ELSEVIER

Pattern Recognition Letters 18 (1997) 343–353

Pattern Recognition Letters

Residuals + directional gaps = skeletons

Rafael Cardoner [1], Federico Thomas *

Institut de Robòtica i Informàtica Industrial (CSIC – UPC), Gran Capità 2-4, 2 planta, 08034 Barcelona, Spain

Received 15 October 1996; revised 3 April 1997

Abstract

Many thinning algorithms for 2D binary images, or modifications of existing ones, have been proposed in recent years. The one given herein is surprisingly simple compared to most of them, and still it has theoretically favorable properties. Actually, it provides a connected – single-pixel in width whenever possible – well-centered homototic skeleton that allows shapes to be nearly reconstructed. In addition to these properties, it is also very attractive because of its generalization to higher dimensions. The presented algorithm is based on the application of directional erosions, while retaining those pixels that introduce disconnections. It is shown how this strategy is specially well-suited for run-length encoded images, leading to a very fast and simple thinning algorithm. © 1997 Elsevier Science B.V.

Keywords: Thinning algorithms; Skeletons; Run-length encoding; Mathematical morphology

1. Introduction

Many thinning algorithms, or modifications of existing ones, have been proposed in recent years. More than 300 published papers on this subject reflect this proliferation (see Lau et al., 1992, or Suen and Wang, 1993).

The common required properties of a skeleton are:

- *Reconstructability.* A skeleton must contain sufficient information which can be used to reconstruct the original binary image.
- *Rotation-invariance.* Because of robustness reasons, a skeleton for a given region must be independent from its orientation.
- *Connectedness.* A skeleton must be homotopic to the region it corresponds to. In other words, it must preserve 8-neighbor connectedness for foreground and 4-neighbor connectedness for background.
- *Thinness.* A skeleton must be single-pixel in width. This is also a common requirement to ease subsequent processes, such as vectorization.

Unfortunately, as recognized by Serra (1992), in the discrete plane these requirements become mutually incompatible, so that practical skeleton methods are invariably a compromise between them. In this sense, thinness may be incompatible with preservation of connectedness, or with reconstructability. Fig. 1 shows an example of both situations, the second one adapted from Arcelli (1985).

Moreover, rotation-invariance is only possible if pseudo-Euclidean distance is used rather than the commonly used city-block and chessboard distances, as explained below.

Two main strategies to obtain the skeleton of a given region have been proposed:

* Corresponding author. E-mail: fthomas@iri.upc.es.

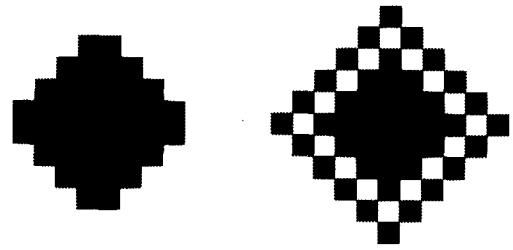
[1] E-mail: cardoner@ic.upc.es.

PII S0167-8655(97)00021-4

- Those that peel off the boundary of an object in a sequence of raster, or parallel operations (see, for example, Jang and Chin, 1990). These algorithms produce skeletons by iteratively deleting pixels from the boundary of regions. The deletion of a pixel in a sequential algorithm is more unpredictable than in a parallel one, because the result depends partly on the order in which the pixels are processed. To retain information the region has to eroded symmetrically from all the directions and the process continues as long as no deletable pixels are present.
- Those based on a previous distance transform (see, for example, Shih and Pu, 1995). A distance transformation converts a binary image, which consists of object (foreground) and non-object (background) pixels, into an image where every object pixel has the value corresponding to the minimum distance to the background. Three types of metrics for digital images are often used: city-block (1-1 metrics), chessboard (1-2 metrics), and pseudo-Euclidean (2-3 or 3-4 metrics). The 1-2 metrics, for example, assigns a distance of 2 between horizontally and vertically adjacent pixels, and 3 between diagonally adjacent pixels. City-block and chessboard distances are easy to compute by iteratively peeling off the border pixels and summing the distances layer-by-layer. If distance values are visualized as the altitude on a surface, then the ridges of the surface constitute the skeleton. In general, these algorithms are not iterative so that the skeleton is produced in a fixed number of passes through the image.

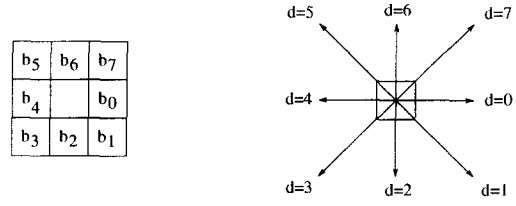
The approach adopted herein, which falls inside the former approach, satisfies the requirements given above in the following order of priority: connectedness, thinness and reconstructability. Whenever possible, these three requirements are satisfied. The peeling approach was adopted because the final algorithm is intended to be directly used on large run-length encoded images, and the exact run-length encoding of a distance transform will seldom reduce the storage significantly (even it is possible it may increase it).

