

A NEW PARALLEL THINNING METHODOLOGY

Y. Y. ZHANG and P. S. P. WANG

*College of Computer Science, Northeastern University
Boston, MA 02115, USA*

A perfectly parallel thinning algorithm (PPTA) is proposed. It can generate perfect skeletons, which consist of end points, break points, and hole points only. Experimental results show that the proposed PPTA can also preserve image connectivity, produce thinner skeletons, and is faster than many existing thinning algorithms. For example, it is twice as fast as one of the fastest parallel thinning algorithm by Holt, Stewart, Clint and Perrorte.

Keywords: Parallel thinning, perfect skeleton, image processing, pattern recognition.

1. INTRODUCTION

Vectorization is a full function in a raster-to-vector drawing conversion system, which converts a broad range of paper drawings into points, lines, arcs, and circles ready for input directly to a CAD database. There are two steps for vectorization, namely: (a) Thinning, which transfers raster drawings to skeletons; (b) Vectorizing, which converts raster data (skeleton) to CAD data (vectors). An example of a vectorizer is shown in Appendix A in which (A) is an original drawing; (B) is the skeleton; (C) is the CAD drawing with vectors.

It is well-known that “thinning” plays a very important role in image processing, pattern recognition, and for the quality of the vectorizer. For a comprehensive survey, see Ref. 4, and for most recent developments, see Refs. 8 and 9. However, one normally cannot directly use some existing thinning algorithms for vectorizing because there may remain redundant pixels in resulting skeletons. Therefore, one needs either to redesign the algorithm, or to do some additional work to remove the redundant pixels after thinning. It means that one should use perfectly 8-connected curve algorithms.⁷ Unfortunately, however, according to Ref. 1, “up to now, no 1-subcycle/iteration parallel thinning algorithms have been successfully developed which can produce the perfectly 8-connected curve”. In this paper we attempt to solve this open problem by proposing algorithm PPTA to produce perfectly 8-connected curves.

2. DEFINITIONS AND NOTATIONS

A binary digitized drawing can be defined as a matrix Q , where each element, $q[i, j]$, is either 1 (dark point) or 0 (white point) and these points are pixels. Here we assume that the drawings consist of these elements which have value 1.

Definition 2.1. 8-neighbors of a pixel p are identified by the eight directions shown in Fig. 1. The four pixels, $p[0]$, $p[2]$, $p[4]$ and $p[6]$ (i.e. north(p), east(p),

south(p), and west(p)), are called the direct neighbors. The four pixels, $p[1]$, $p[3]$, $p[5]$, and $p[7]$ (i.e. north-east(p), south-east(p), south-west(p), and north-west(p)), are called the indirect neighbors.

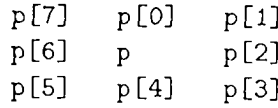


Fig. 1. Pixel p and its neighbors.

Definition 2.2. Neighbor number of p , $NN(p)$, is the number of nonzero neighbors of current pixel p :

$$NN(p) = \sum_{k=0}^7 p[k]$$

Definition 2.3. Weight number of p , $WN(p)$, is defined as follows:

$$WN(p) = \sum_{k=0}^7 p[k] * 2^k$$

$M = WN(Q)$ means $m[i, j] = WN(q[i, j])$ for all i and j . For $3*3$ window the weight numbers are between 0 and 255 as shown in Appendix B. A set which consists of all weight numbers is called $WS = \{0, 1, \dots, 255\}$.

Definition 2.4. Connection number of p , $CN(p)$, is defined as follows:

$$CN(p) = \sum_{k=0,2,4,6} \overline{p[k]} * (p[k+1] \vee p[k+2])$$

where $\overline{p[k]}$ means $1 - p[k]$, $p[8] = p[0]$. From [6], we have

- $CN(p) = 0$ p is the interior (hole) or isolated point
- $= 1$ p is the edge point
- $= 2$ p is the connecting point
- $= 3$ p is the branching point
- $= 4$ p is the crossing point

Definition 2.5. End point set $ES = \{WN(p) \mid p \text{ in } Q \text{ and } NN(p) = 1\}$.

