tiprocessor," *Journal of Pattern Recognition and Artificial Intelligence*, vol. 3, No. 2, 1989, pp. 207-216.

[Hi69] Hilditch, C. J., "Linear skeletons from square cupboards," in *Machine Intelligence*, vol. IV, B. Meltzer and D. Michie, eds., Elsevier, New York, 1969, pp. 403-420.

[NS84a] Naccache, N. J. and Shinghal, R., "An investigation into the skeletonization approach of Hilditch," *Pattern Recognition*, vol. 17, No. 3, 1984, pp. 279-284.

[NS84b] Naccache, N. J. and Shinghal, R., "SPTA: A proposed algorithm for thinning binary patterns," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. SMC-14, No. 3, May/June 1984, pp. 409-418.

[NS86] Naccache, N. J. and Shinghal, R., "In response to A comment on an investigation into the skeletonization approach of Hilditch," *Pattern Recognition*, vol. 19, No. 2, 1986, p. 111.

[Ro75] Rosenfeld, A., "A characterization of parallel thinning algorithms," *Information and Control*, vol. 29, 1975, pp. 286-291.

[Ru66] Rutovitz, D., "Pattern recognition," *Journal of the Royal Statistical Society*, vol. 129, 1966, pp. 504-530.

[SR71] Stefanelli, R. and Rosenfeld, A., "Some parallel thinning algorithms for digital pictures," *Journal of the A.C.M.,* vol. 18, 1971, pp. 255-264.

[St86] Stefanelli, R., "A comment on an investigation into the skeletonization approach of Hilditch," *Pattern Recognition*, vol. 19, No. 1, 1986, pp. 13-14.

a	b	c
d	p	e
f	g	h

(a)

1		1
1	1	1
	1	1

(b)

	1	
1	1	
1	1	1

(c)

1	1	1
	1	1
1	1	1

(d)

	1	
1	1	1
1		1

(e)

Fig. 3 (a) The 8-neighborhood of pixel p. Illustrating when p is a (b) *north,* (c) *east,* (d) *west* and *(e) south* border point, respectively.

```
          are 4-simple but neither 4-isolated nor 4-endpoint.
end repeat
end
```

The 8-connected version of the algorithm may be obtained simply by substituting 8-simple, 8-isolated and 8-endpoint for 4-simple, 4-isolated and 4-endpoint respectively in the four steps. The order of the sequential part of the algorithm, i.e., the north, south, east, west sequence for examining border points is arbitrary but should be consistent. It also helps to alternate in this way in order to obtain skeletons that are centered in the original pattern.

Finally we should point out that this algorithm preserves the connectivity of the input pattern.

Theorem: Given a 4-connected pattern, ALGORITHM 4-CONNECTED yields a 4-connected skeleton.

A similar theorem holds for the 8-connected version.

4. References

[BS89] Beffert, H. and Shinghal, R., "Skeletonizing binary patterns on the homogeneous mul-

	p	1
1		1

(a)

	p	1
	1	

(b)

1	p	1

(c)

	p	1
		1

(d)

Fig. 4 Illustrating the four combinations of 4 and 8 simplicity of pixels.

connected.

3. The Algorithm of Rosenfeld

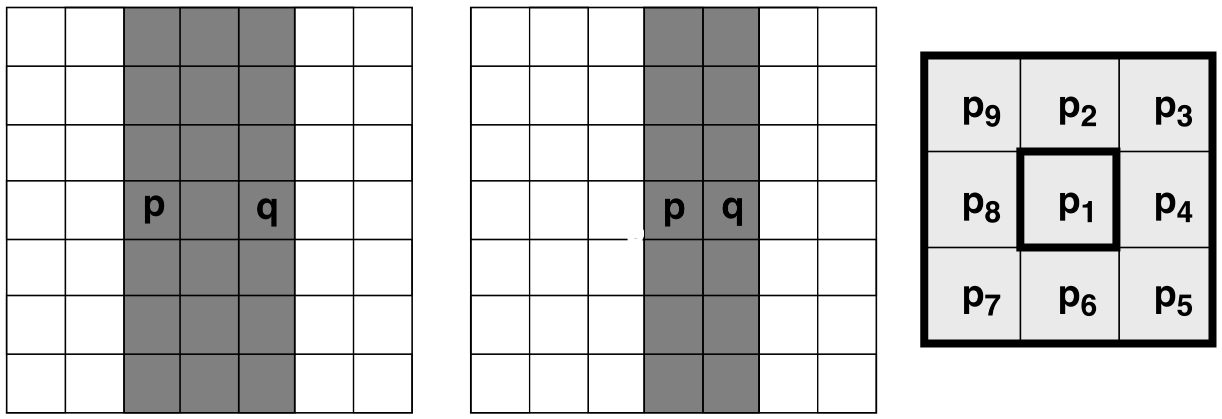
The algorithm described here is a simple parallel algorithm due to Rosenfeld [Ro75]. Like the algorithm of Hilditch this algorithm works by successively discarding in parallel certain subsets of the "boundary" of **P**. We first define their notion of boundary. Let **P** be, as before, the set of black (one's) pixels. The complement of **P** is the background and such pixels are white (zero's sometimes left blank to improve their visibility to the human eye). Consider a pixel p of **P**. Label the 8 neighbors of p as in Fig. 3 (a). We call p a *north border* point if b=0 (Fig. 3 (b)); an *east border* point if e=0 (Fig. 3 (c)); a *west border* point if d=0, (Fig. 3 (d)) and a *south border* point if g=0, (Fig. 3 (e).