The proposed procedure can be outlined as follows. Those pixels whose deletion by a directional erosion might destroy the connectedness are retained and classified as *gaps*, then the region is eroded and the corresponding *residual* computed. Gaps and residuals are retained in the image, and this process is repeated until no progress is made.



(a) (b)

Fig. 1. Two examples in which common requirements for a skeleton become mutually incompatible: (a) thinness and reconstructability, and (b) thinness and connectedness.



(a) (b)

Fig. 2. (a) The eight neighbors of a pixel, and (b) directions associated with them.

This paper is structured as follows. In order to make it as self-contained as possible, the required morphological operations are reviewed in Section 2. Next, the idea of residuals and gaps associated with directional erosions are introduced, two key points for our skeletonization algorithm which are presented in Section 3. Then, it is shown how a slight modification of the algorithm make it well-suited for run-length encoded images, a spatial-encoding technique that is reviewed in Section 4. The same section contains some examples in $\mathbb{Z}^2$ and time comparisons with other well-known algorithms. We conclude in Section 5.

2. Background

2.1. Definitions and properties

When restricted to $\mathbb{Z}^2$, the eight neighbors of a pixel b will be denoted by $b_0, b_1, \ldots, b_7$ as shown

in Fig. 2(a). Then, each neighbor defines a direction from the right horizontal turning clockwise, that can be summed and subtracted in mod 8 (Fig. 2(b)).

Mathematical morphological operations apply to sets of any dimensions, such as Euclidean $\mathbb{R}^n$ or its discrete equivalent, the set of n-tuples of integers $\mathbb{Z}^n$. Dilation and erosion are the primary operations of mathematical morphology. Erosion of a binary region X using structuring element B is defined as

$$X \ominus B = \{y \mid \forall b \in B,\ y + b \in X\}, \tag{1}$$

and its dilation using the same structuring element as

$$X \oplus B = \{y \mid y = x + b,\ x \in X,\ b \in B\}, \tag{2}$$

and its opening as

$$X\ o\ B = ((X \ominus B) \oplus B). \tag{3}$$

The residual, $X \perp B$, is the set made of those points in X which do not belong to its opening using the structuring element B, that is,

$$X \perp B = X \setminus (X\ o\ B) \tag{4}$$

If X_b denotes the translation of X in the direction associated with $b \in B$, then it can be shown that

$$X \ominus B = \bigcap_{b \in B} X_{-b}. \tag{5}$$

In other words, erosion can be accomplished by taking the intersection of all the translates of X, where the shifts in the translates are the negated members of B seen as vectors.

An especially interesting case for B is that in which B consists of two pixels, where one is the origin. Then, the erosion of X using B can be computed simply by

$$X \ominus B = X \cap X_{-b}, \tag{6}$$

and its opening by

$$X\ o\ B = (X \cap X_{-b}) \cup (X \cap X_{-b})_b. \tag{7}$$

Since the following property for dilation and erosion always holds,

$$(B1 \ominus B2) \ominus B3 = B1 \ominus (B2 \oplus B3), \tag{8}$$

then, if

$$B = B1 \oplus B2 \oplus \cdots \oplus Bk, \tag{9}$$

one concludes that

$$X \ominus B = (\cdots [(X \ominus B1) \ominus B2] \ominus \cdots \ominus Bk). \tag{10}$$

Thus, if a structuring element can be broken down to a chain of dilations of smaller substructuring elements, the desired operation may be performed as a string of suboperations. In (Zhuang, 1994) the optimal two-pixel decomposition for a binary structuring element is given.

2.2. Residuals

Sometimes a binary image skeleton is defined as the set of all the residuals of the successive erosions of X, using the following simple algorithm:

algorithm S1;
input: X;
output: S;
$S \leftarrow \emptyset$;
while $X \neq \emptyset$
  $S \leftarrow S \cup (X \perp B)$;
  $X \leftarrow X \ominus B$;
endwhile;
end,

or, equivalently,

algorithm S2;
input: X;
output: S;
$S \leftarrow \emptyset$;
while $X \neq \emptyset$
  $E \leftarrow X \ominus B$;
  $S \leftarrow S \cup (X \setminus (E \oplus B)$;
  $X \leftarrow E$;
endwhile;
end,

which reduces the numbers of required operations.