$$ES = \{1, 2, 4, 8, 16, 32, 64, 128\}$$

Definition 2.6. Break point set $BS = \{WN(p) \mid p \text{ in } Q \text{ and } CN(p) > 1\}$. From Appendix B,

$$BS = BS2 + BS3 + BS4$$

$$BS2 = \{ WN(p) \mid p \text{ in } Q \text{ and } CN(p) = 2 \}$$

$$= \{ 9, 10, 11, 17, 18, 19, 25, 26, 27, 33, 34, 35, 36, 37, 38, 39, 40, 44, 45, 46, 47, 49, 50, 51, 57, 58, 59, 66, 68, 70, 72, 73, 75, 76, 78, 82, 90, 98, 100, 102, 104, 105, 107, 108, 110, 114, 122, 130, 132, 134, 136, 137, 139, 140, 142, 144, 145, 147, 148, 150, 152, 153, 155, 156, 158, 160, 161, 163, 165, 167, 173, 175, 176, 177, 179, 180, 182, 184, 185, 187, 188, 190, 194, 196, 198, 200, 201, 203, 204, 206, 210, 218, 226, 228, 230, 232, 233, 235, 236, 238, 242, 250 \}$$

$$BS3 = \{ WN(p) \mid p \text{ in } Q \text{ and } CN(p) = 3 \}$$

$$= \{ 41, 42, 43, 74, 106, 138, 146, 154, 162, 164, 166, 168, 169, 171, 172, 174, 178, 186, 202, 234 \}$$

$$BS4 = \{ WN(p) \mid p \text{ in } Q \text{ and } CN(p) = 4 \}$$

$$= \{ 170 \}$$

Definition 2.7. Hole point set $HS = \{ WN(p) \mid p \text{ in } Q \text{ and } CN(p) = 0 \}$.

$$HS = \{ 0, 85, 87, 93, 95, 117, 119, 125, 127, 213, 215, 221, 223, 245, 247, 253, 255 \}$$

Definition 2.8. Removal point set $RS = \{ WN(p) \mid p \text{ in } Q \text{ and } CN(p) = 1 \text{ and } NN(p) > 1 \}$

$$RS = WS - ES - BS - HS$$

$$= \{ 3, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23, 24, 28, 29, 30, 31, 48, 52, 53, 54, 55, 56, 60, 61, 62, 63, 65, 67, 69, 71, 77, 79, 80, 81, 83, 84, 86, 88, 89, 91, 92, 94, 96, 97, 99, 101, 103, 109, 111, 112, 113, 115, 116, 118, 120, 121, 123, 124, 126, 129, 131, 133, 135, 141, 143, 149, 151, 157, 159, 181, 183, 189, 191, 192, 193, 195, 197, 199, 205, 207, 208, 209, 211, 212, 214, 216, 217, 219, 220, 222, 224, 225, 227, 229, 231, 237, 239, 240, 241, 243, 244, 246, 248, 249, 251, 252, 254 \}$$

Definition 2.9. Unremovable point set $URS = ES + BS + HS = WS - RS$.

Definition 2.10. Skeleton SS is a set of black points which cannot be deleted by some thinning algorithms.

Definition 2.11. Perfect skeleton is a skeleton which consists of end point, break point, and hole point only. The perfect skeleton defined here is an irreducible set (sometimes also referred to as the perfect 8-connected curve, see e.g. Ref. 7). A thinning algorithm is perfect if it transforms patterns or images to perfect skeletons.

3. PPTA: A PERFECTLY AND FULLY PARALLEL THINNING ALGORITHM

During the past 20 years, researchers designed a lot of thinning algorithms. But “fully parallel thinning algorithms can have difficulty preserving the connectivity of an image.”² In this section we try to design a perfectly and fully parallel thinning algorithm and to prove that it preserves the connectivity of the input pattern or image, and produces the perfectly 8-connected curve (perfect skeleton).

Rosenfeld proved that a parallel thinning algorithm preserves the connectivity if only simple points are removed in iterations.^{5,6} A pixel is called *simple* if changing it from “1” to “0” does not change the connectivity of 1s in its neighborhood. Based on these concepts, the thinning algorithm can be described as follows: The parallel algorithm removes only one type of border point – north, south, east, or west at a time.^{2,7} This is 4-subcycle/iteration algorithm which is certainly not very efficient. However, the authors did not give an actual approach to detect simple points for 2-subcycle/iteration and fully parallel thinning algorithms (later on we will give an algorithm to detect simple points for fully parallel thinning).