We define pixel p as a 4-*endpoint* provided that exactly one of its 4-neighbors is black and an 8-*endpoint* provided that exactly one of its 8-neighbors is black. We define pixel p as 4-*isolated* if none of its 4-neighbors is black and 8-*isolated* if none of its 8-neighbors is black.

We define a border point to be 4-*simple* if changing it from black to white (one to zero) does not alter the 4-connectivity of the remaining black pixels within the Moore neighborhood of p. We define a border point to be 8-*simple* if changing it from black to white (one to zero) does not alter the 8-connectivity of the remaining black pixels within the Moore neighborhood of p. The simplicity of pixels is illustrated in Fig. 4. In Fig. 4 (a) p is 4-simple but not 8-simple. In Fig. 4 (b) p is 8-simple but not 4-simple. In Fig. 4 (c) p is neither 4-simple nor 8-simple and in Fig. 4 (d) p is both 4-simple and 8-simple. It is readily observed that 4-endpoints and 4-isolated points are always 4-simple. Similarly, 8-endpoints and 8-isolated points are always 4-simple.

We are now ready to describe the algorithm which can be viewed in two different modes depending on the type of connectivity used. Like Hilditch's algorithm these algorithms are in part sequential and in part parallel.

```
ALGORITHM 4-CONNECTED

begin

repeat until no pixels are changed from black to white:

        Step 1: (in parallel) Change all black pixels to white if they are north border points that
                are 4-simple but neither 4-isolated nor 4-endpoint.

        Step 2: (in parallel) Change all black pixels to white if they are south border points that
                are 4-simple but neither 4-isolated nor 4-endpoint.

        Step 3: (in parallel) Change all black pixels to white if they are east border points that
                are 4-simple but neither 4-isolated nor 4-endpoint.

        Step 4: (in parallel) Change all black pixels to white if they are west border points that
```



(a) (b) (c)

Fig. 2

pixel that satisfies the following four conditions.

(1) $2 \leq B(p_1) \leq 6$

(2) $A(p_1) = 1$

(3) $p_2 \bullet p_4 \bullet p_8 = 0$ or $A(p_2) \neq 1$

(4) $p_2 \bullet p_4 \bullet p_6 = 0$ or $A(p_4) \neq 1$

The algorithm stops when during a pass no pixels are changed from black to white.

This algorithm tends to yield 8-connected skeletons when the input pattern **P** is 8-connected and is fairly insensitive to contour noise.

Comments

(1) The condition $B(p_1) \leq 6$, ensures that $p_1$is on the boundary of **P**.

(2) The condition $2 \leq B(p_1)$ ensures we keep isolated points as well as skeleton tips.

(3) The condition $A(p_1) = 1$, ensures we do not fragment the skeleton.

(4) Conditions (3) and (4) ensure we do not change the connectivity of "lines" that are two pixels thick. Note that these conditions also imply that a 4×4 window is actually used.