Now, let us assume that B is a 3×3 square structuring element which can be broken down into a chain of three dilations of two-pixel elements, as shown in Fig. 3. Then, the above algorithm can be rewritten as follows:

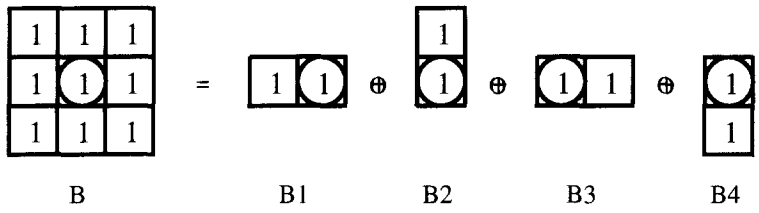


Fig. 3. A 3 × 3 square structuring element can be broken down into a chain of three dilations of two-pixel elements.

algorithm S3;
input: X;
output: S;
$S \leftarrow \emptyset$;
$i \leftarrow 1$;
while $X \neq \emptyset$
  $E \leftarrow ((((X \cap X_0) \cap X_2) \cap X_4) \cap X_6)$;
  $S \leftarrow S + [X \setminus ((((E \cup E_4) \cup E_6) \cup E_0) \cup E_2)]^i$;
  $X \leftarrow E$;
  $i \leftarrow i + 1$;
endwhile;
end,

where $(I)^i$ denotes the operation that sets image I to level i. Then, the result of this algorithm is clearly a grey-level image where each pixel of the skeleton is labeled with its chessboard distance to the nearest region border. The main advantage of this algorithm over the previous one is that it only involves directional erosions and dilations along the coordinate axis. Although the skeleton thus obtained allows us to entirely reconstruct the initial set by simply dilating each pixel according to its grey-level value, it is neither unit-width nor connectivity preserving (Fig. 6(a)). The first drawback can be easily overcome by slightly modifying the above algorithm as follows:

algorithm S4;
input: X;
output: S;
$S \leftarrow \emptyset$;
$A \leftarrow \emptyset$;
$i \leftarrow 1$;
while $X \neq \emptyset$
  for $(d \leftarrow 0; d < 8; d \leftarrow d + 2)$
    $E \leftarrow X \cap X_d$;
    $A \leftarrow A \cup (X \setminus (E \cup E_{d+4}))$;
    $X \leftarrow E$;
  endfor;
  $S \leftarrow S + A^i$;
  $i \leftarrow i + 1$;
endwhile;
end.

Now, since residuals are independently unit-width (they are obtained from single directional erosions) and they do not stack but at some junctions, the obtained skeleton is unit-width except at these junctions (Fig. 6(b)). As a counterpart, the original shape can only be nearly reconstructed but, as it have been already pointed out, thinness and reconstructability are mutually incompatible goals.

In order to overcome the remaining drawback – connectivity – we first introduce the concept of directional gaps.

2.3. Directional gaps

Those pixels required to ensure connectivity in the final skeleton, and not included in the medial axis computed in the above subsection, will be part of a set of disjoined regions that we will call gaps. Contrary to what one might expect, when considering only directional erosions, gaps can be easily computed.

Consider the effect of a directional erosion, from left to right (that is, $X \cap X_{-4}$), on the 3 × 3 configurations of pixels in Fig. 4. While those marked in Fig. 4(a) will be classified as residuals, those marked in Fig. 4(b) will not, leading to disconnections. Now, consider the effect of the same kind of erosion on the 2 × 2 configuration of pixels in Fig. 5(a). In order to ensure connectedness, the two foreground pixels must appear in the skeleton. If the considered erosion removes the SW pixel, it will appear as directional residual (because the SE pixel is background) and thus it will be directly classified as skeleton. If not, it will

(a) (b)

Fig. 4. Marked pixels will be clearly removed by a directional erosion from left to right. While the ones on the left pattern will be classified as residuals, the ones on the right pattern will not.

(a) (b)

Fig. 5. Directional gaps. The foreground pixels in both 2×2 patterns must be retained to ensure connectedness.

remain in the image. The NE pixel will be clearly removed and perhaps it might not be classified as skeleton. Thus, it must be retained to ensure connectedness. The same happens with the configuration of pixels if Fig. 5(b). In this case, following the same reasoning as above, the SE pixel is the one that must be retained. Obviously, for erosions in other directions, the situation is equivalent. Hence, the directional gap of a binary region X in a direction d ($d = \{0,2,4,6\}$) can be obtained by computing

$$X \cap \bar{X}_d \cap [(X_{d+1} \cap \bar{X}_{d+2}) \cup (X_{d-1} \cap \bar{X}_{d-2})]. \quad (11)$$

3. The thinning algorithm

The motivation behind our thinning algorithm is seen as follows. First those pixels whose deletion by a directional erosion might destroy the connectedness are retained and classified as gaps, then the region is effectively eroded and the corresponding residual is computed. Gaps and residual are retained in the image since they are part of the skeleton. This process is repeated until no more progress is made. The following algorithm in pseudo-code implements this idea.