A major problem to detect the simple point for fully parallel algorithm is collision. A *collision* occurs if there are simple points in the neighbors of simple point p such that p is no longer simple if the simple neighbor points are removed in the same iteration. For example in Fig. 2, p is not simple when $p[0]$ is removed.

| | | |
|------|------|------|
| p[7] | p[0] | p[1] |
| p[6] | p | p[2] |
| 0 | 0 | 0 |

Fig. 2. A collision example.

An algorithm “SimplePoint” which can detect the simple points for avoiding collisions is fully parallel iteration is shown in Fig. 3.

```

const WNset:array[0..255] of integer = (
{000} 0,0,0,2,0,6,1,8,0,0, 0,0,1,5,1,5,0,0,0,0, 1,5,1,5,0,0,0,0,1,5,
{030} 1,5,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,1,0, 0,0,1,5,1,5,1,0,0,0,
{060} 1,5,1,5,0,2,0,2,0,2, 0,2,0,0,0,0,0,6,0,6, 1,3,0,4,1,0,1,0,1,3,
{090} 0,4,1,0,1,0,1,3,0,4, 0,4,0,4,0,0,0,0,0,7, 0,7,1,3,0,4,1,0,1,0,
{120} 1,3,0,4,1,0,1,0,0,2, 0,2,0,2,0,2,0,0,0,0, 0,6,0,6,0,0,0,0,0,6,
{150} 0,6,0,0,0,0,0,6,0,6, 0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,
{180} 0,6,0,6,0,0,0,0,0,6, 0,6,1,1,0,2,0,2,0,2, 0,0,0,0,0,6,0,6,1,1,
{210} 0,2,1,0,1,0,1,1,0,2, 1,0,1,0,1,1,0,2,0,2, 0,2,0,0,0,0,0,6,0,6,
{240} 1,1,0,2,1,0,1,0,1,1, 0,2,1,0,1,0);

```

```

function SimplePoint(p:integer):BOOLEAN;
begin
  WNpoint:= WN(p);          WNorth:= WN(north(p));
  WNeast:=  WN(east(p));    WWest:=  WN(west(p));
  WNortheast:= WN(north-east(p));
  SimplePoint:=FALSE;
  case WNset[WNpoint] of
    1:SimplePoint:=TRUE;
    2:if (WNset[WNorth]=0) then SimplePoint:=TRUE;
    3:if (WNset[WWest] =0) then SimplePoint:=TRUE;
    4:if (WNset[WNorth]=0 and WWest=0) then SimplePoint:=TRUE;
    5:if (WNset[WNeast] =0) then SimplePoint:=TRUE;
    6:if (WNset[WNorth]=0 and WNset[WNeast]=0) then SimplePoint:=TRUE;
    7:if (WNset[WNorth]=0 and WNset[WNeast]=0 and WNset[WWest]=0)
        then SimplePoint:=TRUE;
    8:if (WNset[WNortheast]=0) then SimplePoint:=TRUE;
  end; {case}
end;

```

Fig. 3. A simple point detection algorithm.

Now based upon the above observations, we are ready to design a perfectly and fully parallel thinning algorithm PPTA as follows:

```

Procedure PPTA;
begin
  repeat
    NothingChanged := TRUE;
    for i:=1 to i0 do for j:=1 to j0 do
      if (q[i,j]>0) and SimplePoint(q[i,j]) then
        begin q[i,j]:=0; NothingChanged:= FALSE; end;
    until NothingChanged;
  end;

```

Fig. 4. PPTA: Perfectly and fully parallel thinning algorithm.

Note that in the above algorithm, $Q(q[i, j])$, where $i = 1 \dots i_0$; $j = 1 \dots j_0$) is the original pattern or image.

From the above observations, we can show two key properties of PPTA algorithm in the following theorems, namely *PPTA preserves the connectivity and produces perfect skeletons from input images or patterns.*

Theorem 1. PPTA preserves the connectivity.