2.3 Problem

Prove or disprove that the above algorithm never erodes a pattern **P** to nothing if **P** is 8-

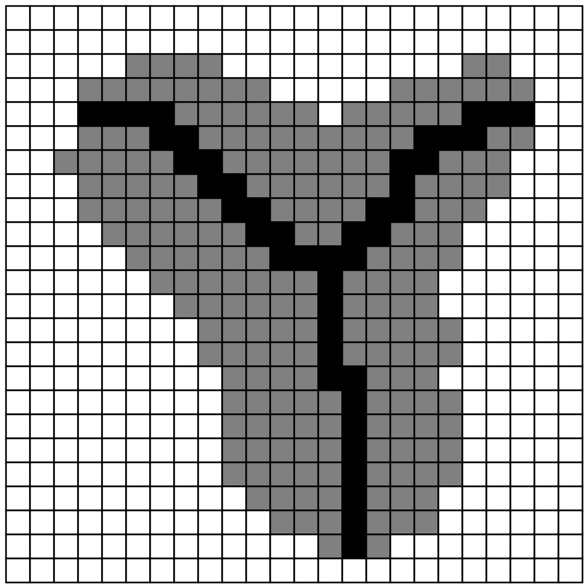


Fig. 1 (c) A binary (shown as white and gray) digitized image of a "Y" chromosome with a reasonable "skeleton" representation shaded in black.

patterns in Fig. 2 representing some portion of a vertical pattern 3-pixels thick in (a) and 2-pixels thick in (b). In Fig. 2 (a) removal of p does not change the resulting connectivity. Neither does removal of q. This is true for all pixels on the columns containing p and q. Therefore all these pixels would be removed yielding a skeleton one pixel thick. In Fig. 2 (b) on the other hand each pixel can be removed on its own without changing the resulting connectivity but if we remove all the pixels above and below p and q we end up with nothing. Therefore care must be taken with skeletonization algorithms lest they behave like expensive erasers for some input patterns. This difficulty can be overcome by demanding that the removal of two adjacent pixels not alter connectivity.

Stability

Once a skeleton (or part of it) is obtained, say after *n* passes, it should not be eroded away by subsequent passes. This can be overcome by including an additional condition to the effect that tips of skeletons may not be removed.

2.2 The algorithm

Consider the 3 × 3 window around a "black" pixel labelled p_1 and label its eight neighbors in a clockwise spiral fashion as illustrated in Fig. 2 (c). Let $A(p_1)$ denote the number of 01 patterns encountered in the ordered set $p_2, ..., p_9, p_2$. Let $B(p_1)$ denote the number of non-zero neighbors of p_1. Then, at each pass in which we remove (in parallel) the outer layer of pixels we remove each