algorithm S5
input: X;
output: S; /* skeleton of X */
$S \leftarrow \emptyset$;
$A \leftarrow \emptyset$;
$S \leftarrow X$;
do
  $X \leftarrow S$;
  for ($d \leftarrow 0$; $d < 8$; $d \leftarrow d + 2$)
    $G \leftarrow S \cap \bar{S}_d \cap [(S_{d+1} \cap \bar{S}_{d+2}) \cup (S_{d-1} \cap \bar{S}_{d-2})]$;
      /* gap */
    $E \leftarrow S \cap S_d$; /* eroded image */
    $R \leftarrow S \setminus (E \cup E_{d+4})$; /* residual */
    $A \leftarrow A \cup R \cup G$;
    $S \leftarrow A \cup E$;
  endfor;
while ($X \neq S$);
end.

Note that the number of iterations is exactly half the chess-board maximum thickness of the region. This is perhaps the simplest proposed algorithm for thinning binary regions – directly expressed in terms of directional simple operations – that provides a connected single-pixel-in-width well-centered homotopic skeleton.

Fig. 6(c) shows the result of applying the above algorithm. When comparing results, the obtained skeletons turn out to be *equivalent* to those generated using "Algorithm B" developed in Jang and Chin (1990). This algorithm consists in iteratively deleting pixels from the boundary of regions that match the rotating structuring elements shown in Fig. 7, denoted as E, L_1 and L_2. If starting by eroding pixels from left to right using our algorithm, all possible configurations in which the central pixel is eroded and is not classified as residual appear in Fig. 8. Some of these pixels are retained because they are found to be gaps, all others are effectively deleted. These pixels coincide with those deleted by Algorithm B, except in configuration appearing in Fig. 8(a). This explains why both algorithms only differ in the skeletal end-points oriented in the starting eroding direction of our algorithm (see Fig. 6(e)).

Finally, in order to label each pixel of the skeleton with its chess-board distance to its nearest region border, the following algorithm can be used:

algorithm S6
input: X;
output: S; /* skeleton of X */

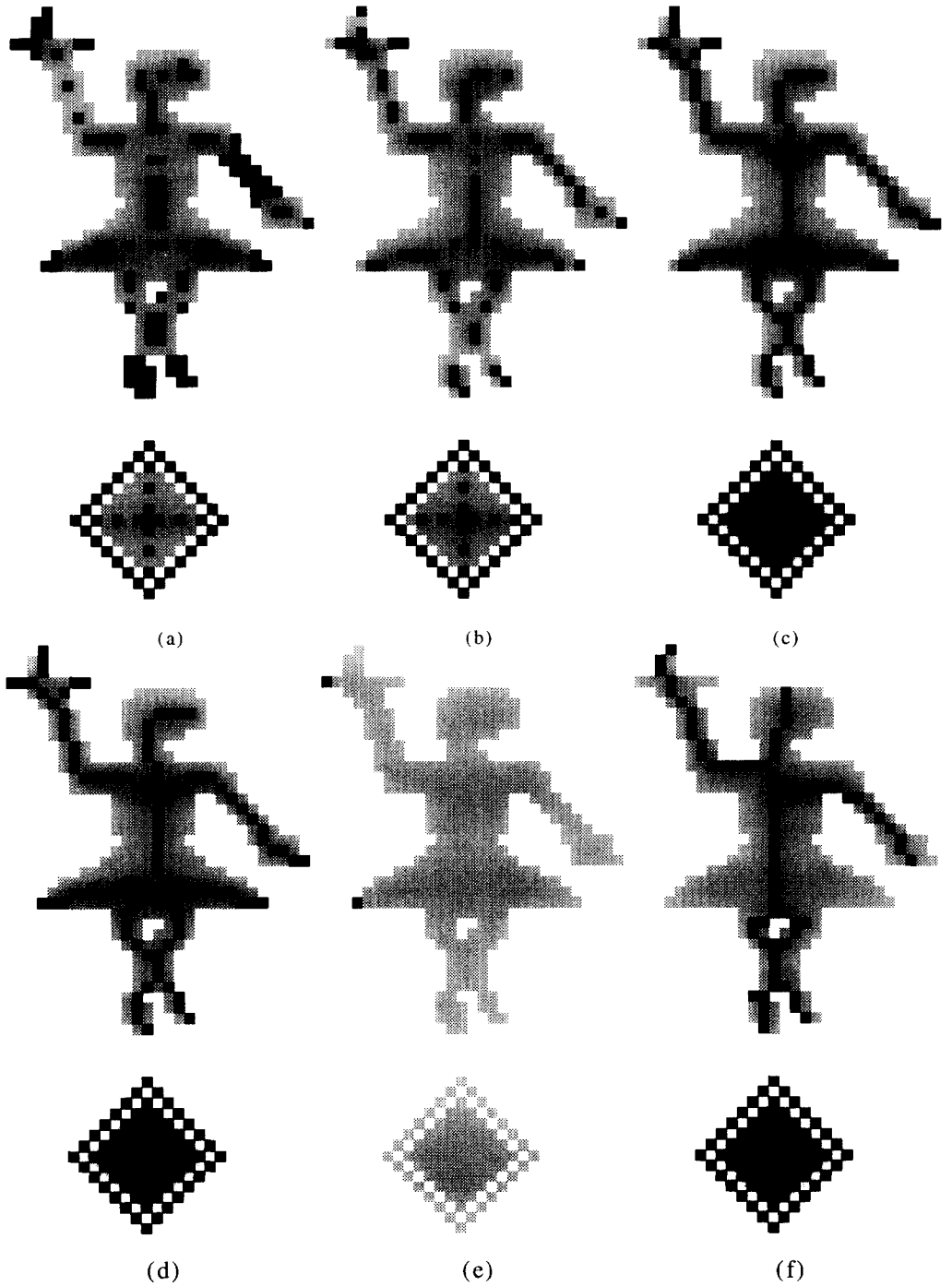


Fig. 6. Binary skeletons obtained using (a) algorithm S3, (b) algorithm S4, (c) algorithm S5, (d) Jang-Chin's algorithm B, and (f) Piper's algorithm. (e) Differences between (c) and (d).

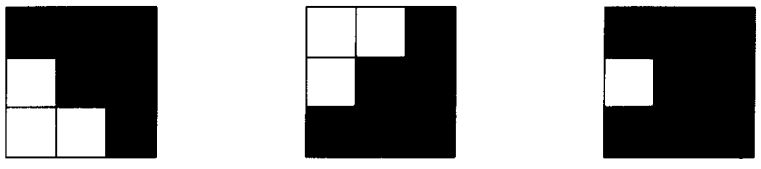


Fig. 7. Rotating structuring elements used by Jang and Chin (1990), in their Algorithm B.

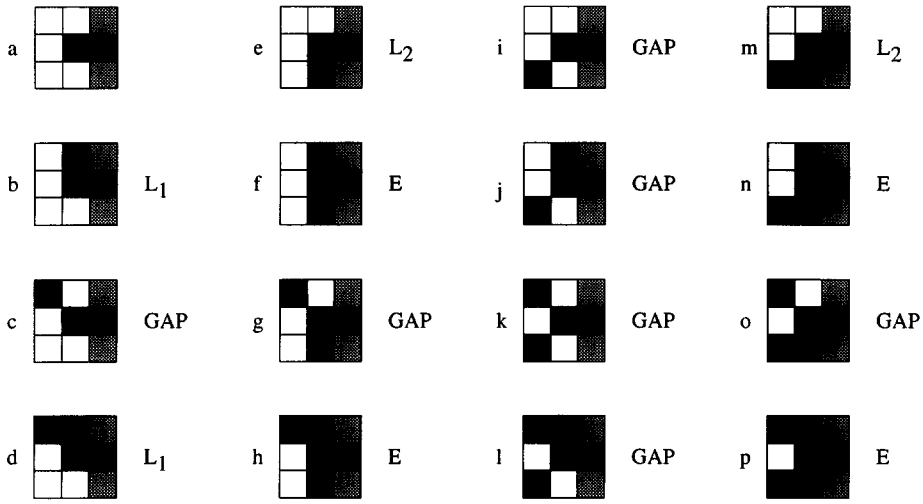
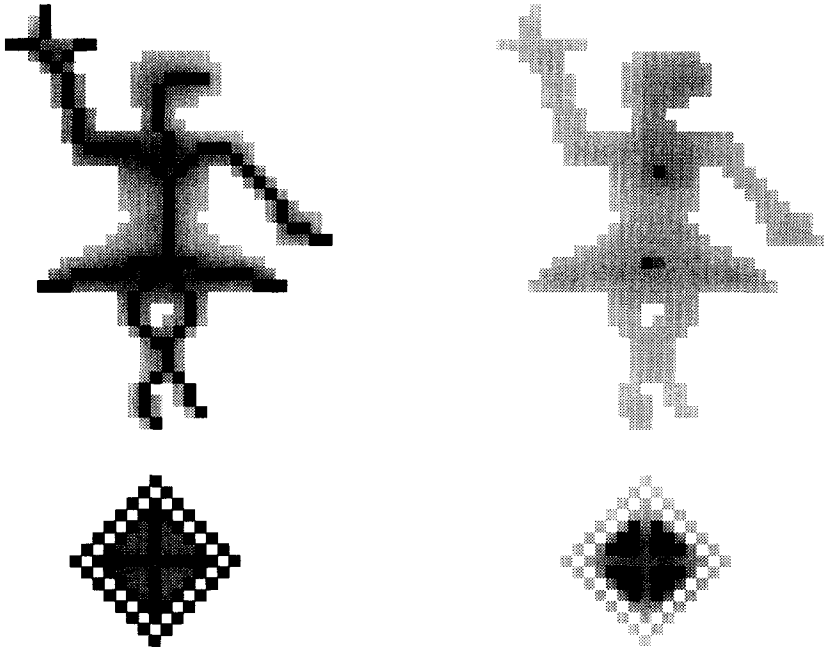