Proof. To design an algorithm for detecting all simple points without collision in fully parallel iteration, we can divide removable point set RS into the following four subsets, based on which we can prove this theorem in four cases:

$$\begin{aligned}
 \text{NorthSet} &= \{ p \mid p \text{ in } RS \text{ and } p[0] = 0 \} \\
 \text{EastSet} &= \{ p \mid p \text{ in } RS \text{ and } p[0]=1 \text{ and } p[2]=0 \} \\
 \text{SouthSet} &= \{ p \mid p \text{ in } RS \text{ and } p[0]=1 \text{ and } p[2]=1 \text{ and } p[4]=0 \} \\
 \text{WestSet} &= \{ p \mid p \text{ in } RS \text{ and } p[0]=1 \text{ and } p[2]=1 \text{ and } p[4]=1 \text{ and } p[6]=0 \} \\
 RS &= \text{NorthSet} + \text{EastSet} + \text{SouthSet} + \text{WestSet}
 \end{aligned}$$

For example, in Fig. 5, n_1, \dots, n_4 are in the NorthSet; e_1, \dots, e_4 are in the EastSet; s_1, \dots, s_6 are in the SouthSet; and w_1 and w_2 are in the WestSet.

| | | | | | | | | | | | | | | | | | | | |
|--------------|--|------|--|------|--|------|--|------|--|------|--|------|--|------|--|------|--|-----|--|
| p | | n1 | | n2 | | n3 | | n4 | | e1 | | e2 | | e3 | | e4 | | | |
| ----- | | 0 0 | | 1 0 | | 0 0 | | 1 0 | | 0 1 | | 0 1 | | 1 0 | | 0 1 | | 1 1 | |
| Window | | 0 n1 | | 1 n2 | | 1 n3 | | 0 n4 | | 0 e1 | | 0 e2 | | 0 e3 | | 0 e4 | | 0 | |
| | | 0 1 | | 1 1 | | 1 0 | | 0 1 | | 0 1 | | 0 0 | | 0 1 | | 0 1 | | 1 0 | |
| ----- | | 30 | | 124 | | 192 | | 240 | | 209 | | 67 | | 97 | | 115 | | | |
| WNset[WN(p)] | | 1 | | 1 | | 1 | | 1 | | 1 | | 2 | | 3 | | 4 | | | |

| | | | | | | | | | | | | | | | | | | | |
|--------------|--|------|--|------|--|------|--|------|--|------|--|------|--|------|--|------|--|-----|--|
| p | | s1 | | s2 | | s3 | | s4 | | s5 | | s6 | | w1 | | w2 | | | |
| ----- | | 0 1 | | 1 0 | | 1 0 | | 0 1 | | 0 1 | | 0 1 | | 0 1 | | 1 1 | | 1 1 | |
| Window | | 1 s1 | | 1 s2 | | 1 s3 | | 1 s4 | | 1 s5 | | 1 s6 | | 1 w1 | | 1 w2 | | 1 | |
| | | 0 0 | | 0 0 | | 1 0 | | 1 0 | | 1 0 | | 1 0 | | 0 1 | | 1 1 | | 0 1 | |
| ----- | | 71 | | 15 | | 77 | | 101 | | 109 | | 7 | | 55 | | 181 | | | |
| WNset[WN(p)] | | 2 | | 5 | | 6 | | 4 | | 7 | | 8 | | 5 | | 6 | | | |

Fig. 5. Sample pixels and their window numbers.

We will detect all simple points for each subset as follows.

Case 1. If all simple points are from NorthSet, then there are no collisions in parallel iteration⁷ and all points in NorthSet are simply called north-0.

$$\text{north-0} = \{ 6, 12, 14, 20, 22, 24, 28, 30, 48, 52, 54, 56, 60, 62, 80, 84, 86, 88, 92, 94, 96, 112, 116, 118, 120, 124, 126, 192, 208, 212, 214, 216, 220, 222, 224, 240, 244, 246, 248, 252, 254 \}$$

$$\text{NorthSet} = \text{north-0} (41 \text{ points})$$

Case 2. If all simple points are from NorthSet and EastSet, some collisions occur between point p (from EastSet) and neighbors of p (from NorthSet). As an example, $e2$ in Fig. 5 is not simple if $north(e2)$ is simple. $e2$ is simple if $north(e2)$ is in URS (see Definition 2.9). We give a simple point condition for that kind of points:

$$\text{east-}n = \{ p \mid p \text{ in EastSet and } p[0] \text{ in URS} \}$$

$\text{east-}n$ is a set of simple points in which p is in EastSet and $north(p)$ is in URS. There are 11 points (see below) in $\text{north-}0$. $e2$ (WN)($e2$) = 67. $WNset(WN(e2)) = 2$ is one of them. Similarly, we can detect all other simple points in $\text{east-}0$, $\text{east-}w$ and $\text{east-}n-w$ as follows:

$$\begin{aligned} \text{(a) east-}0 &= \{ p \mid p \text{ in EastSet} \} \\ &= \{ 193, 209, 217, 225, 241, 249 \} \\ \text{(b) east-}n &= \{ p \mid p \text{ in EastSet and } p[0] \text{ in URS} \} \\ &= \{ 3, 65, 67, 129, 131, 195, 211, 219, 227, 243, 251 \} \\ \text{(c) east-}w &= \{ p \mid p \text{ in EastSet and } p[6] \text{ in URS} \} \\ &= \{ 81, 89, 97, 113, 121 \} \\ \text{(d) east-}n-w &= \{ p \mid p \text{ in EastSet and } p[0] \text{ in URS and } p[6] \text{ in URS} \} \\ &= \{ 83, 91, 99, 115, 123 \} \\ \text{EastSet} &= \text{east-}0 + \text{east-}n + \text{east-}w + \text{east-}n-w \text{ (27 points)} \end{aligned}$$

Case 3. If all simple points are from RS except WestSet, then there are collisions between p (from SouthSet) and the neighbors of p (from NorthSet and EastSet). We detect all simple points in SouthSet as follows:

$$\begin{aligned} \text{(a) south-}n &= \{ p \mid p \text{ in EastSet and } p[0] \text{ in URS} \} \\ &= \{ 69, 71, 133, 135, 197, 199, 229, 231 \} \\ \text{(b) south-}e &= \{ p \mid p \text{ in EastSet and } p[2] \text{ in URS} \} \\ &= \{ 13, 15 \} \\ \text{(c) south-}n-e &= \{ p \mid p \text{ in EastSet and } p[0] \text{ in URS and } p[2] \text{ in URS} \} \\ &= \{ 5, 77, 79, 141, 143, 205, 207, 237, 239 \} \\ \text{(d) south-}n-w &= \{ p \mid p \text{ in EastSet and } p[0] \text{ in URS and } p[6] \text{ in URS} \} \\ &= \{ 101, 103 \} \\ \text{(e) south-}n-e-w &= \{ p \mid p \text{ in EastSet and } p[0] \text{ in URS and } p[2] \text{ in URS} \\ &\quad \text{and } p[6] \text{ in URS} \} \\ &= \{ 109, 111 \} \\ \text{(f) south-}n-e &= \{ p \mid p \text{ in EastSet and } p[1] \text{ in URS} \} \\ &= \{ 7 \} \\ \text{SouthSet} &= \text{south-}n + \text{south-}e + \text{south-}n-e + \text{south-}n-w + \\ &\quad \text{south-}n-e-w + \text{south-}n-e \text{ (24 points)} \end{aligned}$$

Case 4. If all simple points are from RS, then all simple points in WestSet are $\text{west-}e$ and $\text{west-}n-e$.

- (a) $\text{west-}e = \{ p \mid p \text{ in WestSet and } p[2] \text{ in URS} \}$
 $= \{ 21, 23, 29, 31, 53, 55, 61, 63 \}$
- (b) $\text{west-}n\text{-}e = \{ p \mid p \text{ in WestSet and } p[0] \text{ in URS and } p[2] \text{ in URS} \}$
 $= \{ 149, 151, 157, 159, 181, 183, 189, 191 \}$
- $\text{WestSet} = \text{west-}e + \text{west-}n\text{-}e$ (16 points)

Combining these four cases, we get an algorithm to detect all simple points in fully parallel iteration as shown in Fig. 3, where $WNset[WN(p)]$ is:

- $WNset[WN(p)] = 0$ p is an unremovable point
 $= 1$ p is a simple point
 $= 2$ p is a simple point if $p[0]$ in URS
 $= 3$ p is a simple point if $p[6]$ in URS
 $= 4$ p is a simple point if $p[0]$ and $p[6]$ are in URS
 $= 5$ p is a simple point if $p[2]$ in URS
 $= 6$ p is a simple point if $p[0]$ and $p[2]$ are in URS
 $= 7$ p is a simple point if $p[0]$ and $p[2]$ and $p[6]$ are in URS
 $= 8$ p is a simple point if $p[1]$ in URS

Theorem 2. PPTA algorithm produces perfect skeleton.

