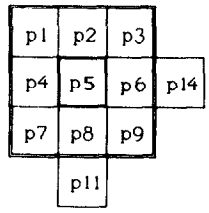


FIG. 6. Neighborhood configuration of the required local operations.

used to acquire local neighbors for the neighborhood operations [13]. The one-dimensional binary data are streamlined and properly registered into storage elements (shift registers) by video delay lines forming a two-dimensional $n \times m$ local window. The outputs of the window are fed into a combinational logic circuit for the thinning operation. Since the algorithm is a one-pass operation, no subsequent pass is necessary; hence no intermediate storage is required. This enables the realization of a cost effective and high speed processing device. Furthermore, a sequence of these processing devices, one for each iteration, is pipelined together for the necessary number of thinning iterations. The throughput of the entire processing pipeline is limited only by the image acquisition rate.

Since the restoring templates consist of a 4×1 and a 1×4 window and all other templates are based on a 3×3 window, a 4×4 memory structure is used to generate the local neighborhood. The pixels in the neighborhood are numbered from P_1 to P_{14} according to the configuration shown in Fig. 6. The processing cell that performs the thinning operation is composed of two sections, the local neighbor-

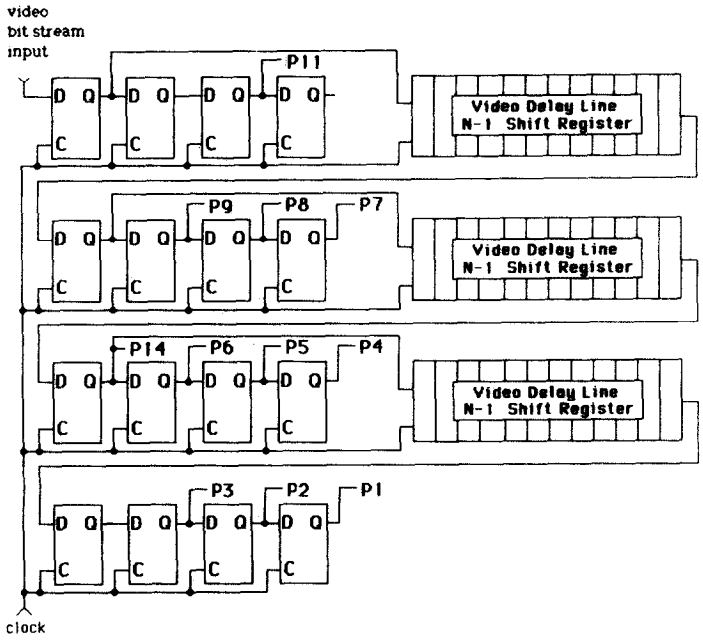


FIG. 7. Memory structure of the 4×4 neighborhood.