Fig. 8. All possible configurations in which the central pixel is eroded by a left-to-right erosion and it is not classified as residual. Pixels in grey can be either black or white.

$i \leftarrow 0$;
$S \leftarrow \emptyset$;
$L \leftarrow \emptyset$; /* increment of skeleton */
do
  $I \leftarrow \emptyset$;
  for $(d \leftarrow 0; j < 8; d \leftarrow d + 2)$
    $G \leftarrow X \cap \bar{X}_d \cap [(X_{d+1} \cap \bar{X}_{d+2})$
      $\cup (X_{d-1} \cap \bar{X}_{d-2})]$; /* gap */
    $E \leftarrow X \cap X_d$; /* eroded image */
    $R \leftarrow X \setminus (E \cup E_{d+4})$; /* residual */
    $I \leftarrow I \cup R \cup G$; /* increment of skeleton */
    $X \leftarrow E \cup I$; /* updated image */
  endfor;
  $X \leftarrow X \cap \bar{L}$;
    /* removal of previous skeleton increment*/
  $S \leftarrow S + (L)^i$;
    /* updated skeleton */
  $L \leftarrow I \cap \bar{L}$;
    /* updated last skeleton increment */
  $i \leftarrow i + 1$;
until $X == \emptyset$;
end.

Actually, this algorithm introduces three main advantages over the basic algorithm S5. First, as mentioned above, a grey-level skeleton is generated, the value of each skeleton pixel being its chess-board distance to the boundary. This eases postprocessing, such as the elimination of non-significant skeletal legs, to make the result simple and useful for object recognition. Second, the iterations continue until an image becomes empty, which is faster that detecting idempotence. Third, the thinning effort is concentrated on the thick region; in other words, once a thin region is obtained, it is removed from the image and hence it is no longer considered. This is of great interest when processing run-length compressed images, as described in the next section.



(a) (b)

Fig. 9. (a) Obtained skeleton applying algorithm S6, and (b) differences with the result of algorithm S5.

Although the obtained skeletal legs are identical for algorithms S5 and S6, differences sometimes arise at skeletal joints. This is a consequence of the deletion-retention strategy used in the latter algorithm: pixels in joints appear as gaps or residuals many times, then they are removed while they have to be retained to get exactly the same results (Fig. 9). This might lead to unexpected behaviors in some pathological cases, but for practical applications it provides excellent results.

4. Spatial encoding and results

The natural redundancy in images can be exploited to speed up morphological operations, often dramatically, through a spatial encoding scheme, such as run-length encoding, a quad-tree representation, or representation of object contours such as polygonal approximations.

Run-length encoding is particularly useful because of its simplicity and generality. Actually, interval coding has provided a particularly simple and effective way of speeding a variety of binary image operations. It has been used to speed up operations such as region labeling (Rosenfeld, 1982), boundary extraction (Pavlidis, 1982), moment invariants calculation (Zakaria et al., 1987), eigenvalue decompositions (Roseborough and Murase, 1995), morphologic computations (Ji et al. 1989) and skeletons (Piper, 1992).

In our case, since ANDing, ORing and shifting are the only three basic operations on images required to compute skeletons, their implementation on run-length encoded images is discussed next.