Proof. To prove a perfect skeleton we only make sure that all points in removable point set RS are deleted when the iteration is terminated. PPTA does that because we remove all possible simple points of RS in each parallel iteration.

4. COMPARISONS OF THINNING ALGORITHMS

To compare the parallel speed, we code algorithm PPTA and HSCP³ in PASCAL language, and run on an IBM PC computer. The tested results of four examples (Appendix C) are shown in Fig. 6. These experimental results show that the proposed algorithm is about twice as fast as an HSCP algorithm.

| | PPTA Algorithm | HSCP Algorithm |
|-------------|----------------|----------------|
| Leaf | t1 | 2.5 * t1 |
| Body | t2 | 2.0 * t2 |
| Character A | t3 | 2.4 * t3 |
| Character B | t4 | 3.3 * t4 |

Fig. 6. The comparisons of PPTA and HSCP.

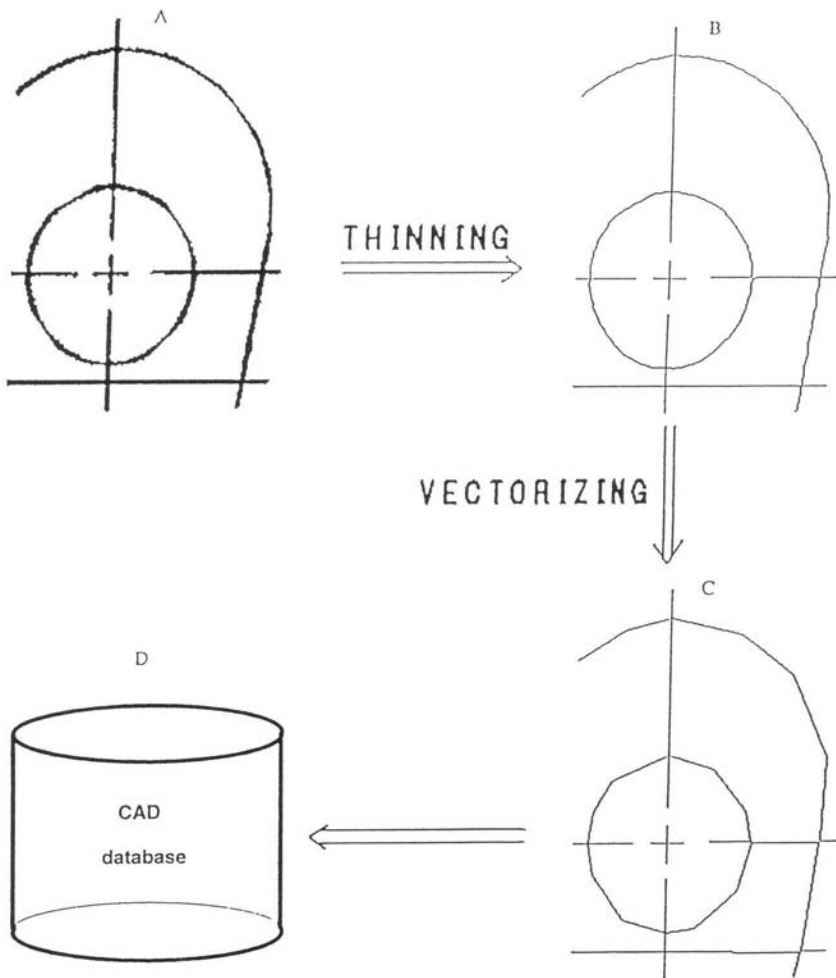
5. CONCLUSIONS

We propose a fast and fully parallel thinning algorithm (PPTA) which preserves the connectivity of patterns and produces thinned and perfect skeletons. Using the same idea, it is easy to design perfectly 2 subcycle/iteration and 4 subcycle/iteration thinning algorithms.