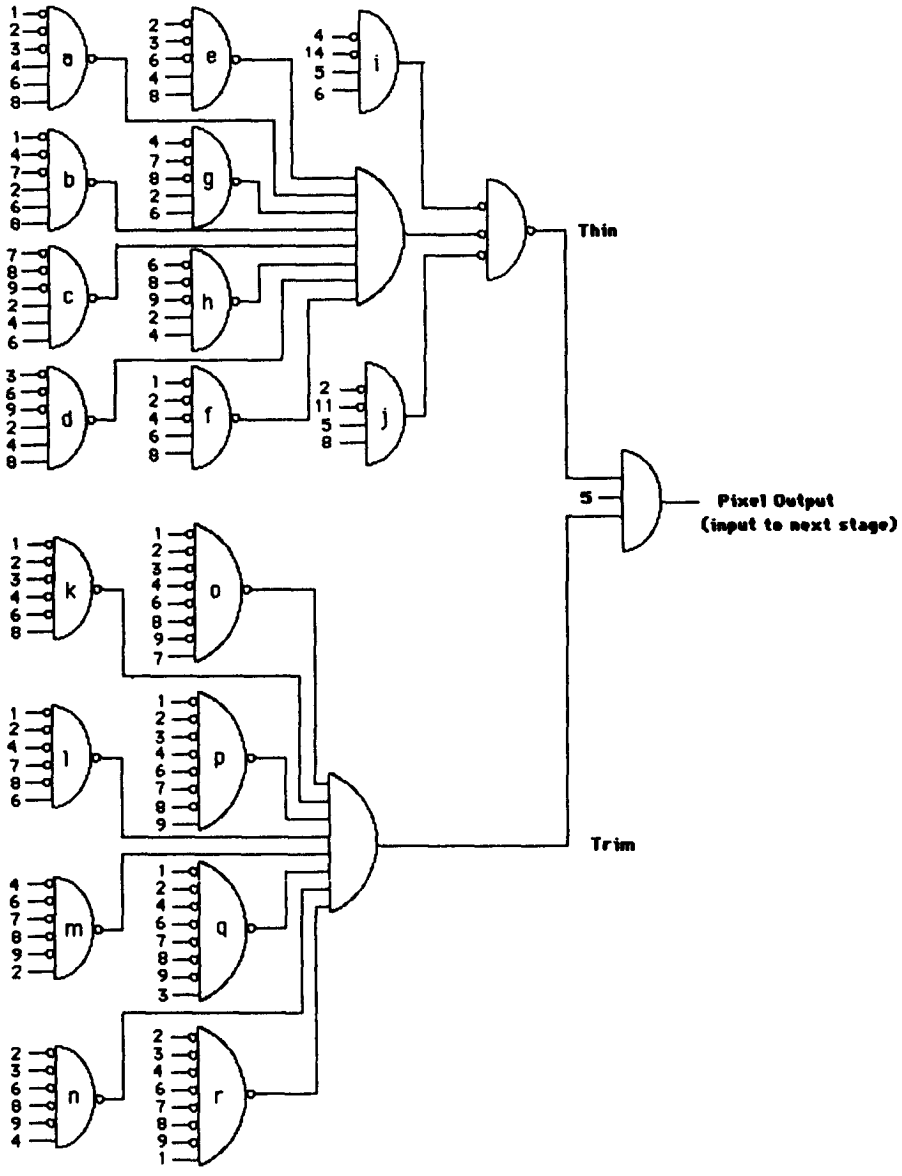


FIG. 8. The realization of the thinning, restoring, and trimming operations.

hood generator and the combinational logic used to implement the templates. The serpentine memory structure for generating the local neighborhood was designed and built as shown in Fig. 7, utilizing D flip-flops and shift registers as video delay lines. Clocking is controlled by the image bit transfer rate and the length of the delay line is determined by the resolution of the image raster scan line. The outputs of the flip-flops of the 4×4 window are connected to the combinational logic as shown in Fig. 8 for the thinning, restoring, and trimming operations. The gates in Fig. 8 are labeled according to the templates as defined in Section 2.

The output of the combinational logic is a video bit stream that represents the result of one iteration of thinning on the input video bit stream. This output data is

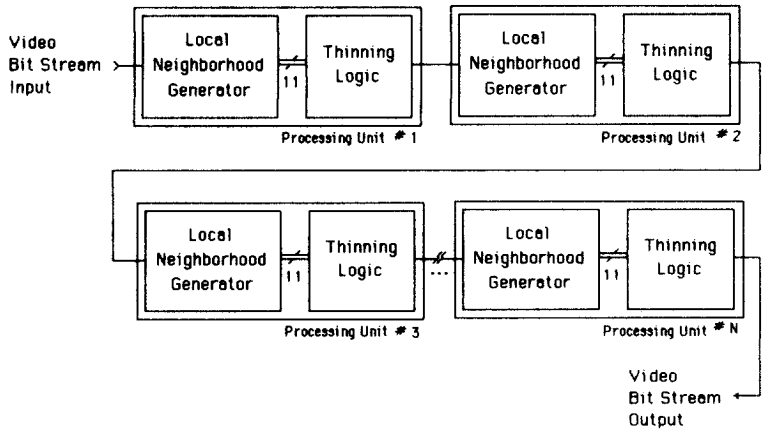


FIG. 9. A pipe containing processing devices for thinning iterations.

delayed by one scan line and two pixels from the input data. Multiple iterations of thinning can be accomplished by cascading stages of thinning processors together as shown in Fig. 9. For example, if 100 pixel widths is the maximum feature size to be thinned, 50 processors (iterations) could be cascaded together to achieve complete thinning. An example of a thinned printed circuit pattern is shown in Fig. 10.