Let us suppose two run-length encoded images to be ANDed. l_1 and l_2 represent the lists of intervals associated with the same line in the first and second image respectively. Then, the list representing the intersection of both lists of intervals can be computed as follows:

algorithm intersect;
input $l_1 = \langle [a_0, b_0], \ldots, [a_n, b_n] \rangle$
and $l_2 = \langle [c_0, d_0], \ldots, [c_m, d_m] \rangle$;
$i \leftarrow 0$; $j \leftarrow 0$;
repeat
if $b_i < c_j$ **then** $i \leftarrow i + 1$;
/* no overlap, skip an interval in l_1 */
else if $d_j < a_i$ **then** $j \leftarrow j + 1$;

Table 1
Performance comparison for the 632 × 750 test image in Fig. 10. All algorithms have been adapted to work on run-length encoded compressed images

Algorithm	Iterations	Time	Nodes	Branches
A	312 + 26	3'45" + 0'25"	188	219
B	33	1'18" + 0'25"	160	190
C	51	1'24" + 0'25"	67	97
S5	32	0'54" + 0'25"	160	191
S6	32	0'19"	160	191

```
      /* no overlap, skip an interval in l2 */
    else  /* overlap, intersect current intervals */
      l3 ← l3 ∘ [max(ai, cj), min(bi, dj)];
      if bi < dj then i ← i + 1;
      else j ← j + 1;
    endelse;
  endif;
until i > n or j > m;
output l3;
```

where ∘ denotes concatenation. The generated list, l_3, can be computed in linear time in the total input size $|l_1| + |l_2|$.

The algorithm for ORing two lines is similar, and also runs in linear time. Note that both operations produce a list l_3 with at most $|l_1| + |l_2|$ intervals.

The evaluation of shifted images is avoided so that displacements of an image are taken into account, introducing the proper indices, directly when ANDed or ORed with another image. In other words, a directional erosion or dilation takes the same time as an AND or OR operation.

The exact run-length encoding will seldom reduce the storage for distance-transformed images, and it is possible it may increase it. This is because adjacent pixels do not have exact correspondence, so the encoded run-length are usually one or a few pixels. This is why thinning processes based on a previous distance transform become quite inefficient when implemented on run-length encoded images.

Algorithms S5 and S6 have been implemented for run-length encoded images in C language running on a PC Pentium/90Mhz with 8 Mbytes of RAM. In order to compare processing times with other well-known algorithms, algorithms A and B described by Jang and Chin (1990) have also been implemented and adapted to work on run-length encoded images. We have also implemented a variation of algorithm B, here called algorithm C, less sensitive to noisy patterns. Performance comparison for the 632 × 750 test image in Fig. 10 are compiled in Table 1. All algorithms, but S6, require 25 seconds to label each pixel of the resulting skeleton with its chess-board distance to the nearest border, as marked in the third column. The time required to convert a bitmap into a run-length representation (typically milliseconds) can be neglected in front of these timings. Then, we can conclude that algorithm S6 clearly outperforms all others.

Fig. 10 shows the resulting skeletons, where the one in Fig. 10(b) corresponds to the result obtained using Piper's algorithm (Piper, 1992) which is highly anisotropic, but whose computation is the fastest known for run-length encoded images (less than 1 second in this example). Since the particular geometric properties a skeleton should preserve may be problem- or application-dependent, Piper's skeleton still remain as an important alternative for some applications.

5. Conclusions

A simple algorithm that computes the skeleton of binary patterns by applying directional erosions, while retaining those pixels that introduce disconnections, has been introduced. The used strategy avoids the main drawbacks associated with rotating thinnings.

Moreover, we have shown the efficiency of this algorithm can be greatly improved and its memory requirements dramatically reduced by run-length encoding the original image. The method uses the natural redundancy occurring in binary images to reduce the number of repeated computations that must be performed and it appears most useful for very large line drawings, where high image resolutions are desired.