ACKNOWLEDGEMENTS

The authors are grateful for the support from the College of Computer Science of Northeastern University.

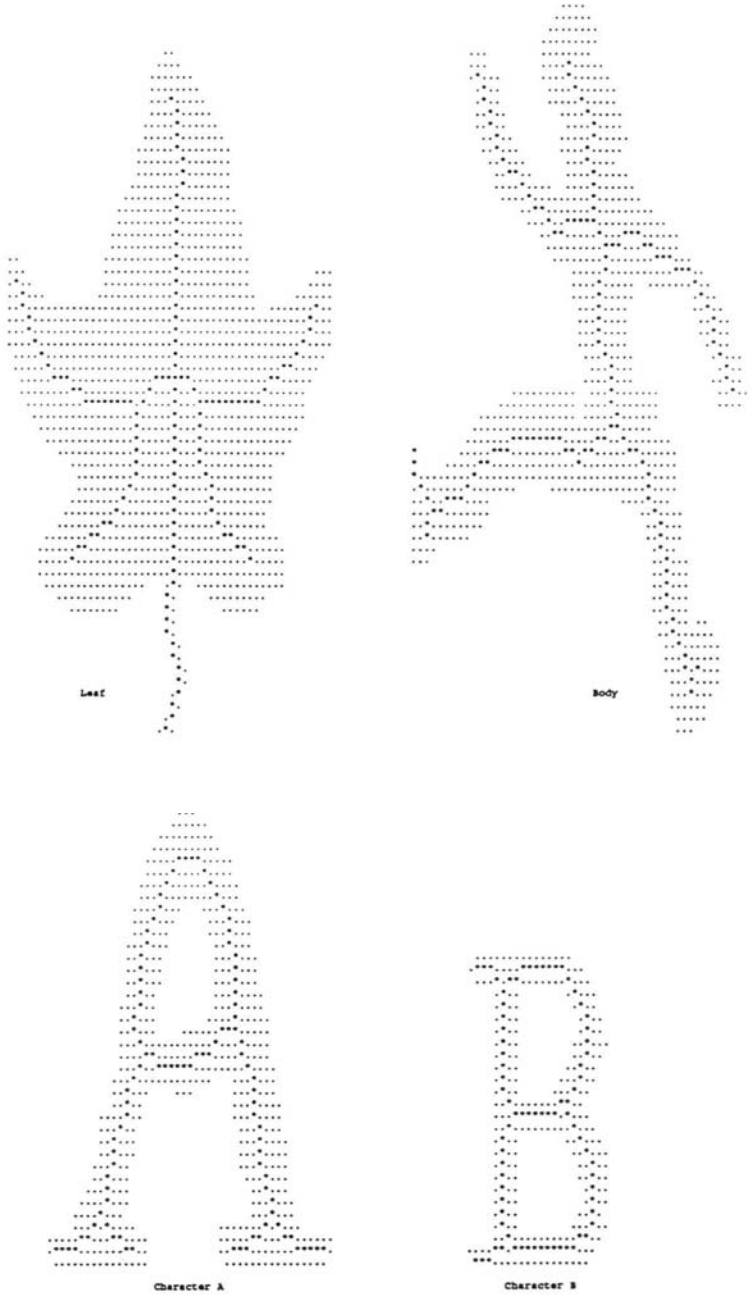
APPENDIX A. An example for vectorizer.



APPENDIX B. 256 weights.

| | | | | | | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 100 | 010 | 001 | 011 | 021 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 001 | 001 | 001 | 000 | 010 | 010 | 010 |
| 000 | 001 | 002 | 003 | 004 | 005 | 006 | 007 | 008 | 009 | 010 | 011 | 012 | 013 | 014 | 015 | 016 | 017 | 018 | 019 | 010 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 020 | 021 | 022 | 023 | 024 | 025 | 026 | 027 | 028 | 029 | 030 | 031 | 032 | 033 | 034 | 035 | 036 | 037 | 038 | 039 | 030 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 040 | 041 | 042 | 043 | 044 | 045 | 046 | 047 | 048 | 049 | 050 | 051 | 052 | 053 | 054 | 055 | 056 | 057 | 058 | 059 | 050 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 080 | 081 | 082 | 083 | 084 | 085 | 086 | 087 | 088 | 089 | 090 | 091 | 092 | 093 | 094 | 095 | 096 | 097 | 098 | 099 | 090 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 080 | 081 | 082 | 083 | 084 | 085 | 086 | 087 | 088 | 089 | 090 | 091 | 092 | 093 | 094 | 095 | 096 | 097 | 098 | 099 | 090 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 110 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 130 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 150 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 170 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 190 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 210 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 230 |
| 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 000 | 010 | 001 | 011 | 011 |
| 010 | 010 | 010 | 010 | 011 | 011 | 011 | 011 | 010 | 010 | 010 | 010 | 010 | 010 | 011 | 011 | 011 | 010 | 010 | 010 | 010 |
| 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 | | | | | 250 |

APPENDIX C. Testing drawings.



REFERENCES

1. Y. S. Chen and W. H. Hsu, "A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms", *Pattern Recogn.* **22**, 3 (1989) 267-282.
2. W. R. Hall, "Fast parallel thinning algorithms: Parallel speed and connectivity preservation", *Commun. Assoc. Computing Mach.* **32** (1989) 124-131.
3. C. M. Holt, A. Stewart, M. Clint, and R. H. Perrorte, "An improved parallel thinning algorithm", *Commun. Assoc. Computing Mach.* **30** (1987) 156-160.
4. L. Lam, S. W. Lee, and C. Y. Suen, "Thinning methodologies - A comprehensive survey", *IEEE PAMI* **14**, 9 (1992) 869-885.
5. A. Rosenfeld, "A characterization of parallel thinning algorithms", *Inform. Control* **29** (1975) 286-291.
6. A. Rosenfeld and L. S. Davis, "A note on thinning", *IEEE Trans. Syst. Man Cybern.* **SMC-6**, 3 (1976) 226-228.
7. H. Tamura, "A comparison of line thinning algorithms from digital geometry viewpoint", *Proc. 4th Int. Conf. Pattern Recognition*, Kyoto, 1978, pp. 715-719.
8. C. Y. Suen and P. S. P. Wang, eds., *Thinning Methodologies for Pattern Recognition*, World Scientific Publishing, 1994.
9. P. S. P. Wang and Y. Y. Zhang, "A fast and flexible thinning algorithm", *IEEE Computer* **38**, 5 (1989) 747-745.

Received July 1992; revised December 1992.



Edward Y. Zhang received his Masters degree in computer science in 1987 from Northeastern University, Boston, MA. He is now a senior software engineer at Zeta Software Company. His current research interests include thinning tool design, vectorization and flow-chart language design.



Patrick S. P. Wang has been tenured full professor of computer science at Northeastern University since 1983, research consultant at MIT Sloan School since 1989, and adjunct faculty master of computer science at Harvard University Extension School since 1985. He received his Ph.D. in computer science from Oregon State University, his M.S. in I.C.S. from Georgia Institute of Technology, his M.S.E.E. from National Taiwan University and his B.S.E.E. from National Chiao Tung University. He was on the faculties of the University of Oregon and Boston University, and senior researcher at Southern Bell, GTE Labs, and Wang Labs prior to his present position. In addition to his research experience at MIT AI Lab, Prof. Wang has been visiting professor and has been invited to give lectures, do research and present papers in more than a dozen countries in Europe and Asia and many universities and industries in the U.S.A. and Canada. He has published over 70 technical papers and 7 books in pattern recognition, AI and imaging technologies and has three OCR patents by US and Europe Patent Bureaus. As IEEE senior member he has organized numerous international conferences and workshops and served as reviewer for many journals and NSF grant proposals. Prof. Wang is currently founding Editor-in-Charge of the *International Journal of Pattern Recognition and Artificial Intelligence*, Editor-in-Chief of the series on *Machine Perception and Artificial Intelligence* by World Scientific Publishing Co., and elected chair of IAPR-SSPR (International Association of Pattern Recognition). In addition to his technical interests, he has also written several articles on the operas of Verdi, Puccini, and Wagner, and the symphonies of Mozart, Beethoven, and Tchaikovsky.