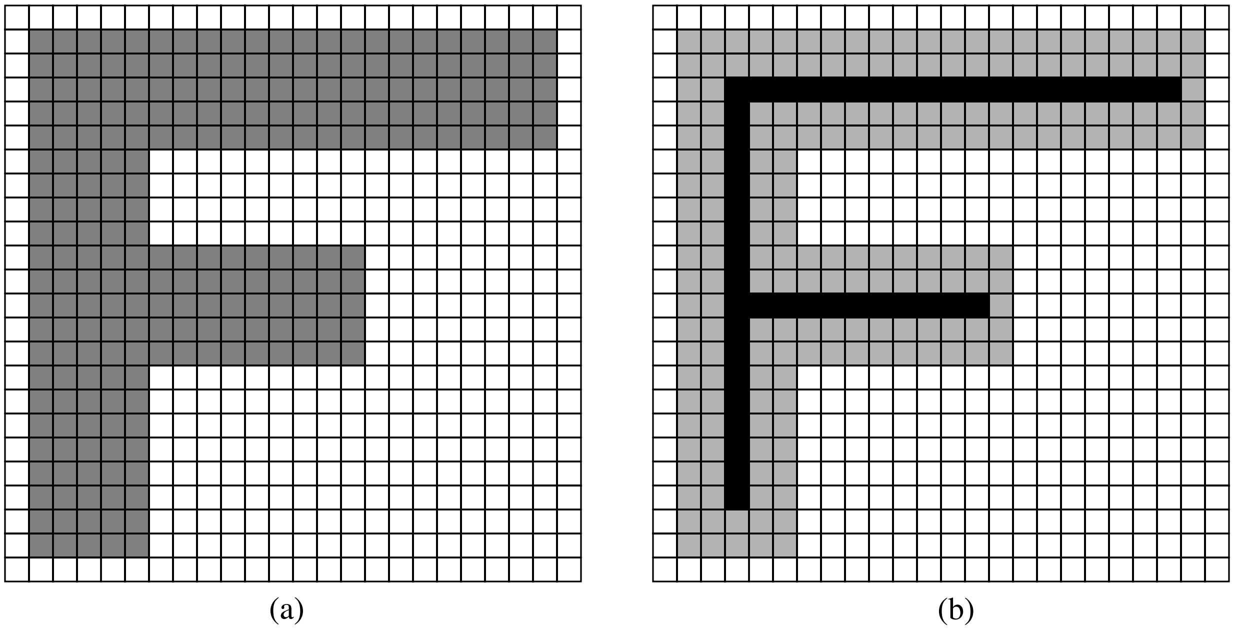


Fig. 1 (a) A “line-like” pattern consisting of three strokes. (b) The *skeleton* of the pattern, i.e., the thinnest representation of the original pattern that preserves the topology.

[BS89].

2.1 Requirements to be met by skeleton

There are some general requirements that affect the design of the algorithm. Let **P** denote a (line-like) binary pattern, i.e., the set of “black” pixels in a digital picture given as a square array.

Thinness

The skeleton of **P** is required to consist of thin lines. This can be achieved by successively removing (changing from black to white) pixels which lie on the boundary of **P** until all that remains are lines which are one point wide.

Position

The skeleton should lie along the center of **P**. This is achieved by removing pixels in a *parallel* fashion. A pixel is removed only if it lies on the boundary of the initial **P** regardless of which other pixels have been removed. At each (parallel) pass the outer layer of **P** (boundary) is removed to give a “thinner” pattern for the next pass. Note that the entire procedure is a parallel algorithm embedded in a sequential algorithm. In order not to thin the pattern **P** away to nothing we impose some more conditions.

Connectivity

The skeleton should have the same connectivity (topology) as the original pattern. This is achieved by testing each pixel that is to be removed to determine whether its removal would alter the connectivity. Obviously, pixels that would alter the connectivity are retained.

Care has to be taken in performing such connectivity tests in parallel. Consider the line like

CHAPTER 7

SKELETONS

Godfried Toussaint

ABSTRACT

This chapter introduces the basic ideas behind the computation of skeletons and illustrates them with two popular algorithms: Hilditch's algorithm and Rosenfeld's algorithm.

1. Introduction

In many pattern recognition applications the input patterns to be analyzed are essentially "line-like." In other words they consist of components which are essentially strokes and the relevant perceptual information is contained not in the thickness of the pieces but in their relative position to the other strokes of the pattern. One such example is character recognition. Another is chromosome recognition where the X and Y chromosomes are essentially topologically the same as the letters "X" and "Y".

Consider the binary digital character "F" in Fig. 1 (a). It consists basically of three strokes or line-like pieces (two horizontal and one vertical) connected in a certain manner. The thickness of the strokes (in this case 5 pixels) is irrelevant to the recognition problem. What is important is the topology of how the three pieces are connected together. In such situations it is convenient to simplify the input as much as possible in order to make the topological analysis as simple as possible. One approach which is quite powerful and popular is to create a version of the pattern that is as thin as possible. For example, for the pattern in Fig. 1 (a) it would be nice to represent it as the one-pixel thin black pattern illustrated in Fig.1 (b). Methods to accomplish this are called *thinning* or *skeletonization* and the resulting patterns are usually refereed to as *skeletons*.

Clearly the above "definition" of the input as "line-like" is vague. Sometimes it is not clear cut whether the input is or is not line-like. For example, the "Y" chromosome in Fig. 1 (c) is quite "blob-like" but nevertheless is considered line-like because it is a "Y" chromosome and it is conceptually attractive to consider it as line like. Furthermore there exists no precise definition of the concept of "skeleton" either, although some related structures (the *medial axis* for example) do have a precise mathematical definition. Hence the algorithms for computing skeletons reflect this vagueness. Below we illustrate two methods to compute skeletons that characterize the scores of algorithms available in the literature.

2. The Algorithm of Hilditch

This algorithm is a parallel algorithm that uses a 4 × 4 window described by Hilditch in a seminar in 1968 and published in [Ru66]. See [NS84a] for an investigation of this algorithm. Hilditch later also published a similar sequential algorithm that uses only a 3 × 3 window [Hi69]. These two algorithms should not be confused [St86], [NS86]. For a comparative survey of skeletonization algorithms see [NS84b]. For efficient implementations of skeletonization algorithms see