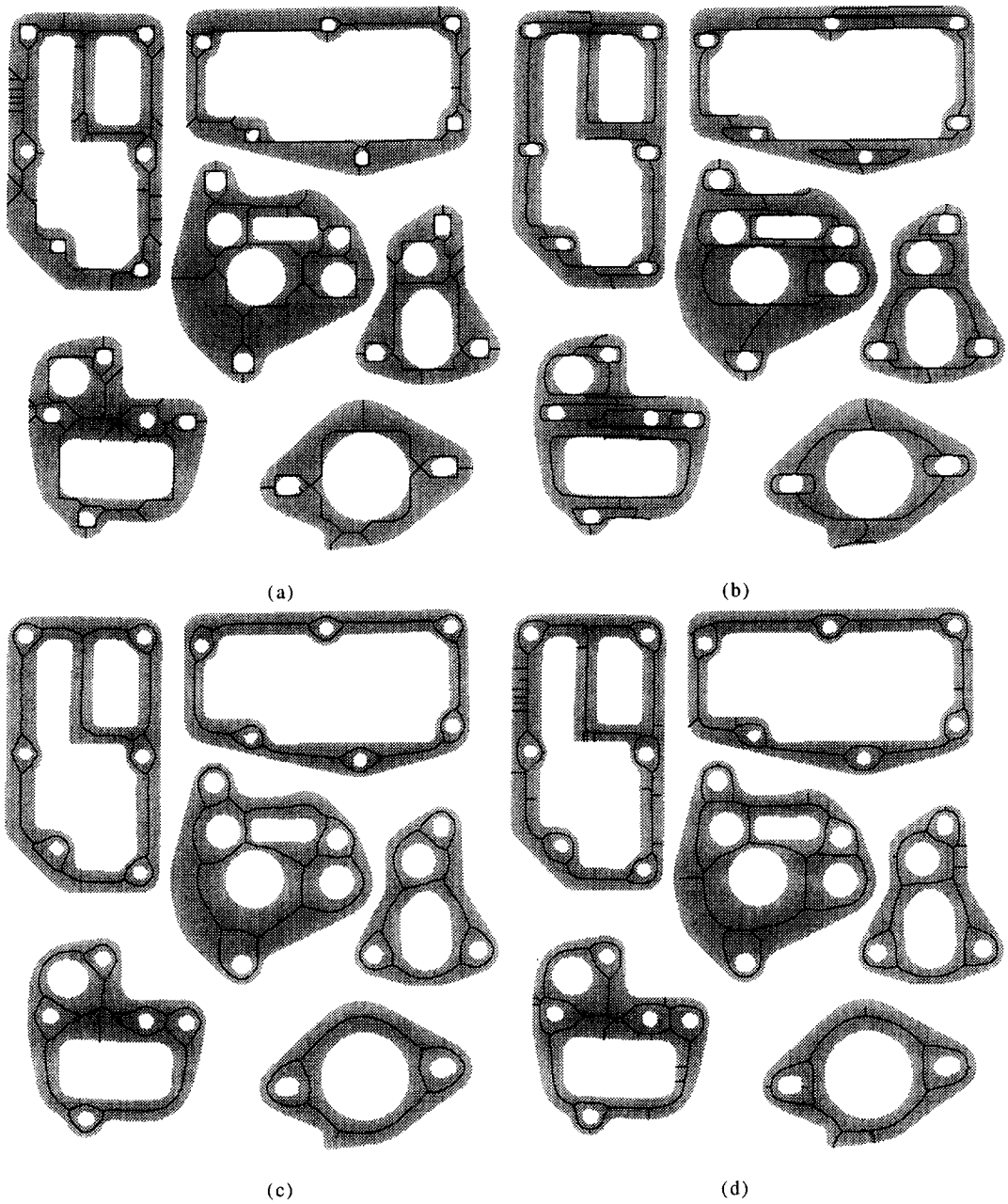


Fig. 10. Obtained results using (a) algorithm A, (b) Piper's algorithm and (c) algorithm C. (d) The slight differences between the results obtained using algorithms B, S5 and S6 cannot be distinguished at the shown resolution level.

References

Arcelli, C. and G. Sanniti di Baja (1985). A width independent fast thinning algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence* 7 (4), 463–474.

Jang, B-K. and R.T. Chin (1990). Analysis of thinning algorithms using mathematical morphology. *Trans. Pattern Analysis and Machine Intelligence* 12 (6), 541–551.

Ji, L., J. Piper and J.-Y. Tang (1989). Erosion and dilation of binary images by arbitrary structuring elements using interval coding. *Pattern Recognition Letters* 9 (3), 201–209.

Lau, L., S-W. Lee and C.Y. Suen (1992). Thinning methodologies. A comprehensive survey, *IEEE Trans. Pattern Analysis and Machine Intelligence* 14 (9), 869–885.

Pavlidis, T. (1982). *Algorithms for Graphics and Image Processing*, Springer, Berlin.

Piper, J. (1992). Interval skeletons. In: *Proc. 11th IAPR Internat. Conf. on Pattern Recognition*, 468–471.

Roseborough, J.B. and H. Murase (1995). Partial eigenvalue decomposition for large image sets using run-length encoding. *Pattern Recognition* 28 (3), 421–430.

Rosenfeld, A. and A.C. Kak (1982). *Digital Picture Processing*. Academic Press, New York.

Serra, J. (1992). *Image Analysis and Mathematical Morphology*. Academic Press, London.

Shih, F.Y. and C.C. Pu (1995). A skeletonization algorithm by maxima tracking on Euclidean distance transform. *Pattern Recognition* 28 (3), 331–341.

Suen, C.Y. and P.S.P. Wang, eds. (1993), *Thinning Methodologies for Pattern Recognition*. Special issue of the Internat. J. Pattern Recognition and Artificial Intelligence 7 (5).

Zakaria, M.F., L.J., Vroomen, P.J.A. Zsombor-Murray and J.M.H.M. van Kessel (1987). Fast algorithm for the computation of moment invariants. *Pattern Recognition* 20, 631–643.

Zhuang, X. (1994). Decomposition of morphological structuring elements. *J. Mathematical Imaging and Vision* 4 (1), 2–8.