Assuming that the processing device is to process a continuous imagery data stream at real time, the required data rate for this application is 512×512 pixels/frame $\times$ 30 frames/second = 7.86 M pixels/second (a 127 nanoseconds

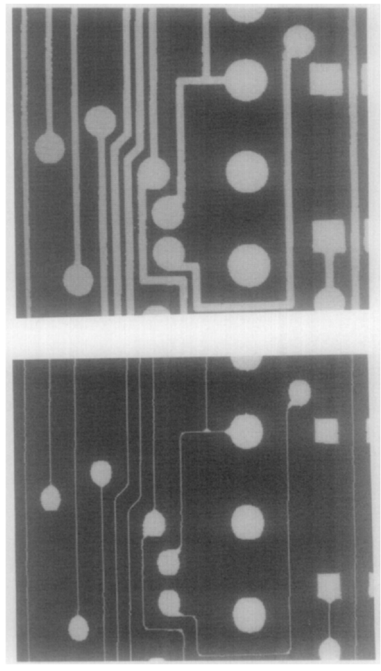


FIG. 10. (a) A printed circuit board image. (b) The resulting skeletons.

pixel period). Since the combinational logic must propagate an output within one pixel period a CMOS gate array was chosen for the realization of the thinning algorithm. The final device was built on a two micron feature-width gate array. The measured propagation delay for neighborhood set-up and combinational logic was 37 ns, giving a usable frequency of about 27 MHz, easily exceeding the above stated design specification.

5. CONCLUSION

This paper describes a one-pass thinning algorithm, its properties and limitations. The connectedness of the generated skeletons has been verified and the boundary noise problem has been dealt with. Furthermore, results generated by this algorithm have been found to be comparable with those by various other methods. On the other hand, the algorithm produces skeletons which are not medial near sharp corners, and it shortens one pixel wide limbs. The major advantage of the presented algorithm is that it requires only a single pass of parallel operations per iteration, enabling us to design and build a cost effective and real-time thinning device which is readily adaptable to most real-time applications. The realization of such a device was briefly described. The device has been tested and applied to printed circuit patterns and a processing speed in the nanosecond range has been achieved.

ACKNOWLEDGMENTS

We wish to thank National Science Foundation (Grant ECS-8352356) and General Motors Foundation, Inc., Warren, Michigan for the support of the research.

REFERENCES

1. H. Tamura, A comparison of line thinning algorithms from digital geometry viewpoint, in *Int. Joint Conf. Pattern Recognition*, 1978, pp. 715–719.
2. E. R. Davies and A. P. N. Plummer, Thinning algorithms: A critique and a new methodology, *Pattern Recognit.* **14**, 1981, 53–63.
3. R. Stefanelli and A. Rosenfeld, Some parallel thinning algorithms for digital pictures, *J. Assoc. Comput. Mach.* **18**, 1971, 255–264.
4. C. J. Hilditch, Comparison of thinning algorithms on a parallel processor, *Image Vision Comput.* **1**, 1983, 115–132.
5. C. Arcelli, L. Cordella, and S. Levialdi, Parallel thinning of binary pictures, *Electron. Lett.* **11**, 1975, 148–149.
6. T. Pavlidis, A thinning algorithm for discrete binary images, *Comput. Graphics Image Process.* **13**, 1980, 142–157.
7. E. S. Deutsch, Thinning algorithms on rectangular, hexagonal, and triangular arrays, *Commun. ACM* **15**, No. 9, 1972, 827–837.
8. D. Rutovitz, Pattern recognition, *J. R. Statist. Soc. Ser. A*, **129**, 1966, 504–530.
9. S. Yokoi, An analysis of topological properties of digitized binary pictures using local features, *Comput. Graphics Image Process.* **4**, 1975, 63–73.
10. C. J. Hilditch, Linear skeletons from square cupboards, in *Machine Intelligence IV* (B. Mertzer and D. Michie, Eds.,), pp. 403–420, University Press, Edinburgh, 1969.
11. U. Montanari, Continuous skeletons from digital images, *J. Assoc. Comput. Mach.* **16**, 1969, 534–549.
12. J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.
13. S. R. Sternberg, Biomedical image processing, *Computer* **16**, No. 1, 1983, 22–35.
14. Special issue in computer architectures for image processing, *Computer* **16**, No. 1, 1983.